CLAITON WERNER KÜSTER
FÁBIO ALEXANDRE IGNÁCIO
FILIPE PAIS LENFERS
LUIZ FABIANO VOIGT GARRETT
SÉRGIO PATRIQUE ZOTTO

EASYFAN

APÊNDICE I : CADERNO DE CÓDIGO FONTE

CURITIBA
2006

SUMÁRIO

1. CADERNO FONTE – EASYFAN

1.1 PACKAGE CONTROLE.EVENTOS;

1.1.1 ConfigurarArquivo.Java

```
package controle.eventos;

import java.awt.Frame;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;
import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;

public class ConfigurarArquivo extends AntiAliasedThinlet{

private static final long serialVersionUID = 1L;
      public ThinletDialog dialogTxt;
      public ThinletDialog dialogXls;
      public ThinletDialog dialog;
    public Thinlet thinlet;
    public int statusCancelar = 0;
    public boolean validacao= false;
      public int contadorLinhas = 0;
      public int contadorTokens = 0;
      public File file = null;
      public Object
txtSeparador,ckTabulacao,txtLinhaInicial,txtLinhaFinal,txtColunaInicial,t
xtColunaFinal,txtColunaClasse,btConfirmar = null;
    public int
intLinhaInicial,intLinhaFinal,intColunaInicial,intColunaFinal,intColunaCl
asse;
    public String stringSeparador;

      public void initParametersTxt(Object separador,Object
ckTabulacao,Object linhaInicial,Object linhaFinal,Object
colunaInicial,Object colunaFinal,Object colunaClasse,Object btConfirmar){
            this.txtSeparador = separador;
            this.ckTabulacao = ckTabulacao;
            this.txtLinhaInicial = linhaInicial;
            this.txtLinhaFinal = linhaFinal;
            this.txtColunaInicial = colunaInicial;
            this.txtColunaFinal = colunaFinal;
            this.txtColunaClasse = colunaClasse;
      }
```

```
      public void initParametersXls(Object xlsLinhaInicial,Object
xlsLinhaFinal,Object xlsColunaInicial,Object xlsColunaFinal,Object
xlsColunaClasse,Object btConfirmarXls){
            this.txtLinhaInicial = xlsLinhaInicial;
            this.txtLinhaFinal = xlsLinhaFinal;
            this.txtColunaInicial = xlsColunaInicial;
            this.txtColunaFinal = xlsColunaFinal;
            this.txtColunaClasse = xlsColunaClasse;
      }

    //carregar arquivos txt
    public void abrirTxt(String type, File file) {
      contadorLinhas = 0;
      this.file = file;
            ChamarDialogoTxt(new Frame(),"Configurações de carregamento
de arquivo *.txt",type);
        dialogTxt.pack();
        dialogTxt.setLocationRelativeTo(this.thinlet);
        if (type.equalsIgnoreCase("Classificacao")){
            dialogTxt.setSize(440,240);
         }
        else{
            dialogTxt.setSize(440,280);
        }
            //contador de linhas e colunas do arquivo
        BufferedReader arqDados = null;
            try {
                  arqDados = new BufferedReader(new
FileReader(this.file.getAbsolutePath()));
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }
                String linha = "";
            try {
                        while ((linha = arqDados.readLine()) != null) {
                              contadorLinhas++;
                        }
                  } catch (NumberFormatException e) {
                        e.printStackTrace();
                  } catch (IOException e) {
                        e.printStackTrace();
                  }

      setString(txtLinhaFinal,"text",Integer.toString(contadorLinhas));
            dialogTxt.setAlwaysOnTop(true);
        dialogTxt.setLocationRelativeTo(dialogTxt.getOwner());
        dialogTxt.setModal(true);
        dialogTxt.setVisible(true);
        dialogTxt.repaint();

    }

    public void ChamarDialogoTxt(Frame owner, String title,String type) {
      this.dialogTxt = new ThinletDialog(owner,title,false);
      this.initTxt(type);
    }
```

```
    private void initTxt(String type) {
      this.thinlet = new AntiAliasedThinlet();
      Object dialogo = null;
      String tela = null;
      if (type.equalsIgnoreCase("Classificacao")){
            tela =
"/visao/gui/telas/carregararquivoTxtClassificacao.xml";
      }
      else{
            tela = "/visao/gui/telas/carregararquivoTxt.xml";
      }
      try {
            dialogo = thinlet.parse(tela, this);
      } catch (IOException e) {
            // TODO Criar mensagem de erro na consturação
            e.printStackTrace();
      }
        thinlet.add(dialogo);
        dialogTxt.setContent(thinlet);
    }

    public void setParametrosTxt(){
      //new MessageDialog(dialog2, "Nenhum arquivo selecionado", "Por
favor selecione um arquivo.").show();
            this.stringSeparador =
getString(thinlet.find("txtSeparador"),"text");
            this.intLinhaInicial =
Integer.parseInt(getString(thinlet.find("txtLinhaInicial"),"text"));
            this.intLinhaFinal =
Integer.parseInt(getString(thinlet.find("txtLinhaFinal"), "text"));
            this.intColunaInicial =
Integer.parseInt(getString(thinlet.find("txtColunaInicial"),"text"));
            this.intColunaFinal =
Integer.parseInt(getString(thinlet.find("txtColunaFinal"),"text"));
            this.intColunaClasse =
Integer.parseInt(getString(thinlet.find("txtColunaClasse"),"text"));
            if (this.stringSeparador == null  ){
                  new MessageDialog(dialog, "Nenhum arquivo selecionado",
"Por favor Prencha corretamente os campos para poder abrir o
arquivo.").show();

            }
                this.file = null;
            dialogTxt.setVisible(false);

    }

    public void cancelarTxt(){
      statusCancelar = 1;
      dialogTxt.setVisible(false);

    }

    //carregar arquivos xls
    public void abrirXls(String tipo,File file){
            this.file = file;
```

```
      ChamarDialogoXls(new Frame(),"Configurações de carregamento de
arquivo *.xls",tipo);
        dialogXls.pack();
        dialogXls.setLocationRelativeTo(this.thinlet);
        if (tipo.equalsIgnoreCase("Classificacao")){
            dialogXls.setSize(450,180);
         }
        else{
            dialogXls.setSize(450,230);
        }
      //contador de linhas e colunas do arquivo
        Workbook workbook = null;
            try {
                  workbook = Workbook.getWorkbook(new
File(this.file.getAbsolutePath()));
            } catch (BiffException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
          Sheet sheet = workbook.getSheet(0);
          int linhaFinal = sheet.getRows();
          int colunaClasse = sheet.getColumns();
          int colunaFinal = colunaClasse - 1;
          setString(txtLinhaFinal,"text",Integer.toString(linhaFinal));
          setString(txtColunaFinal,"text",Integer.toString(colunaFinal));

setString(txtColunaClasse,"text",Integer.toString(colunaClasse));
        dialogXls.setAlwaysOnTop(true);
        dialogXls.setLocationRelativeTo(dialogXls.getOwner());
        dialogXls.setModal(true);
        dialogXls.setVisible(true);
        dialogXls.repaint();
    }

    public void ChamarDialogoXls(Frame owner, String title,String tipo) {
      this.dialogXls = new ThinletDialog(owner,title,false);
      this.initXls(tipo);
    }


    private void initXls(String tipo) {
      this.thinlet = new AntiAliasedThinlet();
      Object dialogo = null;
      String tela = null;
      if (tipo.equalsIgnoreCase("Classificacao")){
            tela =
"/visao/gui/telas/carregararquivoXlsClassificacao.xml";
      }
      else{
            tela = "/visao/gui/telas/carregararquivoXls.xml";
      }
            try {
            dialogo = thinlet.parse(tela, this);
      } catch (IOException e) {
```

```
            // TODO Criar mensagem de erro na consturação
            e.printStackTrace();
      }
        thinlet.add(dialogo);
        dialogXls.setContent(thinlet);
    }
    public void setParametrosXls(){
            this.intLinhaInicial =
Integer.parseInt(getString(thinlet.find("xlsLinhaInicial"),"text"));
            this.intLinhaFinal =
Integer.parseInt(getString(thinlet.find("xlsLinhaFinal"), "text"));
            this.intColunaInicial =
Integer.parseInt(getString(thinlet.find("xlsColunaInicial"),"text"));
            this.intColunaFinal =
Integer.parseInt(getString(thinlet.find("xlsColunaFinal"),"text"));
            this.intColunaClasse =
Integer.parseInt(getString(thinlet.find("xlsColunaClasse"),"text"));
        dialogXls.setVisible(false);

    }

    public void cancelarXls(){
      statusCancelar = 1;
      dialogXls.setVisible(false);
    }

    public void habilitarTabulacao(){
      if (this.getBoolean(ckTabulacao, "selected") == true ){
            setBoolean(txtSeparador,"enabled",false);
            setBoolean(txtSeparador,"editable",false);
            calculaColuna();
            validacao = true;
      }else{
            setBoolean(txtSeparador,"enabled",true);
            setBoolean(txtSeparador,"editable",true);
            validacao = false;
      }
      thinlet.repaint();
    }


    public String getSeparador(){
      if (validacao == true) {
            this.stringSeparador = "\t";
            return this.stringSeparador;
      }else{
      return this.stringSeparador;
      }
    }

    public void calculaColuna(){
      contadorTokens = 0;
      String sp = null;
      if (this.getBoolean(ckTabulacao, "selected") == true ){
            sp = "\t";
      }else{
            sp = getString(txtSeparador,"text").toString();
```

```
      }
      BufferedReader arqDados = null;
            try {
                  arqDados = new BufferedReader(new
FileReader(this.file.getAbsolutePath()));
            } catch (FileNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
                String linha = "";
               try {
                        while ((linha = arqDados.readLine()) != null) {
                                    StringTokenizer t = new
StringTokenizer(linha, sp);
                                    while (t.hasMoreTokens()){
                                          t.nextToken();
                                          contadorTokens++;
                                    }
                              }
                        } catch (IOException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                        }
                    catch (NumberFormatException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                  }
                  if (contadorTokens == 0){

                  }
                      contadorTokens = contadorTokens/contadorLinhas;
                  int colClasse = contadorTokens;
                  int colFinal = colClasse - 1;
                  if (contadorTokens < 2 ){
                        setString(txtColunaFinal,"text","");
                        setString(txtColunaClasse,"text","");
                  }else{

      setString(txtColunaFinal,"text",Integer.toString(colFinal));

      setString(txtColunaClasse,"text",Integer.toString(colClasse));
                  }
                  thinlet.repaint();
    }

    public int getColunaClasse(){
      return this.intColunaClasse;
    }
    public int getColunaInicial(){
      return this.intColunaInicial;
    }
    public int getColunaFinal(){
      return this.intColunaFinal;
    }
    public int getLinhaInicial(){
      return this.intLinhaInicial;
    }
```

```
    public int getLinhaFinal(){
      return this.intLinhaFinal;
    }
    public int getStatusCancelar(){
      return this.statusCancelar;
    }

}
```

1.1.2 EvtAbrirArquivo.java

```
package controle.eventos;

import java.awt.Frame;
import java.io.File;
import java.util.ArrayList;

import javax.swing.JPopupMenu.Separator;

import modelo.RedeConfig;

import jfan.fan.NeuronioFAN;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.CarregaNeuronioXML;
import jfan.io.arquivos.GerenciadorPadroes;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ExtensionFileFilter;
import thinletcommons.FileChooser;
import thinletcommons.FileFilter;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class EvtAbrirArquivo extends AntiAliasedThinlet{

      private static final long serialVersionUID = 1L;
      private String[][] array = null;
    private ConfigurarArquivo conarq = new ConfigurarArquivo();
    public String type = null;
    public String[] allFilesSuported = {"xml","dat","txt","xls"};

    private FileFilter[] filtro = new FileFilter[]{
            new FileFilter(){
              public boolean accept(File dir) {
                  if (dir.getAbsolutePath().indexOf(".dat") > -1 ||
dir.getAbsolutePath().indexOf(".epd") > -1 ||
dir.getAbsolutePath().indexOf(".txt") > -1 ||
dir.getAbsolutePath().indexOf(".xls") > -1) {
                      return true;
                  }
                  return false;
              }
              public String getDescription() { return "Arquivos
suportados (epd,dat,txt,xls)"; }
```

```
          },
            new ExtensionFileFilter("epd", "EasyFAN files (*.epd)"),
                  new ExtensionFileFilter("dat", "DAT files (*.dat)"),
                  new ExtensionFileFilter("txt", "TXT files (*.txt)"),
                  new ExtensionFileFilter("xls", "XLS files (*.xls)"),

        };


public  String[][] abrirTreinamento(){
      this.type = "Treinamento";
      String[][] arrayResult= null;
      try {
            FileChooser f = new FileChooser(new Frame(),"Abrir conjunto
treinamento",FileChooser.MODE_OPEN);
            f.setFileFilters(filtro);
            f.setSelectedFile(App.getUltimoArquivo());
            f.setSelectedFiler(App.getUltimoFiltro());

            f.show();

            File file = f.getSelectedFile();
            if (file != null)
                  App.setUltimoArquivo(file);
            App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());


            if (file != null){
                  String path = file.getAbsolutePath();
                  GerenciadorPadroes ga = App.getGerenciadorPadroes();
                  String formatoArquivo = path.substring(path.length()-
3,path.length()).toUpperCase();
                  if (formatoArquivo.equals("TXT")){
                        conarq.abrirTxt(this.type,file);
                        if (conarq.getStatusCancelar() != 1){

      ga.adicionarConjuntoTreinamento(path,conarq.getSeparador(),conarq.g
etColunaClasse(),conarq.getColunaInicial(),conarq.getColunaFinal(),conarq
.getLinhaInicial(),conarq.getLinhaFinal());
                        }
                  } else      if (formatoArquivo.equals("XLS")){
                        conarq.abrirXls(this.type,file);
                        if (conarq.getStatusCancelar() != 1){

      ga.adicionarConjuntoTreinamento(path,null,conarq.getColunaClasse(),
conarq.getColunaInicial(),conarq.getColunaFinal(),conarq.getLinhaInicial(
),conarq.getLinhaFinal());
                        }
                  }else
                        ga.adicionarConjuntoTreinamento(path,"
",0,0,0,0,0);
                        array = ga.getConjuntoTreinamento();
                        arrayResult=array;

                        if
(App.getProjetoConfig().getRedeConf().getMaximosAutomaticos() == null) {
```

```
                              int fim =
ga.getConjuntoTreinamentoRede().get(0).getQuantasCaracteristicas();
                              boolean[] bb = new boolean[fim];
                              for (int i = 0; i< fim;i++) bb[i] = true;

      App.getProjetoConfig().getRedeConf().setMaximosAutomaticos(bb);

                        }

                        if
(App.getProjetoConfig().getRedeConf().getMinimosAutomaticos() == null) {
                              int fim =
ga.getConjuntoTreinamentoRede().get(0).getQuantasCaracteristicas();
                              boolean[] bb = new boolean[fim];
                              for (int i = 0; i< fim;i++) bb[i] = true;

      App.getProjetoConfig().getRedeConf().setMinimosAutomaticos(bb);

                        }

                        if
(App.getProjetoConfig().getRedeConf().getMediasAutomaticas() == null) {
                              int fim =
ga.getConjuntoTreinamentoRede().get(0).getQuantasCaracteristicas();
                              boolean[] bb = new boolean[fim];
                              for (int i = 0; i< fim;i++) bb[i] = true;

      App.getProjetoConfig().getRedeConf().setMediasAutomaticas(bb);

                        }

                        double[] dd;// = new double[fim];
                        GerenciadorPadroes g =
App.getGerenciadorPadroes();
                        dd =
DetectorMaxMinMean.procurarMaximos(g.getConjuntoClassificacaoRede(),g.get
ConjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());

                        if
(App.getProjetoConfig().getRedeConf().getMaximos() == null) {

      App.getProjetoConfig().getRedeConf().setMaximos(dd);
                        }
                        else {
                              double[] arrayD =
App.getProjetoConfig().getRedeConf().getMaximos();
                              boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMaximosAutomaticos();
                              for (int i =0; i < arrayD.length; i++ ) {
                                    if (arrayB[i]) {
                                          arrayD[i] =  dd[i];
                                    }
                              }
                        }
```

```
                        dd =
DetectorMaxMinMean.procurarMinimos(g.getConjuntoClassificacaoRede(),g.get
ConjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());
                        if
(App.getProjetoConfig().getRedeConf().getMinimos() == null) {

      App.getProjetoConfig().getRedeConf().setMinimos(dd);
                        }
                        else {
                              double[] arrayD =
App.getProjetoConfig().getRedeConf().getMinimos();
                              boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMinimosAutomaticos();
                              for (int i =0; i < arrayD.length; i++ ) {
                                    if (arrayB[i]) {
                                          arrayD[i] =  dd[i];
                                    }
                              }
                        }

                        dd =
DetectorMaxMinMean.procurarMedias(g.getConjuntoClassificacaoRede(),g.getC
onjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());
                        if
(App.getProjetoConfig().getRedeConf().getMedias() == null) {

      App.getProjetoConfig().getRedeConf().setMedias(dd);
                        }
                        else {
                              double[] arrayD =
App.getProjetoConfig().getRedeConf().getMedias();
                              boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMediasAutomaticas();
                              for (int i =0; i < arrayD.length; i++ ) {
                                    if (arrayB[i]) {
                                          arrayD[i] =  dd[i];
                                    }
                              }
                        }


                        App.vericarMonitor();

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
                  }
            }
            catch(Exception e){
                        e.printStackTrace();
                  }
            App.setPrecisaNormalizar(true);
      return arrayResult;
}
public  String[][] abrirTeste(){
      this.type = "Teste";
```

```
	String[][] arrayResult= null;
	try {
	FileChooser f = new FileChooser(new Frame(),"Abrir conjunto de
teste",FileChooser.MODE_OPEN);
	f.setFileFilters(filtro);
	f.setSelectedFile(App.getUltimoArquivo());
		f.setSelectedFiler(App.getUltimoFiltro());

		f.show();

		File file = f.getSelectedFile();
		if (file != null)
			App.setUltimoArquivo(file);
		App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());
		if (file != null){
			String path = file.getAbsolutePath();
			String formatoArquivo = path.substring(path.length()-
3,path.length()).toUpperCase();
			GerenciadorPadroes ga = App.getGerenciadorPadroes();
			if (formatoArquivo.equals("TXT")){
				conarq.abrirTxt(this.type,file);

	ga.adicionarConjuntoTeste(path,conarq.getSeparador(),conarq.getColu
naClasse(),conarq.getColunaInicial(),conarq.getColunaFinal(),conarq.getLi
nhaInicial(),conarq.getLinhaFinal());
			}
			else if (formatoArquivo.equals("XLS")){
				conarq.abrirXls(this.type,file);

	ga.adicionarConjuntoTeste(path,null,conarq.getColunaClasse(),conarq
.getColunaInicial(),conarq.getColunaFinal(),conarq.getLinhaInicial(),cona
rq.getLinhaFinal());
			}else
				ga.adicionarConjuntoTeste(path,"  ",0,0,0,0,0);
			array = ga.getConjuntoTeste();
			arrayResult=array;
			App.vericarMonitor();

			if
(App.getProjetoConfig().getRedeConf().getMaximosAutomaticos() == null) {
				int fim =
ga.getConjuntoTesteRede().get(0).getQuantasCaracteristicas();
				boolean[] bb = new boolean[fim];
				for (int i = 0; i< fim;i++) bb[i] = true;

	App.getProjetoConfig().getRedeConf().setMaximosAutomaticos(bb);

			}

			if
(App.getProjetoConfig().getRedeConf().getMinimosAutomaticos() == null) {
				int fim =
ga.getConjuntoTesteRede().get(0).getQuantasCaracteristicas();
				boolean[] bb = new boolean[fim];
				for (int i = 0; i< fim;i++) bb[i] = true;

	App.getProjetoConfig().getRedeConf().setMinimosAutomaticos(bb);
```

```
                }

                if
(App.getProjetoConfig().getRedeConf().getMediasAutomaticas() == null) {
                      int fim =
ga.getConjuntoTesteRede().get(0).getQuantasCaracteristicas();
                      boolean[] bb = new boolean[fim];
                      for (int i = 0; i< fim;i++) bb[i] = true;

      App.getProjetoConfig().getRedeConf().setMediasAutomaticas(bb);

                }

                double[] dd;// = new double[fim];
                GerenciadorPadroes g =  App.getGerenciadorPadroes();
                dd =
DetectorMaxMinMean.procurarMaximos(g.getConjuntoClassificacaoRede(),g.get
ConjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());

                if (App.getProjetoConfig().getRedeConf().getMaximos()
== null) {

      App.getProjetoConfig().getRedeConf().setMaximos(dd);
                }
                else {
                      double[] arrayD =
App.getProjetoConfig().getRedeConf().getMaximos();
                      boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMaximosAutomaticos();
                      for (int i =0; i < arrayD.length; i++ ) {
                            if (arrayB[i]) {
                                  arrayD[i] =  dd[i];
                            }
                      }
                }

                dd =
DetectorMaxMinMean.procurarMinimos(g.getConjuntoClassificacaoRede(),g.get
ConjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());
                if (App.getProjetoConfig().getRedeConf().getMinimos()
== null) {

      App.getProjetoConfig().getRedeConf().setMinimos(dd);
                }
                else {
                      double[] arrayD =
App.getProjetoConfig().getRedeConf().getMinimos();
                      boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMinimosAutomaticos();
                      for (int i =0; i < arrayD.length; i++ ) {
                            if (arrayB[i]) {
                                  arrayD[i] =  dd[i];
                            }
                      }
```

```
                  }

                  dd =
DetectorMaxMinMean.procurarMedias(g.getConjuntoClassificacaoRede(),g.getC
onjuntoTesteRede(), g.getConjuntoTreinamentoRede(),
g.getConjuntoValidacaoRede());
                  if (App.getProjetoConfig().getRedeConf().getMedias() ==
null) {

      App.getProjetoConfig().getRedeConf().setMedias(dd);
                  }
                  else {
                        double[] arrayD =
App.getProjetoConfig().getRedeConf().getMedias();
                        boolean[] arrayB =
App.getProjetoConfig().getRedeConf().getMediasAutomaticas();
                        for (int i =0; i < arrayD.length; i++ ) {
                              if (arrayB[i]) {
                                    arrayD[i] =  dd[i];
                              }
                        }
                  }


            }
            if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0) {
                  App.vericarMonitor();

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
            }
            else {
                  App.vericarMonitor();

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTeste());
            }


      }
      catch(Exception e){
                  e.printStackTrace();
            }



      App.setPrecisaNormalizar(true);
      return arrayResult;
}
public  String[][] abrirClassificacao(){
      this.type = "Classificacao";
      String[][] arrayResult= null;
      try {

      FileChooser f = new FileChooser(new Frame(),"Abrir conjunto
classificaï¿½ï¿½o",FileChooser.MODE_OPEN);
```

```
	f.setFileFilters(filtro);
	f.setSelectedFile(App.getUltimoArquivo());
		f.setSelectedFiler(App.getUltimoFiltro());

		f.show();

		File file = f.getSelectedFile();
		if (file != null)
			App.setUltimoArquivo(file);
		App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());
		if (file != null){
			file.getName();
			String path = file.getAbsolutePath();
			String formatoArquivo = path.substring(path.length()-
3,path.length()).toUpperCase();
			GerenciadorPadroes ga = App.getGerenciadorPadroes();
			if (formatoArquivo.equals("TXT")){
				conarq.abrirTxt(this.type,file);

	ga.adicionarConjuntoClassificacao(path,conarq.getSeparador(),conarq
.getColunaClasse(),conarq.getColunaInicial(),conarq.getColunaFinal(),cona
rq.getLinhaInicial(),conarq.getLinhaFinal());
			}else if (formatoArquivo.equals("XLS")){
				conarq.abrirXls(this.type,file);

	ga.adicionarConjuntoClassificacao(path,null,conarq.getColunaClasse(
),conarq.getColunaInicial(),conarq.getColunaFinal(),conarq.getLinhaInicia
l(),conarq.getLinhaFinal());
			}else
				ga.adicionarConjuntoClassificacao(path,"
",0,0,0,0,0);
			array = ga.getConjuntoClassificacao();
			arrayResult=array;
		}

		if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0) {
			App.vericarMonitor();

	App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
		}
		else {
			App.vericarMonitor();

	App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoClassificacao());
		}
	}
	catch(Exception e){
			e.printStackTrace();
		}
	App.setPrecisaNormalizar(true);



	return arrayResult;
```

```
}
public  String[][] abrirValidacao(){
      this.type = "Validacao";
      String[][] arrayResult= null;
      try {
      FileChooser f = new FileChooser(new Frame(),"Abrir conjunto
validacaï¿½ï¿½o",FileChooser.MODE_OPEN);
      f.setFileFilters(filtro);
      f.setSelectedFile(App.getUltimoArquivo());
            f.setSelectedFiler(App.getUltimoFiltro());

            f.show();

            File file = f.getSelectedFile();
            if (file != null)
                  App.setUltimoArquivo(file);
            App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());
            if(file != null){
                  file.getName();
                  String path = file.getAbsolutePath();
                  String formatoArquivo = path.substring(path.length()-
3,path.length()).toUpperCase();
                  GerenciadorPadroes ga = App.getGerenciadorPadroes();
                  if (formatoArquivo.equals("TXT")){
                        conarq.abrirTxt(this.type,file);

      ga.adicionarConjuntoValidacao(path,conarq.getSeparador(),conarq.get
ColunaClasse(),conarq.getColunaInicial(),conarq.getColunaFinal(),conarq.g
etLinhaInicial(),conarq.getLinhaFinal());
                  }else if (formatoArquivo.equals("XLS")){
                        conarq.abrirXls(this.type,file);

      ga.adicionarConjuntoValidacao(path,null,conarq.getColunaClasse(),co
narq.getColunaInicial(),conarq.getColunaFinal(),conarq.getLinhaInicial(),
conarq.getLinhaFinal());
                  }else
                        ga.adicionarConjuntoValidacao(path,"
",0,0,0,0,0);
                  array = ga.getConjuntoValidacao();
                  arrayResult=array;
            }
            if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0) {
                  App.vericarMonitor();

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
            }
            else {
                  App.vericarMonitor();

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoValidacao());
            }
      }
      catch(Exception e){
                  e.printStackTrace();
```

```
            }
      App.setPrecisaNormalizar(true);
      return arrayResult;

}

      public CarregaNeuronioXML abrirRede() {
            FileChooser fc = new
FileChooser(App.getTelaEasyFAN().getFrame(),"Abrir Rede",
FileChooser.MODE_OPEN);
            fc.setFileFilters(new FileFilter[]{
                        new FileFilter(){
                              public boolean accept(File dir) {
                                    if
(dir.getAbsolutePath().indexOf(".dat") > -1 ||
dir.getAbsolutePath().indexOf(".enn") > -1) {
                                          return true;
                                    }
                                    return false;
                              }
                              public String getDescription() { return
"Arquivos suportados (enn,dat)"; }
                        },
                          new ExtensionFileFilter("enn", "EasyFAN files
(*.enn)"),
                          new ExtensionFileFilter("dat", "DAT files
(*.dat)"),
             });
            fc.show();
            File selectedFile = fc.getSelectedFile();
            String enderecoDoArquivo = selectedFile.getAbsolutePath();
            String extArquivo =
selectedFile.getName();//fc.getFilterChanged();
            extArquivo = extArquivo.substring(extArquivo.length()-
4,extArquivo.length());
            if (extArquivo.equalsIgnoreCase(".enn")){
                  CarregaNeuronioXML cxml = new
CarregaNeuronioXML(enderecoDoArquivo,App.getGerenciadorPadroes());
                  try {
                        return cxml;
                  } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                        return null;
                  }


            }
            else {
                  MessageDialog msg = new
MessageDialog(fc.getDialog(),"Arquivo invïálido","Tipo de arquivo
invï¿½lido.");
                  msg.show();
            }
            return null;

      }
```

```
	public void abrirRedeTreinamento() {
		try {
			CarregaNeuronioXML cxml = abrirRede();
			RedeConfig rc = App.getProjetoConfig().getRedeConf();
			rc.setMaximos(cxml.getMaximos());
			rc.setMinimos(cxml.getMinimos());
			rc.setMedias(cxml.getMedias());
			rc.setTipoNormalizador(cxml.getTipoNormalizacao());
		}
		catch (Exception e) {
			e.printStackTrace();
		}

	}

	public ArrayList<NeuronioFAN> abrirRedeSomenteNeuronios() {
		try {
			CarregaNeuronioXML cxml = abrirRede();
			ArrayList<NeuronioFAN> arr = cxml.getNeuroniosFAN();
			int fim = arr.size();
			int[] classes = new int[fim];
			for(int i = 0; i < fim; i++) {
				classes[i] = arr.get(i).getClasseAssociada();
			}
			RecalculadorClasses.indexarNeuronios(classes, arr);
			return cxml.getNeuroniosFAN();
		}
		catch (Exception e) {
			e.printStackTrace();
		}
		return null;
	}

	public CarregaNeuronioXML abrirRedeCarregaXML() {
		try {
			CarregaNeuronioXML cxml = abrirRede();
			ArrayList<NeuronioFAN> arr = cxml.getNeuroniosFAN();
			//RecalculadorClasses.indexarNeuronios(classes, arr);
			return cxml;
		}
		catch (Exception e) {
			e.printStackTrace();
		}
		return null;
	}

}
```

1.1.3 EvtClassificacao.Java

```
package controle.eventos;

import java.util.ArrayList;

import controle.main.App;
```

```
import visao.gui.telas.classes.EasyFan;
import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;

public class EvtClassificacao {

	private final EasyFan easyfan;

	public EvtClassificacao(EasyFan easyfan) {
		this.easyfan = easyfan;
	}


	public void classificar(StoredNetworkType tipoRede) {
		App.requisitarNormalizacao();
		App.getMonitorFAN().classificar(tipoRede);
		App.getGerenciadorPadroes().atualizarConjuntoClassificacao();
		easyfan.atualizarTabelaClassificacao();

	}

	public void classificar(ArrayList<NeuronioFAN> neuronios) {
		App.requisitarNormalizacao();
		App.getMonitorFAN().classificar(neuronios);
		App.getGerenciadorPadroes().atualizarConjuntoClassificacao();
		easyfan.atualizarTabelaClassificacao();

	}

}
```

1.1.4 EvtLimparConjunto.Java

```
package controle.eventos;

import thinletcommons.MessageDialog;
import visao.gui.telas.classes.ReiniciarConfig;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.GerenciadorPadroes;
import controle.main.App;

public class EvtLimparConjunto {

	public static void limparConjuntoTreinamento(){
		App.getGerenciadorPadroes().limparConjuntoTreinamento();
		GerenciadorPadroes gp = App.getGerenciadorPadroes();
		try {
			if ((gp.getConjuntoClassificacaoRede().size() == 0) &&
(gp.getConjuntoTesteRede().size() == 0) &&
(gp.getConjuntoValidacaoRede().size() == 0)) {

	App.getProjetoConfig().getRedeConf().setMaximos(null);

	App.getProjetoConfig().getRedeConf().setMinimos(null);

	App.getProjetoConfig().getRedeConf().setMedias(null);
```

```
      RecalculadorClasses.getClassesMapIndexReal().clear();

      RecalculadorClasses.getClassesMapRealIndex().clear();
                        MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenção","Todos os
conjuntos estãoo vazios.\n Deseja reiniciar a
rede?",MessageDialog.MODE_YES_NO);
                        int resposta = msg.show();
                        if (resposta == MessageDialog.ACTION_YES) {
                              App.reiniciarGerenciadorPadroes();
                              App.iniciaRede();
                        }
                  }
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }
      public static  void limparConjuntoTeste(){
            App.getGerenciadorPadroes().limparConjuntoTeste();
            GerenciadorPadroes gp = App.getGerenciadorPadroes();
            try {
                  if ((gp.getConjuntoClassificacaoRede().size() == 0) &&
(gp.getConjuntoTreinamentoRede().size() == 0) &&
(gp.getConjuntoValidacaoRede().size() == 0)) {

      App.getProjetoConfig().getRedeConf().setMaximos(null);

      App.getProjetoConfig().getRedeConf().setMinimos(null);

      App.getProjetoConfig().getRedeConf().setMedias(null);

      RecalculadorClasses.getClassesMapIndexReal().clear();

      RecalculadorClasses.getClassesMapRealIndex().clear();
                        MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenção","Todos os
conjuntos estão vazios.\n Deseja reiniciar a
rede?",MessageDialog.MODE_YES_NO);
                        int resposta = msg.show();
                        if (resposta == MessageDialog.ACTION_YES) {
                              App.reiniciarGerenciadorPadroes();
                              App.iniciaRede();
                        }
                  }
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }
    public static void limparConjuntoValidacao(){
      App.getGerenciadorPadroes().limparConjuntoValidacao();
      GerenciadorPadroes gp = App.getGerenciadorPadroes();
            try {
```

```
                  if ((gp.getConjuntoClassificacaoRede().size() == 0) &&
(gp.getConjuntoTreinamentoRede().size() == 0) &&
(gp.getConjuntoTesteRede().size() == 0)) {

      App.getProjetoConfig().getRedeConf().setMaximos(null);

      App.getProjetoConfig().getRedeConf().setMinimos(null);

      App.getProjetoConfig().getRedeConf().setMedias(null);

      RecalculadorClasses.getClassesMapIndexReal().clear();

      RecalculadorClasses.getClassesMapRealIndex().clear();
                        MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenção","Todos os
conjuntos estão vazios.\n Deseja reiniciar a
rede?",MessageDialog.MODE_YES_NO);
                        int resposta = msg.show();
                        if (resposta == MessageDialog.ACTION_YES) {
                              App.reiniciarGerenciadorPadroes();
                              App.iniciaRede();
                        }
                  }
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }
    public static void limparConjuntoClassificacao(){
      App.getGerenciadorPadroes().limparConjuntoClassificacao();
      GerenciadorPadroes gp = App.getGerenciadorPadroes();
            try {
                  if ((gp.getConjuntoValidacaoRede().size() == 0) &&
(gp.getConjuntoTreinamentoRede().size() == 0) &&
(gp.getConjuntoTesteRede().size() == 0)) {

      App.getProjetoConfig().getRedeConf().setMaximos(null);

      App.getProjetoConfig().getRedeConf().setMinimos(null);

      App.getProjetoConfig().getRedeConf().setMedias(null);

      RecalculadorClasses.getClassesMapIndexReal().clear();

      RecalculadorClasses.getClassesMapRealIndex().clear();
                        MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenção","Todos os
conjuntos estãoo vazios.\n Deseja reiniciar a
rede?",MessageDialog.MODE_YES_NO);
                        int resposta = msg.show();
                        if (resposta == MessageDialog.ACTION_YES) {
                              App.reiniciarGerenciadorPadroes();
                              App.iniciaRede();
                        }
                  }
            }
            catch (Exception e) {
```

```
                e.printStackTrace();
            }
}
}
```

1.1.5 EvtNormalização.Java

```
package controle.eventos;

import java.util.ArrayList;

import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import controle.main.App;

public class EvtNormalizacao {

      public static void arrumarParametrosNormalizacao() {
            ArrayList<IPadrao> tr,ts,vl,cl;
            try {
                  tr =
App.getGerenciadorPadroes().getConjuntoTreinamentoRede();
                  ts =
App.getGerenciadorPadroes().getConjuntoTesteRede();
                  vl =
App.getGerenciadorPadroes().getConjuntoValidacaoRede();
                  cl =
App.getGerenciadorPadroes().getConjuntoClassificacaoRede();

                  double[] detectMax =
DetectorMaxMinMean.procurarMaximos(tr, ts, vl, cl);
                  double[] detectMin =
DetectorMaxMinMean.procurarMinimos(tr, ts, vl, cl);
                  double[] detectMean =
DetectorMaxMinMean.procurarMedias(tr, ts, vl, cl);

                  boolean[] autoMax =
App.getProjetoConfig().getRedeConf().getMaximosAutomaticos();
                  boolean[] autoMin =
App.getProjetoConfig().getRedeConf().getMinimosAutomaticos();
                  boolean[] autoMean =
App.getProjetoConfig().getRedeConf().getMediasAutomaticas();

                  double[] max =
App.getProjetoConfig().getRedeConf().getMaximos();
                  double[] min =
App.getProjetoConfig().getRedeConf().getMinimos();
                  double[] mean =
App.getProjetoConfig().getRedeConf().getMedias();
```

```
                  for (int i = 0; i < autoMax.length; i++) {
                        if (autoMax[i]){
                              max[i] = detectMax[i];
                        }
                  }

                  for (int i = 0; i < autoMin.length; i++) {
                        if (autoMin[i]){
                              min[i] = detectMin[i];
                        }
                  }

                  for (int i = 0; i < autoMean.length; i++) {
                        if (autoMean[i]){
                              mean[i] = detectMean[i];
                        }
                  }
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public static void requisitarNormalizacao() {
            if(App.isPrecisaNormalizar()) {
                  //firstRunning = false;
                  ArrayList<IPadrao> tr,ts,vl,cl;
                  try {
                        tr =
App.getGerenciadorPadroes().getConjuntoTreinamentoRede();
                        ts =
App.getGerenciadorPadroes().getConjuntoTesteRede();
                        vl =
App.getGerenciadorPadroes().getConjuntoValidacaoRede();
                        cl =
App.getGerenciadorPadroes().getConjuntoClassificacaoRede();

                        INormalizator norm = null;

                        NormalizatorConfiguration config =
App.getProjetoConfig().getRedeConf().getNormalizatorConfiguration();

//

      config.setMaximos(App.getProjetoConfig().getRedeConf().getMaximos()
);

      config.setMinimos(App.getProjetoConfig().getRedeConf().getMinimos()
);

      config.setMedias(App.getProjetoConfig().getRedeConf().getMedias());


                        NormalizatorTypes tipo =
App.getProjetoConfig().getRedeConf().getTipoNormalizador();
                        //Os testes para saber qual normalizador usar,
```

```
                        //o normalizador, caso nao se tenha escolhido
normalizador, é  o NormalizatorMax, por isso ele será considerado no
else.
                        //O padrão do easyfan é o NormalizatorMaxMin.
                        if (tipo == NormalizatorTypes.NormalizatorMaxMin)
{
                              norm = new NormalizatorMaxMin(config);
                        }
                        else if (tipo ==
NormalizatorTypes.NormalizatorMaxMean) {
                              norm = new NormalizatorMaxMean(config);
                        }
                        else if (tipo ==
NormalizatorTypes.NormalizatorArcSinMaxMin) {
                        }
                        else{
                              norm = new NormalizatorMax(config);
                        }

                        if (tr.size() > 0) norm.normalizarThreaded(tr);
                        if (ts.size() > 0) norm.normalizarThreaded(ts);
                        if (vl.size() > 0) norm.normalizarThreaded(vl);
                        if (cl.size() > 0) norm.normalizarThreaded(cl);


                        CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(App.getProjetoConfig().getRedeConf().getR
aioDifuso(),
App.getProjetoConfig().getRedeConf().getSuporteConjuntosDifusos());
                        if (tr.size() > 0)
calc.calcularCaracteristicasFAN(tr);
                        if (ts.size() > 0)
calc.calcularCaracteristicasFAN(ts);
                        if (vl.size() > 0)
calc.calcularCaracteristicasFAN(vl);
                        if (cl.size() > 0)
calc.calcularCaracteristicasFAN(cl);

                              App.setPrecisaNormalizar(false);
                  }
                  catch (Exception e) {
                        e.printStackTrace();
                  }

            }
      }
}
```

1.1.6 EvtSalvarArquivo.Java

```
package controle.eventos;

import java.awt.Frame;
import java.io.File;
import jfan.io.arquivos.PersistorDados;
import controle.main.App;
```

```
import thinletcommons.ExtensionFileFilter;
import thinletcommons.FileChooser;
import thinletcommons.FileFilter;
import thinletcommons.MessageDialog;

public class EvtSalvarArquivo {


      public static void salvarConjuntoTreinamento() {
            FileChooser f = new FileChooser(new Frame(), "Salvar conjunto
de treinamento",  FileChooser.MODE_SAVE);
            f.setFileFilters(new FileFilter[] {
                        new ExtensionFileFilter("epd", "EasyFAN files
(*.epd)"),
                        new ExtensionFileFilter("dat", "DAT files
(*.dat)")}

                  );

      f.show();

            File file = f.getSelectedFile();
            if (file != null)
                  App.setUltimoArquivo(file);
            App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());

            if (file != null) {
                  PersistorDados pd = new PersistorDados();
                  try {
                        if (f.getSelectedFileFilter().getDescription()
                                    .equalsIgnoreCase("DAT files
(*.dat)"))

      pd.gravaPadroesDat(App.getGerenciadorPadroes()
                                          .getConjuntoTreinamentoRede(),
file
                                          .getAbsolutePath()
                                          + ".dat");
                        else if
(f.getSelectedFileFilter().getDescription()
                                    .equalsIgnoreCase("EasyFAN files
(*.epd)")) {
                              String[] nomeCarac;
                              if (App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento() == null) {

                                    nomeCarac = new
String[App.getGerenciadorPadroes()

      .getConjuntoTreinamentoRede().get(0)

      .getQuantasCaracteristicas()];
                                    for (int i = 0; i < nomeCarac.length;
i++)
                                          nomeCarac[i] = "Caracteristica "
```

```
                                                        + String.valueOf(i +
1);
                              } else
                                    nomeCarac =
App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento();



      pd.gravaPadroesXml(App.getGerenciadorPadroes(),App.getGerenciadorPa
droes()
                                          .getConjuntoTreinamentoRede(),
file
                                          .getAbsolutePath()
                                          + ".epd",
f.getSelectedFile().getName(),
                                          "Treinamento", nomeCarac);

                        }
                  } catch (Exception e) {
                        new MessageDialog(f.getDialog(), "Erro ao
persistir",
                                    "Um erro ocorreu ao tentar salvar o
arquivo.").show();
                        e.printStackTrace();
                  }
            }
      }

      public static void salvarConjuntoTeste() {
            if (permitePersistir()){

                  FileChooser f = new FileChooser(new Frame(), "Salvar
conjunto de teste",
                              FileChooser.MODE_SAVE);
                  f.setFileFilters(new FileFilter[] {
                              new ExtensionFileFilter("epd", "EasyFAN
files (*.epd)"),
                              new ExtensionFileFilter("dat", "DAT files
(*.dat)")
             });

                  f.setSelectedFile(App.getUltimoArquivo());
                  f.setSelectedFiler(App.getUltimoFiltro());

                  f.show();

                  File file = f.getSelectedFile();
                  if (file != null)
                        App.setUltimoArquivo(file);
                  App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());

                  if (file != null) {
                        PersistorDados pd = new PersistorDados();
                        try {
```

```
                                if
(f.getSelectedFileFilter().getDescription()
                                                .equalsIgnoreCase("DAT files
(*.dat)"))

      pd.gravaPadroesDat(App.getGerenciadorPadroes()
                                                      .getConjuntoTesteRede(),
file
                                                      .getAbsolutePath()
                                                      + ".dat");
                                else if
(f.getSelectedFileFilter().getDescription()
                                                .equalsIgnoreCase("EasyFAN files
(*.epd)")) {
                                          String[] nomeCarac = null;
                                          if (App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento() == null) {

                                                nomeCarac = new
String[App.getGerenciadorPadroes()

      .getConjuntoTesteRede().get(0)

      .getQuantasCaracteristicas()];
                                                for (int i = 0; i <
nomeCarac.length; i++)
                                                      nomeCarac[i] =
"Caracteristica "
                                                                  +
String.valueOf(i + 1);
                                          } else
                                                nomeCarac =
App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento();


      pd.gravaPadroesXml(App.getGerenciadorPadroes(),App.getGerenciadorPa
droes()

      .getConjuntoTesteRede(), file
                                                            .getAbsolutePath()
                                                            + ".epd",
f.getSelectedFile().getName(),
                                                            "Teste", nomeCarac);
                                }
                          } catch (Exception e) {
                                new MessageDialog(f.getDialog(), "Erro ao
persistir",
                                                "Um erro ocorreu ao tentar
salvar o arquivo.").show();
                                e.printStackTrace();
                          }
                    }
              }
      }
```

```
      public static void salvarConjuntoClassificacao() {
            if (permitePersistir()){

                  FileChooser f = new FileChooser(new Frame(), "Salvar
conjunto de classificacï¿½o",
                              FileChooser.MODE_SAVE);
                  f.setFileFilters(new FileFilter[] {
                              new ExtensionFileFilter("epd", "EasyFAN
files (*.epd)"),
                              new ExtensionFileFilter("dat", "DAT files
(*.dat)")
                  });

                  f.setSelectedFile(App.getUltimoArquivo());
                  f.setSelectedFiler(App.getUltimoFiltro());

                  f.show();

                  File file = f.getSelectedFile();
                  if (file != null)
                        App.setUltimoArquivo(file);
                  App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());

                  if (file != null) {
                        PersistorDados pd = new PersistorDados();
                        try {
                              if
(f.getSelectedFileFilter().getDescription()
                                          .equalsIgnoreCase("DAT files
(*.dat)"))

      pd.gravaPadroesDat(App.getGerenciadorPadroes()

      .getConjuntoClassificacaoRede(), file
                                                .getAbsolutePath()
                                                + ".dat");
                              else if
(f.getSelectedFileFilter().getDescription()
                                          .equalsIgnoreCase("EasyFAN files
(*.epd)")) {
                                    String[] nomeCarac;
                                    if (App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento() == null) {
                                          nomeCarac = new
String[App.getGerenciadorPadroes()

      .getConjuntoClassificacaoRede().get(0)

      .getQuantasCaracteristicas()];
                                          for (int i = 0; i <
nomeCarac.length; i++)
                                                nomeCarac[i] =
"Caracteristica "
                                                      +
String.valueOf(i + 1);
```

```
                                    } else
                                          nomeCarac =
App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento();


      pd.gravaPadroesXml(App.getGerenciadorPadroes(),App.getGerenciadorPa
droes()

      .getConjuntoTesteRede(), file
                                                      .getAbsolutePath()
                                                      + ".epd",
f.getSelectedFile().getName(),
                                                      "Classificacao",
nomeCarac);
                              }
                        } catch (Exception e) {
                              new MessageDialog(f.getDialog(), "Erro ao
persistir",
                                          "Um erro ocorreu ao tentar
salvar o arquivo.").show();
                              e.printStackTrace();
                        }
                  }
            }
      }

      private static boolean permitePersistir(){
            boolean permitir = true;
            try {
                  if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size()<1){
                        new MessageDialog(App.getTelaEasyFAN().getFrame(),
"Verificaï¿½ï¿½o",
                        "ï¿½ necessï¿½rio carregar um conjunto de
treinamento").show();
                        permitir = false;
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return permitir;
      }



      public static void salvarConjuntoValidacao() {
            if (permitePersistir()){
                  FileChooser f = new FileChooser(new Frame(), "Salvar
conjunto de validaï¿½ï¿½o",
                              FileChooser.MODE_SAVE);
                  f.setFileFilters(new FileFilter[] {
                              new ExtensionFileFilter("epd", "EPD files
(*.epd)"),
                              new ExtensionFileFilter("dat", "DAT files
(*.dat)")
```

```
                });

                f.setSelectedFile(App.getUltimoArquivo());
                f.setSelectedFiler(App.getUltimoFiltro());

                f.show();

                File file = f.getSelectedFile();
                if (file != null)
                      App.setUltimoArquivo(file);
                App.setUltimoFiltro(f.getIndexOfSelectedFileFilter());

                if (file != null) {
                      PersistorDados pd = new PersistorDados();
                      try {
                            if
(f.getSelectedFileFilter().getDescription()
                                        .equalsIgnoreCase("DAT files
(*.dat)"))

      pd.gravaPadroesDat(App.getGerenciadorPadroes()

      .getConjuntoValidacaoRede(), file
                                              .getAbsolutePath()
                                              + ".dat");
                            else if
(f.getSelectedFileFilter().getDescription()
                                        .equalsIgnoreCase("EasyFAN files
(*.epd)")) {
                                  String[] nomeCarac;
                                  if (App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento() == null) {
                                        nomeCarac = new
String[App.getGerenciadorPadroes()

      .getConjuntoValidacaoRede().get(0)

      .getQuantasCaracteristicas()];
                                        for (int i = 0; i <
nomeCarac.length; i++)
                                              nomeCarac[i] =
"Caracteristica "
                                                    +
String.valueOf(i + 1);
                                  } else
                                        nomeCarac =
App.getGerenciadorPadroes()

      .getNomeCaracteristicasTreinamento();


      pd.gravaPadroesXml(App.getGerenciadorPadroes(),App.getGerenciadorPa
droes()

      .getConjuntoTesteRede(), file
                                                    .getAbsolutePath()
```

```
                                                        + ".epd",
f.getSelectedFile().getName(),
                                                        "Validacao",
nomeCarac);


                                }
                            } catch (Exception e) {
                                new MessageDialog(f.getDialog(), "Erro ao
persistir", "Um erro ocorreu ao tentar salvar o arquivo.").show();
                                e.printStackTrace();
                            }
                        }
                    }

            }

}
```

1.1.7 EvtTeste.Java

```
package controle.eventos;

import java.text.NumberFormat;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;
import jfan.fan.beans.StatisticBean;
import jfan.fan.padroes.IPadrao;
import modelo.AbaTesteBean;
import thinlet.Thinlet;
import thinletcommons.MessageDialog;
import visao.gui.telas.classes.MatrizConfusaoTela;
import controle.main.App;



public class EvtTeste {

      private final AbaTesteBean testeBean = new AbaTesteBean();
      private final Object lblHarmonica,lblMaxMin,lblAritmetica;
      private Thinlet thinlet;
      private final NumberFormat numFormat = NumberFormat.getInstance();
      private StatisticBean estatisticas;

      public EvtTeste(Thinlet thinlet,Object lblHarmonica,Object
lblMaxMin, Object lblAritmetica) {
            this.lblAritmetica = lblAritmetica;
            this.lblHarmonica = lblHarmonica;
            this.lblMaxMin = lblMaxMin;
            numFormat.setMaximumFractionDigits(4);
            numFormat.setMinimumFractionDigits(4);
            numFormat.setMaximumIntegerDigits(3);
            numFormat.setMinimumIntegerDigits(3);
            this.thinlet = thinlet;
```

```
	}

	public void testarRede(StoredNetworkType tipo) {
		App.requisitarNormalizacao();
		switch (tipo) {
			case Actual:
				estatisticas = App.getMonitorFAN().testar();
				break;
			case BestAritmethicMean:
				estatisticas =
App.getMonitorFAN().testar(App.getMonitorFAN().getNeuroniosMelhorAritmeti
ca());
				break;
			case BestHarmonicMean:
				estatisticas =
App.getMonitorFAN().testar(App.getMonitorFAN().getNeuroniosMelhorHarmonic
a());
				break;
			case BestMaxMin:
				estatisticas =
App.getMonitorFAN().testar(App.getMonitorFAN().getNeuroniosMelhorMaximoMi
nimo());
				break;
			default:
				break;
		}
		atualizarDados();
	}

	public void testarRede(ArrayList<NeuronioFAN> neuronios) {
		App.requisitarNormalizacao();
		estatisticas = App.getMonitorFAN().testar(neuronios);
		atualizarDados();
	}

	public void testarRede(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes) {
		estatisticas = App.getMonitorFAN().testar(neuronios,
padroes);
		atualizarDados();
	}

	public void chamarTelaMatrizConfusao() {
		if (testeBean.getMatrizConfusao() != null) {
			MatrizConfusaoTela mct = new
MatrizConfusaoTela(App.getTelaEasyFAN().getFrame(),"Matriz de confusï¿½o:
Teste",-
1,testeBean.getMatrizConfusao(),testeBean.getHarmonica(),testeBean.getAri
tmetica(),testeBean.getMaxmin(), null);
			mct.show();
		}
		else {
			MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenï¿½iï¿½o","ï¿½
necessï¿½rio testar uma rede antes.");
			msg.show();
		}
```

```
	}

	private void atualizarDados() {
		testeBean.setAritmetica(estatisticas.getAritmetica());
		testeBean.setHarmonica(estatisticas.getHarmonica());
		testeBean.setMaxmin(estatisticas.getMaxmin());

	testeBean.setMatrizConfusao(estatisticas.getMatrizConfusao());
		thinlet.setString(lblAritmetica, "text",
numFormat.format(testeBean.getAritmetica()));
		thinlet.setString(lblHarmonica, "text",
numFormat.format(testeBean.getHarmonica()));
		thinlet.setString(lblMaxMin, "text",
numFormat.format(testeBean.getMaxmin()));

	App.getTelaEasyFAN().setCertosTeste(estatisticas.getAcertos());
		App.getTelaEasyFAN().setErrosTeste(estatisticas.getErros());
	}

}
```

1.1.8 EvtTreinamento.Java

```
package controle.eventos;

import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Timer;
import java.util.TimerTask;

import javax.management.monitor.Monitor;

import jfan.exceptions.JfanPorcentagemException;
import jfan.fan.MonitorFAN;
import jfan.fan.events.FaseRedeEvent;
import jfan.fan.events.FaseRedeListener;
import jfan.fan.events.RedeParouEvent;
import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import sun.security.x509.AVA;
import thinlet.Thinlet;
import thinletcommons.MessageDialog;
import utils.AWTUtils;
import visao.gui.telas.classes.EasyFan;
import controle.eventos.auxiliares.AtualizadorEpocas;
import controle.eventos.auxiliares.AtualizadorMelhores;
import controle.eventos.auxiliares.AtualizadorTempera;
import controle.eventos.auxiliares.EnfileiradorEventos;
```

```
import controle.eventos.auxiliares.EvtQueues;
import controle.main.App;

public class EvtTreinamento implements FaseRedeListener{
	Thread threadAtualizadorEpocas;
	AtualizadorEpocas atualizadorEpocas;
	Thread threadAtualizadorTempera;
	AtualizadorTempera atualizadorTempera;
	Thread threadAtualizadorMelhores;
	AtualizadorMelhores atualizadorMelhores;

	private final EvtQueues filasEventos = App.getEventQueues();

	private final EnfileiradorEventos enfileirador =
App.getEnfileiradorEventos();

	/**
	 * Referencia da tela thinlet para alterar os componentes.
	 */
	private Thinlet tela;

	/**
	 * Caso true o treinamento deve rodar, caso false ele deve parar.
	 */
	private boolean keepGoing = true;

	private boolean firstRunning = true;

	/**
	 * Campos do thinlet
	 */
	private Object lblFaseRede, lblEpoca, lblTempo, lblMelhorHarmonica,
lblMelhorMaxMin, lblTemperaValorAtual, lblHarmonicaAtual, lblMaxMinAtual,
lblMelhorAritmetica, lblAritmeticaAtual;

	private Timer timer = new Timer(true);

	/**
	 * Tarefa para o timer.
	 */
	private AtualizadorTempo atualizadorTempo = new AtualizadorTempo();

	/**
	 * Construtor que captura a tela que será atualizada.
	 */
	public EvtTreinamento(EasyFan screen) {
		tela = screen;

		lblFaseRede = tela.find("lblFaseRede");
		lblEpoca = tela.find("lblEpoca");
		lblTempo = tela.find("lblTempo");
		lblMelhorHarmonica = tela.find("lblMelhorHarmonica");
		lblMelhorMaxMin = tela.find("lblMelhorMaxMin");
		lblTemperaValorAtual = tela.find("lblTemperaValorAtual");
		lblMaxMinAtual = tela.find("lblMaxMinAtual");
		lblHarmonicaAtual = tela.find("lblHarmonicaAtual");
		lblAritmeticaAtual = tela.find("lblAritmeticaAtual");
```

```
            lblMelhorAritmetica = tela.find("lblMelhorAritmetica");
      }

      /**
       * Iniciar o treinamento da rede.
       */
      public void iniciarTreinamento() {
            verificarMonitorFAN();
            //App.vericarMonitor();
            App.requisitarNormalizacao();
            App.getMonitorFAN().addFaseRedeListener(this);


            App.getMonitorFAN().execute();
            if (atualizadorTempo != null)
                  atualizadorTempo = new
AtualizadorTempo(atualizadorTempo);
            timer.scheduleAtFixedRate(atualizadorTempo, 1000, 1000);

            if (atualizadorEpocas == null ||
!threadAtualizadorEpocas.isAlive()) {
                  atualizadorEpocas = new
AtualizadorEpocas(App.getEventQueues().getQueueTrocaEpocaEvent(),tela,lbl
Epoca,lblHarmonicaAtual, lblMaxMinAtual,lblAritmeticaAtual);
            }
            if (threadAtualizadorEpocas == null ||
!threadAtualizadorEpocas.isAlive()) {
                  threadAtualizadorEpocas = new
Thread(atualizadorEpocas);

                  threadAtualizadorEpocas.start();
            }

            if (atualizadorMelhores == null ||
!threadAtualizadorMelhores.isAlive()) {
                  atualizadorMelhores = new
AtualizadorMelhores(App.getEventQueues().getQueueMelhorAritmeticaEvent(),
App.getEventQueues().getQueueMelhorHarmonicaEvent(),App.getEventQueues().
getQueueMelhorMaxMinEvent(),tela,lblMelhorHarmonica, lblMelhorMaxMin,
lblMelhorAritmetica);
            }
            if (threadAtualizadorMelhores == null ||
!threadAtualizadorMelhores.isAlive()) {
                  threadAtualizadorMelhores = new
Thread(atualizadorMelhores);

                  threadAtualizadorMelhores.start();
            }

            if (atualizadorTempera == null ||
!threadAtualizadorTempera.isAlive()) {
                  atualizadorTempera = new
AtualizadorTempera(App.getEventQueues().getQueueTrocaValorTemperaEvent(),
lblTemperaValorAtual,tela);
            }
            if (threadAtualizadorTempera == null ||
!threadAtualizadorTempera.isAlive()) {
```

```
                  threadAtualizadorTempera = new
Thread(atualizadorTempera);

                  threadAtualizadorTempera.start();
            }



      }

      private void verificarMonitorFAN() {
            if (App.getMonitorFAN() == null) {
                  App.vericarMonitor();
                  MonitorFAN monitor = App.getMonitorFAN();
                  monitor.addTrocaEpocaListener(enfileirador);
                  monitor.addMaximoMinimoListener(enfileirador);
                  monitor.addMelhorAritimeticaListener(enfileirador);
                  monitor.addMelhorHarmonicaListener(enfileirador);
                  ITemperaSimulada temp =
App.getProjetoConfig().getTreinoConf().getTempera();
                  if (temp != null)
                        temp.addTrocaValorTemperaListener(enfileirador);

      monitor.addValidacaoListener(((EasyFan)tela).getEvtValidacao());

      monitor.setCadaEpocaValidar(App.getProjetoConfig().getValidacaoConf
ig().getCadaEpocaValidar());

      monitor.setTipoRedeValidar(App.getProjetoConfig().getValidacaoConfi
g().getTipoRedeValidacao());
            }
            else {
                  MonitorFAN monitor = App.getMonitorFAN();
                  monitor.addTrocaEpocaListener(enfileirador);
                  monitor.addMaximoMinimoListener(enfileirador);
                  monitor.addMelhorAritimeticaListener(enfileirador);
                  monitor.addMelhorHarmonicaListener(enfileirador);
                  ITemperaSimulada temp =
App.getProjetoConfig().getTreinoConf().getTempera();
                  if (temp != null)
                        temp.addTrocaValorTemperaListener(enfileirador);

      monitor.addValidacaoListener(((EasyFan)tela).getEvtValidacao());

      monitor.setCadaEpocaValidar(App.getProjetoConfig().getValidacaoConf
ig().getCadaEpocaValidar());

      monitor.setTipoRedeValidar(App.getProjetoConfig().getValidacaoConfi
g().getTipoRedeValidacao());
            }
      }

      public void pararTreinamento() {

            App.getMonitorFAN().stop();

            Thread.yield();
```

```
            //Para o timer
            atualizadorTempo.cancel();
            timer.purge();
      Thread.yield();
            atualizadorEpocas.stop();
            atualizadorMelhores.stop();
            atualizadorTempera.stop();

      }

      public void trocarTempera(ITemperaSimulada tempera) {
            verificarMonitorFAN();
            //TODO averiguar se asincronizaï¿½ï¿½o estï¿½ ocorrendo bem.
            App.getProjetoConfig().getTreinoConf().setTempera(tempera);
            App.getMonitorFAN().setTemperaSimulada(tempera);
            if (tempera != null)
                  tempera.addTrocaValorTemperaListener(enfileirador);
      }

      public void alterarMinimoTempera(double min) {
            ITemperaSimulada t =
App.getProjetoConfig().getTreinoConf().getTempera();
            if (t != null)
                  t.setLimiteMinimo(min);
      }

      public void alterarMaximoTempera(double max) {
            ITemperaSimulada t =
App.getProjetoConfig().getTreinoConf().getTempera();
            if (t != null)
                  t.setLimiteMaximo(max);
      }

      public void alterarStepTempera(double step) {
            ITemperaSimulada t =
App.getProjetoConfig().getTreinoConf().getTempera();
            if (t != null)
                  t.setStep(step);
      }

      public void alterarCadaEpocaEmbaralharPadores(int epocas) {
            verificarMonitorFAN();
            App.getMonitorFAN().setCadaEpocaEmbaralharPadroes(epocas);

      App.getProjetoConfig().getTreinoConf().setCadaEpocaEmbaralharPadroe
s(epocas);
      }

      public void desativarCadaEpocaEmbaralharPadores() {
            verificarMonitorFAN();
            App.getMonitorFAN().setCadaEpocaEmbaralharPadroes(-1);

      App.getProjetoConfig().getTreinoConf().setCadaEpocaEmbaralharPadroe
s(-1);
      }
```

```
	public void alterarCadaEpocaRenormalizar(int epocas) {
		verificarMonitorFAN();
		App.getMonitorFAN().setCadaEpocaRenormalizar(epocas);

	App.getProjetoConfig().getTreinoConf().setCadaEpocaRenormalizar(epo
cas);
	}

	public void desativarCadaEpocaRenormalizar() {
		verificarMonitorFAN();
		//Troca a opï¿½iï¿½o no monitorFAN
		App.getMonitorFAN().setCadaEpocaRenormalizar(-1);

	App.getProjetoConfig().getTreinoConf().setCadaEpocaRenormalizar(-
1);
	}

	public void ativarPesosAleatorios(int pesoBase) {
		verificarMonitorFAN();
		App.getMonitorFAN().ativarPesosAleatorios(pesoBase);
		App.getProjetoConfig().getTreinoConf().setPesoBase(pesoBase);

	App.getProjetoConfig().getTreinoConf().setRandomizarPesos(true);
	}

	public void desativarPesosAleatorios() {
		verificarMonitorFAN();
		App.getMonitorFAN().desativarPesosAleatorios();

	App.getProjetoConfig().getTreinoConf().setRandomizarPesos(false);
		App.getProjetoConfig().getTreinoConf().setPesoBase(-1);
	}

	public void ativarPorcentagemPadroesTreino(int porcentagem) {
		verificarMonitorFAN();
		int padroes = 0;
		try {

	App.getProjetoConfig().getTreinoConf().setPorcentagemPadroesTreino(
porcentagem);
			padroes =
App.getMonitorFAN().getNumeroPadroesPorcentagem(porcentagem);

	App.getMonitorFAN().setNumeroPadroesTreinamento(padroes);

	App.getProjetoConfig().getTreinoConf().setNumeroPadroesTreino(padro
es);
		} catch (Exception e) {
			MessageDialog m = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Ocorreu um erro.","Ocorreu
o seguinte problema: " + e.getMessage());
			m.show();
			//e.printStackTrace();
		}
	}

	public void desativarPorcentagemPadroesTreino() {
```

```
            verificarMonitorFAN();

      App.getProjetoConfig().getTreinoConf().setPorcentagemPadroesTreino(
0);
            App.getMonitorFAN().setNumeroPadroesTreinamento(-1);

      App.getProjetoConfig().getTreinoConf().setNumeroPadroesTreino(-1);
      }

      public void ativarChanceRenormalizar(int porcentagem) {
            verificarMonitorFAN();
            try {

      App.getMonitorFAN().setChanceRenormalizacao(porcentagem);

      App.getProjetoConfig().getTreinoConf().setChanceRenormalizar(porcen
tagem);
            } catch (JfanPorcentagemException e) {
                  MessageDialog m = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Ocorreu um erro.","Ocorreu
o seguinte problema: " + e.getMessage());
                  m.show();
                  //e.printStackTrace();
            }
      }

      public void desativarChanceRenormalizar() {
            verificarMonitorFAN();
            try {
                  App.getMonitorFAN().setChanceRenormalizacao(0);

      App.getProjetoConfig().getTreinoConf().setChanceRenormalizar(0);
            } catch (JfanPorcentagemException e) {
                  MessageDialog m = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Ocorreu um erro.","Ocorreu
o seguinte problema: " + e.getMessage());
                  m.show();
                  //e.printStackTrace();
            }
      }

      public void trocarTipoNormalizacao(NormalizatorTypes type) {

      App.getProjetoConfig().getRedeConf().setTipoNormalizador(type);
            App.setPrecisaNormalizar(true);
      }

      public void trocarEpocaStepTempera(int i) {
            App.getMonitorFAN().setCadaEpocaStepTempera(i);

      App.getProjetoConfig().getTreinoConf().setCadaEpocaStepTempera(i);
      }

      public void trocarEpocaReiniciarTempera(int i) {
            App.getMonitorFAN().setCadaEpocaReiniciarTempera(i);

      App.getProjetoConfig().getTreinoConf().setCadaEpocaResetTempera(i);
```

```
	}

	public void trocarLimiteMaximoTempera(double d) {

	App.getProjetoConfig().getTreinoConf().getTempera().setLimiteMaximo
(d);
	}

	public void trocarLimiteMinimoTempera(double d) {

	App.getProjetoConfig().getTreinoConf().getTempera().setLimiteMinimo
(d);
	}

	public void trocarStepTempera(double d) {

	App.getProjetoConfig().getTreinoConf().getTempera().setStep(d);
	}

	public void setNumeroEpocasTreinar(int i) {

	App.getProjetoConfig().getTreinoConf().setNumeroEpocasTreinar(i);
		App.getMonitorFAN().setNumeroEpocasTreinar(i);
	}

	public void treinarContinuamente() {

	App.getProjetoConfig().getTreinoConf().setNumeroEpocasTreinar(-1);
		App.getMonitorFAN().setNumeroEpocasTreinar(-1);
	}

	public void ativarParadaPorTempo(int dias, int horas, int minutos,
int segundos) {
		atualizadorTempo.setParada(dias, horas, minutos, segundos);
		App.getProjetoConfig().getTreinoConf().setTempoParada(dias,
horas, minutos, segundos);
	}

	public void desativarParadaPorTempo() {
		atualizadorTempo.desativarParada();
		App.getProjetoConfig().getTreinoConf().setTempoParada(0, 0,
0, 0);
	}

	private class AtualizadorTempo extends TimerTask {

		int segundos, minutos, horas, dias, segundosParada,
minutosParada, horasParada, diasParada;
		boolean validarParada = false;

		StringBuilder strBuilder = new StringBuilder();

		NumberFormat numFormat = NumberFormat.getInstance();
		private boolean naoDesenhar;

		{
			numFormat.setMaximumFractionDigits(0);
```

```
                numFormat.setMaximumIntegerDigits(2);
                numFormat.setMinimumFractionDigits(0);
                numFormat.setMinimumIntegerDigits(2);
                numFormat.setParseIntegerOnly(true);
        }

        public AtualizadorTempo(AtualizadorTempo atualizadorTempo) {
                segundos = atualizadorTempo.segundos;
                minutos = atualizadorTempo.minutos;
                horas = atualizadorTempo.horas;
                dias = atualizadorTempo.dias;
                segundosParada = atualizadorTempo.segundosParada;
                minutosParada = atualizadorTempo.minutosParada;
                horasParada = atualizadorTempo.horasParada;
                diasParada = atualizadorTempo.diasParada;
                validarParada = atualizadorTempo.validarParada;
                if (validarParada)
                        avaliaParada();
        }

        public AtualizadorTempo() {}

        public void setParada(int dias, int horas, int minutos, int
segundos) {
                diasParada = dias;
                horasParada = horas;
                minutosParada = minutos;
                segundosParada = segundos;
                validarParada = true;
        }

        public void desativarParada() {
                validarParada = false;
        }

        private void addSegundo() {
                if (segundos >= 59) {
                        addMinuto();
                        segundos = 0;
                }
                else {
                        segundos++;
                }
        }

        private void addMinuto() {
                if (minutos >= 59) {
                        addHora();
                        minutos = 0;
                }
                else {
                        minutos++;
                }
        }

        private void addHora() {
                if (horas >= 23) {
```

```
                        addDia();
                        horas = 0;
                  }
                  else {
                        horas++;
                  }
            }

            private void addDia() {
                  dias++;
            }

            private void avaliaParada() {
                  boolean b = segundos >= segundosParada && minutos >=
minutosParada && horas >= horasParada && dias >= diasParada;
                  if (b){
                        pararTreinamento();
                  }
                  this.naoDesenhar = b;
            }

            @Override
            public void run() {

                  if (!naoDesenhar) {
                        addSegundo();

                        strBuilder.setLength(0);
                        strBuilder.append(numFormat.format(dias));
                        strBuilder.append(" dias ");
                        strBuilder.append(numFormat.format(horas));
                        strBuilder.append(":");
                        strBuilder.append(numFormat.format(minutos));
                        strBuilder.append(":");
                        strBuilder.append(numFormat.format(segundos));

                        tela.setString(lblTempo, "text",
strBuilder.toString());
                  }


                  if (validarParada)
                        avaliaParada();
            }

      }

      public void mudouFaseRede(FaseRedeEvent event) {
      }

      public void redeParou(RedeParouEvent event) {
            //Para o timer
            atualizadorTempo.cancel();
            timer.purge();
      atualizadorEpocas.stop();
            atualizadorMelhores.stop();
            atualizadorTempera.stop();
```

```
			App.getTelaEasyFAN().configurarTelaPararTreinamento();
		}
}
```

1.1.9 EvtValidacao.Java

```
package controle.eventos;

import java.text.NumberFormat;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;
import jfan.fan.beans.StatisticBean;
import jfan.fan.events.ValidacaoEvent;
import jfan.fan.events.ValidacaoListener;
import jfan.fan.padroes.IPadrao;
import thinletcommons.MessageDialog;
import visao.gui.telas.classes.EasyFan;
import visao.gui.telas.classes.MatrizConfusaoTela;
import controle.main.App;

public class EvtValidacao implements ValidacaoListener {
	private EasyFan owner;
	private final NumberFormat numFormat = NumberFormat.getInstance();
	private StatisticBean statisticas;
	private int epoca = -1;

	public EvtValidacao(EasyFan owner) {
		numFormat.setMaximumFractionDigits(4);
		numFormat.setMinimumFractionDigits(4);
		numFormat.setMaximumIntegerDigits(3);
		numFormat.setMinimumIntegerDigits(3);
		this.owner = owner;
	}

	public void validarRede(StoredNetworkType tipo) {
		App.requisitarNormalizacao();
		statisticas = App.getMonitorFAN().validar(tipo);
		atualizarDados(tipo.toString(),"--");
		epoca = -1;
	}

	public void validarRede(ArrayList<NeuronioFAN> neuronios) {
		App.requisitarNormalizacao();
		statisticas = App.getMonitorFAN().validar(neuronios);
		atualizarDados("Carregada","--");
		epoca = -1;
	}

	public void chamarTelaMatrizConfusao() {
		if (statisticas.getMatrizConfusao() != null) {
			MatrizConfusaoTela mct = new
MatrizConfusaoTela(App.getTelaEasyFAN().getFrame(),"Matriz de confusÃ£o:
Teste",epoca,statisticas.getMatrizConfusao(),statisticas.getHarmonica(),s
tatisticas.getAritmetica(),statisticas.getMaxmin(), "validacao");
```

```
                mct.show();
            }
            else {
                MessageDialog msg = new
MessageDialog(App.getTelaEasyFAN().getFrame(),"Atenção","É necessário
testar uma rede antes.");
                msg.show();
            }
      }

      private void atualizarDados(String tipoRede, String epocaValidada)
{

      owner.setAcertosValidacao(Integer.toString(statisticas.getAcertos()
));

      owner.setErrosValidacao(Integer.toString(statisticas.getErros()));

      owner.setHarmonicaValidacao(numFormat.format(statisticas.getHarmoni
ca()));

      owner.setAritmeticaValidacao(numFormat.format(statisticas.getAritme
tica()));

      owner.setMaxMinValidacao(numFormat.format(statisticas.getMaxmin()))
;
            owner.setTipoRedeValidacao(tipoRede);
            owner.setEpocaValidada(epocaValidada);

      }

      public void validarCadaEpocas(int i) {

      App.getProjetoConfig().getValidacaoConfig().setCadaEpocaValidar(i);
            App.getMonitorFAN().setCadaEpocaValidar(i);
      }

      public void desativarValidarCadaEpocas() {

      App.getProjetoConfig().getValidacaoConfig().setCadaEpocaValidar(0);
            App.getMonitorFAN().setCadaEpocaValidar(0);
      }

      public void setTipoRedeValidacao(StoredNetworkType tipoRede) {

      App.getProjetoConfig().getValidacaoConfig().setTipoRedeValidacao(ti
poRede);
            App.getMonitorFAN().setTipoRedeValidar(tipoRede);
      }


      public void validou(ValidacaoEvent event) {
            statisticas = event.getStatisticas();
            epoca = event.getEpocaValidacao() ;
            atualizarDados(event.getTipoRedeValidacao().toString(),
Integer.toString(epoca));
      }
```

```
	public void validarRede(ArrayList<NeuronioFAN> neurons,
ArrayList<IPadrao> conjunto) {
		statisticas = App.getMonitorFAN().validar(neurons,conjunto);
		atualizarDados("Carregada","--");
	}

}
```

1.2 PACKAGE CONTROLE.EVENTOS.AUXILIARES;

1.2.1 AtualizadorEpocas.Java

```
package controle.eventos.auxiliares;

import java.text.NumberFormat;
import java.util.Arrays;
import java.util.Comparator;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

import controle.main.App;

import modelo.DataSetPool;

import jfan.fan.events.TrocaEpocaEvent;
import thinlet.Thinlet;

public class AtualizadorEpocas implements Runnable {

	private LinkedBlockingQueue<TrocaEpocaEvent> fila;

	private TrocaEpocaEvent eventoAtual;

	private Thinlet tela;

	NumberFormat numformat = NumberFormat.getInstance();

	private boolean keepGoing = true;

	private DataSetPool pool = App.getDatasetpool();

	private Object lblEpoca, lblMediaHarmonica, lblMaximoMinimo,
			lblMediaAritmetica;

	public AtualizadorEpocas(LinkedBlockingQueue<TrocaEpocaEvent>
queue,
			Thinlet tela, Object lblEpoca, Object
lblMediaHarmonica,
			Object lblMaximoMinimo, Object lblMediaAritmetica) {
		this.tela = tela;
		this.lblEpoca = lblEpoca;
		this.lblMediaHarmonica = lblMediaHarmonica;
		this.lblMaximoMinimo = lblMaximoMinimo;
		this.lblMediaAritmetica = lblMediaAritmetica;
```

```
            fila = queue;
            numformat.setMinimumFractionDigits(4);
            numformat.setMaximumFractionDigits(4);
            numformat.setMinimumIntegerDigits(3);
            numformat.setMaximumIntegerDigits(3);
      }

      public void stop(){
            keepGoing = false;
      }

      public void run() {
            while (keepGoing) {
                        if (fila.size() > 0) {
                              try {
                                    eventoAtual = fila.poll(100,
TimeUnit.MILLISECONDS);
                                    if (eventoAtual != null && tela !=
null) {
                                          tela.setString(lblEpoca, "text",
Integer

      .toString(eventoAtual.getEpoca()));

      tela.setString(lblMediaHarmonica, "text", numformat

      .format(eventoAtual.getMediaHarmonica()));
                                          tela.setString(lblMaximoMinimo,
"text", numformat

      .format(eventoAtual.getMaximoDoMinimo()));

tela.setString(lblMediaAritmetica, "text",

numformat.format(eventoAtual.getMediaAritmetica()));


ContainerMatrizConfusao.setMatrizEpoca(eventoAtual.getEpoca(),
eventoAtual.getMatrizConfusao(),eventoAtual.getMediaHarmonica(),eventoAtu
al.getMediaAritmetica(),eventoAtual.getMaximoDoMinimo());


pool.addEstatisticData(eventoAtual.getEpoca(),
eventoAtual.getMediaHarmonica(), eventoAtual.getMaximoDoMinimo(),
eventoAtual.getMediaAritmetica());
                                    }
                              } catch (InterruptedException e) {
                                    e.printStackTrace();
                              }
                        } else {
                                    Thread.yield();
                        }

            }
            if (fila.size() > 0) {
                  TrocaEpocaEvent[] array = new
TrocaEpocaEvent[fila.size()];
```

```
                    fila.toArray(array);
                    Arrays.sort(array, new Comparator<TrocaEpocaEvent>() {

                          public int compare(TrocaEpocaEvent o1,
TrocaEpocaEvent o2) {
                                return o1.getEpoca() - o2.getEpoca();
                          }

                    });
                    eventoAtual = array[array.length-1];

                    tela.setString(lblEpoca, "text", Integer
                                .toString(eventoAtual.getEpoca()));
                    tela.setString(lblMediaHarmonica, "text", numformat
                                .format(eventoAtual.getMediaHarmonica()));
                    tela.setString(lblMaximoMinimo, "text", numformat
                                .format(eventoAtual.getMaximoDoMinimo()));
                     tela.setString(lblMediaAritmetica,
"text",numformat.format(eventoAtual.getMediaAritmetica()));

ContainerMatrizConfusao.setMatrizEpoca(eventoAtual.getEpoca(),
eventoAtual.getMatrizConfusao(),eventoAtual.getMediaHarmonica(),eventoAtu
al.getMediaAritmetica(),eventoAtual.getMaximoDoMinimo());

                     fila.clear();
                     array = null;
               }
         }
}
```

1.2.2 AtualizadorMelhores.Java

```
package controle.eventos.auxiliares;

import java.text.NumberFormat;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

import thinlet.Thinlet;

import jfan.fan.events.MelhorAritimeticaEvent;
import jfan.fan.events.MelhorHarmonicaEvent;
import jfan.fan.events.MelhorMaximoMinimoEvent;

public class AtualizadorMelhores implements Runnable {

      private LinkedBlockingQueue<MelhorAritimeticaEvent>
filaAritimetica;

      private LinkedBlockingQueue<MelhorHarmonicaEvent> filaHarmonica;

      private LinkedBlockingQueue<MelhorMaximoMinimoEvent> filaMaxMin;

      private MelhorAritimeticaEvent aritimeticaAtual;

      private MelhorHarmonicaEvent harmonicaAtual;
```

```
	private MelhorMaximoMinimoEvent maxMinAtual;

	NumberFormat numformat = NumberFormat.getInstance();

	private boolean keepGoing = true;

	/**
	 * Tela thinlet, para poder atualizar os campos.
	 */
	private Thinlet tela;

	/**
	 * Campos thinlet.
	 */
	Object lblMelhorHarmonica, lblMelhorMaxMin, lblMelhorAritimetica;

	public AtualizadorMelhores(
			LinkedBlockingQueue<MelhorAritimeticaEvent>
queueAritimetica,
			LinkedBlockingQueue<MelhorHarmonicaEvent>
queueHarmonica,
			LinkedBlockingQueue<MelhorMaximoMinimoEvent>
queueMaxMin,
			Thinlet tela, Object lblHarmonica, Object lblMaxMin,
			Object lblAritimetica) {

		this.tela = tela;

		lblMelhorAritimetica = lblAritimetica;
		lblMelhorHarmonica = lblHarmonica;
		lblMelhorMaxMin = lblMaxMin;

		filaAritimetica = queueAritimetica;
		filaHarmonica = queueHarmonica;
		filaMaxMin = queueMaxMin;

		numformat.setMinimumFractionDigits(4);
		numformat.setMaximumFractionDigits(4);
		numformat.setMinimumIntegerDigits(3);
		numformat.setMaximumIntegerDigits(3);
	}

	public void stop() {
		keepGoing = false;
	}

	public void run() {
		while (keepGoing) {
			synchronized (this) {
				if (filaAritimetica.size() == 0 &&
filaHarmonica.size() == 0
						&& filaMaxMin.size() == 0) {
					try {
						Thread.sleep(400);
					} catch (InterruptedException e) {
						e.printStackTrace();
```

```
                        }
                    } else { // Caso haja algo em alguma fila.
                        if (tela != null) {
                            try {
                                if (filaAritimetica.size() > 0)
{
                                    aritimeticaAtual =
filaAritimetica.poll(250,

      TimeUnit.MILLISECONDS);
                                    if (aritimeticaAtual !=
null) {

      tela.setString(lblMelhorAritimetica, "text",
numformat.format(aritimeticaAtual.getMediaAritimetica()));
                                    }
                                }

                                if (filaHarmonica.size() > 0) {
                                    harmonicaAtual =
filaHarmonica.poll(250,

      TimeUnit.MILLISECONDS);
                                    if (harmonicaAtual !=
null) {

      tela.setString(lblMelhorHarmonica, "text",
numformat.format(harmonicaAtual.getMediaHarmonica()));
                                    }
                                }

                                if (filaMaxMin.size() > 0) {
                                    maxMinAtual =
filaMaxMin.poll(250,

      TimeUnit.MILLISECONDS);
                                    if (maxMinAtual != null) {

      tela.setString(lblMelhorMaxMin, "text",
numformat.format(maxMinAtual.getMaximoDoMinimo()));
                                    }
                                }
                            } catch (InterruptedException e) {
                                e.printStackTrace();
                            }
                        }
                    }
                }
                Thread.yield();
            }
        }

}
```

1.2.3 AtualizadorTempera.Java

```
package controle.eventos.auxiliares;

import java.text.NumberFormat;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

import jfan.fan.events.TrocaValorTemperaEvent;
import thinlet.Thinlet;

public class AtualizadorTempera implements Runnable {

    private LinkedBlockingQueue<TrocaValorTemperaEvent> filaValorTempera;

    private TrocaValorTemperaEvent eventoAtual;

    NumberFormat numformat = NumberFormat.getInstance();

    private boolean keepGoing = true;

    private long ultimaAtualizacao = 0l;

    private Thinlet tela;

    Object lblValorAtualTempera;

    public AtualizadorTempera(
            LinkedBlockingQueue<TrocaValorTemperaEvent>
queueValorTemperaEvent, Object lblValorTempera, Thinlet tela) {

        this.tela = tela;

        lblValorAtualTempera = lblValorTempera;

        filaValorTempera = queueValorTemperaEvent;

        numformat.setMinimumFractionDigits(4);
        numformat.setMaximumFractionDigits(4);
        numformat.setMinimumIntegerDigits(3);
        numformat.setMaximumIntegerDigits(3);
    }

    public void stop() {
        keepGoing = false;
    }

    public void run() {
        while (keepGoing) {
            synchronized (this) {
                if (filaValorTempera.size() == 0) {
                    try {
                        Thread.sleep(400);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                } else {
```

```
                    if (tela != null) {
                        try {
                            if (filaValorTempera.size() > 0) {
                                eventoAtual = filaValorTempera.poll(250,
                                        TimeUnit.MILLISECONDS);
                                if (eventoAtual != null) {
                                    tela.setString(lblValorAtualTempera,
"text", numformat.format(eventoAtual.getValor()));
                                }
                            }
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                    ultimaAtualizacao = System.currentTimeMillis();
                }
            }
            Thread.yield();
        }
    }
}
```

1.2.4 ContainerMatrizConfusao.Java

```
package controle.eventos.auxiliares;

public class ContainerMatrizConfusao  {

      private static int[][] MATRIZ;
      private static int     EPOCA;
      private static float   HARMONICA;
      private static float   ARITMETICA;
      private static float   MAXMIN;

      public synchronized static int getEpoca() {
            return EPOCA;
      }

      public synchronized static int[][] getMatriz() {
            return MATRIZ;
      }

      public synchronized static float getHarmonica() {
            return HARMONICA;
      }

      public synchronized static float getAritmetica() {
            return ARITMETICA;
      }

      public synchronized static float getMaxMin() {
            return MAXMIN;
      }
```

```
	public synchronized static void setMatrizEpoca(int epoca, int[][]
matriz, float harmonica, float aritmetica, float maxmin) {
		EPOCA = epoca;
		MATRIZ = matriz;
		HARMONICA = harmonica;
		ARITMETICA = aritmetica;
		MAXMIN = maxmin;
	}

}
```

1.2.5 EnfileiradorEventos.Java

```
package controle.eventos.auxiliares;

import java.util.concurrent.LinkedBlockingQueue;

import jfan.fan.events.MelhorAritimeticaEvent;
import jfan.fan.events.MelhorAritimeticaListener;
import jfan.fan.events.MelhorHarmonicaEvent;
import jfan.fan.events.MelhorHarmonicaListener;
import jfan.fan.events.MelhorMaximoMinimoEvent;
import jfan.fan.events.MelhorMaximoMinimoListener;
import jfan.fan.events.TrocaEpocaEvent;
import jfan.fan.events.TrocaEpocaListener;
import jfan.fan.events.TrocaValorTemperaEvent;
import jfan.fan.events.TrocaValorTemperaListener;

public class EnfileiradorEventos implements MelhorAritimeticaListener,
		MelhorHarmonicaListener, MelhorMaximoMinimoListener,
TrocaEpocaListener, TrocaValorTemperaListener {

	private final LinkedBlockingQueue<MelhorAritimeticaEvent>
queueMelhorAritmeticaEvent;

	private final LinkedBlockingQueue<MelhorHarmonicaEvent>
queueMelhorHarmonicaEvent;

	private final LinkedBlockingQueue<MelhorMaximoMinimoEvent>
queueMelhorMaxMinEvent;

	private final LinkedBlockingQueue<TrocaEpocaEvent>
queueTrocaEpocaEvent;

	private final LinkedBlockingQueue<TrocaValorTemperaEvent>
queueTrocaValorTemperaEvent;

	private final int MAX_FILA_EPOCA = 20;

	private final int MAX_FILA_OUTROS = 10;



	public EnfileiradorEventos(EvtQueues queues) {
		queueMelhorAritmeticaEvent =
queues.getQueueMelhorAritmeticaEvent();
```

```
            queueMelhorHarmonicaEvent =
queues.getQueueMelhorHarmonicaEvent();
            queueMelhorMaxMinEvent = queues.getQueueMelhorMaxMinEvent();
            queueTrocaEpocaEvent = queues.getQueueTrocaEpocaEvent();
            queueTrocaValorTemperaEvent =
queues.getQueueTrocaValorTemperaEvent();
      }

      public void trocouMelhorMediaAritimetica(MelhorAritimeticaEvent
event) {
            try {
                  if (queueMelhorAritmeticaEvent.size() >=
MAX_FILA_OUTROS) {
                        queueMelhorAritmeticaEvent.poll();
                  }
                  queueMelhorAritmeticaEvent.put(event);
            } catch (InterruptedException e) {
                  e.printStackTrace();          }
      }

      public void trocouMelhorMediaHarmonica(MelhorHarmonicaEvent event)
{
            try {
                  if (queueMelhorHarmonicaEvent.size() >=
MAX_FILA_OUTROS) {
                        queueMelhorHarmonicaEvent.poll();
                  }
                  queueMelhorHarmonicaEvent.put(event);
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }

      public void trocouMelhorMaximoMinimo(MelhorMaximoMinimoEvent event)
{
            try {
                  if (queueMelhorMaxMinEvent.size() >= MAX_FILA_OUTROS) {
                        queueMelhorMaxMinEvent.poll();
                  }
                  queueMelhorMaxMinEvent.put(event);
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
      }

      public void trocouEpoca(TrocaEpocaEvent event) {
            try {
                  if (queueTrocaEpocaEvent.size() >= MAX_FILA_OUTROS) {
                        queueTrocaEpocaEvent.poll();
                  }
                  queueTrocaEpocaEvent.put(event);
                              } catch (InterruptedException e) {
                  e.printStackTrace();          }
      }

      public void trocouValorTempera(TrocaValorTemperaEvent event) {
            try {
```

```
                if (queueTrocaValorTemperaEvent.size() >=
MAX_FILA_OUTROS) {
                      queueTrocaValorTemperaEvent.poll();
                }
                queueTrocaValorTemperaEvent.put(event);
          } catch (InterruptedException e) {
                e.printStackTrace();
          }
     }

}
```

1.2.6 EvtQueues.Java

```
package controle.eventos.auxiliares;

import java.util.concurrent.LinkedBlockingQueue;

import jfan.fan.events.MelhorAritimeticaEvent;
import jfan.fan.events.MelhorHarmonicaEvent;
import jfan.fan.events.MelhorMaximoMinimoEvent;
import jfan.fan.events.TrocaEpocaEvent;
import jfan.fan.events.TrocaValorTemperaEvent;

public class EvtQueues {

     private final LinkedBlockingQueue<MelhorAritimeticaEvent>
queueMelhorAritmeticaEvent = new
LinkedBlockingQueue<MelhorAritimeticaEvent>();

     private final LinkedBlockingQueue<MelhorHarmonicaEvent>
queueMelhorHarmonicaEvent = new
LinkedBlockingQueue<MelhorHarmonicaEvent>();

     private final LinkedBlockingQueue<MelhorMaximoMinimoEvent>
queueMelhorMaxMinEvent = new
LinkedBlockingQueue<MelhorMaximoMinimoEvent>();

     private final LinkedBlockingQueue<TrocaEpocaEvent>
queueTrocaEpocaEvent = new LinkedBlockingQueue<TrocaEpocaEvent>();

     private final LinkedBlockingQueue<TrocaValorTemperaEvent>
queueTrocaValorTemperaEvent = new
LinkedBlockingQueue<TrocaValorTemperaEvent>();

     public LinkedBlockingQueue<MelhorAritimeticaEvent>
getQueueMelhorAritmeticaEvent() {
          return this.queueMelhorAritmeticaEvent;
     }

     public LinkedBlockingQueue<MelhorHarmonicaEvent>
getQueueMelhorHarmonicaEvent() {
          return this.queueMelhorHarmonicaEvent;
     }
```

```
	public LinkedBlockingQueue<MelhorMaximoMinimoEvent>
getQueueMelhorMaxMinEvent() {
		return this.queueMelhorMaxMinEvent;
	}

	public LinkedBlockingQueue<TrocaEpocaEvent>
getQueueTrocaEpocaEvent() {
		return this.queueTrocaEpocaEvent;
	}

	public LinkedBlockingQueue<TrocaValorTemperaEvent>
getQueueTrocaValorTemperaEvent() {
		return this.queueTrocaValorTemperaEvent;
	}
}
```

1.3 PACKAGE CONTROLE.MAIN;

1.3.1 App.Java

```
package controle.main;


import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Label;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import jfan.exceptions.JfanPorcentagemException;
import jfan.fan.MonitorFAN;
import jfan.fan.RedeFAN;
import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import jfan.io.arquivos.GerenciadorPadroes;
import modelo.ApplicationConfig;
import modelo.DataSetPool;
import modelo.Projeto;
import visao.gui.telas.classes.EasyFan;
import controle.eventos.auxiliares.EnfileiradorEventos;
import controle.eventos.auxiliares.EvtQueues;

public class App {

	private static File ultimoArquivo = null;

	private static int ultimoFiltro = 0;
```

```
	private static MonitorFAN monitorFAN = null;

	private static GerenciadorPadroes gerenciadorPad = new
GerenciadorPadroes();

	private static EasyFan tela;

	private static final ApplicationConfig appConfig = new
ApplicationConfig();

	private static final Projeto projetoConfig = new
Projeto("./ArqConfigEasyFAN.xml");

	private static final EvtQueues eventQueues = new EvtQueues();

	private static final EnfileiradorEventos enfileirador = new
EnfileiradorEventos(eventQueues);

	private static boolean precisaNormalizar = true;

	private static final DataSetPool datasetpool = new DataSetPool();

	private static Object lockGraphicStatistic = new Object();
	public static void main(String[] args) {
		try {
			tela = new EasyFan();

			tela.open();
		} catch (IOException e) {
			e.printStackTrace();
			Dialog d = new Dialog(new Frame(),"Erro",true);
			String message = ("Ocorreu um erro ao tentar abrir a
GUI principal, " +
			"o arquivo EasyFAN.xml deve estar corrompido ou
adulterado inadequadamente.");
			Label l = new Label(message,Label.CENTER);
			d.add(l);
			d.setVisible(true);

		}
	}

	public static void iniciaRede() {
		try {
			monitorFAN = new MonitorFAN(new
RedeFAN(projetoConfig.getRedeConf().getRaioDifuso(),
projetoConfig.getRedeConf().getSuporteConjuntosDifusos()),gerenciadorPad)
;
			applyConfigurations();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public static GerenciadorPadroes getGerenciadorPadroes() {
		return gerenciadorPad;
```

```
	}

	public static MonitorFAN getMonitorFAN() {
		return monitorFAN;
	}

	public static ApplicationConfig getApplicationConfig() {
		return appConfig;
	}

	public static EvtQueues getEventQueues() {
		return eventQueues;
	}

	public static Projeto getProjetoConfig() {
		return projetoConfig;
	}

	private static void applyConfigurations() {


	monitorFAN.setTemperaSimulada(projetoConfig.getTreinoConf().getTemp
era());

	monitorFAN.setCadaEpocaEmbaralharPadroes(projetoConfig.getTreinoCon
f().getCadaEpocaEmbaralharPadroes());

	monitorFAN.setCadaEpocaReiniciarTempera(projetoConfig.getTreinoConf
().getCadaEpocaResetTempera());

	monitorFAN.setCadaEpocaRenormalizar(projetoConfig.getTreinoConf().g
etCadaEpocaRenormalizar());

	monitorFAN.setCadaEpocaStepTempera(projetoConfig.getTreinoConf().ge
tCadaEpocaStepTempera());
		try {

	monitorFAN.setChanceRenormalizacao(projetoConfig.getTreinoConf().ge
tChanceRenormalizar());
		} catch (JfanPorcentagemException e) {
			e.printStackTrace();
		}

	monitorFAN.setNumeroPadroesTreinamento(projetoConfig.getTreinoConf(
).getNumeroPadroesTreino());
		monitorFAN.addValidacaoListener(tela.getEvtValidacao());
		projetoConfig.getRedeConf().setNormalizatorConfiguration(new
NormalizatorConfiguration());
	}

	public static boolean isPrecisaNormalizar() {
		return precisaNormalizar;
	}

	public static void setPrecisaNormalizar(boolean precisaNormalizar)
{
		App.precisaNormalizar = precisaNormalizar;
```

```
	}

	public static EasyFan getTelaEasyFAN() {
		return tela;
	}

	public static DataSetPool getDatasetpool() {
		return datasetpool;
	}

	{
	}

	public static EnfileiradorEventos getEnfileiradorEventos() {
		return enfileirador;
	}

	public static void vericarMonitor() {
		if (monitorFAN == null) {
			App.iniciaRede();

	App.getMonitorFAN().addTrocaEpocaListener(enfileirador);

	App.getMonitorFAN().addMaximoMinimoListener(enfileirador);

	App.getMonitorFAN().addMelhorAritimeticaListener(enfileirador);

	App.getMonitorFAN().addMelhorHarmonicaListener(enfileirador);
			App.getMonitorFAN().setCadaEpocaEmbaralharPadroes(50);
			App.getMonitorFAN().setCadaEpocaStepTempera(0);

			App.getMonitorFAN().setCadaEpocaReiniciarTempera(500);
			App.getMonitorFAN().setCadaEpocaRenormalizar(2000);
			ITemperaSimulada temp =
App.getProjetoConfig().getTreinoConf().getTempera();
			if (temp != null)
				temp.addTrocaValorTemperaListener(enfileirador);

	monitorFAN.addValidacaoListener(tela.getEvtValidacao());
		}

	}

	public static Object getLockGraphicStatistic() {
		return lockGraphicStatistic;
	}

	public static void reiniciarGerenciadorPadroes() {
		gerenciadorPad = new GerenciadorPadroes();
	}

	public static void requisitarNormalizacao() {
		if(precisaNormalizar) {
			//firstRunning = false;
			ArrayList<IPadrao> tr,ts, vl, cl;
			try {
				tr = gerenciadorPad.getConjuntoTreinamentoRede();
```

```
                        ts = gerenciadorPad.getConjuntoTesteRede();
                        vl = gerenciadorPad.getConjuntoValidacaoRede();
                        cl =
gerenciadorPad.getConjuntoClassificacaoRede();

                        INormalizator norm = null;

                        NormalizatorConfiguration config =
App.getProjetoConfig().getRedeConf().getNormalizatorConfiguration();


      config.setMaximos(App.getProjetoConfig().getRedeConf().getMaximos()
);

      config.setMinimos(App.getProjetoConfig().getRedeConf().getMinimos()
);

      config.setMedias(App.getProjetoConfig().getRedeConf().getMedias());


                        NormalizatorTypes tipo =
App.getProjetoConfig().getRedeConf().getTipoNormalizador();

                        if (tipo == NormalizatorTypes.NormalizatorMaxMin)
{
                              norm = new NormalizatorMaxMin(config);
                        }
                        else if (tipo ==
NormalizatorTypes.NormalizatorMaxMean) {
                              norm = new NormalizatorMaxMean(config);
                        }
                        else if (tipo ==
NormalizatorTypes.NormalizatorArcSinMaxMin) {
                        }
                        else{
                              norm = new NormalizatorMax(config);
                        }

                        norm.normalizarThreaded(tr);
                        norm.normalizarThreaded(ts);
                        norm.normalizarThreaded(vl);
                        norm.normalizarThreaded(cl);

                        CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(App.getProjetoConfig().getRedeConf().getR
aioDifuso(),
App.getProjetoConfig().getRedeConf().getSuporteConjuntosDifusos());
                        calc.calcularCaracteristicasFAN(tr);
                        calc.calcularCaracteristicasFAN(ts);
                        calc.calcularCaracteristicasFAN(vl);
                        calc.calcularCaracteristicasFAN(cl);
                        precisaNormalizar =  false;
                  }
                  catch (Exception e) {
                        e.printStackTrace();
                  }
            }
```

```
	}

	public static File getUltimoArquivo() {
		return ultimoArquivo;
	}

	public static void setUltimoArquivo(File ultimoArquivo) {
		App.ultimoArquivo = ultimoArquivo;
	}

	public static int getUltimoFiltro() {
		return ultimoFiltro;
	}

	public static void setUltimoFiltro(int ultimoFiltro) {
		App.ultimoFiltro = ultimoFiltro;
	}
}
```

1.4 PACKAGE MODELO;

1.4.1 AbaTesteBean.Java

```
package modelo;

import java.util.ArrayList;

import jfan.fan.NeuronioFAN;

public class AbaTesteBean {

	private int[][] matrizConfusao;

	private ArrayList<NeuronioFAN> neuroniosTeste;

	private float harmonica, aritmetica, maxmin;

	public float getAritmetica() {
		return aritmetica;
	}

	public void setAritmetica(float aritmetica) {
		this.aritmetica = aritmetica;
	}

	public float getHarmonica() {
		return harmonica;
	}

	public void setHarmonica(float harmonica) {
		this.harmonica = harmonica;
	}

	public int[][] getMatrizConfusao() {
		return matrizConfusao;
	}
```

```
	public void setMatrizConfusao(int[][] matrizConfusao) {
		this.matrizConfusao = matrizConfusao;
	}

	public float getMaxmin() {
		return maxmin;
	}

	public void setMaxmin(float maxmin) {
		this.maxmin = maxmin;
	}

	public ArrayList<NeuronioFAN> getNeuroniosTeste() {
		return neuroniosTeste;
	}

	public void setNeuroniosTeste(ArrayList<NeuronioFAN>
neuroniosTeste) {
		this.neuroniosTeste = neuroniosTeste;
	}

}
package modelo;

import java.util.ArrayList;

import jfan.fan.NeuronioFAN;

public class AbaTesteBean {

	private int[][] matrizConfusao;

	private ArrayList<NeuronioFAN> neuroniosTeste;

	private float harmonica, aritmetica, maxmin;

	public float getAritmetica() {
		return aritmetica;
	}

	public void setAritmetica(float aritmetica) {
		this.aritmetica = aritmetica;
	}

	public float getHarmonica() {
		return harmonica;
	}

	public void setHarmonica(float harmonica) {
		this.harmonica = harmonica;
	}

	public int[][] getMatrizConfusao() {
		return matrizConfusao;
	}
```

```
	public void setMatrizConfusao(int[][] matrizConfusao) {
		this.matrizConfusao = matrizConfusao;
	}

	public float getMaxmin() {
		return maxmin;
	}

	public void setMaxmin(float maxmin) {
		this.maxmin = maxmin;
	}

	public ArrayList<NeuronioFAN> getNeuroniosTeste() {
		return neuroniosTeste;
	}

	public void setNeuroniosTeste(ArrayList<NeuronioFAN>
neuroniosTeste) {
		this.neuroniosTeste = neuroniosTeste;
	}

}
```

1.4.2 ApplicationConfig.Java

```
package modelo;

public class ApplicationConfig {

	public static class PropriedadesGUI {

		private static int linhasPorPagina = 400;

		public static int getLinhasPorPagina() {
			return linhasPorPagina;
		}

		public static void setLinhasPorPagina(int linhasPorPagina) {
			PropriedadesGUI.linhasPorPagina = linhasPorPagina;
		}


	}

}
```

1.4.3 DataSetPool.Java

```
package modelo;

import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;

import controle.main.App;
```

```
public class DataSetPool {

	private final CategoryDataset datasetEstatistica = new
DefaultCategoryDataset();
	private String serieMedHarm = "Média Harmônica";
	private String serieMaxMin = "Máximo Mínimo";
	private String serieMedArit = "Média Aritmética";

	public void addEstatisticData(int epoca, double harmonica, double
maxmin, double aritmetica) {
		synchronized (App.getLockGraphicStatistic()) {

	((DefaultCategoryDataset)datasetEstatistica).addValue(harmonica,
serieMedHarm, Integer.toString(epoca));

	((DefaultCategoryDataset)datasetEstatistica).addValue(maxmin,
serieMaxMin, Integer.toString(epoca));

	((DefaultCategoryDataset)datasetEstatistica).addValue(aritmetica,
serieMedArit, Integer.toString(epoca));
			if (datasetEstatistica.getColumnCount() > 20) {
		((DefaultCategoryDataset)datasetEstatistica).removeColumn(0);
				}
		}

	}

	public CategoryDataset getDatasetEstatistica() {
		return datasetEstatistica;
	}

}
```

1.4.4 Projeto.Java

```
package modelo;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import jfan.fan.StoredNetworkType;
import jfan.fan.temperas.TemperaAleatoria;
import jfan.fan.temperas.TemperaSimuladaFAN;
import jfan.fan.temperas.TipoTempera;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
```

```
public class Projeto {

	private String nome;

	private RedeConfig rede = new RedeConfig();

	private TreinamentoConfig treino = new TreinamentoConfig();

	private ValidacaoConfig validacao = new ValidacaoConfig();

	private String arquivoAssociado;

	public Projeto(String arquivoConfig) {
		arquivoAssociado = arquivoConfig;
		loadDefaultConfigurationXML(arquivoConfig);
	}

	public String getNome() {
		return this.nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public RedeConfig getRedeConf() {
		return this.rede;
	}

	public void setRedeConf(RedeConfig rede) {
		this.rede = rede;
	}

	public TreinamentoConfig getTreinoConf() {
		return this.treino;
	}

	public void setTreinoConf(TreinamentoConfig treino) {
		this.treino = treino;
	}

	public ValidacaoConfig getValidacaoConfig() {
		return validacao;
	}

	public void setValidacaoConfig(ValidacaoConfig validacao) {
		this.validacao = validacao;
	}

	private boolean loadDefaultConfigurationXML(String file) {

		if (file == null) {
			return false;
		}

		File arq = new File(file);
```

```
            if (arq.exists()) {
                  try {

                        SAXParser parser =
SAXParserFactory.newInstance().newSAXParser();

                        InputSource input = new InputSource(file);

                        parser.parse(input, new LeitorXmlConfig());

                        return true;
                  }
                  catch (ParserConfigurationException ex) {
                        ex.printStackTrace();
                  }
                  catch (SAXException ex) {
                        ex.printStackTrace();
                  }
                  catch (IOException ex) {
                        ex.printStackTrace();
                  }

            }

            return false;

      }

      public void saveAsDefaultConfiguration() {
            try {
                  BufferedWriter bw = new BufferedWriter(new
FileWriter(arquivoAssociado));
                  bw.write("<?xml version=\"1.0\"?>"); bw.newLine();
                  bw.write("<defaultConfiguration>"); bw.newLine();

                  bw.write("<networkConfiguration>"); bw.newLine();
                  bw.write("\t<suporteConjuntosDifusos>");

      bw.write(Integer.toString(rede.getSuporteConjuntosDifusos()));
                  bw.write("</suporteConjuntosDifusos>"); bw.newLine();
                  bw.write("\t<raioDifuso>");
                  bw.write(Integer.toString(rede.getRaioDifuso()));
                  bw.write("</raioDifuso>"); bw.newLine();
                  bw.write("</networkConfiguration>"); bw.newLine();

                  bw.write("<trainningConfiguration>"); bw.newLine();
                  bw.write("\t<cadaEpocaRenormalizar>");

      bw.write(Integer.toString(treino.getCadaEpocaRenormalizar()));
                  bw.write("</cadaEpocaRenormalizar>"); bw.newLine();
                  bw.write("\t<chanceRenormalizar>");

      bw.write(Integer.toString(treino.getChanceRenormalizar()));
                  bw.write("</chanceRenormalizar>"); bw.newLine();
                  bw.write("\t<cadaEpocaEmbaralharPadroes>");
```

```
      bw.write(Integer.toString(treino.getCadaEpocaEmbaralharPadroes()));
                  bw.write("</cadaEpocaEmbaralharPadroes>");
bw.newLine();
                  bw.write("\t<numeroPadroesTreino>");

      bw.write(Integer.toString(treino.getNumeroPadroesTreino()));
                  bw.write("</numeroPadroesTreino>"); bw.newLine();
                  bw.write("\t<randomizarPesos>");
                  bw.write(treino.isRandomizarPesos()?"true":"false");
                  bw.write("</randomizarPesos>"); bw.newLine();
                  bw.write("\t<pesoBase>");
                  bw.write(Integer.toString(treino.getPesoBase()));
                  bw.write("</pesoBase>"); bw.newLine();
                        bw.write("\t<tempera tipo=\""
+treino.getTipoTempera().toString() + "\">"); bw.newLine();
                        bw.write("\t\t<maximo>");

      bw.write(Double.toString(treino.getTempera().getLimiteMaximo()));
                        bw.write("</maximo>"); bw.newLine();
                        bw.write("\t\t<minimo>");

      bw.write(Double.toString(treino.getTempera().getLimiteMinimo()));
                        bw.write("</minimo>"); bw.newLine();
                        bw.write("\t\t<step>");

      bw.write(Double.toString(treino.getTempera().getStepValue()));
                        bw.write("</step>"); bw.newLine();
                        bw.write("\t</tempera>"); bw.newLine();
                  bw.write("\t<cadaEpocaResetTempera>");

      bw.write(Integer.toString(treino.getCadaEpocaResetTempera()));
                  bw.write("</cadaEpocaResetTempera>"); bw.newLine();
                  bw.write("\t<cadaEpocaStepTempera>");

      bw.write(Integer.toString(treino.getCadaEpocaStepTempera()));
                  bw.write("</cadaEpocaStepTempera>"); bw.newLine();
                  bw.write("\t<porcentagemPadroesTreino>");

      bw.write(Integer.toString(treino.getPorcentagemPadroesTreino()));
                  bw.write("</porcentagemPadroesTreino>"); bw.newLine();
                  bw.write("\t<numeroEpocasTreinar>");

      bw.write(Integer.toString(treino.getNumeroEpocasTreinar()));
                  bw.write("</numeroEpocasTreinar>"); bw.newLine();
                        bw.write("\t<tempoParada>"); bw.newLine();
                        bw.write("\t\t<dias>");

      bw.write(Integer.toString(treino.getDiasParada()));
                        bw.write("</dias>"); bw.newLine();
                        bw.write("\t\t<horas>");

      bw.write(Integer.toString(treino.getHorasParada()));
                        bw.write("</horas>"); bw.newLine();
                        bw.write("\t\t<minutos>");

      bw.write(Integer.toString(treino.getMinutosParada()));
```

```
                        bw.write("</minutos>"); bw.newLine();
                        bw.write("\t\t<segundos>");

      bw.write(Integer.toString(treino.getSegundosParada()));
                        bw.write("</segundos>"); bw.newLine();
                        bw.write("\t</tempoParada>"); bw.newLine();
                  bw.write("</trainningConfiguration>");
                  bw.write("<validationConfiguration>"); bw.newLine();
                  bw.write("\t<cadaEpocaValidar>");

      bw.write(Integer.toString(validacao.getCadaEpocaValidar()));
                  bw.write("</cadaEpocaValidar>"); bw.newLine();
                  bw.write("\t<tipoRedeValidacao>");
                  bw.write(validacao.getTipoRedeValidacao().toString());
                  bw.write("</tipoRedeValidacao>"); bw.newLine();
                  bw.write("</validationConfiguration>"); bw.newLine();

                  bw.write("</defaultConfiguration>");

                  bw.close();

            } catch (IOException e) {
                  e.printStackTrace();
            }

      }

      private class LeitorXmlConfig extends DefaultHandler {

            private String tag;

            private StringBuilder value = new StringBuilder(150);

            private Attributes attributes = null;

            private int dias,horas,minutos,segundos;
            public void startDocument() { }
            public void startElement(String uri,String localName,String
tag,Attributes atributos){
                  this.tag = tag;
                  value.setLength(0);
                  attributes = atributos;
            }

            public void characters(char[] ch, int start, int length)
throws SAXException{
                  if (tag.equalsIgnoreCase("/suporteConjuntosDifusos")) {

      rede.setSuporteConjuntosDifusos(Integer.parseInt(value.toString()))
;
                  }
                  else if (tag.equalsIgnoreCase("/raioDifuso")) {

      rede.setRaioDifuso(Integer.parseInt(value.toString()));
                  }
                  else if
(tag.equalsIgnoreCase("/cadaEpocaRenormalizar")) {
```

```
		treino.setCadaEpocaRenormalizar(Integer.parseInt(value.toString()))
;
			}
			else if (tag.equalsIgnoreCase("/chanceRenormalizar")) {

		treino.setChanceRenormalizar(Integer.parseInt(value.toString()));
			}
			else if
(tag.equalsIgnoreCase("/cadaEpocaEmbaralharPadroes")) {

		treino.setCadaEpocaEmbaralharPadroes(Integer.parseInt(value.toStrin
g()));
			}
			else if (tag.equalsIgnoreCase("/numeroPadroesTreino"))
{

		treino.setNumeroPadroesTreino(Integer.parseInt(value.toString()));
			}
			else if (tag.equalsIgnoreCase("/randomizarPesos")) {
				if (value.toString().equals("true")) {
					treino.setRandomizarPesos(true);
				}
				else {
					treino.setRandomizarPesos(false);
				}
			}
			else if (tag.equalsIgnoreCase("/pesoBase")) {

		treino.setPesoBase(Integer.parseInt(value.toString()));
			}
			else if (tag.equalsIgnoreCase("tempera")) {
				String s = attributes.getValue(0);
				if (s.equals("TemperaAleatorio")) {
					treino.setTempera(new TemperaAleatoria());

		treino.setTipoTempera(TipoTempera.TemperaAleatoria);
				}
				else if (s.equals("TemperaSimulada")) {
					treino.setTempera(new TemperaSimuladaFAN());

		treino.setTipoTempera(TipoTempera.TemperaSimuladaFAN);
				}
			}
			else if (tag.equalsIgnoreCase("/maximo")) {

		treino.getTempera().setLimiteMaximo(Double.parseDouble(value.toStri
ng()));
			}
			else if (tag.equalsIgnoreCase("/minimo")) {

		treino.getTempera().setLimiteMinimo(Double.parseDouble(value.toStri
ng()));
			}
			else if (tag.equalsIgnoreCase("/step")) {

		treino.getTempera().setStep(Double.parseDouble(value.toString()));
```

```
                }
                else if
(tag.equalsIgnoreCase("/cadaEpocaResetTempera")) {

      treino.setCadaEpocaResetTempera(Integer.parseInt(value.toString()))
;
                }
                else if (tag.equalsIgnoreCase("/cadaEpocaStepTempera"))
{

      treino.setCadaEpocaStepTempera(Integer.parseInt(value.toString()));
                }
                else if
(tag.equalsIgnoreCase("/porcentagemPadroesTreino")) {

      treino.setPorcentagemPadroesTreino(Integer.parseInt(value.toString(
)));
                }
                else if (tag.equalsIgnoreCase("/numeroEpocasTreinar"))
{

      treino.setNumeroEpocasTreinar(Integer.parseInt(value.toString()));
                }
                else if (tag.equalsIgnoreCase("/dias")) {
                      dias = Integer.parseInt(value.toString());
                }
                else if (tag.equalsIgnoreCase("/horas")) {
                      horas = Integer.parseInt(value.toString());
                }
                else if (tag.equalsIgnoreCase("/minutos")) {
                      minutos = Integer.parseInt(value.toString());
                }
                else if (tag.equalsIgnoreCase("/segundos")) {
                      segundos = Integer.parseInt(value.toString());
                }
                else if (tag.equalsIgnoreCase("/tempoParada")) {
                      treino.setTempoParada(dias, horas, minutos,
segundos);
                }
                else if (tag.equalsIgnoreCase("/cadaEpocaValidar")) {

      validacao.setCadaEpocaValidar(Integer.parseInt(value.toString()));
                }
                else if (tag.equalsIgnoreCase("/tipoRedeValidacao")) {
                      String s = value.toString();
                      StoredNetworkType snt =
StoredNetworkType.BestHarmonicMean;
                      if (s.equalsIgnoreCase("Actual")) {
                            snt = StoredNetworkType.Actual;
                      }
                      else if (s.equalsIgnoreCase("BestAritmethicMean"))
{
                            snt = StoredNetworkType.BestAritmethicMean;
                      }
                      else if (s.equalsIgnoreCase("BestMaxMin")) {
                            snt = StoredNetworkType.BestMaxMin;
                      }
```

```
                        validacao.setTipoRedeValidacao(snt);
                  }
                  else {
                        value.append(ch,start,length);
                  }
            }
            public void endElement(String uri, String localName, String
tag){        this.tag = "/" + tag;
            }

            public void endDocument() { }
      }



}
```

1.4.5 RedeConfig.Java

```
package modelo;

import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorTypes;

public class RedeConfig {

      private int suporteConjuntosDifusos = 100;

      private int raioDifuso = 6;

      private NormalizatorTypes tipoNormalizador =
NormalizatorTypes.NormalizatorMaxMin;

      private NormalizatorConfiguration normalizatorConfiguration = new
NormalizatorConfiguration();

      double[] maximos = null;

      boolean[] maximosAutomaticos = null;

      double[] minimos = null;

      boolean[] minimosAutomaticos = null;

      double[] medias = null;

      boolean[] mediasAutomaticas = null;

      public int getRaioDifuso() {
            return this.raioDifuso;
      }

      public void setRaioDifuso(int raioDifuso) {
            this.raioDifuso = raioDifuso;
      }
```

```
	public int getSuporteConjuntosDifusos() {
		return this.suporteConjuntosDifusos;
	}

	public void setSuporteConjuntosDifusos(int suporteConjuntosDifusos)
{
		this.suporteConjuntosDifusos = suporteConjuntosDifusos;
	}

	public NormalizatorConfiguration getNormalizatorConfiguration() {
		return this.normalizatorConfiguration;
	}

	public void setNormalizatorConfiguration(
			NormalizatorConfiguration normalizatorConfiguration) {
		this.normalizatorConfiguration = normalizatorConfiguration;
	}

	public NormalizatorTypes getTipoNormalizador() {
		return this.tipoNormalizador;
	}

	public void setTipoNormalizador(NormalizatorTypes tipoNormalizador)
{
		this.tipoNormalizador = tipoNormalizador;
	}

	public double[] getMaximos() {
		return this.maximos;
	}

	public void setMaximos(double[] maximos) {
		this.maximos = maximos;
	}

	public double[] getMedias() {
		return this.medias;
	}

	public void setMedias(double[] medias) {
		this.medias = medias;
	}

	public double[] getMinimos() {
		return this.minimos;
	}

	public void setMinimos(double[] minimos) {
		this.minimos = minimos;
	}

	public boolean[] getMaximosAutomaticos() {
		return maximosAutomaticos;
	}

	public void setMaximosAutomaticos(boolean[] maximosAutomaticos) {
		this.maximosAutomaticos = maximosAutomaticos;
```

```
	}

	public boolean[] getMediasAutomaticas() {
		return mediasAutomaticas;
	}

	public void setMediasAutomaticas(boolean[] mediasAutomaticas) {
		this.mediasAutomaticas = mediasAutomaticas;
	}

	public boolean[] getMinimosAutomaticos() {
		return minimosAutomaticos;
	}

	public void setMinimosAutomaticos(boolean[] minimosAutomaticos) {
		this.minimosAutomaticos = minimosAutomaticos;
	}
}
```

1.4.6 TreinamentoConfig.Java

```
package modelo;

import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.temperas.TemperaAleatoria;
import jfan.fan.temperas.TipoTempera;

public class TreinamentoConfig {

	private int cadaEpocaRenormalizar = 0;

	private int chanceRenormalizar = 0;

	private int cadaEpocaEmbaralharPadroes = 50;

	private int numeroPadroesTreino = 0;

	private boolean randomizarPesos = false;

	private int pesoBase = 500;

	private ITemperaSimulada tempera = new TemperaAleatoria();

	private int cadaEpocaResetTempera = 100;

	private TipoTempera tipoTempera = TipoTempera.TemperaAleatoria;

	private int cadaEpocaStepTempera = 10;

	private int porcentagemPadroesTreino = 0;

	private int numeroEpocasTreinar = -1;

	private int diasParada, horasParada, minutosParada, segundosParada;
```

```
	public int getCadaEpocaEmbaralharPadroes() {
		return this.cadaEpocaEmbaralharPadroes;
	}

	public void setCadaEpocaEmbaralharPadroes(int
cadaEpocaEmbaralharPadroes) {
		this.cadaEpocaEmbaralharPadroes = cadaEpocaEmbaralharPadroes;
	}

	public int getCadaEpocaRenormalizar() {
		return this.cadaEpocaRenormalizar;
	}

	public void setCadaEpocaRenormalizar(int cadaEpocaRenormalizar) {
		this.cadaEpocaRenormalizar = cadaEpocaRenormalizar;
	}

	public int getChanceRenormalizar() {
		return this.chanceRenormalizar;
	}

	public void setChanceRenormalizar(int chanceRenormalizar) {
		this.chanceRenormalizar = chanceRenormalizar;
	}

	public int getNumeroPadroesTreino() {
		return this.numeroPadroesTreino;
	}

	public void setNumeroPadroesTreino(int numeroPadroesTreino) {
		this.numeroPadroesTreino = numeroPadroesTreino;
	}

	public int getPesoBase() {
		return this.pesoBase;
	}

	public void setPesoBase(int pesoBase) {
		this.pesoBase = pesoBase;
	}

	public boolean isRandomizarPesos() {
		return this.randomizarPesos;
	}

	public void setRandomizarPesos(boolean randomizarPesos) {
		this.randomizarPesos = randomizarPesos;
	}

	public ITemperaSimulada getTempera() {
		return this.tempera;
	}

	public void setTempera(ITemperaSimulada tempera) {
		this.tempera = tempera;
	}
```

```
	public int getCadaEpocaResetTempera() {
		return this.cadaEpocaResetTempera;
	}

	public void setCadaEpocaResetTempera(int cadaEpocaResetTempera) {
		this.cadaEpocaResetTempera = cadaEpocaResetTempera;
	}

	public int getCadaEpocaStepTempera() {
		return this.cadaEpocaStepTempera;
	}

	public void setCadaEpocaStepTempera(int cadaEpocaStepTempera) {
		this.cadaEpocaStepTempera = cadaEpocaStepTempera;
	}

	public TipoTempera getTipoTempera() {
		return this.tipoTempera;
	}

	public void setTipoTempera(TipoTempera tipoTempera) {
		this.tipoTempera = tipoTempera;
	}

	public int getPorcentagemPadroesTreino() {
		return porcentagemPadroesTreino ;
	}

	public void setPorcentagemPadroesTreino(int porcentagem) {
		this.porcentagemPadroesTreino = porcentagem;
	}

	public int getNumeroEpocasTreinar() {
		return numeroEpocasTreinar;
	}

	public void setNumeroEpocasTreinar(int numeroEpocasTreinar) {
		this.numeroEpocasTreinar = numeroEpocasTreinar;
	}

	public void setTempoParada(int diasParada,int horasParada,int
minutosParada,int segundosParada) {
		this.diasParada = diasParada;
		this.horasParada = horasParada;
		this.minutosParada = minutosParada;
		this.segundosParada = segundosParada;
	}

	public int getDiasParada() {
		return diasParada;
	}

	public int getHorasParada() {
		return horasParada;
	}
```

```
	public int getMinutosParada() {
		return minutosParada;
	}

	public int getSegundosParada() {
		return segundosParada;
	}

}
```

1.4.7 ValidacaoConfig.Java

```
package modelo;

import jfan.fan.StoredNetworkType;

public class ValidacaoConfig {

	private int cadaEpocaValidar = 0;

	private StoredNetworkType tipoRedeValidacao =
StoredNetworkType.BestHarmonicMean;

	public int getCadaEpocaValidar() {
		return cadaEpocaValidar;
	}

	public void setCadaEpocaValidar(int cadaEpocaValidar) {
		this.cadaEpocaValidar = cadaEpocaValidar;
	}

	public StoredNetworkType getTipoRedeValidacao() {
		return tipoRedeValidacao;
	}

	public void setTipoRedeValidacao(StoredNetworkType
tipoRedeValidacao) {
		this.tipoRedeValidacao = tipoRedeValidacao;
	}

}
```

1.5 PACKAGE VISAO.GUI.TELAS.CLASSES;

1.5.1 CaracteristicasUtilizadas.Java

```
package visao.gui.telas.classes;


	import java.awt.Dialog;
	import java.awt.Frame;
	import java.io.IOException;
	import thinlet.Thinlet;
	import thinletcommons.AntiAliasedThinlet;
	import thinletcommons.ThinletDialog;
```

```
public class CaracteristicasUtilizadas {

	private Thinlet thinlet;
	private ThinletDialog dialog;

	public CaracteristicasUtilizadas(Dialog owner) {
		this.dialog = new ThinletDialog(owner,false);
		this.init();
	}

	public CaracteristicasUtilizadas(Dialog owner, String title)
{
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	public CaracteristicasUtilizadas(Frame owner) {
		this.dialog = new ThinletDialog(owner,false);
		this.init();
	}
	public CaracteristicasUtilizadas(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/CaracteristicasUtilizadas.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
	  thinlet.add(panel);
	  dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setSize(315, 135);
		dialog.setAlwaysOnTop(true);
	  dialog.setLocationRelativeTo(dialog.getOwner());
	  dialog.setVisible(true);
	}
	  public void confirmar(){

	  }

	  public void cancelar(){
		dialog.setVisible(false);
	  }

	  public static void main(String[] args) {
```

```
                    CaracteristicasUtilizadas cu = new
CaracteristicasUtilizadas(new Frame(),"Selecao de Caracteristicas");
                    cu.show();
                    while (cu.dialog.isShowing()) {}
                    cu.dialog.dispose();

                    System.exit(0);

              }

       }
```

1.5.2 CarregarRede.Java

```
package visao.gui.telas.classes;

import java.awt.Frame;
import java.io.File;

import jfan.io.arquivos.CarregaNeuronioXML;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ExtensionFileFilter;
import thinletcommons.FileChooser;
import thinletcommons.FileFilter;
import thinletcommons.ThinletDialog;
import controle.main.App;


public class CarregarRede extends AntiAliasedThinlet {

       private ThinletDialog dialog;
       private AntiAliasedThinlet thinlet;
     private FileChooser fc;

       public void carregarRede(Frame ower, String title) {
             fc = new
FileChooser(App.getTelaEasyFAN().getFrame(),"Carregar Rede",
FileChooser.MODE_OPEN);
             fc.setFileFilters(new FileFilter[]{
                    new FileFilter(){
                           public boolean accept(File dir) {
                               if (dir.getAbsolutePath().indexOf(".enn") >
-1 || dir.getAbsolutePath().indexOf(".fan") > -1 ) {
                                  return true; }
                                 return false; }
                           public String getDescription() { return "Redes
Suportadas (enn,fan)"; }
                          },
                          new ExtensionFileFilter("enn", "EasyFAN files
(*.enn)"),
                          new ExtensionFileFilter("fan", "FAN files
(*.fan)"),
               });
              fc.show();
              File selectedFile = fc.getSelectedFile();
              String enderecoDoArquivo = selectedFile.getAbsolutePath();
```

```
            String extArquivo = fc.getFilterChanged();
            extArquivo = extArquivo.substring(extArquivo.length()-
5,extArquivo.length()-1);
            if (extArquivo.equalsIgnoreCase(".enn")){
                  CarregaNeuronioXML cxml = new
CarregaNeuronioXML(enderecoDoArquivo,App.getGerenciadorPadroes());
                  try {

      System.out.println(cxml.getNumeroCaracteristicas());
                  } catch (Exception e1) {
                        e1.printStackTrace();
                  }
                  try {

      App.getMonitorFAN().setNeuronios(cxml.getNeuroniosFAN());
                  } catch (Exception e) {
                        e.printStackTrace();
                  }


            }
   }

}
```

1.5.3 ClassificarRede.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.eventos.EvtClassificacao;
import controle.eventos.EvtValidacao;
import controle.main.App;

public class ClassificaRede {

      private Thinlet thinlet;

      private ThinletDialog dialog;

      private StoredNetworkType tipoRede =
StoredNetworkType.BestHarmonicMean;

      private boolean carregar = false;

      private EvtClassificacao evtClassificacao =
App.getTelaEasyFAN().getEvtClassificacao();
```

```
	public ClassificaRede(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ClassificaRede(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ClassificaRede(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ClassificaRede(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel = thinlet.parse(
					"/visao/gui/telas/TesteRede.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setModal(true);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
	}

	public void confirmar() {
		if (carregar) {
			ArrayList<NeuronioFAN> neurons =
App.getTelaEasyFAN().getEvtAbrirArquivo().abrirRedeSomenteNeuronios();
			evtClassificacao.classificar(neurons);
		}
		else {
			evtClassificacao.classificar(tipoRede);
		}
		dialog.setVisible(false);
	}

	public void cancelar() {
		dialog.setVisible(false);
	}
```

```
	public void radioHarmonica() {
		this.carregar = false;
		this.tipoRede = StoredNetworkType.BestHarmonicMean;
	}

	public void radioAritmetica() {
		this.carregar = false;
		this.tipoRede = StoredNetworkType.BestAritmethicMean;
	}

	public void radioMaxMin() {
		this.carregar = false;
		this.tipoRede = StoredNetworkType.BestMaxMin;
	}

	public void radioAtual() {
		this.carregar = false;
		this.tipoRede = StoredNetworkType.Actual;
	}

	public void radioCarregada() {
		this.carregar = true;
	}


}
```

1.5.4 ConfigClassificacao.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class ConfigClassificacao {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	public ConfigClassificacao(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ConfigClassificacao(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ConfigClassificacao(Frame owner) {
```

```
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ConfigClassificacao(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel = thinlet.parse(

	"/visao/gui/telas/ParametrosClassificacao.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setSize(445, 160);
		dialog.setAlwaysOnTop(true);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
	}

	public void confirmar() {

	}

	public void cancelar() {
		dialog.setVisible(false);
	}

	public static void main(String[] args) {
		ConfigClassificacao cc = new ConfigClassificacao(new Frame(),
				"Configuraçoes de Classificacao");
		cc.show();
		while (cc.dialog.isShowing()) {
		}
		cc.dialog.dispose();

		System.exit(0);

	}

}
```

1.5.5 ConfigNormalizacao.Java

```
package visao.gui.telas.classes;


import java.awt.Dialog;
import java.awt.Frame;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;

import controle.eventos.EvtNormalizacao;
import controle.main.App;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;

public class ConfigNormalizacao  extends AntiAliasedThinlet {
      private Thinlet thinlet;
      private ThinletDialog dialog;

      private int indiceAtual = 0;

      private double[] arrayValoresMax =
App.getProjetoConfig().getRedeConf().getMaximos();
      private double[] arrayValoresMin =
App.getProjetoConfig().getRedeConf().getMinimos();
      private double[] arrayValoresMean =
App.getProjetoConfig().getRedeConf().getMedias();

      private boolean[] arrayAutoMax =
App.getProjetoConfig().getRedeConf().getMaximosAutomaticos();
      private boolean[] arrayAutoMin =
App.getProjetoConfig().getRedeConf().getMinimosAutomaticos();
      private boolean[] arrayAutoMean =
App.getProjetoConfig().getRedeConf().getMediasAutomaticas();

      private Object txtMax, txtMin, txtMean, cmbCaracteristica, chkMax,
chkMin, chkMean;



      public ConfigNormalizacao(Frame frame,String title) {
            ChamarDialogo(frame,"Configurar Normalização");
      }

    public void ChamarDialogo(Frame owner, String title) {
      this.dialog = new ThinletDialog(owner,title,true);
      this.init();
      this.show();
    }
```

```
    public void initParameters(Object txtMax,Object txtMin,Object
txtMean,Object cmbCaracteristica,Object chkMax,Object chkMin,Object
chkMean) {
      this.txtMax = txtMax;
      this.txtMin = txtMin;
      this.txtMean = txtMean;
      this.chkMax = chkMax;
      this.chkMean = chkMean;
      this.chkMin = chkMin;
      this.cmbCaracteristica = cmbCaracteristica;

      try {
            if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() == 0) {
                  MessageDialog msgDialog = new
MessageDialog(dialog,"Aviso","Deve ser carregado um conjunto de
treinamento \n antes de configurar os parÃ¢metros de normalizaÃ§Ã£o.");
                  msgDialog.show();
                  dialog.setVisible(false);
                  return;
            }
                  int numCar =
App.getGerenciadorPadroes().getConjuntoTreinamentoRede().get(0).getQuanta
sCaracteristicas();
                  for (int i = 0; i < numCar; i++ ) {
                        Object choice = thinlet.create("choice");
                        thinlet.setString(choice, "text",
Integer.toString(i+1));
                        thinlet.add(cmbCaracteristica, choice);
                  }
                  thinlet.setInteger(cmbCaracteristica, "selected", 0);
                  thinlet.setString(cmbCaracteristica, "text", "1");

                  loadData();
            } catch (Exception e) {
                  e.printStackTrace();
            }

    }

    private void loadData() {
            thinlet.setBoolean(chkMax, "selected",
arrayAutoMax[indiceAtual]);
            thinlet.setBoolean(chkMin, "selected",
arrayAutoMin[indiceAtual]);
            thinlet.setBoolean(chkMean, "selected",
arrayAutoMean[indiceAtual]);
            thinlet.setString(txtMax, "text",
Double.toString(arrayValoresMax[indiceAtual]));
            thinlet.setString(txtMin, "text",
Double.toString(arrayValoresMin[indiceAtual]));
            thinlet.setString(txtMean, "text",
Double.toString(arrayValoresMean[indiceAtual]));
            thinlet.setBoolean(txtMax, "enabled",
!arrayAutoMax[indiceAtual]);
            thinlet.setBoolean(txtMax, "editable",
!arrayAutoMax[indiceAtual]);
```

```
            thinlet.setBoolean(txtMin, "enabled",
!arrayAutoMin[indiceAtual]);
            thinlet.setBoolean(txtMin, "editable",
!arrayAutoMin[indiceAtual]);
            thinlet.setBoolean(txtMean, "enabled",
!arrayAutoMean[indiceAtual]);
            thinlet.setBoolean(txtMean, "editable",
!arrayAutoMean[indiceAtual]);
    }

      private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/configurarNormalizacao.xml", this);
            } catch (IOException e) {
                  new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuração de normalizacao").show();
                  e.printStackTrace();

            }
        thinlet.add(panel);
        dialog.setContent(thinlet);
        dialog.setModal(true);
      }

      public void show() {

            try {
                  if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() <= 0) {
                  }
                  else {
                        dialog.pack();
                        dialog.setSize(270,210);
                        dialog.setAlwaysOnTop(true);
                    dialog.setLocationRelativeTo(dialog.getOwner());
                    dialog.setVisible(true);
                  }

            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

    public void confirmarNormalizacao(){
      dialog.setVisible(false);
      dialog.dispose();
    }
    public void cancelar(){
      dialog.setVisible(false);
      dialog.dispose();
    }
    public void comboChanged() {
      indiceAtual = thinlet.getInteger(cmbCaracteristica, "selected");
      loadData();
```

```
}

public void maxCheck() {
  boolean b = thinlet.getBoolean(chkMax, "selected");
  if (b) {
        thinlet.setBoolean(txtMax, "enabled", false);
        thinlet.setBoolean(txtMax, "editable", false);
  }
  else {
        thinlet.setBoolean(txtMax, "enabled", true);
        thinlet.setBoolean(txtMax, "editable", true);
  }
  arrayAutoMax[indiceAtual] = b;
  App.setPrecisaNormalizar(true);
  EvtNormalizacao.arrumarParametrosNormalizacao();
  loadData();
}

public void minCheck() {
  boolean b = thinlet.getBoolean(chkMin, "selected");
  if (b) {
        thinlet.setBoolean(txtMin, "enabled", false);
        thinlet.setBoolean(txtMin, "editable", false);
  }
  else {
        thinlet.setBoolean(txtMin, "enabled", true);
        thinlet.setBoolean(txtMin, "editable", true);
  }
  arrayAutoMin[indiceAtual] = b;
  App.setPrecisaNormalizar(true);
  EvtNormalizacao.arrumarParametrosNormalizacao();
  loadData();
}

public void meanCheck() {
  boolean b = thinlet.getBoolean(chkMean, "selected");
  if (b) {
        thinlet.setBoolean(txtMean, "enabled", false);
        thinlet.setBoolean(txtMean, "editable", false);
  }
  else {
        thinlet.setBoolean(txtMean, "enabled", true);
        thinlet.setBoolean(txtMean, "editable", true);
  }
  arrayAutoMean[indiceAtual] = b;
  App.setPrecisaNormalizar(true);
  EvtNormalizacao.arrumarParametrosNormalizacao();
  loadData();
}

public void maxChange() {
  String s = thinlet.getString(txtMax, "text");

  if (s != null && !s.equalsIgnoreCase("")) {
        try {
              double d = Double.parseDouble(s);
              char ultimo = s.charAt(s.length()-1);
```

```
                    if (ultimo == 'd' || ultimo == 'f') {
                          s = s.substring(0, s.length()-1);
                          thinlet.setString(txtMax,"text", s);
                          return;
                    }
              arrayValoresMax[indiceAtual] = d;
        }
        catch (NumberFormatException e) {
              s = s.substring(0, s.length() - 1);
                    setString(txtMax, "text", s);
        }
  }
  App.setPrecisaNormalizar(true);
}

public void minChange() {
  String s = thinlet.getString(txtMin, "text");

  if (s != null && !s.equalsIgnoreCase("")) {
        try {
              double d = Double.parseDouble(s);
              char ultimo = s.charAt(s.length()-1);
                    if (ultimo == 'd' || ultimo == 'f') {
                          s = s.substring(0, s.length()-1);
                          thinlet.setString(txtMin,"text", s);
                          return;
                    }
              arrayValoresMin[indiceAtual] = d;
        }
        catch (NumberFormatException e) {
              s = s.substring(0, s.length() - 1);
                    setString(txtMin, "text", s);
        }
  }
  App.setPrecisaNormalizar(true);
}

public void meanChange() {
  String s = thinlet.getString(txtMean, "text");

  if (s != null && !s.equalsIgnoreCase("")) {
        try {
              double d = Double.parseDouble(s);
              char ultimo = s.charAt(s.length()-1);
                    if (ultimo == 'd' || ultimo == 'f') {
                          s = s.substring(0, s.length()-1);
                          thinlet.setString(txtMean,"text", s);
                          return;
                    }
              arrayValoresMean[indiceAtual] = d;
        }
        catch (NumberFormatException e) {
              s = s.substring(0, s.length() - 1);
                    setString(txtMean, "text", s);
        }
  }
  App.setPrecisaNormalizar(true);
```

```
    }

}
```

1.5.6 ConfigPesos.Java

```
package visao.gui.telas.classes;

import java.awt.Frame;
import java.io.IOException;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.GerenciadorPadroes;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class ConfigPesos extends AntiAliasedThinlet {

      private Object pnl;
      private Object[] txts;
      private Thinlet thinlet;
      private ThinletDialog dialog;

      private ArrayList<NeuronioFAN> neuronios =
App.getMonitorFAN().getNeuroniosAtuais();

      public void initParameters(Object pnl) {
            this.pnl = pnl;
      }

      public ConfigPesos(Frame ower, String title) {
            this.dialog = new ThinletDialog(ower,title,true);
            this.init();
      }

      private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel = thinlet.parse(
                              "/visao/gui/telas/ConfigPesos.xml", this);
            } catch (IOException e) {
                  e.printStackTrace();

            }
            thinlet.add(panel);
            dialog.setContent(thinlet);
            Object label,label2;
            int fim = neuronios.size();
            txts = new Object[fim];
            GerenciadorPadroes gp = App.getGerenciadorPadroes();
```

```
            for (int i = 0; i < fim; i++) {
                  label = thinlet.create("label");
                  label2 = thinlet.create("label");
                  thinlet.setString(label2, "text", "0 - 1000");
                  int classMapeada =
RecalculadorClasses.getClassesMapIndexReal().get(neuronios.get(i).getClas
seAssociada());
                  String classe = gp.getMapaClasses().get(classMapeada);
                  if (classe == null) {
                        classe = Integer.toString(classMapeada);
                  }
                  thinlet.setString(label, "text", classe);
                  thinlet.add(pnl, label);
                  txts[i] = thinlet.create("textfield");
                  thinlet.setString(txts[i], "text",
Integer.toString(neuronios.get(i).getPesoPenalizacao()));
                  thinlet.setMethod(txts[i], "action",
"validarPeso(this)",thinlet,this);
                  thinlet.add(pnl,txts[i]);
                  thinlet.add(pnl,label2);
            }
      }

      public void validarPeso(Object textField) {
            String s = getString(textField, "text");
            int pesoBase = 1000;
                  try {
                        if (s != null && !s.equals(""))
                              pesoBase = Integer.parseInt(s);
                        if (pesoBase > 1000) {
                              s = s.substring(0, s.length() - 1);
                              setString(textField, "text", s);
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                    j++;
                              } catch (NumberFormatException ex) {
                                    sb.deleteCharAt(j);
                              }
                        }

                        setString(textField, "text", sb.toString());
                  }
      }



      public void show() {
            dialog.pack();
            dialog.setModal(true);
            dialog.setLocationRelativeTo(dialog.getOwner());
```

```
            dialog.setVisible(true);
      }

      public void confirmar() {
            int fim = neuronios.size();
            for (int i = 0; i < fim; i++ ) {
                  String s = thinlet.getString(txts[i], "text");

      neuronios.get(i).setPesoPenalizacao(Integer.parseInt(s));
            }
            cancelar();
      }

      public void cancelar() {
            dialog.setVisible(false);
            dialog.dispose();
      }


}
```

1.5.7 ConfigRede.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;

import controle.main.App;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;

public class ConfigRede {

      private Thinlet thinlet;
      private ThinletDialog dialog;
      private int raioDifuso, suporteConjuntoDifuso;
      private Object txtRaioDifuso, txtSuporteConjuntoDifuso;

      public ConfigRede(Dialog owner) {
            this.dialog = new ThinletDialog(owner,false);
            this.init();
      }

      public ConfigRede(Dialog owner, String title) {
            this.dialog = new ThinletDialog(owner,title,false);
            this.init();
      }

      public ConfigRede(Frame owner) {
            this.dialog = new ThinletDialog(owner,false);
            this.init();
      }
```

```
	public ConfigRede(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/ConfiguracoesRede.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		raioDifuso =
App.getProjetoConfig().getRedeConf().getRaioDifuso();
		thinlet.setString(txtRaioDifuso, "text",
Integer.toString(raioDifuso));
		suporteConjuntoDifuso =
App.getProjetoConfig().getRedeConf().getSuporteConjuntosDifusos();
		thinlet.setString(txtSuporteConjuntoDifuso, "text",
Integer.toString(suporteConjuntoDifuso));

	  thinlet.add(panel);
	  dialog.setContent(thinlet);
	}

	public void initParameters(Object txtRaioDifuso, Object txtSuporte)
{
		this.txtRaioDifuso = txtRaioDifuso;
		this.txtSuporteConjuntoDifuso = txtSuporte;
	}

	public void show() {
		dialog.pack();
		dialog.setSize(315, 135);
		dialog.setAlwaysOnTop(true);
	  dialog.setLocationRelativeTo(dialog.getOwner());
	  dialog.setVisible(true);
    }

	public void changeRaioDifuso() {
		String s = thinlet.getString(txtRaioDifuso, "text");
		try {
			if (s != null && !s.equals(""))
				raioDifuso = Integer.parseInt(s);
		} catch (NumberFormatException e) {
			StringBuilder sb = new StringBuilder(s);
			int j = 0;
			while (j < sb.length()) {
				try {
					int z =
Integer.parseInt(Character.toString(sb
							.charAt(j)));
					j++;
				} catch (NumberFormatException ex) {
```

```
                              sb.deleteCharAt(j);
                        }
                  }
                  thinlet.setString(txtRaioDifuso, "text",
sb.toString());
            }
      }

      public void changeSuporteConjuntoDisfuso() {
            String s = thinlet.getString(txtSuporteConjuntoDifuso,
"text");
            try {
                  if (s != null && !s.equals(""))
                        suporteConjuntoDifuso = Integer.parseInt(s);
            } catch (NumberFormatException e) {
                  StringBuilder sb = new StringBuilder(s);
                  int j = 0;
                  while (j < sb.length()) {
                        try {
                              int z =
Integer.parseInt(Character.toString(sb
                                          .charAt(j)));
                              j++;
                        } catch (NumberFormatException ex) {
                              sb.deleteCharAt(j);
                        }
                  }
                  thinlet.setString(txtSuporteConjuntoDifuso, "text",
sb.toString());
            }
      }

      public void confirmar(){
            MessageDialog msg = new MessageDialog(dialog, "Atenção","Para
aplicar essa configuraÃ§Ã£o a rede será reiniciada.\n Clique em OK para
confirmar.",MessageDialog.MODE_OK_CANCEL);
            int resposta = msg.show();
            if (resposta == MessageDialog.ACTION_OK) {
                  try {

      App.getProjetoConfig().getRedeConf().setRaioDifuso(raioDifuso);

      App.getProjetoConfig().getRedeConf().setSuporteConjuntosDifusos(sup
orteConjuntoDifuso);
                        App.iniciaRede();
                        App.setPrecisaNormalizar(true);
                        if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0)

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
                  }
                  catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            dialog.setVisible(false);
```

```
	}

	public void cancelar(){
		dialog.setVisible(false);
	}

	public static void main(String[] args) {
		ConfigRede cr = new ConfigRede(new Frame(),"Configuraï¿½ï¿½es
de Rede");
		cr.show();
		while (cr.dialog.isShowing()) {}
		cr.dialog.dispose();

		System.exit(0);

	}

}
```

1.5.8 ConfigTempera.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;

import controle.eventos.EvtTreinamento;
import controle.main.App;

public class ConfigTempera extends AntiAliasedThinlet{

	private Thinlet thinlet;
	private ThinletDialog dialog;
	private EvtTreinamento evtTreinamento;
	private Object txtMaxLimit;
	private Object txtMinLimit;
	private Object txtStep;


   public void ConfigTempera(Frame ower, String title, EvtTreinamento
evtTreinamento) {
		this.dialog = new ThinletDialog(ower,title,false);
		this.init();
		this.evtTreinamento = evtTreinamento;
	}

   public void initParameters(Object txtMaxLimit,Object
txtMinLimit,Object txtStep) {
	    this.txtMaxLimit = txtMaxLimit;
	    this.txtMinLimit = txtMinLimit;
	    this.txtStep = txtStep;
```

```
   }


   private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/configurarTempera.xml", this);
            } catch (IOException e) {
                  new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuração de tempera").show();
                  e.printStackTrace();

            }
       thinlet.add(panel);
       dialog.setContent(thinlet);
       dialog.setModal(true);
      }

      public void show() {
            dialog.pack();
            dialog.setSize(200,155);
            dialog.setAlwaysOnTop(true);
        dialog.setLocationRelativeTo(dialog.getOwner());

thinlet.setString(txtMaxLimit,"text",Double.toString(App.getProjetoConfig
().getTreinoConf().getTempera().getLimiteMaximo()));

thinlet.setString(txtMinLimit,"text",Double.toString(App.getProjetoConfig
().getTreinoConf().getTempera().getLimiteMinimo()));

thinlet.setString(txtStep,"text",Double.toString(App.getProjetoConfig().g
etTreinoConf().getTempera().getStepValue()));
        dialog.setVisible(true);
      }

   public void eventTrocarStepTempera(Object txtStepTempera) {
      String s = thinlet.getString(txtStepTempera, "text");
      if (s != null && !s.equals("")) {
            double step;
            try {
                        step = Double.parseDouble(s);
                        char ultimo = s.charAt(s.length()-1);
                        if (ultimo == 'd' || ultimo == 'f') {
                              s = s.substring(0, s.length()-1);
                              thinlet.setString(txtStepTempera,"text", s);
                              return;
                        }
                        evtTreinamento.trocarStepTempera(step);
                  }
                  catch (NumberFormatException e) {
                        s = s.substring(0, s.length()-1);
                        thinlet.setString(txtStepTempera,"text", s);
                  }
      }
    }
```

```
   public void eventTrocarLimiteMaximoTempera(Object txt) {
      String s = thinlet.getString(txt, "text");
      if (s != null && !s.equals("")) {
            double d;
            try {
                        d = Double.parseDouble(s);
                        char ultimo = s.charAt(s.length()-1);
                        if (ultimo == 'd' || ultimo == 'f') {
                              s = s.substring(0, s.length()-1);
                              thinlet.setString(txt,"text", s);
                              return;
                        }
                        evtTreinamento.trocarLimiteMaximoTempera(d);
                  }
                  catch (NumberFormatException e) {
                        s = s.substring(0, s.length()-1);
                        thinlet.setString(txt,"text", s);
                  }
      }
    }

    public void eventTrocarLimiteMinimoTempera(Object txt) {
      String s = thinlet.getString(txt, "text");
      if (s != null && !s.equals("")) {
            double d;
            try {
                        d = Double.parseDouble(s);
                        char ultimo = s.charAt(s.length()-1);
                        if (ultimo == 'd' || ultimo == 'f') {
                              s = s.substring(0, s.length()-1);
                              thinlet.setString(txt,"text", s);
                              return;
                        }
                        evtTreinamento.trocarLimiteMinimoTempera(d);
                  }
                  catch (NumberFormatException e) {
                        s = s.substring(0, s.length()-1);
                        thinlet.setString(txt,"text", s);
                  }
      }
    }

    public void confirmarTempera(){
      dialog.setVisible(false);
    }
    public void cancelar(){
      dialog.setVisible(false);
    }
}
```

1.5.9 ConfigTeste.Java

```
package visao.gui.telas.classes;
import java.awt.Dialog;
import java.awt.Frame;
```

```
import java.io.IOException;

import controle.eventos.EvtTeste;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class ConfigTeste {

	private Thinlet thinlet;
	private ThinletDialog dialog;
	private Object txtNumEpocas,rdContinuamente,
rdNumEpocas,rdTempo,txtDiasTs,txtHorasTs,txtMinutosTs,txtSegundosTs;

	public ConfigTeste(Dialog owner) {
		this.dialog = new ThinletDialog(owner,false);
		this.init();
	}

	public ConfigTeste(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	public ConfigTeste(Frame owner) {
		this.dialog = new ThinletDialog(owner,false);
		this.init();
	}
	public ConfigTeste(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/ParametrosTeste.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
	  thinlet.add(panel);
	  dialog.setContent(thinlet);
	}
	public void initParameters(Object txtNumEpocas, Object
rdContinuamente, Object rdNumEpocas,Object rdTempo,Object
txtDiasTs,Object txtHorasTs,Object txtMinutosTs,Object txtSegundosTs) {
		this.txtNumEpocas = txtNumEpocas;
		this.rdContinuamente = rdContinuamente;
		this.rdNumEpocas = rdNumEpocas;
		this.rdTempo = rdTempo;
		this.txtDiasTs = txtDiasTs;
		this.txtHorasTs = txtHorasTs;
		this.txtMinutosTs = txtMinutosTs;
		this.txtSegundosTs = txtSegundosTs;
```

```
    }

    public void trocarNumEpocasTestar(){

          thinlet.setBoolean(txtNumEpocas, "enabled", true);
          thinlet.setBoolean(txtNumEpocas, "editable", true);
          thinlet.setBoolean(txtDiasTs, "enabled", false);
          thinlet.setBoolean(txtDiasTs, "editable", false);
          thinlet.setBoolean(txtHorasTs, "enabled", false);
          thinlet.setBoolean(txtHorasTs, "editable", false);
          thinlet.setBoolean(txtMinutosTs, "enabled", false);
          thinlet.setBoolean(txtMinutosTs, "editable", false);
          thinlet.setBoolean(txtSegundosTs, "enabled", false);
          thinlet.setBoolean(txtSegundosTs, "editable", false);

          thinlet.repaint();
    }

    public void testarContinuamente() {
          thinlet.setBoolean(txtNumEpocas, "enabled", false);
          thinlet.setBoolean(txtNumEpocas, "editable", false);
          thinlet.setBoolean(txtDiasTs, "enabled", false);
          thinlet.setBoolean(txtDiasTs, "editable", false);
          thinlet.setBoolean(txtHorasTs, "enabled", false);
          thinlet.setBoolean(txtHorasTs, "editable", false);
          thinlet.setBoolean(txtMinutosTs, "enabled", false);
          thinlet.setBoolean(txtMinutosTs, "editable", false);
          thinlet.setBoolean(txtSegundosTs, "enabled", false);
          thinlet.setBoolean(txtSegundosTs, "editable", false);

          thinlet.repaint();

    }

public void testarTempo(){
       thinlet.setBoolean(txtNumEpocas, "enabled", false);
          thinlet.setBoolean(txtNumEpocas, "editable", false);
          thinlet.setBoolean(txtDiasTs, "enabled", true);
          thinlet.setBoolean(txtDiasTs, "editable", true);
          thinlet.setBoolean(txtHorasTs, "enabled", true);
          thinlet.setBoolean(txtHorasTs, "editable", true);
          thinlet.setBoolean(txtMinutosTs, "enabled", true);
          thinlet.setBoolean(txtMinutosTs, "editable", true);
          thinlet.setBoolean(txtSegundosTs, "enabled", true);
          thinlet.setBoolean(txtSegundosTs, "editable", true);
          thinlet.repaint();
}
    public void show() {
          dialog.pack();
          dialog.setSize(270, 160);
          dialog.setModal(true);
          dialog.setAlwaysOnTop(true);
      dialog.setLocationRelativeTo(dialog.getOwner());
      dialog.setVisible(true);
 }
    public void confirmar(){
```

```
	}

	public void cancelar(){
		dialog.setVisible(false);
	}

	public static void main(String[] args) {
		ConfigTeste ct = new ConfigTeste(new Frame(),"Configurações
de Teste");
		ct.show();
		while (ct.dialog.isShowing()) {}
		ct.dialog.dispose();

		System.exit(0);

	}

}
```

1.5.10 ConfigTreinamento.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;

import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.eventos.EvtTreinamento;
import controle.main.App;

public class ConfigTreinamento {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	private EvtTreinamento evtTreinamento;

	private Object txtNumEpocas, rdContinuamente, rdNumEpocas, rdTempo,
			txtDiasTr, txtHorasTr, txtMinutosTr, txtSegundosTr;

	private boolean paradaPorTempo;

	private int diasParada;

	private int horasParada;

	private int segundosParada;

	private int minutosParada;

	public ConfigTreinamento(Dialog owner, EvtTreinamento evt) {
		this.dialog = new ThinletDialog(owner, false);
```

```
            evtTreinamento = evt;
            this.init();
      }

      public ConfigTreinamento(Dialog owner, String title, EvtTreinamento
evt) {
            this.dialog = new ThinletDialog(owner, title, false);
            evtTreinamento = evt;
            this.init();
      }

      public ConfigTreinamento(Frame owner, EvtTreinamento evt) {
            this.dialog = new ThinletDialog(owner, false);
            evtTreinamento = evt;
            this.init();
      }

      public ConfigTreinamento(Frame owner, String title, EvtTreinamento
evt) {
            this.dialog = new ThinletDialog(owner, title, false);
            evtTreinamento = evt;
            this.init();
      }

      private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/ParametrosTreinamento.xml",
                              this);
            } catch (IOException e) {
                  e.printStackTrace();

            }
            if
(App.getProjetoConfig().getTreinoConf().getNumeroEpocasTreinar() > 0) {
                  thinlet.setBoolean(rdNumEpocas, "selected", true);
                  thinlet.setBoolean(txtNumEpocas, "enabled", true);
                  thinlet.setBoolean(txtNumEpocas, "editable", true);
                  thinlet.setString(txtNumEpocas, "text",
Integer.toString(App
                              .getProjetoConfig().getTreinoConf()
                              .getNumeroEpocasTreinar()));
            }
            else if
(App.getProjetoConfig().getTreinoConf().getDiasParada() > 0 ||
App.getProjetoConfig().getTreinoConf().getHorasParada() > 0 ||
App.getProjetoConfig().getTreinoConf().getMinutosParada() > 0 ||
App.getProjetoConfig().getTreinoConf().getSegundosParada() > 0) {
                  thinlet.setBoolean(rdTempo, "selected", true);
                  thinlet.setBoolean(rdContinuamente, "selected", false);
                  thinlet.setBoolean(txtDiasTr, "enabled", true);
                  thinlet.setBoolean(txtDiasTr, "editable", true);
                  thinlet.setBoolean(txtHorasTr, "enabled", true);
                  thinlet.setBoolean(txtHorasTr, "editable", true);
                  thinlet.setBoolean(txtMinutosTr, "enabled", true);
```

```
                thinlet.setBoolean(txtMinutosTr, "editable", true);
                thinlet.setBoolean(txtSegundosTr, "enabled", true);
                thinlet.setBoolean(txtSegundosTr, "editable", true);

            }

            thinlet.setString(txtDiasTr, "text",
Integer.toString(App.getProjetoConfig().getTreinoConf().getDiasParada()))
;
            thinlet.setString(txtHorasTr, "text",
Integer.toString(App.getProjetoConfig().getTreinoConf().getHorasParada())
);
            thinlet.setString(txtMinutosTr, "text",
Integer.toString(App.getProjetoConfig().getTreinoConf().getMinutosParada(
)));
            thinlet.setString(txtSegundosTr, "text",
Integer.toString(App.getProjetoConfig().getTreinoConf().getSegundosParada
()));

            thinlet.add(panel);
            dialog.setContent(thinlet);
      }

      public void initParameters(Object txtNumEpocas, Object
rdContinuamente,
                  Object rdNumEpocas, Object rdTempo, Object txtDiasTr,
                  Object txtHorasTr, Object txtMinutosTr, Object
txtSegundosTr) {
            this.txtNumEpocas = txtNumEpocas;
            this.rdContinuamente = rdContinuamente;
            this.rdNumEpocas = rdNumEpocas;
            this.rdTempo = rdTempo;
            this.txtDiasTr = txtDiasTr;
            this.txtHorasTr = txtHorasTr;
            this.txtMinutosTr = txtMinutosTr;
            this.txtSegundosTr = txtSegundosTr;

      }

      public void show() {
            dialog.pack();
            dialog.setModal(true);
            dialog.setLocationRelativeTo(dialog.getOwner());
            dialog.setVisible(true);
      }

      public void trocarNumEpocasTreinar() {
            String s = thinlet.getString(txtNumEpocas, "text");
            if (!s.trim().equals("") && s != null) {
                  try {
                        int i = Integer.parseInt(s);
                        evtTreinamento.setNumeroEpocasTreinar(i);
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              if (Character.isLetter(sb.charAt(j))) {
```

```
                                    sb.deleteCharAt(j);
                                } else {
                                    j++;
                                }
                            }
                            thinlet.setString(txtNumEpocas, "text",
sb.toString());
                        }
                }
                thinlet.setBoolean(txtNumEpocas, "enabled", true);
                thinlet.setBoolean(txtNumEpocas, "editable", true);
                thinlet.setBoolean(txtDiasTr, "enabled", false);
                thinlet.setBoolean(txtDiasTr, "editable", false);
                thinlet.setBoolean(txtHorasTr, "enabled", false);
                thinlet.setBoolean(txtHorasTr, "editable", false);
                thinlet.setBoolean(txtMinutosTr, "enabled", false);
                thinlet.setBoolean(txtMinutosTr, "editable", false);
                thinlet.setBoolean(txtSegundosTr, "enabled", false);
                thinlet.setBoolean(txtSegundosTr, "editable", false);
                this.paradaPorTempo = false;
                thinlet.repaint();
        }

        public void treinarContinuamente() {
                evtTreinamento.treinarContinuamente();
                thinlet.setBoolean(txtNumEpocas, "enabled", false);
                thinlet.setBoolean(txtNumEpocas, "editable", false);
                thinlet.setBoolean(txtDiasTr, "enabled", false);
                thinlet.setBoolean(txtDiasTr, "editable", false);
                thinlet.setBoolean(txtHorasTr, "enabled", false);
                thinlet.setBoolean(txtHorasTr, "editable", false);
                thinlet.setBoolean(txtMinutosTr, "enabled", false);
                thinlet.setBoolean(txtMinutosTr, "editable", false);
                thinlet.setBoolean(txtSegundosTr, "enabled", false);
                thinlet.setBoolean(txtSegundosTr, "editable", false);
                this.paradaPorTempo = false;
                thinlet.repaint();

        }

        public void treinarTempo() {
                thinlet.setBoolean(txtNumEpocas, "enabled", false);
                thinlet.setBoolean(txtNumEpocas, "editable", false);
                thinlet.setBoolean(txtDiasTr, "enabled", true);
                thinlet.setBoolean(txtDiasTr, "editable", true);
                thinlet.setBoolean(txtHorasTr, "enabled", true);
                thinlet.setBoolean(txtHorasTr, "editable", true);
                thinlet.setBoolean(txtMinutosTr, "enabled", true);
                thinlet.setBoolean(txtMinutosTr, "editable", true);
                thinlet.setBoolean(txtSegundosTr, "enabled", true);
                thinlet.setBoolean(txtSegundosTr, "editable", true);
                this.paradaPorTempo = true;
                thinlet.repaint();
        }

        public void diasEdited() {
                String s = thinlet.getString(txtDiasTr, "text");
```

```
            if (!s.trim().equals("") && s != null) {
                  try {
                        int i = Integer.parseInt(s);
                        if (s.length() > 2) {
                              thinlet.setString(txtDiasTr, "text",
s.substring(0, s
                                          .length() - 1));
                        } else {
                              this.diasParada = i;
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              if (Character.isLetter(sb.charAt(j))) {
                                    sb.deleteCharAt(j);
                              } else {
                                    j++;
                              }
                        }
                        thinlet.setString(txtDiasTr, "text",
sb.toString());
                  }
            }
      }

      public void horasEdited() {
            String s = thinlet.getString(txtHorasTr, "text");
            if (!s.trim().equals("") && s != null) {
                  try {
                        int i = Integer.parseInt(s);
                        if (s.length() > 2) {
                              thinlet.setString(txtHorasTr, "text",
s.substring(0, s
                                          .length() - 1));
                        } else {
                              this.horasParada = i;
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              if (Character.isLetter(sb.charAt(j))) {
                                    sb.deleteCharAt(j);
                              } else {
                                    j++;
                              }
                        }
                        thinlet.setString(txtHorasTr, "text",
sb.toString());
                  }
            }
      }

      public void minutosEdited() {
            String s = thinlet.getString(txtMinutosTr, "text");
            if (!s.trim().equals("") && s != null) {
```

```
                  try {
                        int i = Integer.parseInt(s);
                        if (s.length() > 2) {
                              thinlet.setString(txtMinutosTr, "text",
s.substring(0, s
                                          .length() - 1));
                        } else {
                              this.minutosParada = i;
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              if (Character.isLetter(sb.charAt(j))) {
                                    sb.deleteCharAt(j);
                              } else {
                                    j++;
                              }
                        }
                        thinlet.setString(txtMinutosTr, "text",
sb.toString());
                  }
            }
      }

      public void segundosEdited() {
            String s = thinlet.getString(txtSegundosTr, "text");
            if (!s.trim().equals("") && s != null) {
                  try {
                        int i = Integer.parseInt(s);
                        if (s.length() > 2) {
                              thinlet.setString(txtSegundosTr, "text",
s.substring(0, s
                                          .length() - 1));
                        } else {
                              this.segundosParada = i;
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              if (Character.isLetter(sb.charAt(j))) {
                                    sb.deleteCharAt(j);
                              } else {
                                    j++;
                              }
                        }
                        thinlet.setString(txtSegundosTr, "text",
sb.toString());
                  }
            }
      }

      public void confirmar() {

            if (paradaPorTempo) {
```

```
                  evtTreinamento.ativarParadaPorTempo(diasParada,
horasParada, minutosParada, segundosParada);
            }
            else {
                  evtTreinamento.desativarParadaPorTempo();
            }

            dialog.setVisible(false);
      }

}
```

1.5.11 ConfigValidacao.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;

import jfan.fan.StoredNetworkType;

import controle.main.App;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class ConfigValidacao {

      private Thinlet thinlet;

      private ThinletDialog dialog;

      private Object
txtNumEpocasVl,rdNuncaVl,rdNumEpocasVl,lbEpocas,rdValidaAtual,rdValidaMed
Harm,rdValidaMedArit,rdValidaMaxMin;

      private StoredNetworkType tipoRede =
StoredNetworkType.BestHarmonicMean;

      int epocas = 0;

      public void initParameters(Object txtNumEpocasVl,Object
rdNuncaVl,Object rdNumEpocasVl,Object lbEpocas,Object rdValidaAtual,
Object rdValidaMedHarm,Object rdValidaMedArit, Object rdValidaMaxMin) {
          this.txtNumEpocasVl = txtNumEpocasVl;
          this.rdNuncaVl = rdNuncaVl;
          this.rdNumEpocasVl = rdNumEpocasVl;
          this.lbEpocas = lbEpocas;
          this.rdValidaAtual = rdValidaAtual;
          this.rdValidaMedHarm = rdValidaMedHarm;
          this.rdValidaMedArit = rdValidaMedArit;
          this.rdValidaMaxMin = rdValidaMaxMin;

      }
```

```
	public void selecionarTipoRedeAtual() {
		tipoRede = StoredNetworkType.Actual;
	}

	public void selecionarTipoRedeHarmonica() {
		tipoRede = StoredNetworkType.BestHarmonicMean;
	}

	public void selecionarTipoRedeAritmetica() {
		tipoRede = StoredNetworkType.BestAritmethicMean;
	}

	public void selecionarTipoRedeMaxMin() {
		tipoRede = StoredNetworkType.BestMaxMin;
	}

	public void nuncaValidar() {
		epocas = -1;
		thinlet.setBoolean(txtNumEpocasVl, "enabled", false);
		thinlet.setBoolean(txtNumEpocasVl, "editable", false);
		thinlet.repaint();

	}

	public void trocarNumEpocasValidar(){

		thinlet.setBoolean(txtNumEpocasVl, "enabled", true);
		thinlet.setBoolean(txtNumEpocasVl, "editable", true);
		thinlet.setBoolean(lbEpocas, "enabled", true);
		thinlet.repaint();

	}

	public void trocarTxtNumEpocasValidar() {
		String s = thinlet.getString(txtNumEpocasVl, "text");
		epocas = 0;
		try {
			if (s != null && !s.equals(""))
				epocas = Integer.parseInt(s);

		} catch (NumberFormatException e) {
			StringBuilder sb = new StringBuilder(s);
			int j = 0;
			while (j < sb.length()) {
				try {
					int z =
Integer.parseInt(Character.toString(sb
							.charAt(j)));
					j++;
				} catch (NumberFormatException ex) {
					sb.deleteCharAt(j);
				}
			}
			thinlet.setString(txtNumEpocasVl, "text",
sb.toString());
		}
	}
```

```
	public ConfigValidacao(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ConfigValidacao(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ConfigValidacao(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ConfigValidacao(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/ParametrosValidacao.xml",
					this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);

		int numEpocas =
App.getProjetoConfig().getValidacaoConfig().getCadaEpocaValidar();

		if (numEpocas > 0) {
			trocarNumEpocasValidar();
			thinlet.setString(txtNumEpocasVl, "text",
Integer.toString(numEpocas));
			thinlet.setBoolean(rdNumEpocasVl, "selected", true);
			thinlet.setBoolean(rdNuncaVl, "selected", false);
		}

		tipoRede =
App.getProjetoConfig().getValidacaoConfig().getTipoRedeValidacao();
		switch (tipoRede) {
		case Actual:
			thinlet.setBoolean(rdValidaAtual, "selected", true);
			thinlet.setBoolean(rdValidaMaxMin, "selected", false);
			thinlet.setBoolean(rdValidaMedArit, "selected", false);
			thinlet.setBoolean(rdValidaMedHarm, "selected", false);
			break;
		case BestAritmethicMean:
			thinlet.setBoolean(rdValidaAtual, "selected", false);
```

```
                thinlet.setBoolean(rdValidaMaxMin, "selected", false);
                thinlet.setBoolean(rdValidaMedArit, "selected", true);
                thinlet.setBoolean(rdValidaMedHarm, "selected", false);
                break;
          case BestHarmonicMean:
                thinlet.setBoolean(rdValidaAtual, "selected", false);
                thinlet.setBoolean(rdValidaMaxMin, "selected", false);
                thinlet.setBoolean(rdValidaMedArit, "selected", false);
                thinlet.setBoolean(rdValidaMedHarm, "selected", true);
                break;
          case BestMaxMin:
                thinlet.setBoolean(rdValidaAtual, "selected", false);
                thinlet.setBoolean(rdValidaMaxMin, "selected", true);
                thinlet.setBoolean(rdValidaMedArit, "selected", false);
                thinlet.setBoolean(rdValidaMedHarm, "selected", false);
                break;
          }

    }

    public void show() {
          dialog.pack();
          dialog.setAlwaysOnTop(true);
          dialog.setLocationRelativeTo(dialog.getOwner());
          dialog.setVisible(true);
          dialog.setModal(true);
    }

    public void confirmar() {
          if (epocas <= 0) {

    App.getTelaEasyFAN().getEvtValidacao().desativarValidarCadaEpocas()
;
          }
          else {

    App.getTelaEasyFAN().getEvtValidacao().validarCadaEpocas(epocas);
          }

    App.getTelaEasyFAN().getEvtValidacao().setTipoRedeValidacao(tipoRed
e);
          dialog.setVisible(false);
    }

    public void cancelar() {
          dialog.setVisible(false);
    }

    public static void main(String[] args) {
          ConfigValidacao cv = new ConfigValidacao(new Frame(),
                      "Configuracoes de Validacao");
          cv.show();
          while (cv.dialog.isShowing()) {
          }
          cv.dialog.dispose();

          System.exit(0);
```

```
	}

}
```

1.5.12 ConjuntoNulo.Java

```
package visao.gui.telas.classes;

	import java.awt.Dialog;
	import java.awt.Frame;
	import java.io.IOException;
	import thinlet.Thinlet;
	import thinletcommons.AntiAliasedThinlet;
	import thinletcommons.ThinletDialog;

	public class ConjuntoNulo {

		private Thinlet thinlet;
		private ThinletDialog dialog;

		public ConjuntoNulo(Dialog owner) {
			this.dialog = new ThinletDialog(owner,false);
			this.init();
		}

		public ConjuntoNulo(Dialog owner, String title) {
			this.dialog = new ThinletDialog(owner,title,false);
			this.init();
		}

		public ConjuntoNulo(Frame owner) {
			this.dialog = new ThinletDialog(owner,false);
			this.init();
		}
		public ConjuntoNulo(Frame owner, String title) {
			this.dialog = new ThinletDialog(owner,title,false);
			this.init();
		}

		private void init() {
			this.thinlet = new AntiAliasedThinlet();
			Object panel = null;
			try {
				panel =
thinlet.parse("/visao/gui/telas/ConjuntoNulo.xml", this);
			} catch (IOException e) {
				e.printStackTrace();

			}
		  thinlet.add(panel);
		  dialog.setContent(thinlet);
		}

		public void show() {
			dialog.pack();
```

```
                dialog.setSize(380,100);
                dialog.setAlwaysOnTop(true);
              dialog.setLocationRelativeTo(dialog.getOwner());
              dialog.setVisible(true);
          }

            public void cancelar(){
                dialog.setVisible(false);
            }

            public static void main(String[] args) {
                ConjuntoNulo cn = new ConjuntoNulo(new
Frame(),"Conjunto Vazio");
                cn.show();
                while (cn.dialog.isShowing()) {}
                cn.dialog.dispose();

                System.exit(0);

            }

      }
```

1.5.13 EasyFan.Java

```
package visao.gui.telas.classes;

import java.awt.Frame;
import java.awt.Image;
import java.awt.Toolkit;
import java.io.File;
import java.io.IOException;

import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.temperas.TemperaAleatoria;
import jfan.fan.temperas.TemperaSimuladaFAN;
import thinlet.FrameLauncher;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.FileChooser;
import thinletcommons.MessageDialog;
import visao.gui.telas.graficos.classes.GraphicChooser;
import visao.gui.telas.utils.CarregarTabela;
import visao.gui.telas.utils.DataView;
import visao.gui.wizard.Wizard;
import controle.eventos.EvtAbrirArquivo;
import controle.eventos.EvtClassificacao;
import controle.eventos.EvtLimparConjunto;
import controle.eventos.EvtNormalizacao;
import controle.eventos.EvtSalvarArquivo;
import controle.eventos.EvtTeste;
import controle.eventos.EvtTreinamento;
import controle.eventos.EvtValidacao;
import controle.eventos.auxiliares.ContainerMatrizConfusao;
import controle.main.App;
```

```
public class EasyFan extends AntiAliasedThinlet {

      private static final long serialVersionUID = 1L;

      private Frame frame;

      private CarregarTabela carregarTabelaTeste = null;

      private CarregarTabela carregarTabelaValidacao = null;

      private CarregarTabela carregarTabelaClassificacao = null;

      private final EvtTreinamento evtTreinamento;

      private final EvtTeste evtTeste;

      private final EvtValidacao evtValidacao;

      private final EvtClassificacao evtClassificacao;

      private boolean pararTreinamento = false;

      public String[][] array = null;

      private EvtAbrirArquivo evtAbrirArquivo = new EvtAbrirArquivo();

      private Object screen;

      private viewHelp vHelp;

      private Wizard abrirWizard = new Wizard();

      private ConfigTempera confTempera = new ConfigTempera();

      private salvarRede salvarRede = new salvarRede();

      private CarregarRede carregarRede = new CarregarRede();

      // CheckBox
      private Object chkEmbaralharPadroes, chkPorcentagemPadroes,
                  chkNormalizarCadaEpoca, chkChanceNormalizar,
chkRandomizarPesos,
                  chkVisualizarTs,chkVisualizarVl,
                  chkVisualizarCl;

      // TextField
      private Object txtPorcentagemPadroes, txtEmbaralharPadroes,
                  txtNormalizarCadaEpoca, txtChanceNormalizar,
txtRandomizarPesos, txtStep, txtReiniciarTempera;

      // Label
      private Object lblEmbaralharPadroes1, lblPorcentagemPadroes1,
                  lblNormalizarCadaEpoca1, lblChanceNormalizar1,
lblRandomizarPesos,
                  lblNormalizacaoUtilizada, lblHarmonicaTeste,
lblMaxMinTeste,
```

```
                  lblAritmeticaTeste, lblAcertosTs, lblErrosTs,
lblHarmonicaVl,lblMaxMinVl,

      lblAritmeticaVl,lblAcertosVl,lblErrosVl,lblEpocaValidada,lblTipoRed
eValidacao;

 //Botao
      private Object btConfigRede, btAbrirProjeto, btWizard, btAbaTreino,
                  btAbaTeste, btAbaValida, btAbaClassifica,
                  btnConfigNormalizacaoUtilizada, btSalvarTr,
btConfigurarTr,
                  btVisualizarDadosTr, btTreinar, btDescarregarTr,
btAbrirTr,
                  btMatrizConfusaoTr, btTempera, btConfigNormalizacao,
btAbrirTs,
                  btDescarregarTs, btSalvarTs, btConfigurarTs,
btTestarRede,
                  btMatrizConfusaoTs, btGrafico, btAbrirVl,
btDescarregarVl,
                  btSalvarVl, btConfigurarVl, btResultadosVl, btValidar,
btAbrirCl,
                  btDescarregarCl, btSalvarCl, btCarregarRedeCl,
btResultadosCl,
                  btClassificar, antPagTs, proxPagTs, antPagVl,
proxPagVl, antPagCl,
                  proxPagCl,btValidarRede,btMatrizConfusaoVl,
btCarregarRedeTr, btSalvarRedeTr ;

      // ComboBox
      private Object cmbNormalizacaoUtilizada, cmbNormaUtil;

      // Tabelas
      private Object tabelaTr, tabelaTs, tabelaVl, tabelaCl;

      public EasyFan() throws IOException {
            screen = parse("/visao/gui/telas/EasyFan.xml");
            add(screen);
            // Prepara evtTreinamento
            evtTreinamento = new EvtTreinamento(this);
            evtTeste = new EvtTeste(this, this.find("lblHarmonicaTeste"),
this
                        .find("lblMaxMinTeste"),
this.find("lblAritmeticaTeste"));
            evtValidacao = new EvtValidacao(this);
            evtClassificacao = new EvtClassificacao(this);
            init();
      }

      public Frame getFrame() {
            return frame;
      }

      public Object getSreen() {
            return screen;
      }

      public void initParameters(Object chkEmbaralharPadroes,
```

```
                  Object txtEmbaralharPadroes, Object
lblEmbaralharPadroes1,
                  Object chkPorcentagemPadroes, Object
txtPorcentagemPadroes,
                  Object lblPorcentagemPadroes1, Object
chkNormalizarCadaEpoca,
                  Object txtNormalizarCadaEpoca, Object
lblNormalizarCadaEpoca1,
                  Object chkChanceNormalizar, Object txtChanceNormalizar,
                  Object lblChanceNormalizar1, Object chkRandomizarPesos,
                  Object txtRandomizarPesos, Object lblRandomizarPesos1,
                  Object lblNormalizacaoUtilizada, Object
cmbNormalizacaoUtilizada,
                  Object btnConfigNormalizacaoUtilizada, Object
btSalvarTr,
                  Object btConfigurarTr, Object btVisualizarDadosTr,
                  Object btTreinar, Object btDescarregarTr, Object
btCarregarRedeTr,
                  Object btMatrizConfusaoTr, Object cmbNormaUtil,
                  Object btConfigNormalizacao, Object btSalvarRedeTr,
Object tabelaTs,
                  Object chkVisualizarTs, Object btDescarregarTs, Object
btSalvarTs,
                  Object btConfigurarTs, Object btGrafico, Object
tabelaTr,
                  Object btConfigRede, Object btAbrirProjeto, Object
btWizard,
                  Object btAbaTreino, Object btAbaTeste, Object
btAbaValida,
                  Object btAbaClassifica, Object btAbrirTr, Object
btAbrirTs,
                  Object btTestarRede, Object btMatrizConfusaoTs,
                  Object lblHarmonicaTeste, Object lblMaxMinTeste,
                  Object lblAritmeticaTeste, Object lblAcertosTs, Object
lblErrosTs,
                  Object btAbrirVl, Object btDescarregarVl, Object
btSalvarVl,
                  Object btConfigurarVl, Object btResultadosVl,
                  Object chkVisualizarVl, Object btValidar, Object
tabelaVl,
                  Object btAbrirCl, Object btDescarregarCl, Object
btSalvarCl,
                  Object btCarregarRedeCl, Object btResultadosCl,
                  Object chkVisualizarCl, Object btClassificar, Object
tabelaCl,
                  Object antPagTs, Object proxPagTs, Object antPagVl,
                  Object proxPagVl, Object antPagCl, Object proxPagCl,
                  Object lblHarmonicaVl, Object lblMaxMinVl, Object
lblAritmeticaVl,
                  Object lblAcertosVl, Object lblErrosVl, Object
btValidarRede, Object btMatrizConfusaoVl,
                  Object txtStep, Object txtReiniciarTempera, Object
lblEpocaValidada, Object lblTipoRedeValidacao) {

            this.chkEmbaralharPadroes = chkEmbaralharPadroes;
            this.txtEmbaralharPadroes = txtEmbaralharPadroes;
            this.lblEmbaralharPadroes1 = lblEmbaralharPadroes1;
```

```
            this.chkPorcentagemPadroes = chkPorcentagemPadroes;
            this.txtPorcentagemPadroes = txtPorcentagemPadroes;
            this.lblPorcentagemPadroes1 = lblPorcentagemPadroes1;

            this.chkNormalizarCadaEpoca = chkNormalizarCadaEpoca;
            this.txtNormalizarCadaEpoca = txtNormalizarCadaEpoca;
            this.lblNormalizarCadaEpoca1 = lblNormalizarCadaEpoca1;

            this.chkChanceNormalizar = chkChanceNormalizar;
            this.txtChanceNormalizar = txtChanceNormalizar;
            this.lblChanceNormalizar1 = lblChanceNormalizar1;

            this.chkRandomizarPesos = chkRandomizarPesos;
            this.txtRandomizarPesos = txtRandomizarPesos;
            this.lblRandomizarPesos = lblRandomizarPesos1;

            this.lblNormalizacaoUtilizada = lblNormalizacaoUtilizada;
            this.cmbNormalizacaoUtilizada = cmbNormalizacaoUtilizada;
            this.btnConfigNormalizacaoUtilizada =
btnConfigNormalizacaoUtilizada;

            this.btConfigRede = btConfigRede;
            this.btAbrirProjeto = btAbrirProjeto;
            this.btWizard = btWizard;
            this.btAbaTreino = btAbaTreino;
            this.btAbaTeste = btAbaTeste;
            this.btAbaValida = btAbaValida;
            this.btAbaClassifica = btAbaClassifica;
            this.btAbrirTr = btAbrirTr;
            this.btDescarregarTr = btDescarregarTr;
            this.btSalvarTr = btSalvarTr;
            this.btConfigurarTr = btConfigurarTr;
            this.btVisualizarDadosTr = btVisualizarDadosTr;
            this.btGrafico = btGrafico;

            this.btTreinar = btTreinar;
            this.btCarregarRedeTr = btCarregarRedeTr;

            this.btMatrizConfusaoTr = btMatrizConfusaoTr;
            this.cmbNormaUtil = cmbNormaUtil;

            this.btConfigNormalizacao = btConfigNormalizacao;
            this.btSalvarRedeTr = btSalvarRedeTr;

            this.tabelaTs = tabelaTs;
            this.btAbrirTs = btAbrirTs;
            this.btDescarregarTs = btDescarregarTs;
            this.btSalvarTs = btSalvarTs;
            this.btConfigurarTs = btConfigurarTs;
            this.chkVisualizarTs = chkVisualizarTs;
            this.btTestarRede = btTestarRede;
            this.btMatrizConfusaoTs = btMatrizConfusaoTs;
            this.lblHarmonicaTeste = lblHarmonicaTeste;
            this.lblMaxMinTeste = lblMaxMinTeste;
            this.lblAritmeticaTeste = lblAritmeticaTeste;
            this.lblAcertosTs = lblAcertosTs;
```

```
            this.lblErrosTs = lblErrosTs;
            this.antPagTs = antPagTs;
            this.proxPagTs = proxPagTs;

            this.btAbrirVl = btAbrirVl;
            this.btDescarregarVl = btDescarregarVl;
            this.btSalvarVl = btSalvarVl;
            this.btConfigurarVl = btConfigurarVl;
            this.btResultadosVl = btResultadosVl;
            this.chkVisualizarVl = chkVisualizarVl;
            this.btValidar = btValidar;
            this.tabelaVl = tabelaVl;
            this.antPagVl = antPagVl;
            this.proxPagVl = proxPagVl;
            this.btValidarRede = btValidarRede;
            this.btMatrizConfusaoVl = btMatrizConfusaoVl;
            this.lblHarmonicaVl = lblHarmonicaVl;
            this.lblMaxMinVl = lblMaxMinVl;
            this.lblAritmeticaVl = lblAritmeticaVl;
            this.lblAcertosVl = lblAcertosVl;
            this.lblErrosVl = lblErrosVl;
            this.lblEpocaValidada = lblEpocaValidada;
            this.lblTipoRedeValidacao = lblTipoRedeValidacao;

            this.btAbrirCl = btAbrirCl;
            this.btDescarregarCl = btDescarregarCl;
            this.btSalvarCl = btSalvarCl;
            this.btCarregarRedeCl = btCarregarRedeCl;
            this.btResultadosCl = btResultadosCl;
            this.chkVisualizarCl = chkVisualizarCl;
            this.btClassificar = btClassificar;
            this.tabelaCl = tabelaCl;
            this.antPagCl = antPagCl;
            this.proxPagCl = proxPagCl;

            this.txtStep = txtStep;
            this.txtReiniciarTempera = txtReiniciarTempera;
      }

      private void init() {
            initAbaTreinamento();
      }

      private void initAbaTreinamento() {
            setBoolean(chkEmbaralharPadroes, "selected",
App.getProjetoConfig()
                        .getTreinoConf().getCadaEpocaEmbaralharPadroes() >
0);
            setBoolean(chkChanceNormalizar, "selected",
App.getProjetoConfig()
                        .getTreinoConf().getChanceRenormalizar() > 0);
            setBoolean(chkNormalizarCadaEpoca, "selected",
App.getProjetoConfig()
                        .getTreinoConf().getCadaEpocaRenormalizar() > 0);
            setBoolean(chkRandomizarPesos, "selected",
App.getProjetoConfig()
                        .getTreinoConf().isRandomizarPesos());
```

```
            setBoolean(chkPorcentagemPadroes, "selected",
App.getProjetoConfig()
                        .getTreinoConf().getNumeroPadroesTreino() > 0);

            if
(App.getProjetoConfig().getTreinoConf().getChanceRenormalizar() > 0) {
                  setString(txtChanceNormalizar, "text", Integer

      .toString(App.getProjetoConfig().getTreinoConf()
                                          .getChanceRenormalizar()));
                  setBoolean(txtChanceNormalizar, "enabled", true);
                  setBoolean(txtChanceNormalizar, "editable", true);
                  setBoolean(lblChanceNormalizar1, "enabled", true);
                  setBoolean(lblChanceNormalizar1, "editable", true);
            }

            if (App.getProjetoConfig().getTreinoConf()
                        .getCadaEpocaEmbaralharPadroes() > 0) {
                  setString(txtEmbaralharPadroes, "text",
Integer.toString(App
                              .getProjetoConfig().getTreinoConf()
                              .getCadaEpocaEmbaralharPadroes()));
                  setBoolean(txtEmbaralharPadroes, "enabled", true);
                  setBoolean(txtEmbaralharPadroes, "editable", true);
                  setBoolean(lblEmbaralharPadroes1, "enabled", true);
            }

            if
(App.getProjetoConfig().getTreinoConf().getCadaEpocaRenormalizar() > 0) {
                  setString(txtNormalizarCadaEpoca, "text",
Integer.toString(App
                              .getProjetoConfig().getTreinoConf()
                              .getCadaEpocaRenormalizar()));
                  setBoolean(txtNormalizarCadaEpoca, "enabled", true);
                  setBoolean(txtNormalizarCadaEpoca, "editable", true);
                  setBoolean(lblNormalizarCadaEpoca1, "enabled", true);
            }

            if
(App.getProjetoConfig().getTreinoConf().getNumeroPadroesTreino() > 0) {
                  setString(txtPorcentagemPadroes, "text",
Integer.toString(App
                              .getProjetoConfig().getTreinoConf()
                              .getPorcentagemPadroesTreino()));
                  setBoolean(txtPorcentagemPadroes, "enabled", true);
                  setBoolean(txtPorcentagemPadroes, "editable", true);
                  setBoolean(lblPorcentagemPadroes1, "enabled", true);
            }

            if
(App.getProjetoConfig().getTreinoConf().isRandomizarPesos()) {
                  setString(txtRandomizarPesos, "text",
Integer.toString(App

      .getProjetoConfig().getTreinoConf().getPesoBase()));
                  setBoolean(txtRandomizarPesos, "enabled", true);
                  setBoolean(txtRandomizarPesos, "editable", true);
```

```
                  setBoolean(lblRandomizarPesos, "enabled", true);
            }

            NormalizatorTypes norm = App.getProjetoConfig().getRedeConf()
                        .getTipoNormalizador();
            int index = 0;
            if (norm == NormalizatorTypes.NormalizatorMax) {
                  index = 1;
            } else if (norm == NormalizatorTypes.NormalizatorMaxMean) {
                  index = 2;
            }
            Object choice = getItem(cmbNormalizacaoUtilizada, index);
            setInteger(cmbNormalizacaoUtilizada, "selected", index);
            setString(cmbNormalizacaoUtilizada, "text", getString(choice,
"text"));


      setString(txtStep,"text",Integer.toString(App.getProjetoConfig().ge
tTreinoConf().getCadaEpocaStepTempera()));

            setString(txtReiniciarTempera, "text",
Integer.toString(App.getProjetoConfig().getTreinoConf().getCadaEpocaReset
Tempera()));
      }
      public void abrirProjeto() {
            try {
                  FileChooser f = new FileChooser(new Frame(), "Abrir",
                              FileChooser.MODE_OPEN);
                  f.show();
                  this.repaint();
                  File file;
                  file = f.getSelectedFile();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public void mudaAba(Object tabbedpane, Object button) {
            String s = this.getProperty(button, "indice").toString();
            this.setInteger(tabbedpane, "selected", Integer.parseInt(s));
      }


      public void abrirTreinamento() {
            array = evtAbrirArquivo.abrirTreinamento();
            if (array != null) {
                  setBoolean(btSalvarTr, "enabled", true);
                  setBoolean(btConfigurarTr, "enabled", true);
                  setBoolean(btVisualizarDadosTr, "enabled", true);
                  setBoolean(btTreinar, "enabled", true);
                  setBoolean(btDescarregarTr, "enabled", true);
            }
      }

      public void descarregarArquivo() {
            array = null;
            setBoolean(btDescarregarTr, "enabled", false);
```

```
            setBoolean(btSalvarTr, "enabled", false);
            setBoolean(btConfigurarTr, "enabled", false);
            setBoolean(btVisualizarDadosTr, "enabled", false);
            setBoolean(btMatrizConfusaoTr, "enabled", false);
            setBoolean(btTreinar, "enabled", false);
            setBoolean(btGrafico, "enabled", false);
            setBoolean(btSalvarRedeTr, "enabled", false);
            setBoolean(btCarregarRedeTr, "enabled", false);
            EvtLimparConjunto.limparConjuntoTreinamento();
      }

      public void limparDados(Object button) {
            array = null;
            String opcao = this.getProperty(button, "indice").toString();
            if ((opcao.equals("btDescarregarTs")) && (carregarTabelaTeste
!= null)) {
                  carregarTabelaTeste.clearTable();
                  EvtLimparConjunto.limparConjuntoTeste();
                  setBoolean(chkVisualizarTs, "enabled", false);
                  setBoolean(chkVisualizarTs, "selected", false);
                  setBoolean(btDescarregarTs, "enabled", false);
                  setBoolean(btSalvarTs, "enabled", false);
                  setBoolean(btConfigurarTs, "enabled", false);
                  this.setString(this.find("pagAtualTs"), "text", "");
                  this.setString(this.find("pagMaxTs"), "text", "");
                  setBoolean(antPagTs, "enabled", false);
                  setBoolean(proxPagTs, "enabled", false);
                  setBoolean(btTestarRede, "enabled", false);
                  setBoolean(btMatrizConfusaoTs, "enabled", false);

            } else if ((opcao.equals("btDescarregarVl"))
                        && (carregarTabelaValidacao != null)) {
                  carregarTabelaValidacao.clearTable();
                  EvtLimparConjunto.limparConjuntoValidacao();
                  setBoolean(chkVisualizarVl, "enabled", false);
                  setBoolean(chkVisualizarVl, "selected", false);
                  setBoolean(btDescarregarVl, "enabled", false);
                  setBoolean(btSalvarVl, "enabled", false);
                  setBoolean(btConfigurarVl, "enabled", false);
                  this.setString(this.find("pagAtualVl"), "text", "");
                  this.setString(this.find("pagMaxVl"), "text", "");
                  setBoolean(antPagVl, "enabled", false);
                  setBoolean(proxPagVl, "enabled", false);

            } else if ((opcao.equals("btDescarregarCl"))
                        && (carregarTabelaClassificacao != null)) {
                  carregarTabelaClassificacao.clearTable();
                  EvtLimparConjunto.limparConjuntoClassificacao();
                  setBoolean(chkVisualizarCl, "enabled", false);
                  setBoolean(chkVisualizarCl, "selected", false);
                  setBoolean(btDescarregarCl, "enabled", false);
                  setBoolean(btSalvarCl, "enabled", false);
                  setBoolean(btCarregarRedeCl, "enabled", false);
                  setBoolean(btResultadosCl, "enabled", false);
                  setBoolean(btClassificar, "enabled", false);
                  this.setString(this.find("pagAtualCl"), "text", "");
                  this.setString(this.find("pagMaxCl"), "text", "");
```

```
                  setBoolean(antPagCl, "enabled", false);
                  setBoolean(proxPagCl, "enabled", false);

            }

            this.repaint();
      }

      public void abrirTeste() {
            array = evtAbrirArquivo.abrirTeste();
            if (array != null) {
                  try {
                        if (carregarTabelaTeste == null)
                              carregarTabelaTeste = new
CarregarTabela(this, tabelaTs,
                                          this.find("pagAtualTs"),
this.find("pagMaxTs"));
                        carregarTabelaTeste.carregaDados(array);
                        this.repaint();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
                  setBoolean(chkVisualizarTs, "enabled", true);
                  setBoolean(chkVisualizarTs, "selected", true);
                  setBoolean(btDescarregarTs, "enabled", true);
                  setBoolean(btSalvarTs, "enabled", true);
                  setBoolean(btConfigurarTs, "enabled", true);
                  setBoolean(antPagTs, "enabled", true);
                  setBoolean(proxPagTs, "enabled", true);
                  setBoolean(btTestarRede, "enabled", true);
                  setBoolean(btMatrizConfusaoTs, "enabled", true);
                  this.repaint();
            }
      }

      public void abrirValidacao(Object table) {
            array = evtAbrirArquivo.abrirValidacao();
            if (array != null) {
                  try {
                        if (carregarTabelaValidacao == null)
                              carregarTabelaValidacao = new
CarregarTabela(this, table,
                                          this.find("pagAtualVl"),
this.find("pagMaxVl"));
                        carregarTabelaValidacao.carregaDados(array);
                        this.repaint();

                  } catch (Exception e) {
                        e.printStackTrace();
                  }
                  setBoolean(chkVisualizarVl, "enabled", true);
                  setBoolean(chkVisualizarVl, "selected", true);
                  setBoolean(btDescarregarVl, "enabled", true);
                  setBoolean(btSalvarVl, "enabled", true);
                  setBoolean(btConfigurarVl, "enabled", true);
                  setBoolean(antPagVl, "enabled", true);
                  setBoolean(proxPagVl, "enabled", true);
```

```
                this.repaint();

            }
        }

        public void abrirClassificacao(Object table) {
            array = evtAbrirArquivo.abrirClassificacao();
            if (array != null) {
                try {
                    if (carregarTabelaClassificacao == null)
                        carregarTabelaClassificacao = new
CarregarTabela(this,
                                table, this.find("pagAtualCl"),
this
                                        .find("pagMaxCl"));
                    carregarTabelaClassificacao.carregaDados(array);
                    this.repaint();
                } catch (Exception e) {
                    e.printStackTrace();
                }
                setBoolean(chkVisualizarCl, "enabled", true);
                setBoolean(chkVisualizarCl, "selected", true);
                setBoolean(btDescarregarCl, "enabled", true);
                setBoolean(btSalvarCl, "enabled", true);
                setBoolean(btResultadosCl, "enabled", true);
                setBoolean(btClassificar, "enabled", true);
                setBoolean(btCarregarRedeCl, "enabled", true);
                setBoolean(antPagCl, "enabled", true);
                setBoolean(proxPagCl, "enabled", true);

                this.repaint();
            }
        }


        public void confirmar() {

        }

        public void cancelar(Object dialog) throws IOException {
            remove(dialog);
        }

        public void sairSistema() {
            SaidaSistema saidaSistema = new SaidaSistema(new Frame(),
                    "Fechar o Sistema");
            saidaSistema.show();
        }

        public void configurarRede() {
            ConfigRede configRede = new ConfigRede(frame,
                    "Configurações de Rede");
            configRede.show();
        }

        public void selecionarCaracteristica() {
```

```
            CaracteristicasUtilizadas caracteristicasUtilizadas = new
CaracteristicasUtilizadas(
                        new Frame(), "Seleçao de Caracteristicas");
            caracteristicasUtilizadas.show();
      }

      public void reiniciarConfiguracao() {
            ReiniciarConfig reiniciarConfig = new ReiniciarConfig(new
Frame(),
                        "Reiniciar Configuraçoes");
            reiniciarConfig.show();
      }

      public void configurar(Object Button) {
            String Aba = this.getProperty(Button, "indice").toString();
            if (Aba.equals("configurarTr")) {
                  ConfigTreinamento configTreinamento = new
ConfigTreinamento(frame,
                              "Configuraçoes de Treinamento",
evtTreinamento);
                  configTreinamento.show();

            }
            if (Aba.equals("configurarTs")) {
                  ConfigTeste configTeste = new ConfigTeste(frame,
                              "Configuraçoes de Teste");
                  configTeste.show();

            }
            if (Aba.equals("configurarVl")) {
                  ConfigValidacao configValidacao = new
ConfigValidacao(frame,
                              "Configuraçoes de Validaçao");
                  configValidacao.show();

            }
      }

      public void exibirResultados() {
                  ResultadosClassificacao resultadosClassificacao = new
ResultadosClassificacao(
                              new Frame(), "Resultados - Classificacao");
                  resultadosClassificacao.show();
            }

      public void chamarMatriz() {

            if (ContainerMatrizConfusao.getMatriz() != null) {
                  int epoca = ContainerMatrizConfusao.getEpoca();
                  int[][] matriz = ContainerMatrizConfusao.getMatriz();
                  float harmonica =
ContainerMatrizConfusao.getHarmonica();
                  float aritmetica =
ContainerMatrizConfusao.getAritmetica();
                  float maxmin = ContainerMatrizConfusao.getMaxMin();
                  MatrizConfusaoTela matrizConfusao = new
MatrizConfusaoTela(frame,
```

```
                                    "Matriz de Confusao", epoca,
                                    matriz,
                                    harmonica,
                                    aritmetica,
                                    maxmin, null);
                        matrizConfusao.show();
                  } else {

                        MessageDialog msgdlg = new MessageDialog(frame,
"Atençao",
                                    "Para exibir a matriz de confusaodeve-se
treinar a rede.");
                        msgdlg.show();

                  }

            }

            public void abrirMatrixConfusaoTeste() {
                  evtTeste.chamarTelaMatrizConfusao();
            }

            public void abrirMatrixConfusaoValidacao() {
                  evtValidacao.chamarTelaMatrizConfusao();
            }

            // habilitar e desabilitar campos com checkbox Treinamento
            public void habilitaCampo(Object checkbox, Object text1, Object
label1) {
                  boolean s = this.getBoolean(checkbox, "selected");
                  if (s == true) {
                        setBoolean(text1, "enabled", true);
                        setBoolean(text1, "editable", true);
                        setBoolean(label1, "enabled", true);
                  } else {
                        setBoolean(text1, "enabled", false);
                        setBoolean(text1, "editable", false);
                        setBoolean(label1, "enabled", false);
                  }

            }

            // habilitar e desabilitar campos com combobox tempera treinamento
            public void habilitarTempera(Object combobox, Object label1, Object
text1,
                        Object label2, Object label3, Object text2, Object
label4) {
                  int i = this.getInteger(combobox, "selected");
                  if (i == 0) {
                        setBoolean(label1, "enabled", false);
                        setBoolean(text1, "editable", false);
                        setBoolean(text1, "enabled", false);
                        setBoolean(label2, "enabled", false);
                        setBoolean(label3, "enabled", false);
                        setBoolean(text2, "editable", false);
                        setBoolean(text2, "enabled", false);
                        setBoolean(label4, "enabled", false);
```

```
            } else {
                  setBoolean(label1, "enabled", true);
                  setBoolean(text1, "editable", true);
                  setBoolean(text1, "enabled", true);
                  setBoolean(label2, "enabled", true);
                  setBoolean(label3, "enabled", true);
                  setBoolean(text2, "editable", true);
                  setBoolean(text2, "enabled", true);
                  setBoolean(label4, "enabled", true);
            }
      }

      private void iniciarTreinamento() {
            evtTreinamento.iniciarTreinamento();
      }

      private void pararTreinamento(Object button1) {
            // Ativa a combo de renormalizaçao
            evtTreinamento.pararTreinamento();
            Object cmbNormalizador = find("cmbNormaUtil");

            setBoolean(cmbNormalizador, "enabled", true);
      }

      public void clickBotaoTreinarParar() {
      setBoolean(btMatrizConfusaoTr, "enabled", true);
            setBoolean(btGrafico, "enabled", true);
          try {

      if(App.getGerenciadorPadroes().getConjuntoTesteRede().size()==0){
                        MessageDialog ms = new MessageDialog(frame,"Erro
de Treinamento","O Conjunto de Teste está nulo. Adicione um conjunto de
Teste antes de iniciar o treinamento");
                        ms.show();
                  }
                  else{
                  if (pararTreinamento) {
                        setBoolean(btSalvarRedeTr, "enabled", true);
                        setBoolean(btCarregarRedeTr, "enabled", true);
                        setBoolean(btConfigurarTr, "enabled", true);
                        setBoolean(btCarregarRedeTr, "enabled", true);
                        try {

      if(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size()>
0){
                                    setBoolean(btDescarregarTr, "enabled",
true);
                              }

      if(App.getGerenciadorPadroes().getConjuntoTesteRede().size()>0){
                                    setBoolean(btDescarregarTs, "enabled",
true);
                              }

      if(App.getGerenciadorPadroes().getConjuntoValidacaoRede().size()>0)
{
```

```
                              setBoolean(btDescarregarVl, "enabled",
true);
                              }
                        } catch (Exception e) {
                              e.printStackTrace();
                        }

                        this.pararTreinamento(btTreinar);
                  } else {
                        setBoolean(cmbNormaUtil, "enabled", false);
                        setBoolean(btConfigNormalizacao, "enabled",
false);
                        setBoolean(btDescarregarTr, "enabled", false);
                        setBoolean(btDescarregarTs, "enabled", false);
                        setBoolean(btDescarregarVl, "enabled", false);
                        setBoolean(btConfigurarTr, "enabled", false);
                        setBoolean(btCarregarRedeTr, "enabled", false);
                        this.iniciarTreinamento();
                        this.pararTreinamento = true;
                  }
                  }
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public void configurarTelaPararTreinamento() {
            setBoolean(cmbNormaUtil, "enabled", true);
            setBoolean(btConfigNormalizacao, "enabled", true);
            this.pararTreinamento = false;
      }

      public void visualizarDadosTr(Object table, Object button) {
            String nomeDialogo = getProperty(button,
"nomeDialogo").toString();
            final DataView dv = new visao.gui.telas.utils.DataView(new
Frame(),
                        nomeDialogo);
            try {
                  final String[][] arrayDados =
App.getGerenciadorPadroes()
                              .getConjuntoTreinamento();

                  Runnable rodar = new Runnable() {
                        public void run() {
                              dv.addRows(arrayDados);
                        }
                  };

                  Thread thread = new Thread(rodar);
                  thread.start();
                  dv.show();
                  this.repaint();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }
```

```
	public void visualizarGrafico() {
		GraphicChooser gc = new GraphicChooser(frame,
				"Escolha o Grafico");
		gc.show();
		this.repaint();
	}

	public void iniciarSobre() {
		SobreEasyFAN sef = new SobreEasyFAN(new Frame(), "EasyFAN");
		sef.show();
	}


	public void open() {
		Frame f = new FrameLauncher("EasyFAN", this, 850, 600);
		f.setExtendedState(Frame.MAXIMIZED_BOTH);
		frame = f;
		File file = new
File("./src/visao/gui/telas/icones/logo1.png");
		Image bf = Toolkit.getDefaultToolkit().createImage(
				file.getAbsolutePath());
		frame.setIconImage(bf);
	}

	public void proximaPaginaTeste() {
		this.carregarTabelaTeste.nextPage();
	}

	public void anteriorPaginaTeste() {
		this.carregarTabelaTeste.beforePage();
	}

	public void proximaPaginaValidacao() {
		this.carregarTabelaValidacao.nextPage();
	}

	public void anteriorPaginaValidacao() {
		this.carregarTabelaValidacao.beforePage();
	}

	public void proximaPaginaClassificacao() {
		this.carregarTabelaClassificacao.nextPage();
	}

	public void anteriorPaginaClassificacao() {
		this.carregarTabelaClassificacao.beforePage();
	}

	public void eventTrocaNormalizador(Object cmbNormalizador) {
		int i = this.getInteger(cmbNormalizador, "selected");
		switch (i) {
		case 0:
			evtTreinamento

	.trocarTipoNormalizacao(NormalizatorTypes.NormalizatorMaxMin);
			break;
```

```
            case 1:
                  evtTreinamento

      .trocarTipoNormalizacao(NormalizatorTypes.NormalizatorMax);
                  break;
            case 2:
                  evtTreinamento

      .trocarTipoNormalizacao(NormalizatorTypes.NormalizatorMaxMean);
                  break;
            }
      }

      public void eventTrocaTempera(Object cmbTempera, Object label1,
                  Object text1, Object label2, Object label3, Object
text2,
                  Object label4) {
            habilitarTempera(cmbTempera, label1, text1, label2, label3,
text2,
                        label4);
            int i = this.getInteger(cmbTempera, "selected");
            ITemperaSimulada tempera = null;
            switch (i) {
            case 1:
                  tempera = new TemperaAleatoria();
                  break;
            case 2:                tempera = new TemperaSimuladaFAN();
                  break;
            }
            evtTreinamento.trocarTempera(tempera);
            if (tempera != null) {
            }
      }

      public void eventEmbaralharPadroes(Object checkbox, Object
textField,
                  Object label) {
            String s = getString(textField, "text");

            habilitaCampo(checkbox, textField, label);
            if (getBoolean(checkbox, "selected")) {
                  int epocas = 0;
                  try {
                        if (s != null && !s.equals(""))
                              epocas =
Integer.parseInt(getString(textField, "text"));

      evtTreinamento.alterarCadaEpocaEmbaralharPadores(epocas);
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                    j++;
```

```
                        } catch (NumberFormatException ex) {
                              sb.deleteCharAt(j);
                        }
                  }
                  setString(textField, "text", sb.toString());
            }

      } else {
            evtTreinamento.desativarCadaEpocaEmbaralharPadores();
      }
}

public void eventRenormalizar(Object checkbox, Object textField,
            Object label) {
      String s = getString(textField, "text");
      ;
      habilitaCampo(checkbox, textField, label);
      if (getBoolean(checkbox, "selected")) {
            int epocas = 0;
            try {
                  if (s != null && !s.equals(""))
                        epocas =
Integer.parseInt(getString(textField, "text"));

evtTreinamento.alterarCadaEpocaRenormalizar(epocas);
            } catch (NumberFormatException e) {
                  StringBuilder sb = new StringBuilder(s);
                  int j = 0;
                  while (j < sb.length()) {
                        try {
                              int z =
Integer.parseInt(Character.toString(sb
                                          .charAt(j)));
                              j++;
                        } catch (NumberFormatException ex) {
                              sb.deleteCharAt(j);
                        }
                  }

            }

      } else {
            evtTreinamento.desativarCadaEpocaRenormalizar();
      }
}

public void eventPesosAleatorios(Object checkbox, Object textField,
            Object label) {
      String s = getString(textField, "text");
      habilitaCampo(checkbox, textField, label);
      if (getBoolean(checkbox, "selected")) {
            int pesoBase = -1;
            try {
                  if (s != null && !s.equals(""))
                        pesoBase = Integer.parseInt(s);
                  if (pesoBase > 1000) {
                        s = s.substring(0, s.length() - 1);
```

```
                              setString(textField, "text", s);
                        } else {

      evtTreinamento.ativarPesosAleatorios(pesoBase);
                        }
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                    j++;
                              } catch (NumberFormatException ex) {
                                    sb.deleteCharAt(j);
                              }
                        }

                        setString(textField, "text", sb.toString());
                  }

            } else {
                  evtTreinamento.desativarPesosAleatorios();
            }
      }

      public void eventPorcentagemPadroesTreino(Object checkbox,
                  Object textField, Object label) {
            String s = getString(textField, "text");
            habilitaCampo(checkbox, textField, label);
            int porcentagem = 0;
            if (getBoolean(checkbox, "selected")) {
                  if (s != null && !s.equals("")) {
                        try {
                              porcentagem = Integer.parseInt(s);
                              if (porcentagem > 100 || porcentagem < 1) {
                                    s = s.substring(0, s.length() - 1);
                                    setString(textField, "text", s);
                              } else {
                                    if (porcentagem == 100) {

      evtTreinamento.desativarPorcentagemPadroesTreino();
                                    } else {
                                          evtTreinamento

      .ativarPorcentagemPadroesTreino(porcentagem);
                                    }
                              }
                        } catch (NumberFormatException e) {
                              StringBuilder sb = new StringBuilder(s);
                              int j = 0;
                              while (j < sb.length()) {
                                    try {
                                          int z =
Integer.parseInt(Character.toString(sb
                                                      .charAt(j)));
```

```
                                        j++;
                                    } catch (NumberFormatException ex) {
                                        sb.deleteCharAt(j);
                                    }
                                }

                                setString(textField, "text", sb.toString());
                            }
                        }
                    } else {
                        evtTreinamento.desativarPorcentagemPadroesTreino();
                    }
                }

                public void eventChanceRenormalizar(Object checkbox, Object
textField,
                        Object label) {
                    String s = getString(textField, "text");
                    habilitaCampo(checkbox, textField, label);
                    int porcentagem = 0;
                    if (getBoolean(checkbox, "selected")) {
                        if (s != null && !s.equals("")) {
                            try {
                                porcentagem = Integer.parseInt(s);
                                if (porcentagem > 100) {
                                    s = s.substring(0, s.length() - 1);
                                    setString(textField, "text", s);
                                } else {

                evtTreinamento.ativarChanceRenormalizar(porcentagem);
                                }
                            } catch (NumberFormatException e) {
                                StringBuilder sb = new StringBuilder(s);
                                int j = 0;
                                while (j < sb.length()) {
                                    try {
                                        int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                        j++;
                                    } catch (NumberFormatException ex) {
                                        sb.deleteCharAt(j);
                                    }
                                }

                                setString(textField, "text", sb.toString());
                            }
                        }
                    } else {
                        evtTreinamento.desativarChanceRenormalizar();
                    }
                }

                // abrir tela para configuraçao da normalizaçao
                public void parametrosNormalizacao() {
                    new visao.gui.telas.classes.ConfigNormalizacao(frame,
                            "Configurar Normalizaçao");
```

```
            EvtNormalizacao.arrumarParametrosNormalizacao();
      }

      // abrir tela para configuraçao da tempera
      public void abrirConfTempera() {
            confTempera.ConfigTempera(App.getTelaEasyFAN().getFrame(),
                        "Configurar Tempera", this.evtTreinamento);
            confTempera.show();
      }

      // abrir modo wizard
      public void abrirWizard() {
            abrirWizard.AbrirWizard(App.getTelaEasyFAN().getFrame(),
                        "EasyFAN - Modo Wizard");
            frame.setVisible(false);
            abrirWizard.show();
            frame.setVisible(true);
      }

      // configurar habilitaçao conjunto treinamento wizard
      public void configurarHabilitacaoConjuntoTreinamento() {
            setBoolean(btDescarregarTr,"enabled",true);
            setBoolean(btSalvarTr, "enabled", true);
            setBoolean(btConfigurarTr, "enabled", true);
            setBoolean(btVisualizarDadosTr, "enabled", true);
            setBoolean(btTreinar, "enabled", true);
      }

      public void configurarHabilitacaoConjuntoTeste() {
            try {
                  if (App.getGerenciadorPadroes().getConjuntoTeste() !=
null) {
                        try {
                              if (carregarTabelaTeste == null)
                                    carregarTabelaTeste = new
CarregarTabela(this,
                                                tabelaTs,
this.find("pagAtualTs"), this

      .find("pagMaxTs"));
                              carregarTabelaTeste.carregaDados(App

      .getGerenciadorPadroes().getConjuntoTeste());
                              this.repaint();
                        } catch (Exception e) {
                              e.printStackTrace();
                        }
                        setBoolean(chkVisualizarTs, "enabled", true);
                        setBoolean(chkVisualizarTs, "selected", true);
                        setBoolean(btDescarregarTs, "enabled", true);
                        setBoolean(btSalvarTs, "enabled", true);
                        setBoolean(btConfigurarTs, "enabled", true);
                        setBoolean(antPagTs, "enabled", true);
                        setBoolean(proxPagTs, "enabled", true);
                        // setBoolean(btConfigurarTs, "enabled", true);

                        this.repaint();
```

```
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        //visualizar conjuntos de teste, classificaçao e validacao
        public void visualizarConjuntos(Object tipoConjunto){
            String s = this.getProperty(tipoConjunto,
"indice").toString();
            Boolean b = getBoolean(tipoConjunto,"selected");
            if (s.equalsIgnoreCase("chkVisualizarTs")){
                if (b == false){
                    carregarTabelaTeste.clearTable();
                    this.setString(this.find("pagAtualTs"), "text",
"");
                    this.setString(this.find("pagMaxTs"), "text", "");
                    setBoolean(antPagTs, "enabled", false);
                    setBoolean(proxPagTs, "enabled", false);
                }else{
                    try {
                        carregarTabelaTeste = new
CarregarTabela(this, tabelaTs,
                        this.find("pagAtualTs"),
this.find("pagMaxTs"));
                        setBoolean(antPagTs, "enabled", true);
                        setBoolean(proxPagTs, "enabled", true);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    if (array == null){
                        array = abrirWizard.getArrayTeste();
                        if (array == null){
                            array = abrirWizard.getArrayTreino();
                        }
                    }
                   carregarTabelaTeste.carregaDados(array);
                }
            }else if (s.equalsIgnoreCase("chkVisualizarVl")){
                if (b == false){
                    carregarTabelaValidacao.clearTable();
                    this.setString(this.find("pagAtualVl"), "text",
"");
                    this.setString(this.find("pagMaxVl"), "text", "");
                    setBoolean(antPagVl, "enabled", false);
                    setBoolean(proxPagVl, "enabled", false);
                }else{
                    try {
                        carregarTabelaValidacao = new
CarregarTabela(this, tabelaVl,
                        this.find("pagAtualVl"),
this.find("pagMaxVl"));
                        setBoolean(antPagVl, "enabled", true);
                        setBoolean(proxPagVl, "enabled", true);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
```

```
                        carregarTabelaValidacao.carregaDados(array);
                  }
            }else if (s.equalsIgnoreCase("chkVisualizarCl")){
                  if (b == false){
                        carregarTabelaClassificacao.clearTable();
                        this.setString(this.find("pagAtualCl"), "text",
"");
                        this.setString(this.find("pagMaxCl"), "text", "");
                        setBoolean(antPagCl, "enabled", false);
                        setBoolean(proxPagCl, "enabled", false);
                  }else{
                        try {
                              carregarTabelaClassificacao= new
CarregarTabela(this, tabelaCl,
                              this.find("pagAtualCl"),
this.find("pagMaxCl"));
                              setBoolean(antPagCl, "enabled", true);
                              setBoolean(proxPagCl, "enabled", true);
                        } catch (IOException e) {
                              e.printStackTrace();
                        }
                        carregarTabelaClassificacao.carregaDados(array);
                  }



            }
            this.repaint();
      }

      // visualizar help
      public void criaHelp() {
            vHelp = new viewHelp(frame);// procura o arquivo EasyHelp.sh
            vHelp.visualizarHelp();
      }

      // Salvar Rede
      public void salvarRede() {
            salvarRede.salvarRede(frame, "Salvar Rede");
            salvarRede.show();
      }

      // carregar rede
      public void carregarRede() {
            carregarRede.carregarRede(App.getTelaEasyFAN().getFrame(),
                        "Carregar Rede");
      }

      public void eventTrocarEpocaStepTempera(Object txtStepTempera) {
            String s = getString(txtStepTempera, "text");
            if (s != null && !s.equals("")) {
                  int step;
                  try {
                        step = Integer.parseInt(s);
                        evtTreinamento.trocarEpocaStepTempera(step);
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
```

```
                        int j = 0;
                        while (j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                    j++;
                              } catch (NumberFormatException ex) {
                                    sb.deleteCharAt(j);
                              }
                        }
                        setString(txtStepTempera, "text", sb.toString());
                  }
            }
      }

      public void eventTrocarReinicializarTempera(Object
txtReiniciarEpoca) {
            String s = getString(txtReiniciarEpoca, "text");
            if (s != null && !s.equals("")) {
                  try {
                        int i = Integer.parseInt(s);
                        evtTreinamento.trocarEpocaReiniciarTempera(i);
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(s);
                        int j = 0;
                        while (j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb
                                                .charAt(j)));
                                    j++;
                              } catch (NumberFormatException ex) {
                                    sb.deleteCharAt(j);
                              }
                        }

                        setString(txtReiniciarEpoca, "text",
sb.toString());
                  }
            }
      }

      public void testarRede() {
            TesteRede testeRede = new TesteRede(frame, "Testar Rede");
            testeRede.show();
      }

      public void validarRede() {
            ValidaRede validaRede = new ValidaRede(frame, "Testar Rede");
            validaRede.show();
      }

      public void salvarConjuntoTreinamento(){
            EvtSalvarArquivo.salvarConjuntoTreinamento();
      }
```

```
	public void salvarConjuntoTeste() {
		EvtSalvarArquivo.salvarConjuntoTeste();
	}

	public void salvarConjuntoValidacao(){
		EvtSalvarArquivo.salvarConjuntoValidacao();
	}

	public void salvarConjuntoClassificacao(){
		EvtSalvarArquivo.salvarConjuntoClassificacao();
	}

	public EvtTeste getEvtTeste() {
		return evtTeste;
	}

	public EvtAbrirArquivo getEvtAbrirArquivo() {
		return evtAbrirArquivo;
	}

	public void setCertosTeste(int i) {
		try {
			this.setString(lblAcertosTs, "text",
Integer.toString(i));
		} catch (NumberFormatException e) {
			this.setString(lblAcertosTs, "text", "");
		}

	}

	public void setErrosTeste(int i) {
		try {
			this.setString(lblErrosTs, "text",
Integer.toString(i));
		} catch (NumberFormatException e) {
			this.setString(lblErrosTs, "text", "");
		}
	}

	public void nomearCaracteristicas(){
		NomeadorCaracteristica nc = new NomeadorCaracteristica(new
Frame(),"Nomeie as caracteristicas");
		nc.show();
	}

	public void setHarmonicaValidacao(String s) {
		setString(lblHarmonicaVl, "text", s);
	}

	public void setMaxMinValidacao(String s) {
		setString(lblMaxMinVl, "text", s);
	}

	public void setAritmeticaValidacao(String s) {
		setString(lblAritmeticaVl, "text", s);
	}
```

```
	public void setAcertosValidacao(String s) {
		setString(lblAcertosVl, "text", s);
	}

	public void setErrosValidacao(String s) {
		setString(lblErrosVl, "text", s);
	}

	public void setEpocaValidada(String s) {
		setString(lblEpocaValidada, "text", s);
	}

	public void setTipoRedeValidacao(String s) {
		setString(lblTipoRedeValidacao, "text", s);
	}

	public EvtValidacao getEvtValidacao() {
		return evtValidacao;
	}

	public EvtClassificacao getEvtClassificacao() {
		return evtClassificacao;
	}

	public void atualizarTabelaClassificacao() {
		this.carregarTabelaClassificacao.draw();
	}

	public void classificar() {
		ClassificaRede cr = new ClassificaRede(frame,"Escolhe a rede
que sera usada");
		cr.show();
		cr = null;
	}

	public void configurarPesos() {
		 if(App.getMonitorFAN() == null) {
			MessageDialog md = new MessageDialog(frame,"Atenção
","Nao existem dados para configurar os pesos.");
			md.show();
			return;
		 }
		ConfigPesos cp = new ConfigPesos(frame,"Configurar o peso de
cada classe");
		cp.show();
		cp = null;
	}

	public void salvarConfiguracoesDefault(){
		App.getProjetoConfig().saveAsDefaultConfiguration();
	}
}
```

1.5.14 InputBox.Java

```
package visao.gui.telas.classes;
```

```
import java.awt.Frame;
import java.io.IOException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class InputBox{


	public static int OK = 0;
	public static int CANCEL = 1;
	public static int OPTION = -1;

	private Thinlet thinlet;
	private ThinletDialog dialog;

	private Object tLabelRotulo,tTextFieldValor;

	public void initParameters(Object tLabelRotulo, Object
tTextFieldValor){
		this.tLabelRotulo = tLabelRotulo;
		this.tTextFieldValor = tTextFieldValor;
	}

	public String getStringTextFieldValor(){
		return thinlet.getString(tTextFieldValor,"text");
	}

	public String getStringTLabelRotulo(){
		return thinlet.getString(tLabelRotulo,"text");
	}

	public void setStringTLabelRotulo(String texto){
		thinlet.setString(tLabelRotulo,"text",texto);
	}

	public InputBox(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel = thinlet.parse("/visao/gui/telas/InputBox.xml",
this);
		} catch (IOException e) {
			e.printStackTrace();

		}
	  thinlet.add(panel);
	  dialog.setModal(true);
	  dialog.setContent(thinlet);
	}
```

```
	public void show() {
		dialog.pack();
		dialog.setSize(315, 135);
		dialog.setAlwaysOnTop(true);
	  dialog.setLocationRelativeTo(dialog.getOwner());
	  dialog.setVisible(true);
    }

	public static void main(String[] args) {
		InputBox ip = new InputBox(new Frame(),"Teste");
		ip.show();
		while (ip.dialog.isShowing()) {}
		ip.dialog.dispose();
		System.exit(0);
	}

	public void confirmar(){
		OPTION = OK;
		dialog.setVisible(false);
	}

	public void cancelar(){
		OPTION = CANCEL;
		dialog.setVisible(false);
	}

}
```

1.5.15 MatrizConfusao.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class MatrizConfusao {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	public MatrizConfusao(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public MatrizConfusao(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public MatrizConfusao(Frame owner) {
```

```
            this.dialog = new ThinletDialog(owner, false);
            this.init();
      }

      public MatrizConfusao(Frame owner, String title) {
            this.dialog = new ThinletDialog(owner, title, false);
            this.init();
      }

      private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/MatrizConfusao.xml", this);
            } catch (IOException e) {
                  e.printStackTrace();

            }
            thinlet.add(panel);
            dialog.setContent(thinlet);
      }

      public void show() {
            dialog.pack();
            dialog.setSize(300, 335);
            dialog.setAlwaysOnTop(true);
            dialog.setLocationRelativeTo(dialog.getOwner());
            dialog.setVisible(true);
      }

      public void cancelar() {
            dialog.setVisible(false);
      }

      public static void main(String[] args) {
            MatrizConfusao mc = new MatrizConfusao(new Frame(),
                        "Matriz de Confusao");
            mc.show();
            while (mc.dialog.isShowing()) {
            }
            mc.dialog.dispose();

            System.exit(0);

      }

}
```

1.5.16 MatrizConfusaoTela.Java

```
package visao.gui.telas.classes;

import java.awt.Color;
import java.awt.Dialog;
import java.awt.Frame;
```

```
import java.io.IOException;
import java.text.NumberFormat;
import java.util.Arrays;
import java.util.HashMap;

import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.GerenciadorPadroes;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;

import controle.eventos.EvtTreinamento;
import controle.main.App;

public class MatrizConfusaoTela extends AntiAliasedThinlet{

      private Thinlet thinlet;
      private ThinletDialog dialog;
      private Object pnlMatrizConfusao, lblAritmetica, lblHarmonica,
lblMaxMin, rdgOpcao, rdQuantidade, rdPorcentagem,lblEpocaTitulo ,
lblEpoca;

      private int epoca;
      private float harmonica,aritmetica,maxmin;
      private int[][] matrizConfusao;
      private double[][] matrizConfusaoPorcentagem = null;
      private String[] classesReais;
      private NumberFormat numformat = NumberFormat.getInstance();
      private String tipoRede;

      private HashMap<Integer,Integer> mapaMapeadosIndices =
RecalculadorClasses.getClassesMapRealIndex();
      private HashMap<String,Integer> mapaReaisMapeados =
App.getGerenciadorPadroes().getMapaClassesReaisMapeadas();


   public MatrizConfusaoTela(Frame ower, String title, int epoca, int[][]
matrizConfusao, float harmonica,float aritmetica,float maxmin, String
tipoRede) {
            this.dialog = new ThinletDialog(ower,title,true);
            this.epoca = epoca;
            this.matrizConfusao = matrizConfusao;
            this.harmonica = harmonica;
            this.aritmetica = aritmetica;
            this.maxmin = maxmin;
            this.tipoRede = tipoRede;
            this.init();
            if (epoca < 1) {
                  thinlet.remove(lblEpocaTitulo);
                  thinlet.remove(lblEpoca);
            }
      }

   public void initParameters(Object pnlMatrizConfusao,Object
lblAritmetica,Object lblHarmonica,Object lblMaxMin,Object rdgOpcao,Object
```

```
rdQuantidade,Object rdPorcentagem, Object lblEpoca, Object
lblEpocaTitulo) {
         this.pnlMatrizConfusao = pnlMatrizConfusao;
         this.lblAritmetica = lblAritmetica;
         this.lblHarmonica = lblHarmonica;
         this.lblMaxMin = lblMaxMin;
         this.rdgOpcao = rdgOpcao;
         this.rdQuantidade = rdQuantidade;
         this.rdPorcentagem = rdPorcentagem;
         this.lblEpoca = lblEpoca;
         this.lblEpocaTitulo = lblEpocaTitulo;
   }

   private void init() {
         if (tipoRede != null && tipoRede.equals("validacao")) {
               classesReais =
App.getGerenciadorPadroes().getClassesConjuntoValidacaoReais();
         }
         else {
               classesReais =
App.getGerenciadorPadroes().getClassesConjuntoTesteReais();
         }


          numformat.setMinimumFractionDigits(4);
            numformat.setMaximumFractionDigits(4);
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/MatrizConfusao.xml", this);
            } catch (IOException e) {
                  new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a Configuraçao de Tempera").show();
                  e.printStackTrace();

            }
       thinlet.add(panel);
       dialog.setContent(thinlet);
       dialog.setModal(true);
       //Seta a label da epoca
       thinlet.setString(lblEpoca, "text", Integer.toString(epoca));
       thinlet.setString(lblHarmonica, "text",
Double.toString(harmonica));
       thinlet.setString(lblAritmetica, "text",
Double.toString(aritmetica));
       thinlet.setString(lblMaxMin, "text", Double.toString(maxmin));
       //Cria a matriz no painel
       criarMatriz();

      }

      public void criarMatriz() {
            thinlet.removeAll(pnlMatrizConfusao);
            Arrays.sort(classesReais);
            //criar as labels de titulo
            Object[] label = new Object[matrizConfusao.length];
```

```
            Object[] label2 = new Object[matrizConfusao.length];
            int fim = label.length;
            for (int i = 0; i < fim;i++) {
                  label[i] = thinlet.create("label");
                  label2[i] = thinlet.create("label");
                  thinlet.setString(label[i],"text",classesReais[i]);
                  thinlet.setString(label2[i],"text",classesReais[i]);
            }
            Object blankLabel = thinlet.create("label");

            //Seta o numero de colunas do painel
            int colunas = (fim*2)+3;
            Object sep1 = thinlet.create("separator");
            thinlet.setInteger(sep1, "colspan", colunas);
            thinlet.add(pnlMatrizConfusao, sep1);
            thinlet.setInteger(pnlMatrizConfusao,"columns",colunas);
            //Adiciona a primeira linha ao painel
            thinlet.add(pnlMatrizConfusao, thinlet.create("separator"));
            thinlet.add(pnlMatrizConfusao, blankLabel);
            thinlet.add(pnlMatrizConfusao, thinlet.create("separator"));
            for (Object o : label) {
                  thinlet.add(pnlMatrizConfusao, o);
                  thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
            }
            Object sep = thinlet.create("separator");
            thinlet.setInteger(sep, "colspan", colunas);
            thinlet.add(pnlMatrizConfusao, sep);

            //Adiciona as outras linhas com seus cabeçalhos e dados
            int mapeado,indexado, mapeadoSub, indexadoSub;
            Object lblData, sepLine;
            for (int i = 0; i < fim; i++) {
                  thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
                  thinlet.add(pnlMatrizConfusao, label2[i]);
                  if (mapaReaisMapeados.size() > 0) {
                        mapeado = mapaReaisMapeados.get(classesReais[i]);
                        indexado = mapaMapeadosIndices.get(mapeado);
                  }
                  else {
                        indexado =
mapaMapeadosIndices.get((int)Double.parseDouble(classesReais[i]));
                  }
                  for (int j = 0; j < fim;j++) {
                        if (mapaReaisMapeados.size() > 0) {
                              mapeadoSub =
mapaReaisMapeados.get(classesReais[j]);
                              indexadoSub =
mapaMapeadosIndices.get(mapeadoSub);
                        }
                        else {
                              indexadoSub =
mapaMapeadosIndices.get((int)Double.parseDouble(classesReais[j]));
                        }
                         thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
```

```
                        lblData = thinlet.create("label");
                        thinlet.setColor(lblData, "background",
Color.WHITE);
                        thinlet.setChoice(lblData, "alignment",
"center");
                        thinlet.setString(lblData, "text",
Integer.toString(matrizConfusao[indexado][indexadoSub]));
                        thinlet.add(pnlMatrizConfusao, lblData);
                  }
                  thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
                  sepLine = thinlet.create("separator");
                  thinlet.setInteger(sepLine, "colspan", colunas);
                  thinlet.add(pnlMatrizConfusao, sepLine);
            }

      }

      public void show() {
            dialog.pack();
            dialog.setAlwaysOnTop(false);
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);
      }

      public void cancelar() {
            dialog.setVisible(false);
            dialog.dispose();
      }

      public void criarMatrizPorcentagem() {
            thinlet.removeAll(pnlMatrizConfusao); //Remove tudo do painel
            //Cria a matriz de porcentagem
            int[] divisor = new int[matrizConfusao.length];
            for (int i = 0; i < divisor.length; i++) {
                  divisor[i] = 0;
                  for (int j = 0; j < matrizConfusao[i].length; j++) {
                        divisor[i] += matrizConfusao[i][j];
                  }
            }

            if (matrizConfusaoPorcentagem == null) {
                  matrizConfusaoPorcentagem = new
double[matrizConfusao.length][matrizConfusao[0].length];
                  for (int i = 0; i < matrizConfusao.length; i++)
                        for (int j = 0; j < matrizConfusao[i].length; j++)
                              matrizConfusaoPorcentagem[i][j] =
(((double)matrizConfusao[i][j])/divisor[i]) * 100.0;
            }
            Arrays.sort(classesReais); //Ordena o array
            //criar as labels de titulo
            Object[] label = new Object[matrizConfusao.length];
            Object[] label2 = new Object[matrizConfusao.length];
            int fim = label.length;
            for (int i = 0; i < fim;i++) {
                  label[i] = thinlet.create("label");
                  label2[i] = thinlet.create("label");
```

```
                thinlet.setString(label[i],"text",classesReais[i]);
                thinlet.setString(label2[i],"text",classesReais[i]);
            }
            Object blankLabel = thinlet.create("label");

            //Seta o numero de colunas do painel
            int colunas = (fim*2)+3;
            Object sep1 = thinlet.create("separator");
            thinlet.setInteger(sep1, "colspan", colunas);
            thinlet.add(pnlMatrizConfusao, sep1);
            thinlet.setInteger(pnlMatrizConfusao,"columns",colunas);
            //Adiciona a primeira linha ao painel
            thinlet.add(pnlMatrizConfusao, thinlet.create("separator"));
            thinlet.add(pnlMatrizConfusao, blankLabel);
            thinlet.add(pnlMatrizConfusao, thinlet.create("separator"));
            for (Object o : label) {
                thinlet.add(pnlMatrizConfusao, o);
                thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
            }
            Object sep = thinlet.create("separator");
            thinlet.setInteger(sep, "colspan", colunas);
            thinlet.add(pnlMatrizConfusao, sep);

            //Adiciona as outras linhas com seus cabeçalhos e dados
            int mapeado,indexado, mapeadoSub, indexadoSub;
            Object lblData, sepLine;
            for (int i = 0; i < fim; i++) {
                thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
                thinlet.add(pnlMatrizConfusao, label2[i]); //cabeÃ§alho
                if (mapaReaisMapeados.size() > 0) {
                    mapeado = mapaReaisMapeados.get(classesReais[i]);
                    indexado = mapaMapeadosIndices.get(mapeado);
                }
                else {
                    indexado =
mapaMapeadosIndices.get((int)Double.parseDouble(classesReais[i]));
                }
                for (int j = 0; j < fim;j++) {
                    if (mapaReaisMapeados.size() > 0) {
                        mapeadoSub =
mapaReaisMapeados.get(classesReais[j]);
                        indexadoSub =
mapaMapeadosIndices.get(mapeadoSub);
                    }
                    else {
                        indexadoSub =
mapaMapeadosIndices.get((int)Double.parseDouble(classesReais[j]));
                    }
                     thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
                     lblData = thinlet.create("label");
                     thinlet.setColor(lblData, "background",
Color.WHITE);
                     thinlet.setChoice(lblData, "alignment",
"center");
```

```
                        thinlet.setString(lblData, "text",
numformat.format(matrizConfusaoPorcentagem[indexado][indexadoSub])+"%");
                        thinlet.add(pnlMatrizConfusao, lblData);
                  }
                  thinlet.add(pnlMatrizConfusao,
thinlet.create("separator"));
                  sepLine = thinlet.create("separator");
                  thinlet.setInteger(sepLine, "colspan", colunas);
                  thinlet.add(pnlMatrizConfusao, sepLine);
            }
      }

      public static void main(String[] args) {
            MatrizConfusaoTela mc = new MatrizConfusaoTela(new Frame(),
                        "Matriz de Confusao", 0, null,0,0,0,  null);
            mc.show();
            while (mc.dialog.isShowing()) {
            }
            mc.dialog.dispose();

            System.exit(0);

      }
}
```

1.5.17 NomeadorCaracteristicas.Java

```
package visao.gui.telas.classes;

      import java.awt.Dialog;
      import java.awt.Frame;
      import java.io.IOException;
      import thinlet.Thinlet;
      import thinletcommons.AntiAliasedThinlet;
      import thinletcommons.ThinletDialog;

public class NomeadorCaracteristica {

            private Thinlet thinlet;
            private ThinletDialog dialog;

            public NomeadorCaracteristica(Dialog owner) {
                  this.dialog = new ThinletDialog(owner,false);
                  this.init();
            }

            public NomeadorCaracteristica(Dialog owner, String title) {
                  this.dialog = new ThinletDialog(owner,title,false);
                  this.init();
            }

            public NomeadorCaracteristica(Frame owner) {
                  this.dialog = new ThinletDialog(owner,false);
                  this.init();
            }
            public NomeadorCaracteristica(Frame owner, String title) {
```

```
                this.dialog = new ThinletDialog(owner,title,false);
                this.init();
            }

            private void init() {
                this.thinlet = new AntiAliasedThinlet();
                Object panel = null;
                try {
                    panel =
thinlet.parse("/visao/gui/telas/EscolherNomesCaracteristicas.xml", this);
                } catch (IOException e) {
                    e.printStackTrace();

                }
              thinlet.add(panel);
              dialog.setContent(thinlet);
            }

            public void show() {
                dialog.pack();
                dialog.setSize(315, 135);
                dialog.setAlwaysOnTop(true);
              dialog.setLocationRelativeTo(dialog.getOwner());
              dialog.setVisible(true);
          }
            public void confirmar(){

            }

            public void cancelar(){
                dialog.setVisible(false);
            }

            public static void main(String[] args) {
                NomeadorCaracteristica nc = new
NomeadorCaracteristica(new Frame(),"Nomeie as características");
                nc.show();
                while (nc.dialog.isShowing()) {}
                nc.dialog.dispose();

                System.exit(0);

            }

      }
```

1.5.18 ReiniciarConfig.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
```

```
public class ReiniciarConfig {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	public ReiniciarConfig(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ReiniciarConfig(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ReiniciarConfig(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ReiniciarConfig(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/ReiniciarConf.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setSize(375, 100);
		dialog.setAlwaysOnTop(true);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
	}

	public void confirmar() {

	}

	public void cancelar() {
		dialog.setVisible(false);
	}
```

```
	public static void main(String[] args) {
		ReiniciarConfig rc = new ReiniciarConfig(new Frame(),
				"Reiniciar Configuraçoes ");
		rc.show();
		while (rc.dialog.isShowing()) {
		}
		rc.dialog.dispose();

		System.exit(0);

	}

}
```

1.5.19 ResultadosClassificacao.Java

```
package visao.gui.telas.classes;

	import java.awt.Dialog;
	import java.awt.Frame;
	import java.io.IOException;
	import thinlet.Thinlet;
	import thinletcommons.AntiAliasedThinlet;
	import thinletcommons.ThinletDialog;

	public class ResultadosClassificacao {

		private Thinlet thinlet;

		private ThinletDialog dialog;

		public ResultadosClassificacao(Dialog owner) {
			this.dialog = new ThinletDialog(owner, false);
			this.init();
		}

		public ResultadosClassificacao(Dialog owner, String title) {
			this.dialog = new ThinletDialog(owner, title, false);
			this.init();
		}

		public ResultadosClassificacao(Frame owner) {
			this.dialog = new ThinletDialog(owner, false);
			this.init();
		}

		public ResultadosClassificacao(Frame owner, String title) {
			this.dialog = new ThinletDialog(owner, title, false);
			this.init();
		}

		private void init() {
			this.thinlet = new AntiAliasedThinlet();
			Object panel = null;
			try {
```

```
                        panel =
thinlet.parse("/visao/gui/telas/ResultadosClassificacao.xml", this);
                  } catch (IOException e) {
                        e.printStackTrace();

                  }
                  thinlet.add(panel);
                  dialog.setContent(thinlet);
            }

            public void show() {
                  dialog.pack();
                  dialog.setSize(220, 170);
                  dialog.setAlwaysOnTop(true);
                  dialog.setLocationRelativeTo(dialog.getOwner());
                  dialog.setVisible(true);
            }

            public void cancelar() {
                  dialog.setVisible(false);
            }

            public static void main(String[] args) {
                  ResultadosClassificacao rc = new
ResultadosClassificacao(new Frame(),
                              "Resultados - Classificacao");
                  rc.show();
                  while (rc.dialog.isShowing()) {
                  }
                  rc.dialog.dispose();

                  System.exit(0);

            }

      }
```

1.5.20 ResultadosValidacao.Java

```
package visao.gui.telas.classes;

      import java.awt.Dialog;
      import java.awt.Frame;
      import java.io.IOException;
      import thinlet.Thinlet;
      import thinletcommons.AntiAliasedThinlet;
      import thinletcommons.ThinletDialog;

      public class ResultadosValidacao {

            private Thinlet thinlet;

            private ThinletDialog dialog;

            public ResultadosValidacao(Dialog owner) {
                  this.dialog = new ThinletDialog(owner, false);
```

```
                this.init();
        }

        public ResultadosValidacao(Dialog owner, String title) {
                this.dialog = new ThinletDialog(owner, title, false);
                this.init();
        }

        public ResultadosValidacao(Frame owner) {
                this.dialog = new ThinletDialog(owner, false);
                this.init();
        }

        public ResultadosValidacao(Frame owner, String title) {
                this.dialog = new ThinletDialog(owner, title, false);
                this.init();
        }

        private void init() {
                this.thinlet = new AntiAliasedThinlet();
                Object panel = null;
                try {
                        panel =
thinlet.parse("/visao/gui/telas/ResultadosValidacao.xml", this);
                } catch (IOException e) {
                        e.printStackTrace();

                }
                thinlet.add(panel);
                dialog.setContent(thinlet);
        }

        public void show() {
                dialog.pack();
                dialog.setSize(200, 190);
                dialog.setAlwaysOnTop(true);
                dialog.setLocationRelativeTo(dialog.getOwner());
                dialog.setVisible(true);
        }

        public void cancelar() {
                dialog.setVisible(false);
        }

        public static void main(String[] args) {
                ResultadosValidacao rv = new ResultadosValidacao(new
Frame(),
                                "Resultados - Validaçao");
                rv.show();
                while (rv.dialog.isShowing()) {
                }
                rv.dialog.dispose();

                System.exit(0);

        }
```

```
	}
```

1.5.21 SaidaSistema.Java

```
package visao.gui.telas.classes;

	import java.awt.Dialog;
	import java.awt.Frame;
	import java.io.IOException;
	import thinlet.Thinlet;
	import thinletcommons.AntiAliasedThinlet;
	import thinletcommons.ThinletDialog;

public class SaidaSistema {

		private Thinlet thinlet;

		private ThinletDialog dialog;

		public SaidaSistema(Dialog owner) {
			this.dialog = new ThinletDialog(owner, false);
			this.init();
		}

		public SaidaSistema(Dialog owner, String title) {
			this.dialog = new ThinletDialog(owner, title, false);
			this.init();
		}

		public SaidaSistema(Frame owner) {
			this.dialog = new ThinletDialog(owner, false);
			this.init();
		}

		public SaidaSistema(Frame owner, String title) {
			this.dialog = new ThinletDialog(owner, title, false);
			this.init();
		}

		private void init() {
			this.thinlet = new AntiAliasedThinlet();
			Object panel = null;
			try {
				panel =
thinlet.parse("/visao/gui/telas/Saida.xml", this);
			} catch (IOException e) {
				e.printStackTrace();

			}
			thinlet.add(panel);
			dialog.setContent(thinlet);
		}

		public void show() {
			dialog.pack();
			dialog.setSize(260, 100);
```

```
                dialog.setAlwaysOnTop(true);
                dialog.setLocationRelativeTo(dialog.getOwner());
                dialog.setVisible(true);
        }

        public void terminar(){
                System.exit(0);
        }
        public void cancelar() {
                dialog.setVisible(false);
        }

        public static void main(String[] args) {
                SaidaSistema ss = new SaidaSistema(new Frame(),
                                "Fechar o Sistema");
                ss.show();
                while (ss.dialog.isShowing()) {
                }
                ss.dialog.dispose();

                System.exit(0);

        }

}
```

1.5.22 SalvarRede.Java

```
package visao.gui.telas.classes;

import java.awt.Frame;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.io.arquivos.PersistorDados;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ExtensionFileFilter;
import thinletcommons.FileChooser;
import thinletcommons.FileFilter;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class salvarRede extends AntiAliasedThinlet {
       private ThinletDialog dialog;
       private AntiAliasedThinlet thinlet;
     private FileChooser fc;
       private static int tipNormalizacao;

      public void salvarRede(Frame ower, String title) {
            this.dialog = new ThinletDialog(ower,false);
            ChamarDialogo(App.getTelaEasyFAN().getFrame(),title);

      }
```

```
	private void ChamarDialogo(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner,title,false);
		this.init();

	  }

	private void init() {
			this.thinlet = new AntiAliasedThinlet();
			Object panel = null;
			try {
				panel =
thinlet.parse("/visao/gui/telas/SalvarRede2.xml", this);
			} catch (IOException e) {
				new MessageDialog(dialog, "Erro ao abrir a
janela", "Erro ao tentar abrir a configuração de normalização").show();
				e.printStackTrace();

			}
		  thinlet.add(panel);
		  dialog.setContent(thinlet);
		  dialog.setModal(true);
		}

	public void show() {
			dialog.pack();
			dialog.setSize(310,190);
			dialog.setAlwaysOnTop(true);
		  dialog.setLocationRelativeTo(dialog.getOwner());
		  dialog.setVisible(true);
		}



	public void fileChooserSR(){
		String tipNeuronio = null;
		Object Item1 = thinlet.find("cbAtual");
		Object Item2 = thinlet.find("cbMelhorMediaAritmetica");
		Object Item3 = thinlet.find("cbMelhorMediaHarmonica");
		Object Item4 = thinlet.find("cbMelhorMaximodoMinimo");
		boolean Item11 = this.getBoolean(Item1,"selected");
		boolean Item12 = this.getBoolean(Item2,"selected");
		boolean Item13 = this.getBoolean(Item3,"selected");
		boolean Item14 = this.getBoolean(Item4,"selected");
		if (Item11 == true){
			tipNeuronio = this.getString(Item1,"text");
		}else if(Item12 == true){
			tipNeuronio = this.getString(Item2,"text");
		}else if(Item13 == true){
			tipNeuronio = this.getString(Item3,"text");
		}else if(Item14 == true){
			tipNeuronio = this.getString(Item4,"text");
		}
		dialog.setVisible(false);
		fc = new FileChooser(App.getTelaEasyFAN().getFrame(),"Salvar
Rede", FileChooser.MODE_SAVE);
		fc.setFileFilters(new FileFilter[]{
```

```
                        new ExtensionFileFilter("enn", "EasyFAN files
(*.enn)"),
                        new ExtensionFileFilter("fan", "FAN files
(*.fan)")
                 });
              fc.show();
              File selectedFile = fc.getSelectedFile();
              if (selectedFile != null){
                    String enderecoDoArquivo = selectedFile.getPath();
                    String nomeArquivo = this.fc.getFileName();
                    String extArquivo = fc.getFilterChanged();
                    extArquivo = extArquivo.substring(extArquivo.length()-
5,extArquivo.length()-1);

App.getProjetoConfig().getRedeConf().getTipoNormalizador();
                    PersistorDados pd = new PersistorDados();
                    ArrayList<NeuronioFAN> neuronios = null;

                    if (tipNeuronio.equalsIgnoreCase("Atual"))
                          neuronios =
App.getMonitorFAN().getNeuroniosAtuais();
                    else if (tipNeuronio.equalsIgnoreCase("Melhor Média
Aritmética"))
                          neuronios =
App.getMonitorFAN().getNeuroniosMelhorAritmetica();
                    else if (tipNeuronio.equalsIgnoreCase("Melhor Média
Harmônica"))
                          neuronios =
App.getMonitorFAN().getNeuroniosMelhorHarmonica();
                    else if (tipNeuronio.equalsIgnoreCase("Melhor Máximo
do Mínimo"))
                         neuronios =
App.getMonitorFAN().getNeuroniosMelhorMaximoMinimo();


                    if(nomeArquivo == null || nomeArquivo.trim().length()
== 0 || extArquivo.equalsIgnoreCase("(*.*"))
                   new MessageDialog(App.getTelaEasyFAN().getFrame(),
"Nada selecionado", "Por favor selecione um arquivo.").show();
                    else{
                          if (extArquivo.equalsIgnoreCase(".enn")){
                                try {
                                      enderecoDoArquivo += extArquivo;

pd.gravaNeuroniosFanXml(neuronios,enderecoDoArquivo,"Teste",App.getProjet
oConfig().getRedeConf().getTipoNormalizador(),
App.getProjetoConfig().getRedeConf().getMinimos(),
App.getProjetoConfig().getRedeConf().getMaximos(),
App.getProjetoConfig().getRedeConf().getMedias(),App.getGerenciadorPadroe
s());
                                } catch (Exception e) {
                                      e.printStackTrace();
                                }
                          }else if(extArquivo.equalsIgnoreCase( ".fan" )){
                                try {
                                      enderecoDoArquivo += extArquivo;
```

```
pd.gravaNeuroniosFanDat(neuronios,enderecoDoArquivo);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                            } else{
                                new
MessageDialog(App.getTelaEasyFAN().getFrame(), "Nada selecionado", "Por
favor selecione um arquivo.").show();
                            }
                }
                }
        }
}
```

1.5.23 SobreEasyFAN.Java

```
package visao.gui.telas.classes;

      import java.awt.Dialog;
      import java.awt.Frame;
      import java.io.IOException;
      import thinlet.Thinlet;
      import thinletcommons.AntiAliasedThinlet;
      import thinletcommons.ThinletDialog;

      public class SobreEasyFAN {

            private Thinlet thinlet;
            private ThinletDialog dialog;

            public SobreEasyFAN(Dialog owner) {
                  this.dialog = new ThinletDialog(owner,false);
                  this.init();
            }

            public SobreEasyFAN(Dialog owner, String title) {
                  this.dialog = new ThinletDialog(owner,title,false);
                  this.init();
            }

            public SobreEasyFAN(Frame owner) {
                  this.dialog = new ThinletDialog(owner,false);
                  this.init();
            }
            public SobreEasyFAN(Frame owner, String title) {
                  this.dialog = new ThinletDialog(owner,title,false);
                  this.init();
            }

            private void init() {
                  this.thinlet = new AntiAliasedThinlet();
                  Object panel = null;
                  try {
                        panel =
thinlet.parse("/visao/gui/telas/SobreEasyFAN.xml", this);
```

```
                } catch (IOException e) {
                      e.printStackTrace();

                }
              thinlet.add(panel);
              dialog.setContent(thinlet);
            }

            public void show() {
                  dialog.pack();
                  dialog.setSize(315, 200);
                  dialog.setAlwaysOnTop(true);
              dialog.setLocationRelativeTo(dialog.getOwner());
              dialog.setVisible(true);
          }

            public void cancelar(){
                  dialog.setVisible(false);
            }

            public static void main(String[] args) {
                  SobreEasyFAN sef = new SobreEasyFAN(new
Frame(),"EasyFAN");
                  sef.show();
                  while (sef.dialog.isShowing()) {}
                  sef.dialog.dispose();

                  System.exit(0);

            }

      }
```

1.5.24 TesteRede.Java

```
package visao.gui.telas.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import java.util.ArrayList;
import controle.eventos.EvtTeste;
import controle.main.App;
import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;
import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.CarregaNeuronioXML;
import thinlet.Thinlet;
```

```
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

      public class TesteRede {

            private Thinlet thinlet;

            private ThinletDialog dialog;

            private StoredNetworkType tipoRede =
StoredNetworkType.BestHarmonicMean;

            private boolean carregar = false;

            private EvtTeste evtTeste =
App.getTelaEasyFAN().getEvtTeste();

            public TesteRede(Dialog owner) {
                  this.dialog = new ThinletDialog(owner, false);
                  this.init();
            }

            public TesteRede(Dialog owner, String title) {
                  this.dialog = new ThinletDialog(owner, title, false);
                  this.init();
            }

            public TesteRede(Frame owner) {
                  this.dialog = new ThinletDialog(owner, false);
                  this.init();
            }

            public TesteRede(Frame owner, String title) {
                  this.dialog = new ThinletDialog(owner, title, false);
                  this.init();
            }

            private void init() {
                  this.thinlet = new AntiAliasedThinlet();
                  Object panel = null;
                  try {
                        panel = thinlet.parse(
                                    "/visao/gui/telas/TesteRede.xml",
this);
                  } catch (IOException e) {
                        e.printStackTrace();

                  }
                  thinlet.add(panel);
                  dialog.setContent(thinlet);
            }

            public void show() {
                  dialog.pack();
                  dialog.setModal(true);
                  dialog.setLocationRelativeTo(dialog.getOwner());
                  dialog.setVisible(true);
```

```
		}

		private void allToFalse(boolean[] arr) {
			for (int i = 0; i < arr.length; i++) arr[i] = false;
		}

		public void confirmar() {
			if (carregar) {
				try {
					CarregaNeuronioXML carregada =
App.getTelaEasyFAN().getEvtAbrirArquivo().abrirRedeCarregaXML();
				NormalizatorConfiguration config = new
NormalizatorConfiguration();
					config.setMaximos(carregada.getMaximos());
					config.setMinimos(carregada.getMinimos());
					config.setMedias(carregada.getMedias());
					NormalizatorTypes tipoNorma =
carregada.getTipoNormalizacao();

					ArrayList<NeuronioFAN> neurons =
carregada.getNeuroniosFAN();

					int fim = neurons.size();
					int[] classes = new int[fim];
					for(int i = 0; i < fim; i++) {
						classes[i] =
neurons.get(i).getClasseAssociada();
					}

	RecalculadorClasses.indexarNeuronios(classes, neurons);
					ArrayList<IPadrao> conjunto =
App.getGerenciadorPadroes().getConjuntoTesteRede();
					ArrayList<IPadrao> conjuntoClonado = new
ArrayList<IPadrao>(conjunto.size());
					for(IPadrao p : conjunto) {
						conjuntoClonado.add(p.clone());
					}

					normalizarConjunto(config, conjuntoClonado,
tipoNorma);

	evtTeste.testarRede(neurons,conjuntoClonado);
				}
				catch (Exception e) {
					e.printStackTrace();
				}
			}
			else {
				evtTeste.testarRede(tipoRede);
			}
			dialog.setVisible(false);
		}

		public void cancelar() {
			dialog.setVisible(false);
		}
```

```
            public void radioHarmonica() {
                  this.carregar = false;
                  this.tipoRede = StoredNetworkType.BestHarmonicMean;
            }

            public void radioAritmetica() {
                  this.carregar = false;
                  this.tipoRede = StoredNetworkType.BestAritmethicMean;
            }

            public void radioMaxMin() {
                  this.carregar = false;
                  this.tipoRede = StoredNetworkType.BestMaxMin;
            }

            public void radioAtual() {
                  this.carregar = false;
                  this.tipoRede = StoredNetworkType.Actual;
            }

            public void radioCarregada() {
                  this.carregar = true;
            }

            public void normalizarConjunto(NormalizatorConfiguration
config,ArrayList<IPadrao> conjunto, NormalizatorTypes tipo) {
                  INormalizator norm = null;
                  if (tipo == NormalizatorTypes.NormalizatorMaxMin) {
                        norm = new NormalizatorMaxMin(config);
                  }
                  else if (tipo == NormalizatorTypes.NormalizatorMaxMean)
{
                        norm = new NormalizatorMaxMean(config);
                  }
                  else if (tipo ==
NormalizatorTypes.NormalizatorArcSinMaxMin) {
                  }
                  else{
                        norm = new NormalizatorMax(config);
                  }

                  norm.normalizarThreaded(conjunto);

                  CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(App.getProjetoConfig().getRedeConf().getR
aioDifuso(),
App.getProjetoConfig().getRedeConf().getSuporteConjuntosDifusos());
                  calc.calcularCaracteristicasFAN(conjunto);
            }

      }
```

1.5.25 ValidarRede.Java

```
package visao.gui.telas.classes;
```

```
import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import java.util.ArrayList;

import jfan.fan.NeuronioFAN;
import jfan.fan.StoredNetworkType;
import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.CarregaNeuronioXML;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.eventos.EvtValidacao;
import controle.main.App;

public class ValidaRede {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	private StoredNetworkType tipoRede =
StoredNetworkType.BestHarmonicMean;

	private boolean carregar = false;

	private EvtValidacao evtValidacao =
App.getTelaEasyFAN().getEvtValidacao();

	public ValidaRede(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ValidaRede(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ValidaRede(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ValidaRede(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}
```

```
	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel = thinlet.parse(
					"/visao/gui/telas/TesteRede.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setModal(true);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
	}

	public void confirmar() {
		if (carregar) {
			try {
				CarregaNeuronioXML carregada =
App.getTelaEasyFAN().getEvtAbrirArquivo().abrirRedeCarregaXML();
				NormalizatorConfiguration config = new
NormalizatorConfiguration();
				config.setMaximos(carregada.getMaximos());
				config.setMinimos(carregada.getMinimos());
				config.setMedias(carregada.getMedias());
				NormalizatorTypes tipoNorma =
carregada.getTipoNormalizacao();

				ArrayList<NeuronioFAN> neurons =
carregada.getNeuroniosFAN();

				int fim = neurons.size();
				int[] classes = new int[fim];
				for(int i = 0; i < fim; i++) {
					classes[i] =
neurons.get(i).getClasseAssociada();
				}
				RecalculadorClasses.indexarNeuronios(classes,
neurons);
				ArrayList<IPadrao> conjunto =
App.getGerenciadorPadroes().getConjuntoValidacaoRede();
				ArrayList<IPadrao> conjuntoClonado = new
ArrayList<IPadrao>(conjunto.size());
				for(IPadrao p : conjunto) {
					conjuntoClonado.add(p.clone());
				}

				normalizarConjunto(config, conjuntoClonado,
tipoNorma);
				evtValidacao.validarRede(neurons,conjuntoClonado);
			}
```

```
                  catch (Exception e) {
                        e.printStackTrace();
                  }

            }
            else {
                  evtValidacao.validarRede(tipoRede);
            }
            dialog.setVisible(false);
      }

      public void cancelar() {
            dialog.setVisible(false);
      }

      public void radioHarmonica() {
            this.carregar = false;
            this.tipoRede = StoredNetworkType.BestHarmonicMean;
      }

      public void radioAritmetica() {
            this.carregar = false;
            this.tipoRede = StoredNetworkType.BestAritmethicMean;
      }

      public void radioMaxMin() {
            this.carregar = false;
            this.tipoRede = StoredNetworkType.BestMaxMin;
      }

      public void radioAtual() {
            this.carregar = false;
            this.tipoRede = StoredNetworkType.Actual;
      }

      public void radioCarregada() {
            this.carregar = true;
      }

      public void normalizarConjunto(NormalizatorConfiguration
config,ArrayList<IPadrao> conjunto, NormalizatorTypes tipo) {
            INormalizator norm = null;
            if (tipo == NormalizatorTypes.NormalizatorMaxMin) {
                  norm = new NormalizatorMaxMin(config);
            }
            else if (tipo == NormalizatorTypes.NormalizatorMaxMean) {
                  norm = new NormalizatorMaxMean(config);
            }
            else if (tipo == NormalizatorTypes.NormalizatorArcSinMaxMin)
{
            }
            else{
                  norm = new NormalizatorMax(config);
            }

            norm.normalizarThreaded(conjunto);
```

```
            CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(App.getProjetoConfig().getRedeConf().getR
aioDifuso(),
App.getProjetoConfig().getRedeConf().getSuporteConjuntosDifusos());
            calc.calcularCaracteristicasFAN(conjunto);
      }
}
```

1.5.26 ViewHelp.Java

```
package visao.gui.telas.classes;

import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import java.awt.Dialog;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.Window;
import java.io.File;
import java.io.IOException;
import java.net.*;
import javax.help.*;
import javax.swing.JDialog;
import javax.swing.JFrame;


public class viewHelp extends JDialog{


      public viewHelp(Frame owner) {
            super(owner);
            this.dialog = new Dialog(owner);
      }

      public viewHelp(Dialog owner) {
            super(owner);
            this.dialog = owner;
      }


      private Dialog dialog;

      public String helpHS = "EasyHelp.hs";


       public void visualizarHelp() {
             ClassLoader cl = this.getClass().getClassLoader();
                  try {
                  //procura caminho do EasyHelp.hs atraves do ClassLoader
                  URL hsURL = HelpSet.findHelpSet(cl,helpHS);//cl e o
arquivo jar
                  //helpHS e uma variavel que contem o nome do helpset
EasyFan
                  //sabendo o caminho do helpset cria um novo objeto
HelpSet
```

```
                    //passa o local onde esta o EasyHelp.hs(hsURL)
                    HelpSet hs = new HelpSet(cl,hsURL);

                    //cria o HelpBroker para visualizar o EasyHelp
                    HelpBroker hb = hs.createHelpBroker();
                    Dimension ds = new Dimension(700,650);
                    hb.setSize(ds);
                    ((DefaultHelpBroker) hb).setActivationWindow(dialog);
                    this.setAlwaysOnTop(false);
                    hb.setDisplayed(true);

                } catch (Exception e) {
                    //Se nao achar o arquivo helpset informa o erro
                    System.out.println("HelpSet " + e.getMessage());
                    System.out.println("HelpSet " + helpHS + " nao
localizado");
                    return;
                }
            }
}
```

1.6 PACKAGE VISAO.GUI.TELAS;

1.6.1 CaracteristicasUtilizadas.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" modal="true" name="dialog"
right="4" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" gap="4" left="4" right="4"
top="4"/>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.2 CarregarArquivoTxt.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel
init="initParametersTxt(txtSeparador,ckTabulacao,txtLinhaInicial,txtLinha
Final,txtColunaInicial,txtColunaFinal,txtColunaClasse,btConfirmarTxt)"
bottom="4" gap="4" left="4" name="dialog" right="4" text="" top="4">
    <panel columns="2" gap="4" halign="center">
        <label text="Escolha as configurações iniciais e finais para
carregar o seu  arquivo  *.txt:"/>
        <label/>
```

```
        <panel border="true" bottom="4" colspan="2" columns="4" gap="4"
left="4" right="4" text="Determine qual será o separador do arquivo"
top="4">
            <label text="Separador: "/>
            <textfield columns="4" halign="center" name="txtSeparador"
action="calculaColuna()"/>
           <checkbox action="habilitarTabulacao()" name="ckTabulacao"
text="Tabulação" selected="false"/>
        </panel>
        <panel bottom="4" columns="4" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Linha Inicial:  "/>
            <textfield columns="4" end="1" halign="center"
name="txtLinhaInicial" start="1" text="1"/>
            <label text="Linha Final :  "/>
            <textfield columns="4" name="txtLinhaFinal"/>
            <label text="Coluna Inicial:"/>
            <textfield columns="4" end="1" halign="center"
name="txtColunaInicial" start="1" text="1"/>
            <label text="Coluna Final:  "/>
            <textfield columns="4" halign="center"
name="txtColunaFinal"/>
        </panel>
        <panel border="true" bottom="4" colspan="2" columns="3" gap="4"
left="4" right="4" text="Escolha a Coluna que servirá como classe"
top="4">
            <label alignment="center" halign="center" text="Coluna"/>
            <label/>
            <textfield columns="4" end="1" halign="center"
name="txtColunaClasse" start="1" text="1"/>
            <label/>
        </panel>
        <panel bottom="4" columns="3" gap="4" halign="right" left="4"
right="4" top="4">
            <label/>
            <button action="setParametrosTxt()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar" name =
"btConfirmarTxt"/>
            <button action="cancelarTxt()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.3 CarregarArquivoTxtClassificacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel
init="initParametersTxt(txtSeparador,ckTabulacao,txtLinhaInicial,txtLinha
Final,txtColunaInicial,txtColunaFinal,txtColunaClasse,btConfirmarTxt)"
bottom="4" gap="4" left="4" name="dialog" right="4" text="" top="4">
    <panel columns="2" gap="4" halign="center">
        <label text="Escolha as configurações iniciais e finais para
carregar o seu  arquivo  *.txt:"/>
        <label name= "txtColunaClasse" visible="false" text=""/>
        <label/>
```

```
        <panel border="true" bottom="4" colspan="2" columns="3" gap="4"
left="4" right="4" text="Determine qual será o separador do arquivo"
top="4">
            <label alignment="center" halign="center" text="Separador"/>
            <textfield columns="4" halign="center" name="txtSeparador"/>
            <checkbox action="habilitarTabulacao()" name="ckTabulacao"
text="Tabulação" selected="false"/>
        </panel>
        <panel bottom="4" columns="4" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Linha Inicial:  "/>
            <textfield columns="4" end="1" halign="center"
name="txtLinhaInicial" start="1" text="1"/>
            <label text="Linha Final :  "/>
            <textfield columns="4" name="txtLinhaFinal"/>
            <label text="Coluna Inicial:"/>
            <textfield columns="4" end="1" halign="center"
name="txtColunaInicial" start="1" text="1"/>
            <label text="Coluna Final:  "/>
            <textfield columns="4" halign="center"
name="txtColunaFinal"/>
        </panel>
        <panel bottom="4" colspan="2" columns="3" gap="4" left="4"
right="4" top="4"/>
        <panel bottom="4" columns="3" gap="4" halign="right" left="4"
right="4" top="4">
            <label/>
            <button action="setParametrosTxt()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"
name="btConfirmarTxt"/>
            <button action="cancelarTxt()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.4 CarregarArquivoXls.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel
init="initParametersXls(xlsLinhaInicial,xlsLinhaFinal,xlsColunaInicial,xl
sColunaFinal,xlsColunaClasse,btConfirmarXls)" bottom="4" gap="4" left="4"
name="dialog" right="4" text="" top="4">
    <panel columns="2" gap="4" halign="center">
        <label text="Escolha as configurações iniciais e finais para
carregar o seu  arquivo  *.xls:"/>
        <label/>
        <panel bottom="4" columns="4" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Linha Inicial:  "/>
            <textfield columns="4" end="1" halign="center"
name="xlsLinhaInicial" start="1" text="1"/>
            <label text="Linha Final :  "/>
            <textfield columns="4" name="xlsLinhaFinal"/>
            <label text="Coluna Inicial:"/>
            <textfield columns="4" end="1" halign="center"
name="xlsColunaInicial" start="1" text="1"/>
```

```
            <label text="Coluna Final:  "/>
            <textfield columns="4" halign="center"
name="xlsColunaFinal"/>
        </panel>
        <panel border="true" bottom="4" colspan="2" columns="3" gap="4"
left="4" right="4" text="Escolha a Coluna que servirá como classe"
top="4">
            <label alignment="center" halign="center" text="Coluna"/>
            <label/>
            <textfield columns="4" end="1" halign="center"
name="xlsColunaClasse" start="1" text="1"/>
            <label/>
        </panel>
        <panel bottom="4" columns="3" gap="4" halign="right" left="4"
right="4" top="4">
            <label/>
            <button action="setParametrosXls()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"
name="btConfirmarXls" />
            <button action="cancelarXls()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.5 CarregarArquivoXlsClassificacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel
init="initParametersXls(xlsLinhaInicial,xlsLinhaFinal,xlsColunaInicial,xl
sColunaFinal,xlsColunaClasse,btConfirmarXls)" bottom="4" gap="4" left="4"
name="dialog" right="4" text="" top="4">
    <panel columns="2" gap="4" halign="center">
        <label text="Escolha as configurações iniciais e finais para
carregar o seu  arquivo  *.xml:"/>
        <label name="xlsColunaClasse" text="" visible = "false" />
        <label/>
        <panel bottom="4" columns="4" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Linha Inicial:  "/>
            <textfield columns="4" end="1" halign="center"
name="xlsLinhaInicial" start="1" text="1"/>
            <label text="Linha Final :  "/>
            <textfield columns="4" name="xlsLinhaFinal"/>
            <label text="Coluna Inicial:"/>
            <textfield columns="4" end="1" halign="center"
name="xlsColunaInicial" start="1" text="1"/>
            <label text="Coluna Final:  "/>
            <textfield columns="4" halign="center"
name="xlsColunaFinal"/>
        </panel>
        <panel bottom="4" colspan="2" columns="3" gap="4" left="4"
right="4" top="4"/>
        <panel bottom="4" columns="3" gap="4" halign="right" left="4"
right="4" top="4">
            <label/>
```

```
            <button action="setParametrosXls()"
icon="/visao/gui/telas/icones/confirmar.png" text="btConfirmarXls"/>
            <button action="cancelarXls()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.6 ClassificarRede.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
    <checkbox group="tipoRede" action="radioCarregada()"
text="Carregada"/>
    <checkbox group="tipoRede" action="radioAtual()" text="Atual"/>
    <checkbox group="tipoRede" action="radioHarmonica()" selected="true"
text="Melhor média harmônica"/>
    <checkbox group="tipoRede" action="radioMaxMin()" text="Melhor máximo
do mínimo"/>
    <checkbox group="tipoRede" action="radioAritmetica()" text="Melhor
média aritmética"/>
    <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
        <label/>
        <button action="confirmar()" alignment="left"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
        <button action="cancelar()" alignment="left"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
    </panel>
</panel>
```

1.6.7 ConfigPesos.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4"
init="initParameters(pnlPesos)">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel name="pnlPesos" columns="3" border="true" bottom="4"
gap="4" left="4" right="4" top="4"/>
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label text="                "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.8 ConfiguracoesRede.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
```

```
<panel init="initParameters(txtRaioDifuso,txtSuporte)" bottom="4" gap="4"
left="4" name="dialog" right="4" top="4">
    <panel columns="1" gap="4" halign="center">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Raio Difuso:   "/>
            <textfield action="changeRaioDifuso()" columns="4" end="1"
halign="center" start="1" name="txtRaioDifuso"/>
            <label text="Suporte do Conjunto Difuso:"/>
            <textfield action="changeSuporteConjuntoDisfuso()"
columns="4" end="2" start="2" name="txtSuporte"/>
        </panel>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                                  "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.9 ConfigurarNormalizacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4"
init="initParameters(txtMax,txtMin,txtMean,cmbCaracteristica,chkMax,chkMi
n,chkMean)" left="4" name="dialog" right="4" top="4" weightx="1">
    <panel columns="1" gap="4" halign="center">
        <panel border="true" bottom="4" colspan="2" columns="3" gap="4"
left="4" right="4" text="Característica a ser configurada" top="4">
            <label alignment="center" halign="center"
text="Característica"/>
            <label/>
            <combobox action="comboChanged()" columns="3"
name="cmbCaracteristica"/>
            <label/>
        </panel>
        <panel bottom="4" columns="4" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Minimo:"/>
            <textfield action="minChange()" columns="15" editable="false"
enabled="false" halign="center" name="txtMin"/>
            <checkbox action="minCheck()" name="chkMin" selected="true"/>
            <label text="Automático"/>
            <label text="Média:"/>
            <textfield action="meanChange()" columns="15"
editable="false" enabled="false" halign="center" name="txtMean"/>
            <checkbox action="meanCheck()" name="chkMean"
selected="true"/>
            <label text="Automático"/>
            <label text="Máximo:"/>
            <textfield action="maxChange()" columns="15" editable="false"
enabled="false" name="txtMax"/>
            <checkbox action="maxCheck()" name="chkMax" selected="true"/>
            <label text="Automático"/>
            <label/>
```

```
            <label/>
            <label/>
            <label/>
        </panel>
    </panel>
    <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4" top="4">
        <button action="confirmarNormalizacao()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
        <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
    </panel>
</panel>
```

1.6.10 ConfigurarTempera.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" gap="4"
init="initParameters(txtLimMaximo,txtLimMinimo,txtStep)" left="4"
name="dialog" right="4" top="4">
    <panel columns="1" gap="4" halign="center">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label alignment="center" text="Limite mínimo: "/>
            <textfield action="eventTrocarLimiteMinimoTempera(this)"
columns="7" halign="center" name="txtLimMinimo"/>
            <label text="Limite Máximo:"/>
            <textfield action="eventTrocarLimiteMaximoTempera(this)"
columns="7" halign="center" name="txtLimMaximo"/>
            <label text="Step: "/>
            <textfield action="eventTrocarStepTempera(this)" columns="7"
name="txtStep"/>
        </panel>
        <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4" top="4">
            <button action="confirmarTempera()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.11 ConjuntoNulo.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<dialog>
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <label text="Conjunto nulo, carregue este conjunto ou escolha
outro conjunto."/>
        <panel bottom="4" columns="1" gap="4" halign="center" left="4"
right="4">
            <button action="cancelar()"
icon="/visao/gui/telas/icones/confirmar.png" text="   OK    "/>
        </panel>
```

```
    </panel>
</dialog>
```

1.6.12 EasyFan.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel border="true" bottom="4" columns="1" gap="4"
init="initParameters(ckEmbaralharPadroes,txtEmbaralharPadroes,labelEmbara
lharPadroes,ckPadroesTreino,txtPadroesTreino,labelPorcentagemPadroes,ckNo
rmalizarCada,txtNormalizarCada,labelNormalizarEpocas,ckChanceNormalizar,t
xtChanceNormalizar,labelChanceNormalizar,ckPesoBase,txtPesoBase,labelRamd
omizarPeso,lblNormalizacaoUtilizada,cmbNormaUtil,btnConfigNormalizacao,bt
SalvarTr,btConfigurarTr,btVisualizarDadosTr,btTreinar,btDescarregarTr,btC
arregarRedeTr,btMatrizConfusaoTr,cmbNormaUtil,btnConfigNormalizacao,btSal
varRedeTr,tabelaTs,chkVisualizarTs,btDescarregarTs,btSalvarTs,btConfigura
rTs,btGrafico,tabelaTr,btConfigRede, btAbrirProjeto, btWizard,
btAbaTreino, btAbaTeste, btAbaValida, btAbaClassifica, btAbrirTr,
btAbrirTs, btTestarRede, btMatrizConfusaoTs, lblHarmonicaTeste,
lblMaxMinTeste, lblAritmeticaTeste, lblAcertosTs,
lblErrosTs,btAbrirVl,btDescarregarVl,btSalvarVl,btConfigurarVl,btResultad
osVl,chkVisualizarVl,btValidar,tabelaVl,btAbrirCl,btDescarregarCl,btSalva
rCl,btCarregarRedeCl,btResultadosCl,chkVisualizarCl,btClassificar,tabelaC
l,antPagTs,proxPagTs,antPagVl,proxPagVl,antPagCl,proxPagCl,lblHarmonicaVl
,lblMaxMinVl,lblAritmeticaVl,lblAcertosVl,lblErrosVl,btValidarRede,btMatr
izConfusaoVl,txtStep,txtReiniciarTempera,lblEpocaValidada,lblTipoRedeVali
dacao)" left="4" name="principal" right="4" top="4" weightx="1"
weighty="1">
    <menubar>
        <menu text="Arquivo">
            <menuitem action="abrirTreinamento()"
icon="/visao/gui/telas/icones/abrirProjeto.png" text="Abrir Arquivo
Treino"/>
            <menuitem action="abrirTeste()"
icon="/visao/gui/telas/icones/abrirProjeto.png" text="Abrir Arquivo
Teste"/>
            <menuitem action="abrirValidacao(tabelaVl)"
icon="/visao/gui/telas/icones/abrirProjeto.png" text="Abrir Arquivo
Validação"/>
            <menuitem action="abrirClassificacao(tabelaCl)"
icon="/visao/gui/telas/icones/abrirProjeto.png" text="Abrir Arquivo
Classificação"/>
            <menuitem action="salvarConfiguracoesDefault()"
icon="/visao/gui/telas/icones/salvar.png" text="Salvar Configurações
Atuais"/>
            <separator/>
            <menuitem action="sairSistema()"
icon="/visao/gui/telas/icones/sair.png" text="Sair"/>
        </menu>
        <menu text="Configurações">
            <menuitem action="configurarRede()"
icon="/visao/gui/telas/icones/configuracoes.png" text="Parâmetros da
Rede"/>
            <menuitem action="selecionarCaracteristica()"
icon="/visao/gui/telas/icones/caracteristicaUtilizada.png"
text="Características Utilizadas"/>
```

```
            <menuitem action="nomearCaracteristicas()"
icon="/visao/gui/telas/icones/nomearCaracteristica.png" text="Nomear
Características"/>
            <menuitem action="configurarPesos()"
icon="/visao/gui/telas/icones/bigorna.png" text="Configurar Pesos"/>
            <menuitem action="reiniciarConfiguracao()"
icon="/visao/gui/telas/icones/recarregar.png" text="Reiniciar
Configurações"/>
        </menu>
        <menu text="Ajuda">
            <menuitem action="criaHelp()"
icon="/visao/gui/telas/icones/ajuda.png"
property="indice=EmDesenvolvimento.xml" text="Ajuda"/>
            <menuitem action="iniciarSobre()"
icon="/visao/gui/telas/icones/neuronio.png" text="Sobre"/>
        </menu>
    </menubar>
    <panel columns="13" weightx="1" weighty="1">
        <tabbedpane name="geral" placement="bottom" weightx="1"
weighty="1">
            <tab name="tInicial" text="EasyFAN">
                <panel bottom="4" columns="1" gap="4" left="4" right="4"
top="4">
                    <panel bottom="4" columns="8" gap="4" height="4"
left="4" right="4" top="4">
                        <button action="configurarRede()"
icon="/visao/gui/telas/icones/novo.png" name="btConfigRede" tooltip="Novo
Projeto"/>
                        <button action="abrirProjeto()"
icon="/visao/gui/telas/icones/abrirProjeto.png" name="btAbrirProjeto"
tooltip="Abrir Projeto"/>
                        <button action="abrirWizard()"
icon="/visao/gui/telas/icones/wizard.png" name="btWizard" tooltip="Modo
Wizard"/>
                        <button action="mudaAba(geral,this)"
icon="/visao/gui/telas/icones/treinamento.png" name="btAbaTreino"
property="indice=1" tooltip="Treinamento"/>
                        <button action="mudaAba(geral,this)"
icon="/visao/gui/telas/icones/teste.png" name="btAbaTeste"
property="indice=2" tooltip="Teste"/>
                        <button action="mudaAba(geral,this)"
icon="/visao/gui/telas/icones/validacao.png" name="btAbaValida"
property="indice=3" tooltip="Validação"/>
                        <button action="mudaAba(geral,this)"
icon="/visao/gui/telas/icones/classificacao.png" name="btAbaClassifica"
property="indice=4" tooltip="Classificação"/>
                    </panel>
                    <label alignment="center" font="14 bold"
halign="center" weightx="1"/>
                    <label alignment="center" halign="center"
icon="/visao/gui/telas/icones/neuron.png" weightx="1"/>
                </panel>
            </tab>
            <tab name="tTreinamento" text="Treinamento">
                <splitpane divider="160">
                    <panel columns="1" gap="4" left="3" right="2"
top="4">
```

```
                        <label font="bold" text="Opções"/>
                        <separator/>
                        <panel columns="1" gap="3" top="4">
                            <label text=" "/>
                            <button action="abrirTreinamento()"
alignment="left" icon="/visao/gui/telas/icones/carregar.png"
name="btAbrirTr" text="Adicionar Arquivo" weightx="1"/>
                            <button action="descarregarArquivo()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/descarregar.png" name="btDescarregarTr"
text="Limpar Conjunto" weightx="1"/>
                            <button action="salvarConjuntoTreinamento()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/salvar.png" name="btSalvarTr" text="Salvar
Conjunto"/>
                            <button action="configurar(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/configuracoes.png" name="btConfigurarTr"
property="indice=configurarTr" text="Configurar Parâmetros"/>
                            <button action="chamarMatriz()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/matriz.png" name="btMatrizConfusaoTr"
text="Matriz de Confusão"/>
                            <button
action="visualizarDadosTr(tabela,btVisualizarDadosTr)" alignment="left"
enabled="false" icon="/visao/gui/telas/icones/dados.png"
name="btVisualizarDadosTr" property="nomeDialogo=Dados do Arquivo de
Treinamento" text="Visualizar Conjunto"/>
                            <button action="visualizarGrafico()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/grafico.png" name="btGrafico"
text="Gráficos"/>
                            <button action="salvarRede()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/salvar.png" name="btSalvarRedeTr"
text="Salvar Rede"/>
                            <button action="carregarRede()"
alignment="left" icon="/visao/gui/telas/icones/abrirProjeto.png"
name="btCarregarRedeTr" text="Carregar Rede"/>
                        </panel>
                    </panel>
                    <panel background="#cfcfcf" border="true" bottom="4"
columns="1" gap="4" left="4" right="4" scrollable="true" top="4">
                        <label font="bold" text="Estratégias"/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
                            <panel columns="1" gap="4" halign="left">
                                <panel bottom="4" columns="3" gap="4"
left="4" right="4" top="4">
                                    <checkbox
action="eventEmbaralharPadroes(ckEmbaralharPadroes,txtEmbaralharPadroes,l
abelEmbaralharPadroes)" name="ckEmbaralharPadroes" selected="true"
text="Embaralhar padrões a cada "/>
                                    <textfield
action="eventEmbaralharPadroes(ckEmbaralharPadroes,txtEmbaralharPadroes,l
abelEmbaralharPadroes)" background="#cccccc" columns="4" end="3"
halign="left" name="txtEmbaralharPadroes" start="3" text="100"/>
```

```
                                    <label halign="left"
name="labelEmbaralharPadroes" text=" época(s)"/>
                                    <checkbox
action="eventPorcentagemPadroesTreino(ckPadroesTreino,txtPadroesTreino,la
belPorcentagemPadroes)" name="ckPadroesTreino" text="Porcentagem de
padrões para treino: "/>
                                    <textfield
action="eventPorcentagemPadroesTreino(ckPadroesTreino,txtPadroesTreino,la
belPorcentagemPadroes)" columns="4" editable="false" enabled="false"
halign="left" name="txtPadroesTreino"/>
                                    <label enabled="false"
name="labelPorcentagemPadroes" text=" 0% - 100%"/>
                                    <checkbox
action="eventRenormalizar(ckNormalizarCada,txtNormalizarCada,labelNormali
zarEpocas)" name="ckNormalizarCada" text="Normalizar a cada "/>
                                    <textfield
action="eventRenormalizar(ckNormalizarCada,txtNormalizarCada,labelNormali
zarEpocas)" columns="4" editable="false" enabled="false" halign="left"
name="txtNormalizarCada"/>
                                    <label enabled="false"
name="labelNormalizarEpocas" text=" época(s)"/>
                                    <checkbox
action="eventChanceRenormalizar(ckChanceNormalizar,txtChanceNormalizar,la
belChanceNormalizar)" name="ckChanceNormalizar" text="Chance de
normalizar "/>
                                    <textfield
action="eventChanceRenormalizar(ckChanceNormalizar,txtChanceNormalizar,la
belChanceNormalizar)" columns="4" editable="false" enabled="false"
halign="left" name="txtChanceNormalizar"/>
                                    <label enabled="false"
name="labelChanceNormalizar" text=" 0% - 100%"/>
                                    <checkbox
action="eventPesosAleatorios(ckPesoBase,txtPesoBase,labelRamdomizarPeso)"
name="ckPesoBase" text="Randonizar peso: "/>
                                    <textfield
action="eventPesosAleatorios(ckPesoBase,txtPesoBase,labelRamdomizarPeso)"
columns="4" editable="false" enabled="false" halign="left"
name="txtPesoBase"/>
                                    <label enabled="false"
name="labelRamdomizarPeso" text=" 0 - 1000"/>
                                    <panel colspan="2" columns="2"
gap="4" halign="left">
                                        <label
name="lblNormalizacaoUtilizada" text="Normalização utilizada "/>
                                        <combobox
action="eventTrocaNormalizador(this)" background="#ffffff" columns="10"
editable="false" halign="left" name="cmbNormaUtil" selected="0"
start="20" text="Normalizador Max Min">
                                            <choice text="Normalizador
Max Min"/>
                                            <choice text="Normalizador
Max"/>
                                            <choice text="Normalizador
Max Média"/>
                                        </combobox>
                                    </panel>
                                    <panel columns="1">
```

```
                                        <button
action="parametrosNormalizacao()" alignment="left"
name="btnConfigNormalizacao" property="indice=EmDesenvolvimento.xml"
text="Configurações..."/>
                                    </panel>
                                </panel>
                            </panel>
                            <label/>
                        </panel>
                        <panel background="#fbffff" border="true"
colspan="10" columns="10" weightx="1"/>
                        <label font="bold" text="Têmpera"/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
                            <panel columns="2" gap="4" halign="left"
weightx="1">
                                <panel bottom="4" columns="3" gap="4"
left="4" right="4" top="4" weightx="1">
                                    <label text="Têmpera utilizada "/>
                                    <combobox
action="eventTrocaTempera(this,lbStepTempera,txtStep,lbEpocaStep,reinicia
rTempera,txtReiniciarTempera,labelReiniciarTempera)" background="#ffffff"
columns="15" editable="false" halign="left" name="cmbTemperaUtil"
selected="1" start="17" text="Têmpera Randômica">
                                        <choice text="Nenhuma"/>
                                        <choice text="Têmpera
Randômica"/>
                                        <choice text="Têmpera Simulada
Padrão"/>
                                    </combobox>
                                    <button action="abrirConfTempera()"
name="btConfTempera" text="Configurar Têmpera"/>
                                    <panel bottom="4" colspan="2"
columns="3" gap="4" left="4" right="4" top="4">
                                        <label name="lbStepTempera"
text="Step têmpera a cada "/>
                                        <textfield
action="eventTrocarEpocaStepTempera(this)" columns="4" halign="left"
name="txtStep"/>
                                        <label name="lbEpocaStep" text="
época(s)"/>
                                        <label name="reiniciarTempera"
text="Reiniciar têmpera a cada "/>
                                        <textfield columns="4"
halign="left" name="txtReiniciarTempera"/>
                                        <label
name="labelReiniciarTempera" text=" época(s)"/>
                                    </panel>
                                </panel>
                                <panel bottom="4" columns="2" gap="4"
left="4" right="4" top="4" weightx="1">
                                    <label colspan="2"/>
                                    <label weightx="1"/>
                                    <label/>
                                    <label text="Valor atual:  "
weightx="1"/>
```

```
                                    <label name="lblTemperaValorAtual"
weightx="1"/>
                                    <label weightx="1"/>
                                    <label weightx="1"/>
                                </panel>
                            </panel>
                        </panel>
                        <panel background="#fbffff" border="true"
colspan="10" columns="10" weightx="1"/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
                            <label/>
                            <label name="lblFaseRede"/>
                            <label text="Época:"/>
                            <label name="lblEpoca"/>
                            <label text="Tempo de Treinamento: "/>
                            <label name="lblTempo"/>
                        </panel>
                        <panel background="#fbffff" border="true"
colspan="10" columns="10" weightx="1"/>
                        <label font="bold" text="Estatísticas"/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
                            <label/>
                            <label/>
                            <label/>
                            <label/>
                            <panel bottom="4" columns="2" gap="4"
weightx="1">
                                <panel bottom="4" columns="2" gap="4"
weightx="1">
                                    <label text="Melhor média harmônica:"
weightx="1"/>
                                    <label name="lblMelhorHarmonica"
weightx="1"/>
                                    <label text="Melhor máximo do
mínimo:" weightx="1"/>
                                    <label name="lblMelhorMaxMin"
weightx="1"/>
                                    <label text="Melhor média
aritmética:" weightx="1"/>
                                    <label name="lblMelhorAritmetica"
weightx="1"/>
                                </panel>
                                <panel bottom="4" columns="2" gap="4"
weightx="1">
                                    <label text="Média harmônica atual:"
weightx="1"/>
                                    <label name="lblHarmonicaAtual"
weightx="1"/>
                                    <label text="Máximo do mínimo atual:"
weightx="1"/>
                                    <label name="lblMaxMinAtual"
weightx="1"/>
                                    <label text="Média aritmética atual:"
weightx="1"/>
```

```
                                    <label name="lblAritmeticaAtual"
weightx="1"/>
                                </panel>
                            </panel>
                        </panel>
                        <panel bottom="4" colspan="10" columns="10"
gap="4" left="4" right="4" top="4">
                            <label weightx="1"/>
                            <label weightx="1"/>
                            <label weightx="1"/>
                            <label weightx="1"/>
                            <label weightx="1"/>
                            <label weightx="1"/>
                            <button action="clickBotaoTreinarParar()"
enabled="false" halign="right"
icon="/visao/gui/telas/icones/treinar2.png" name="btTreinar"
text="Treinar/Parar"/>
                        </panel>
                    </panel>
                </splitpane>
            </tab>
            <tab name="tTeste" text="Teste">
                <splitpane divider="170">
                    <panel columns="1" gap="4" left="3" right="2"
top="4">
                        <label font="bold" text="Opções"/>
                        <separator/>
                        <panel columns="1" gap="3" top="4">
                            <label text=" "/>
                            <button action="abrirTeste()"
alignment="left" icon="/visao/gui/telas/icones/carregar.png"
name="btAbrirTs" text="Adicionar Arquivo" weightx="1"/>
                            <button action="limparDados(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/descarregar.png" name="btDescarregarTs"
property="indice=btDescarregarTs" text="Limpar Conjunto" weightx="1"/>
                            <button action="salvarConjuntoTeste()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/salvar.png" name="btSalvarTs" text="Salvar
Conjunto"/>
                            <button action="configurar(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/configuracoes.png" name="btConfigurarTs"
property="indice=configurarTs" text="Configurar Parâmetros"/>
                            <label/>
                        </panel>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="   "/>
                        <label text="  "/>
                        <label text="   "/>
                        <label text="   "/>
```

```
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label font="bold" text="  Opções de Rede"/>
                        <separator/>
                        <panel bottom="4" columns="1" font="bold" gap="4"
left="4" top="4">
                            <button action="testarRede()"
alignment="left" enabled="false" name="btTestarRede" text="Testar Rede"
weightx="1"/>
                            <button action="abrirMatrixConfusaoTeste()"
alignment="left" enabled="false" name="btMatrizConfusaoTs" text="Matriz
de Confusão" weightx="1"/>
                        </panel>
                    </panel>
                    <panel background="#cfcfcf" border="true" bottom="4"
columns="10" gap="4" left="4" right="4" scrollable="true" top="4">
                        <label text="Conjunto de Teste             "/>
                        <label text="            "/>
                        <label text="N° de Padrões da Página:  "/>
                        <label name="padroesTsN" text="             "/>
                        <label text="     "/>
                        <label text="N° de Caracerísticas:  "/>
                        <label name="caracTsN" text="       "/>
                        <label text="    "/>
                        <label text="  "/>
                        <label name="arquivoTs" text="
"/>
                        <table colspan="10" name="tabelaTs"
rowspan="83"/>
                        <panel bottom="4" colspan="10" columns="10"
gap="4" left="4" right="4" top="4">
                            <checkbox action="visualizarConjuntos(this)"
enabled="false" name="chkVisualizarTs" property="indice=chkVisualizarTs"
text=" Visualizar Conjunto de Teste"/>
                            <label weightx="1"/>
                            <label text="Pagina: "/>
                            <label name="pagAtualTs" text="  "/>
                            <label text="/"/>
                            <label name="pagMaxTs" text="  "/>
                            <button action="anteriorPaginaTeste()"
enabled="false" halign="left"
icon="/visao/gui/telas/icones/anteriorPagina.png" name="antPagTs"
text="Página Anterior"/>
                            <button action="proximaPaginaTeste()"
enabled="false" halign="left"
icon="/visao/gui/telas/icones/proximaPagina.png" name="proxPagTs"
text="Próxima Página"/>
                            <label text="  " weightx="1"/>
                        </panel>
                        <label text="  "/>
                        <separator colspan="10"/>
                        <label/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
```

```
                            <panel bottom="4" columns="2" gap="4"
left="4" right="4" top="4" weightx="1">
                                <label text="Média harmônica: "/>
                                <label name="lblHarmonicaTeste"/>
                                <label text="Máximo do mínimo: "/>
                                <label name="lblMaxMinTeste"/>
                                <label text="Média aritmética: "/>
                                <label name="lblAritmeticaTeste"/>
                            </panel>
                            <panel bottom="4" columns="2" gap="4"
height="4" left="4" right="4" top="4" weightx="1">
                                <label text="Acertos:"/>
                                <label name="lblAcertosTs"/>
                                <label text="Erros: "/>
                                <label name="lblErrosTs"/>
                            </panel>
                        </panel>
                    </panel>
                </splitpane>
            </tab>
            <tab name="tValidação" text="Validação">
                <splitpane divider="169">
                    <panel columns="1" gap="4" left="3" right="2"
top="4">
                        <label font="bold" text="Opções"/>
                        <separator/>
                        <panel columns="1" gap="3" top="4">
                            <label text=" "/>
                            <button action="abrirValidacao(tabelaVl)"
alignment="left" icon="/visao/gui/telas/icones/carregar.png"
name="btAbrirVl" text="Adicionar Arquivo" weightx="1"/>
                            <button action="limparDados(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/descarregar.png" name="btDescarregarVl"
property="indice=btDescarregarVl" text="Limpar Conjunto" weightx="1"/>
                            <button action="salvarConjuntoValidacao()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/salvar.png" name="btSalvarVl" text="Salvar
Conjunto"/>
                            <button action="configurar(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/configuracoes.png" name="btConfigurarVl"
property="indice=configurarVl" text="Configurar Parâmetros"/>
                        </panel>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label text="   "/>
                        <label text="  "/>
                        <label text="   "/>
                        <label text="   "/>
                        <label text="  "/>
                        <label text="  "/>
```

```
                        <label text="  "/>
                        <label text="  "/>
                        <label text="  "/>
                        <label font="bold" text="  Opções de Rede"/>
                        <separator/>
                        <panel bottom="4" columns="1" font="bold" gap="4"
left="4" top="4">
                            <button action="validarRede()"
alignment="left" name="btValidarRede" text="Validar Rede" weightx="1"/>
                            <button
action="abrirMatrixConfusaoValidacao()" alignment="left"
name="btMatrizConfusaoVl" text="Matriz de Confusão" weightx="1"/>
                        </panel>
                    </panel>
                    <panel background="#cfcfcf" border="true" bottom="4"
columns="10" gap="4" left="4" right="4" scrollable="true" top="4">
                        <label text="Conjunto de Validação     "/>
                        <label text="            "/>
                        <label text="N° de Padrões:  "/>
                        <label name="padroesVlN" text="             "/>
                        <label text="     "/>
                        <label text="N° de Classes:  "/>
                        <label name="classeVlN" text="       "/>
                        <label text="    "/>
                        <label/>
                        <label name="arquivoVl" text="
"/>
                        <table colspan="10" name="tabelaVl"
rowspan="85"/>
                        <panel bottom="4" colspan="10" columns="10"
gap="4" left="4" right="4" top="4">
                            <checkbox action="visualizarConjuntos(this)"
enabled="false" name="chkVisualizarVl" property="indice=chkVisualizarVl"
text=" Visualizar Conjunto de Validação"/>
                            <label weightx="1"/>
                            <label text="Pagina: "/>
                            <label name="pagAtualVl" text="  "/>
                            <label text="/"/>
                            <label name="pagMaxVl" text="  "/>
                            <button action="anteriorPaginaValidacao()"
enabled="false" halign="left"
icon="/visao/gui/telas/icones/anteriorPagina.png" name="antPagVl"
text="Página Anterior"/>
                            <button action="proximaPaginaValidacao()"
enabled="false" halign="left"
icon="/visao/gui/telas/icones/proximaPagina.png" name="proxPagVl"
text="Próxima Página"/>
                            <label text="  " weightx="1"/>
                        </panel>
                        <label colspan="10"/>
                        <separator colspan="10"/>
                        <label colspan="10"/>
                        <panel border="true" bottom="4" colspan="10"
columns="2" gap="4" left="4" right="4" top="4" weightx="1">
                            <panel bottom="4" colspan="2" columns="2"
gap="4" left="4">
                                <label text="Época Validada: "/>
```

```
                                <label name="lblEpocaValidada"/>
                                <label text="Tipo de Rede Usada: "/>
                                <label name="lblTipoRedeValidacao"/>
                            </panel>
                            <panel bottom="4" columns="2" gap="4"
left="4" right="4" top="4" weightx="1">
                                <label text="Média harmônica: "/>
                                <label name="lblHarmonicaVl"/>
                                <label text="Máximo do mínimo: "/>
                                <label name="lblMaxMinVl"/>
                                <label text="Média aritmética: "/>
                                <label name="lblAritmeticaVl"/>
                            </panel>
                            <panel bottom="4" columns="2" gap="4"
height="4" left="4" right="4" top="4" weightx="1">
                                <label text="Acertos:"/>
                                <label name="lblAcertosVl"/>
                                <label text="Erros: "/>
                                <label name="lblErrosVl"/>
                            </panel>
                        </panel>
                    </panel>
                </splitpane>
            </tab>
            <tab name="tClassificação" text="Classificação">
                <splitpane divider="141">
                    <panel columns="1" gap="4" left="3" right="2"
top="4">
                        <label font="bold" text="Opções"/>
                        <separator/>
                        <panel columns="1" gap="3" top="4">
                            <label text=" "/>
                            <button action="abrirClassificacao(tabelaCl)"
alignment="left" icon="/visao/gui/telas/icones/carregar.png"
name="btAbrirCl" text="Adicionar Arquivo" weightx="1"/>
                            <button action="limparDados(this)"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/descarregar.png" name="btDescarregarCl"
property="indice=btDescarregarCl" text="Limpar Conjunto" weightx="1"/>
                            <button
action="salvarConjuntoClassificacao()" alignment="left" enabled="false"
icon="/visao/gui/telas/icones/salvar.png" name="btSalvarCl" text="Salvar
Conjunto"/>
                            <button action="classificar()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/abrirProjeto.png" name="btCarregarRedeCl"
text="Classificar"/>
                            <button action="exibirResultados()"
alignment="left" enabled="false"
icon="/visao/gui/telas/icones/resultados.png" name="btResultadosCl"
text="Exibir Resultados"/>
                        </panel>
                    </panel>
                    <panel background="#cfcfcf" border="true" bottom="4"
columns="10" gap="4" left="4" right="4" scrollable="true" top="4">
                        <label text="Conjunto de Classifição  "/>
                        <label text="            "/>
```

```
                        <label text="N° de Padrões:  "/>
                        <label name="padroesClN" text="             "/>
                        <label text="     "/>
                        <label text="N° de Classes:  "/>
                        <label name="classeClN" text="       "/>
                        <label text="    "/>
                        <label/>
                        <label name="arquivoCl" text="
"/>
                        <table colspan="10" name="tabelaCl" weighty="1"/>
                        <panel bottom="4" colspan="10" columns="10"
gap="4" left="4" right="4" top="4">
                            <checkbox action="visualizarConjuntos(this)"
enabled="false" name="chkVisualizarCl" property="indice=chkVisualizarCl"
text=" Visualizar Conjunto de Classificação"/>
                            <label weightx="1"/>
                            <label text="Pagina: "/>
                            <label name="pagAtualCl" text="  "/>
                            <label text="/"/>
                            <label name="pagMaxCl" text="  "/>
                            <button
action="anteriorPaginaClassificacao()" enabled="false" halign="left"
icon="/visao/gui/telas/icones/anteriorPagina.png" name="antPagCl"
text="Página Anterior"/>
                            <button action="proximaPaginaClassificacao()"
enabled="false" halign="left"
icon="/visao/gui/telas/icones/proximaPagina.png" name="proxPagCl"
text="Próxima Página"/>
                            <label text="  " weightx="1"/>
                            <button enabled="false" halign="right"
icon="/visao/gui/telas/icones/classificacao.png" name="btClassificar"
text="Classificar"/>
                        </panel>
                    </panel>
                </splitpane>
            </tab>
        </tabbedpane>
    </panel>
</panel>
```

1.6.13 EscolherNomesCaracteristicas.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<dialog bottom="4" columns="1" gap="4" left="4" right="4" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" name="pnlNomes"
top="4"/>
    <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4">
        <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
        <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
    </panel>
</dialog>
```

1.6.14 InputBox.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" gap="4"
init="initParameters(tLabelRotulo,tTextFieldValor)" left="4"
name="dialog" right="4" top="4">
    <panel columns="1" gap="4" halign="center">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label alignment="center" name="tLabelRotulo"
text="Característica:"/>
            <textfield columns="25" end="1" halign="center"
name="tTextFieldValor" start="1" text=""/>
        </panel>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                                    "/>
            <button action="confirmar()" alignment="right"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()" alignment="right"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.15 MatrizConfusao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="2" columns="1" gap="4"
init="initParameters(pnlMatrizConfusao,lblAritmetica,lblHarmonica,lblMaxM
in,rdgOpcao,rdQuantidade,rdPorcentagem,lblEpoca,lblEpocaTitulo)" left="2"
name="dialog" right="2" top="2">
    <panel border="true" bottom="4" columns="1" gap="4" left="4"
right="4" scrollable="true" top="4" weightx="1">
        <panel border="true" bottom="4" columns="1" gap="4" left="4"
name="pnlMatrizConfusao" right="4" scrollable="true" top="4"
weightx="1"/>
        <separator/>
        <panel bottom="4" columns="1" gap="4">
            <checkbox action="criarMatriz()" group="rdgOpcao"
name="rdQuantidade" selected="true" text="Quantidade de Ocorrências"/>
            <checkbox action="criarMatrizPorcentagem()" group="rdgOpcao"
name="rdPorcentagem" text="Porcentagem"/>
        </panel>
    </panel>
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
        <label name="lblEpocaTitulo" text="Época capturada:  "/>
        <label name="lblEpoca" text="                    "/>
        <label text="Média Harmônica da Época:"/>
        <label name="lblHarmonica"/>
        <label text="Máximo do Mínimo da Época:"/>
        <label name="lblMaxMin"/>
        <label text="Média Aritmética da Época: "/>
        <label name="lblAritmetica"/>
    </panel>
```

```
    <panel columns="1" gap="4" halign="right" left="4" right="4" top="4">
        <button action="cancelar()" alignment="left" colspan="2"
halign="left" icon="/visao/gui/telas/icones/sair.png" text="    Sair
"/>
    </panel>
</panel>
```

1.6.16 NovoProjeto.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
text="Novo Projeto" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label text="Nome do Projeto: "/>
            <textfield columns="20"/>
        </panel>
        <panel border="true" bottom="4" columns="2" gap="4" left="4"
right="4" text="Configurações de Rede" top="4">
            <label text="Tamanho do Conjuto Difuso (Raio Difuso): "/>
            <textfield columns="5" end="1" start="1" text="6"/>
            <label text="Suporte do Conjunto Difuso (Range): "/>
            <textfield columns="5" end="3" start="3" text="100"/>
            <panel border="true" bottom="4" colspan="2" columns="3"
gap="4" left="4" right="4" text="Função de Pertinência" top="4">
                <label alignment="center" text="Classe"/>
                <label text="                "/>
                <label alignment="center" text="Peso"/>
                <combobox/>
                <label/>
                <textfield/>
            </panel>
        </panel>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                                          "/>
            <button icon="/visao/gui/telas/icones/confirmar.png"
text="Confirmar"/>
            <button action="cancelar(dialog)"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.17 ParametrosClassificacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="2" columns="44" gap="2" left="2" name="dialog" right="2"
top="2">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" columns="1" gap="4" left="4"
right="4" top="4"/>
        <panel bottom="4" columns="3" gap="4" halign="left" left="4"
right="4" top="4">
```

```
            <button action="confirmar()" halign="left"
icon="/visao/gui/telas/icones/confirmar.png" name="confirmar"
text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" name="cancelar"
text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.18 ParametrosTeste.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" height="4"
init="initParameters(txtNumEpocas,rdContinuamente,rdNumEpocas,rdTempo,txt
DiasTs,txtHorasTs,txtMinutosTs,txtSegundosTs)" left="4" name="dialog"
right="4" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" columns="4" gap="4" left="4"
right="4" text="Modo de Teste" top="4">
            <checkbox action="testarContinuamente()" colspan="10"
group="cTeste" name="rdContinuamente" selected="true" text="Testar
Sempre"/>
            <checkbox action="trocarNumEpocasTestar()" group="cTeste"
name="rdNumEpocas" text="Testar a partir da "/>
            <textfield columns="3" editable="false" enabled="false"
name="txtNumEpocas"/>
            <label colspan="2" text=" época(s)"/>
            <label colspan="6"/>
        </panel>
        <panel bottom="4" gap="4" halign="right" left="4" right="4"
top="4">
            <label text="                    "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.19 ParametrosTreinamento.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="2" columns="1" gap="2"
init="initParameters(txtNumEpocas,rdContinuamente,rdNumEpocas,rdTempo,txt
DiasTr,txtHorasTr,txtMinutosTr,txtSegundosTr)" left="2" name="dialog"
right="2" top="2">
    <panel border="true" bottom="4" columns="3" gap="4" left="4" top="4">
        <panel bottom="4" columns="1" left="4" right="4" top="4">
            <label text="Critérios para Treinamento"/>
            <panel border="true" bottom="4" columns="10" gap="4" left="4"
right="4" top="4">
```

```
                <checkbox action="treinarContinuamente()" group="cTreino"
name="rdContinuamente" selected="true"/>
                <label colspan="9" text="Treinar continuamente"/>

                <checkbox action="trocarNumEpocasTreinar()"
group="cTreino" name="rdNumEpocas"/>
                <label text="Treinar  "/>
                <textfield action="trocarNumEpocasTreinar()" columns="4"
editable="false" enabled="false" name="txtNumEpocas"/>
                <label text="  época(s)" colspan="2"/>
                <label colspan="5"/>
                    <checkbox action="treinarTempo()" group="cTreino"
name="rdTempo"/>
                    <label text="Treinar"/>
                    <textfield columns="4" editable="false"
enabled="false" name="txtDiasTr" action="diasEdited()"/>
                    <label text="dias"/>
                    <textfield columns="3" editable="false"
enabled="false" name="txtHorasTr" action="horasEdited()"/>
                    <label text="horas"/>
                    <textfield columns="3" editable="false"
enabled="false" name="txtMinutosTr" action="minutosEdited()"/>
                    <label text="minutos"/>
                    <textfield columns="3" editable="false"
enabled="false" name="txtSegundosTr" action="segundosEdited()"/>
                    <label text="segundos"/>

            </panel>
        </panel>
    </panel>
    <label/>
    <label/>
    <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4" top="4">
        <label text=" " weightx="1"/>
        <button action="confirmar()" halign="right"
icon="/visao/gui/telas/icones/confirmar.png" name="confirmar"
text="Confirmar"/>
    </panel>
</panel>
```

1.6.20 ParametrosValidacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" height="4"
init="initParameters(txtNumEpocasVl,rdNuncaVl,rdNumEpocasVl,lbEpocas,rdVa
lidaAtual,rdValidaMedHarm,rdValidaMedArit,rdValidaMaxMin)" left="4"
name="dialog" right="4" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" columns="4" gap="4" left="4"
right="4" text="Modo de Validação" top="4">
            <checkbox action="nuncaValidar()" colspan="10"
group="cValidar" name="rdNuncaVl" selected="true" text="Nunca Validar"/>
            <checkbox action="trocarNumEpocasValidar()" group="cValidar"
name="rdNumEpocasVl" text="Validar a cada "/>
```

```
            <textfield action="trocarTxtNumEpocasValidar()" columns="3"
editable="false" enabled="false" name="txtNumEpocasVl"/>
            <label colspan="2" name="lbEpocas" text=" época(s)"/>
            <label colspan="6"/>
        </panel>
        <panel border="true" bottom="4" columns="1" gap="4" left="4"
right="4" text="Tipo de Rede para Validação" top="4">
            <checkbox group="validacaoRede" name="rdValidaAtual"
text="Atual" action="selecionarTipoRedeAtual()"/>
            <checkbox group="validacaoRede" selected="true"
name="rdValidaMedHarm" text="Melhor média harmônica"
action="selecionarTipoRedeHarmonica()"/>
            <checkbox group="validacaoRede" name="rdValidaMedArit"
text="Melhor média aritmética" action="selecionarTipoRedeAritmetica()"/>
            <checkbox group="validacaoRede" name="rdValidaMaxMin"
text="Melhor máximo do mínimo" action="selecionarTipoRedeMaxMin()"/>
        </panel>
        <panel bottom="4" gap="4" halign="right" left="4" right="4"
top="4">
            <label text="                    "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.21 ProgressoCarregar.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel name="dialog" bottom="4" columns="1" gap="4" height="4" left="4"
right="4" top="4">
    <panel bottom="4" columns="1" gap="4" height="4" left="4" right="4"
top="4">
        <label text="Carregando Arquivo"/>
        <progressbar/>
    </panel>
</panel>
```

1.6.22 ReiniciarConf.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <label text="Você tem certeza que deseja voltar às configurações
padrões?"/>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                                          "/>
            <button action="confirmar()"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
```

```
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.6.23 ResultadosClassificacao.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" columns="2" gap="4" left="4"
right="4" top="4">
            <label text="N° de Padrões: "/>
            <label name="padroesTrN"/>
            <label text="N° de Classes: "/>
            <label name="ClasseTrN"/>
            <separator colspan="2"/>
            <label text="Taxa de Classificação: "/>
            <label name="taxaClassificacao"/>
            <label text="Taxa de acerto: "/>
            <label name="taxaAcerto"/>
        </panel>
        <panel columns="2">
            <label weightx="1"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/sair.png" text="Sair"/>
        </panel>
    </panel>
</panel>
```

1.6.24 Saida.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" gap="4" left="4" name="dialog" right="4" top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <label text="Tem certeza que deseja sair do EasyFAN?"/>
        <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
            <label text="                    "/>
            <button action="terminar()"
icon="/visao/gui/telas/icones/confirmar.png" name="dialog" text="Sim"/>
            <button action="cancelar()"
icon="/visao/gui/telas/icones/cancelar.png" text="Não"/>
        </panel>
    </panel>
</panel>
```

1.6.25 SalvarRede2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
```

```
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
text="Escolha o tipo de rede que deseje salvar :" top="4" weightx="1">
    <panel border="true" bottom="4" columns="1" gap="4" left="4"
right="4" top="4" weightx="1">
        <checkbox group="gp" name="cbAtual" selected="true"
text="Atual"/>
        <checkbox group="gp" name="cbMelhorMediaAritmetica" text="Melhor
Média Aritmética"/>
        <checkbox group="gp" name="cbMelhorMediaHarmonica" text="Melhor
Média Harmônica"/>
        <checkbox group="gp" name="cbMelhorMaximodoMinimo" text="Melhor
Máximo do Mínimo"/>
    </panel>
    <panel bottom="4" columns="1" gap="4" halign="right" left="4"
right="4" top="4">
        <button action="fileChooserSR()" alignment="right"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
    </panel>
</panel>
```

1.6.26 SobreEasyFAN.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4"
weightx="1">
    <label text="EasyFAN"/>
    <textarea columns="1" editable="false" text="EasyFAN" weightx="1"/>
    <panel bottom="4" columns="1" gap="4" halign="right" left="4"
right="4" top="4">
        <button action="cancelar()"
icon="/visao/gui/telas/icones/sair.png" text="Sair"/>
    </panel>
</panel>
```

1.6.27 TesteRede.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
    <checkbox group="tipoRede" action="radioCarregada()" text="Carregar
Rede"/>
    <checkbox group="tipoRede" action="radioAtual()" text="Atual"/>
    <checkbox group="tipoRede" action="radioHarmonica()" selected="true"
text="Melhor média harmônica"/>
    <checkbox group="tipoRede" action="radioMaxMin()" text="Melhor máximo
do mínimo"/>
    <checkbox group="tipoRede" action="radioAritmetica()" text="Melhor
média aritmética"/>
    <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
        <label/>
        <button action="confirmar()" alignment="left"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
        <button action="cancelar()" alignment="left"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
```

```
    </panel>
</panel>
```

1.6.28 ValidarRede.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
    <checkbox group="tipoRede" action="radioCarregada()"
text="Carregada"/>
    <checkbox group="tipoRede" action="radioAtual()" text="Atual"/>
    <checkbox group="tipoRede" action="radioHarmonica()" selected="true"
text="Melhor média harmônica"/>
    <checkbox group="tipoRede" action="radioMaxMin()" text="Melhor máximo
do mínimo"/>
    <checkbox group="tipoRede" action="radioAritmetica()" text="Melhor
média aritmética"/>
    <panel bottom="4" columns="3" gap="4" left="4" right="4" top="4">
        <label/>
        <button action="confirmar()" alignment="left"
icon="/visao/gui/telas/icones/confirmar.png" text="Confirmar"/>
        <button action="cancelar()" alignment="left"
icon="/visao/gui/telas/icones/cancelar.png" text="Cancelar"/>
    </panel>
</panel>
```

1.7 PACKAGE VISAO.GUI.TELAS.GRAFICOS;

1.7.1 ComparateFeatureGraphic.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="6" gap="4"
init="initParameters(labelImagem,cmbCaracteristica,cmbTipoNeuronio)"
left="4" right="4" top="4">
     <label text="Tipo de Neurônios"/>
    <combobox action="updateTipoNeuronio()" name="cmbTipoNeuronio"
editable="false" selected="0" text="Atuais">
        <choice text="Atuais"/>
        <choice text="Melhor Média Aritmética"/>
        <choice text="Melhor Média Harmônica"/>
        <choice text="Melhor Máximo do Mínimo"/>
    </combobox>
    <label text="     "/>
     <label text="Característica"/>
    <combobox action="updateCaracteristica" editable ="false"
name="cmbCaracteristica"/>
    <label text=""/>
    <panel colspan="6" columns="1" weightx="1" weighty="1">
        <label name="labelImagem" weightx="1" weighty="1"/>
    </panel>
</panel>
```

1.7.2 FeatureGraphic.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="5" gap="4"
init="initParameters(labelImagem,cmbConjunto,cmbCaracteristica)" left="4"
right="4" top="4">
    <label text="Conjunto"/>
    <combobox action="updateConjunto()" editable="false"
name="cmbConjunto" selected="0" text="Treinamento">
        <choice text="Treinamento"/>
        <choice text="Teste"/>
        <choice text="Validação"/>
        <choice text="Classificação"/>
    </combobox>
    <label text="Caracteristica"/>
    <combobox action="updateCaracteristica()" editable="false"
name="cmbCaracteristica"/>
    <label/>
    <panel colspan="5" weightx="1" weighty="1">
        <label name="labelImagem" weightx="1" weighty="1"/>
    </panel>
</panel>
```

1.7.3 Grafico.xml

```
<panel gap="4" top="4" right="4" columns="1">
<bean name="chart" bean="visao.gui.telas.ChartBean" weightx="1"
colspan="9" rowspan="25" />
</panel>
```

1.7.4 GraphicChooser.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
    <button action="statisticGraphic()" alignment="left" text="Evolução
de Erros"/>
    <button action="neuronGraphic()" alignment="left" text="Gráfico de
Neurônios"/>
    <button action="comparateFeatureGraphic()" alignment="left"
text="Comparação de Características"/>
    <button action="featureGraphic()" alignment="left" text="Visualização
de Características"/>
    <panel bottom="4" columns="2" gap="4">
        <label text="                                  "/>
        <button icon="/visao/gui/telas/icones/cancelar.png"
action="cancelar()" text="Cancelar"/>
    </panel>
</panel>
```

1.7.5 NeuronGraphic.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="9" gap="4"
init="initParameters(labelImagem,cmbNeuronio,cmbTipoNeuronio,cmbCaracteri
stica)" left="4" right="4" top="4">
    <label text="Tipo de Neurônios"/>
    <combobox action="updateTipoNeuronio()" name="cmbTipoNeuronio"
editable="false" selected="0" text="Atuais">
        <choice text="Atuais"/>
        <choice text="Melhor Média Aritmética"/>
        <choice text="Melhor Média Harmônica"/>
        <choice text="Melhor Máximo do Mínimo"/>
    </combobox>
    <label text="     "/>
    <label text="Neurônio"/>
    <combobox action="updateNeuronio()" editable="false"
name="cmbNeuronio"/>
    <label text="     "/>
    <label text="Característica"/>
    <combobox action="updateCaracteristica" editable ="false"
name="cmbCaracteristica"/>
    <label/>
    <panel colspan="9" weightx="1" weighty="1">
        <label name="labelImagem" weightx="1" weighty="1"/>
    </panel>
</panel>
```

1.7.6 StatisticGraphic.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" init="initParameters(labelImagem)"
left="4" right="4" top="4">
    <panel colspan="1" weightx="1" weighty="1">
        <label name="labelImagem" weightx="1" weighty="1"/>
    </panel>
</panel>
```

1.8 PACKAGE VISAO.GUI.TELAS.GRAFICOS.CLASSES;

1.8.1 ComparateFeatureGraphic.Java

```
package visao.gui.telas.graficos.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.util.ArrayList;
import jfan.fan.NeuronioFAN;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
```

```
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class ComparateFeatureGraphic {
	private Thinlet thinlet;

	private ThinletDialog dialog;

	private Object lblGrafico;

	private JFreeChart freeChart;

	private CategoryDataset dataset = new DefaultCategoryDataset();

	private ArrayList<NeuronioFAN> array;

	private Object cmbCaracteristica;

	private Object cmbTipoNeuronio;

	private int indiceCaracteristica;

	private String[] neuronio = null;

	public ComparateFeatureGraphic(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ComparateFeatureGraphic(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public ComparateFeatureGraphic(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public ComparateFeatureGraphic(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;

		try {
			panel =
thinlet.parse("/visao/gui/telas/graficos/ComparateFeatureGraphic.xml",
					this);
```

```
            } catch (IOException e1) {
                  e1.printStackTrace();
            }

            thinlet.add(panel);
            dialog.setContent(thinlet);
            createChart();
            int fim = 0;

            try {
                  double[][] numeroCarac =
App.getMonitorFAN().getNeuroniosAtuais()
                              .get(0).getMatrizNeural();
                  fim = numeroCarac.length;

            } catch (Exception e) {
                  e.printStackTrace();
            }
            Object choice;

            for (int i = 0; i < fim; i++) {
                  choice = thinlet.create("choice");
                  thinlet.setString(choice, "text", Integer.toString(i +
1));
                  thinlet.add(cmbCaracteristica, choice, i);
            }
            thinlet.setInteger(cmbCaracteristica, "selected", 0);

            updateTipoNeuronio();
            dialog.addComponentListener(new ComponentAdapter() {
                  public void componentResized(ComponentEvent e) {
                        createImage();
                  }
            });
      }

      private void createImage() {
            int height = dialog.getHeight() - 60;
            if (height <= 0)
                  height = 1;
            thinlet.setIcon(lblGrafico, "icon", (Image) freeChart
                        .createBufferedImage(thinlet.getWidth() - 10,
height));
      }

      public void show() {
            dialog.pack();
            dialog.setSize(1000, 400);
            dialog.setAlwaysOnTop(true);
            dialog.setLocationRelativeTo(dialog.getOwner());
            dialog.setVisible(true);
            createImage();
      }

      public void initParameters(Object labelImagem, Object
cmbCaracteristica,
                  Object cmbTipoNeuronio) {
```

```
            this.cmbCaracteristica = cmbCaracteristica;
            this.lblGrafico = labelImagem;
            this.cmbTipoNeuronio = cmbTipoNeuronio;
      }

      private void createChart() {

            // create the chart...
            freeChart = ChartFactory.createLineChart(
                        "Gráfico de Comparacao de Caracteristicas", //
chart
                        // title
                        "Range", // domain axis label
                        "Valor", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips
                        false // urls
                        );

      }

      public void updateCaracteristica() {
            indiceCaracteristica =
thinlet.getSelectedIndex(cmbCaracteristica);
            updateTipoNeuronio();
      }

      public void updateTipoNeuronio() {
            int indiceTipo = thinlet.getSelectedIndex(cmbTipoNeuronio);
            ((DefaultCategoryDataset) dataset).clear();
            if (indiceTipo == 0) {
                  array = App.getMonitorFAN().getNeuroniosAtuais();
            }
            if (indiceTipo == 1) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorAritmetica();
            }
            if (indiceTipo == 2) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorHarmonica();
            }
            if (indiceTipo == 3) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorMaximoMinimo();
            }
            int fim2 = array.size();
            int fim3 = array.get(0).getMatrizNeural()[0].length;
            double[][] array3 = new double[fim2][fim3];
            double[][] nf = null;
            for (int j=0; j<fim2;j++){
                  nf=array.get(j).getMatrizNeural();
             for (int jj=0; jj<fim3; jj++){
                   array3[j][jj]= nf[indiceCaracteristica][jj];
             }
            }
```

```
            neuronio = new String[array3.length];
            int fim4 = array3.length;
            int fim5 = array3[0].length;
            for (int i = 0; i < fim4; i++) {
                  neuronio[i] = "Neuronio" + Integer.toString(i + 1);
                  double somatorio =
array.get(0).getSomatorio(indiceCaracteristica+1);
                  for (int ii = 0; ii < fim5; ii++) {
                        String columns = Integer.toString(ii);
                              ((DefaultCategoryDataset)
dataset).addValue(array3[i][ii]/somatorio,
                                          neuronio[i], columns);
                        }
                  }

            createImage();
      }

      public static void main(String[] args) {
            final ComparateFeatureGraphic cfg = new
ComparateFeatureGraphic(
                        new Frame(), "Comparacao de Caracteristicas");
            cfg.show();
            cfg.dialog.addWindowListener(new WindowListener() {

                  public void windowActivated(WindowEvent e) {

                  }

                  public void windowClosed(WindowEvent e) {
                        System.exit(0);

                  }

                  public void windowClosing(WindowEvent e) {

                  }

                  public void windowDeactivated(WindowEvent e) {

                  }

                  public void windowDeiconified(WindowEvent e) {

                  }

                  public void windowIconified(WindowEvent e) {

                  }

                  public void windowOpened(WindowEvent e) {

                  }

            });
      }
}
```

1.8.2 FeatureGraphic.Java

```
	package visao.gui.telas.graficos.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import jfan.fan.padroes.IPadrao;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;
import visao.gui.telas.classes.ConjuntoNulo;
import controle.main.App;

public class FeatureGraphic {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	private Object lblGrafico;

	private JFreeChart freeChart;

	private CategoryDataset dataset = new DefaultCategoryDataset();

	private ArrayList<IPadrao> array;

	private double[][] arrayGrafico;

	private String[] caracteristica = null;

	private String serieCaracteristica = "Característica";

	private Object cmbConjunto;

	private Object cmbCaracteristica;

	private int tamanhoArray;

	private int lastCaracteristica = -10;
```

```
	private int lastConjunto = -10;

	public FeatureGraphic(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public FeatureGraphic(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public FeatureGraphic(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public FeatureGraphic(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;

		try {
			panel = thinlet.parse(

	"/visao/gui/telas/graficos/FeatureGraphic.xml", this);
		} catch (IOException e1) {
			e1.printStackTrace();
		}

		thinlet.add(panel);
		dialog.setContent(thinlet);
		createChart();
		int fim = 0;
		try {
			fim =
App.getGerenciadorPadroes().getConjuntoTreinamentoRede().get(
					0).getQuantasCaracteristicas();
		} catch (Exception e) {
			e.printStackTrace();
		}
		Object choice;
		for (int i = 0; i < fim; i++) {
			choice = thinlet.create("choice");
			thinlet.setString(choice, "text", Integer.toString(i +
1));
			thinlet.add(cmbCaracteristica, choice, i);
		}
		thinlet.setInteger(cmbCaracteristica, "selected", 0);

		updateConjunto();
		dialog.addComponentListener(new ComponentAdapter() {
			public void componentResized(ComponentEvent e) {
```

```
                        createImage();
                  }
            });
      }

      private void createImage() {
            int height = dialog.getHeight() - 60;
            if (height <= 0)
                  height = 1;
            thinlet.setIcon(lblGrafico, "icon", (Image) freeChart
                        .createBufferedImage(thinlet.getWidth() - 10,
height));
      }

      public void show() {
            dialog.pack();
            dialog.setSize(1010, 500);
            dialog.setAlwaysOnTop(true);
            dialog.setLocationRelativeTo(dialog.getOwner());
            dialog.setVisible(true);
            createImage();
      }

      public void initParameters(Object labelImagem, Object cmbConjunto,
                  Object cmbCaracteristica) {
            this.cmbConjunto = cmbConjunto;
            this.cmbCaracteristica = cmbCaracteristica;
            this.lblGrafico = labelImagem;
      }

      public void updateConjunto() {
            int itemIndice = thinlet.getSelectedIndex(cmbConjunto);
            if (itemIndice == lastConjunto)
                  return;

            if (itemIndice == 0) {

                  try {
                        if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede()
                                    .size() == 0) {
                              ConjuntoNulo conjuntoNulo = new
ConjuntoNulo(new Frame(),
                                          "Conjunto de Treinamento
Vazio");
                              conjuntoNulo.show();
                              thinlet.setBoolean(cmbCaracteristica,
"enabled", false);
                              thinlet.repaint();
                        } else {
                              thinlet.setBoolean(cmbCaracteristica,
"enabled", true);
                              thinlet.repaint();
                              try {
                                    array = App.getGerenciadorPadroes()

      .getConjuntoTreinamentoRede();
```

```
                                } catch (Exception e) {
                                      e.printStackTrace();
                                }
                          }
                    } catch (Exception e) {
                          e.printStackTrace();
                    }
              }
              if (itemIndice == 1) {
                    try {
                          if
(App.getGerenciadorPadroes().getConjuntoTesteRede().size() == 0) {
                                ConjuntoNulo conjuntoNulo = new
ConjuntoNulo(new Frame(),
                                            "Conjunto de Teste Vazio");
                                conjuntoNulo.show();
                                thinlet.setBoolean(cmbCaracteristica,
"enabled", false);
                                thinlet.repaint();
                          } else {
                                thinlet.setBoolean(cmbCaracteristica,
"enabled", true);
                                thinlet.repaint();
                                try {
                                      array = App.getGerenciadorPadroes()
                                                  .getConjuntoTesteRede();
                                } catch (Exception e) {
                                      e.printStackTrace();
                                }
                          }
                    } catch (Exception e) {
                          e.printStackTrace();
                    }
              }
              if (itemIndice == 2) {
                    try {
                          if
(App.getGerenciadorPadroes().getConjuntoValidacaoRede()
                                      .size() == 0) {
                                ConjuntoNulo conjuntoNulo = new
ConjuntoNulo(new Frame(),
                                            "Conjunto de Validação Vazio");
                                conjuntoNulo.show();
                                thinlet.setBoolean(cmbCaracteristica,
"enabled", false);
                                thinlet.repaint();
                          } else {
                                thinlet.setBoolean(cmbCaracteristica,
"enabled", true);
                                thinlet.repaint();
                                try {
                                      array = App.getGerenciadorPadroes()

      .getConjuntoValidacaoRede();
                                } catch (Exception e) {
                                      e.printStackTrace();
                                }
```

```
                        }
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            if (itemIndice == 3) {
                  try {
                        if
(App.getGerenciadorPadroes().getConjuntoClassificacaoRede()
                                    .size() == 0) {
                              ConjuntoNulo conjuntoNulo = new
ConjuntoNulo(new Frame(),
                                          "Conjunto de Classificação
Vazio");
                              conjuntoNulo.show();
                              thinlet.setBoolean(cmbCaracteristica,
"enabled", false);
                              thinlet.repaint();
                        } else {
                              thinlet.setBoolean(cmbCaracteristica,
"enabled", true);
                              thinlet.repaint();
                              try {
                                    array = App.getGerenciadorPadroes()

      .getConjuntoClassificacaoRede();
                              } catch (Exception e) {
                                    e.printStackTrace();
                              }
                        }
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            updateCaracteristica();
            lastConjunto = itemIndice;
      }

      public void updateCaracteristica() {
            ((DefaultCategoryDataset) dataset).clear();
            App.requisitarNormalizacao();
            Object item = thinlet.getSelectedItem(cmbCaracteristica);
            int caracSize = array.get(0).getQuantasCaracteristicas();
            double[][] arrayOrdenado = new
double[array.get(0).getQuantasCaracteristicas()][array.size()];
            double[][] arrayTodas = new
double[array.get(0).getQuantasCaracteristicas()][array.size()];
            int arraySize = array.size();
            caracteristica = new String[caracSize];
            for (int i = 0; i < caracSize; i++) {
                  for (int ii = 0; ii < arraySize; ii++) {
                        arrayOrdenado[i][ii] = array.get(ii)
                                    .getCaracteristicaNormalizada(i + 1);
                        arrayTodas[i][ii] =
array.get(ii).getCaracteristicaNormalizada(i + 1);
                  }
            }
```

```
            for (int i = 0; i < arrayOrdenado.length; i++) {
                  Arrays.sort(arrayOrdenado[i]);
            }
            if (arraySize > 50) {

                  tamanhoArray = arraySize / 50;
                  if (!(arraySize % 50 == 0)) {
                        arrayGrafico = new double[caracSize][51];
                  } else {
                        arrayGrafico = new double[caracSize][50];
                  }
                  int cont = 0;
                  int indiceArray = 0;
                  double valor = 0.0;
                  for (int j = 0; j < caracSize; j++) {
                        for (int jj = 0; jj < arrayOrdenado[0].length;
jj++) {
                              valor = valor + arrayOrdenado[j][jj];
                              cont = cont + 1;
                              if (cont % tamanhoArray == 0) {
                                    arrayGrafico[j][indiceArray] = valor /
cont;
                                    valor = 0;
                                    cont = 0;
                                    indiceArray = indiceArray + 1;
                                    if (indiceArray == 50) {
                                          indiceArray = 0;
                                    }
                              }

                        }
                        if (!(valor == 0)) {
                              arrayGrafico[j][50] = valor / cont;
                        }
                        valor = 0;
                        cont = 0;
                  }
            } else {
                  arrayGrafico = new double[caracSize][array.size()];
                  for (int i = 0; i < caracSize; i++) {
                        for (int ii = 0; ii < arrayOrdenado[0].length;
ii++) {
                              arrayGrafico[i][ii] = arrayOrdenado[i][ii];
                        }
                  }

            }
            // *******Criação do Dataset*************
            if (item != null) {
                  int itemIndice =
thinlet.getSelectedIndex(cmbCaracteristica);
                  String itemTexto = thinlet.getString(item, "text");
                  if (itemIndice == lastCaracteristica)
                        return;
                  ((DefaultCategoryDataset) dataset).clear();
                  if (itemTexto.equals("Todas")) {
```

```
                        if (arraySize > 50) {

                              ((DefaultCategoryDataset) dataset).clear();
                        } else {
                              for (int i = 0; i < caracSize; i++) {
                                    caracteristica[i] = "Característica"
                                                + Integer.toString(i + 1);
                                    for (int ii = 0; ii < array.size();
ii++) {
                                          String columns =
Integer.toString(ii);
                                          ((DefaultCategoryDataset)
dataset).addValue(array
                                                      .get(ii)

      .getCaracteristicaNormalizada(i + 1),
                                                      caracteristica[i],
columns);
                                    }
                              }
                        }
                  } else {
                        int fim = arrayGrafico[itemIndice].length;
                        for (int i = 0; i < fim; i++) {
                              String columns = Integer.toString(i + 1);
                              ((DefaultCategoryDataset) dataset)

      .addValue(arrayGrafico[itemIndice][i],
                                                      serieCaracteristica
                                                                  +
Integer.toString(itemIndice + 1),
                                                      columns);

                        }
                  }
                  lastCaracteristica = itemIndice;
                  createImage();

            }
      }

      private void createChart() {

            // create the chart...
            freeChart = ChartFactory.createLineChart(
                        "Visualização de Característica", // chart
                        // title
                        "Padrões", // domain axis label
                        "Característica Normalizada", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips
                        false // urls
                        );

      }
```

```
	public static void main(String[] args) {
		final FeatureGraphic fg = new FeatureGraphic(new Frame(),
				"Gráfico de Característica");
		fg.show();
		fg.dialog.addWindowListener(new WindowListener() {

			public void windowActivated(WindowEvent e) {

			}

			public void windowClosed(WindowEvent e) {
				System.exit(0);

			}

			public void windowClosing(WindowEvent e) {

			}

			public void windowDeactivated(WindowEvent e) {

			}

			public void windowDeiconified(WindowEvent e) {

			}

			public void windowIconified(WindowEvent e) {

			}

			public void windowOpened(WindowEvent e) {

			}

		});
	}
}
```

1.8.3 GraphicChooser.Java

```
package visao.gui.telas.graficos.classes;

import java.awt.Frame;
import java.io.IOException;

import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class GraphicChooser extends AntiAliasedThinlet {

	private static final long serialVersionUID = 1L;

	private Thinlet thinlet;
```

```
	private ThinletDialog dialog;

	public GraphicChooser(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;
		try {
			panel =
thinlet.parse("/visao/gui/telas/graficos/GraphicChooser.xml", this);
		} catch (IOException e) {
			e.printStackTrace();

		}
		thinlet.add(panel);
		dialog.setContent(thinlet);
	}

	public void show() {
		dialog.pack();
		dialog.setSize(205, 170);
		dialog.setAlwaysOnTop(false);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
	}

	public void statisticGraphic() {
		StatisticGraphic sg = new StatisticGraphic(new Frame(),
"Evolução de Erros");
		sg.show();
		this.repaint();
	}

	public void featureGraphic() {
		FeatureGraphic fg = new FeatureGraphic(new Frame(),
"Visualização de Características");
		fg.show();
		this.repaint();
	}

	public void neuronGraphic() {
		NeuronGraphic ng = new NeuronGraphic(new Frame(),"Gráfico de
Neurônio");
		ng.show();
		this.repaint();
		}
    public void comparateFeatureGraphic(){
      ComparateFeatureGraphic cfg = new ComparateFeatureGraphic(new
Frame(), "Comparação de Características");
      cfg.show();
      this.repaint();
    }
```

```
	public void cancelar(){
		dialog.setVisible(false);
	}
	public static void main(String[] args) {
		GraphicChooser gc = new GraphicChooser(new Frame(), "Escolha
o Grafico");
		gc.show();
	}
}
```

1.8.4 NeuronChooser.Java

```
package visao.gui.telas.graficos.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.util.ArrayList;
import jfan.fan.NeuronioFAN;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.GerenciadorPadroes;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class NeuronGraphic {
	private Thinlet thinlet;

	private ThinletDialog dialog;

	private Object lblGrafico;

	private JFreeChart freeChart;

	private CategoryDataset dataset = new DefaultCategoryDataset();

	private ArrayList<NeuronioFAN> array;

	private Object cmbNeuronio;

	private Object cmbTipoNeuronio;

	private Object cmbCaracteristica;
```

```
	private int indiceNeuronio;

	private int indiceCaracteristica;

	private String[] caracteristica = null;

	public NeuronGraphic(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public NeuronGraphic(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public NeuronGraphic(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public NeuronGraphic(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;

		try {
			panel =
thinlet.parse("/visao/gui/telas/graficos/NeuronGraphic.xml",
					this);
		} catch (IOException e1) {
			e1.printStackTrace();
		}

		thinlet.add(panel);
		dialog.setContent(thinlet);
		createChart();
		int fim = 0;
		int fim2 = 0;

		try {
			double[][] numeroCarac =
App.getMonitorFAN().getNeuroniosAtuais()
					.get(0).getMatrizNeural();
			fim = App.getMonitorFAN().getNeuroniosAtuais().size();
			fim2 = numeroCarac.length;

		} catch (Exception e) {
			e.printStackTrace();
		}
		Object choice;
		GerenciadorPadroes gp = App.getGerenciadorPadroes();
		for (int i = 0; i < fim; i++) {
```

```
                choice = thinlet.create("choice");
                int classMapeada =
RecalculadorClasses.getClassesMapIndexReal().get(i);
                String classe = gp.getMapaClasses().get(classMapeada);
                if (classe == null) {
                      classe = Integer.toString(classMapeada);
                }
                thinlet.setString(choice, "text", classe);
                thinlet.add(cmbNeuronio, choice, i);
                thinlet.setInteger(cmbNeuronio, "selected", 0);
          }
          for (int i = 0; i < fim2; i++) {
                choice = thinlet.create("choice");
                thinlet.setString(choice, "text", Integer.toString(i +
1));
                thinlet.add(cmbCaracteristica, choice, i);
          }
          choice = thinlet.create("choice");
          thinlet.setString(choice, "text", "Todas");
          thinlet.add(cmbCaracteristica, choice, fim2);
          thinlet.setInteger(cmbCaracteristica, "selected", 0);

          updateTipoNeuronio();
          dialog.addComponentListener(new ComponentAdapter() {
                public void componentResized(ComponentEvent e) {
                      createImage();
                }
          });
    }

    private void createImage() {
          int height = dialog.getHeight() - 60;
          if (height <= 0)
                height = 1;
          thinlet.setIcon(lblGrafico, "icon", (Image) freeChart
                      .createBufferedImage(thinlet.getWidth() - 10,
height));
    }

    public void show() {
          dialog.pack();
          dialog.setSize(1000, 400);
          dialog.setLocationRelativeTo(dialog.getOwner());
          dialog.setVisible(true);
          createImage();
    }

    public void initParameters(Object labelImagem, Object cmbNeuronio,
                Object cmbTipoNeuronio, Object cmbCaracteristica) {
          this.cmbNeuronio = cmbNeuronio;
          this.lblGrafico = labelImagem;
          this.cmbTipoNeuronio = cmbTipoNeuronio;
          this.cmbCaracteristica = cmbCaracteristica;

    }

    private void createChart() {
```

```
            // create the chart...
            freeChart = ChartFactory.createLineChart("Gráfico de
Neurônio", // chart
                        // title
                        "Range", // domain axis label
                        "Valor", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips
                        false // urls
                        );

      }

      public void updateCaracteristica() {
            indiceCaracteristica =
thinlet.getSelectedIndex(cmbCaracteristica);
            updateTipoNeuronio();
      }

      public void updateNeuronio() {
            indiceNeuronio = thinlet.getSelectedIndex(cmbNeuronio);
            updateTipoNeuronio();
      }

      public void updateTipoNeuronio() {
            int indiceTipo = thinlet.getSelectedIndex(cmbTipoNeuronio);
            ((DefaultCategoryDataset) dataset).clear();
            if (indiceTipo == 0) {
                  array = App.getMonitorFAN().getNeuroniosAtuais();
            }
            if (indiceTipo == 1) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorAritmetica();
            }
            if (indiceTipo == 2) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorHarmonica();
            }
            if (indiceTipo == 3) {
                  array =
App.getMonitorFAN().getNeuroniosMelhorMaximoMinimo();
            }
            double[][] array2 =
array.get(indiceNeuronio).getMatrizNeural();

            caracteristica = new String[array2.length];
            if (indiceCaracteristica == array2.length) {
                  for (int i = 0; i < array2.length; i++) {
                        double somatorio =
array.get(indiceNeuronio).getSomatorio(i+1);
                        caracteristica[i] = "Característica" +
Integer.toString(i + 1);
                        for (int ii = 0; ii < array2[0].length; ii++) {
                              String columns = Integer.toString(ii);
```

```
                                ((DefaultCategoryDataset)
dataset).addValue(array2[i][ii]/somatorio,
                                                caracteristica[i], columns);
                        }
                  }
            } else {
                  double somatorio =
array.get(indiceNeuronio).getSomatorio(indiceCaracteristica+1);
                  for (int i = 0; i < array2[0].length; i++) {
                        String columns = Integer.toString(i);
                        ((DefaultCategoryDataset) dataset).addValue(

      array2[indiceCaracteristica][i]/somatorio, "Característica",
                                    columns);
                  }
            }
            createImage();
      }

      public static void main(String[] args) {
            final NeuronGraphic ng = new NeuronGraphic(new Frame(),
                        "Grafico de Neuronio");
            ng.show();
            ng.dialog.addWindowListener(new WindowListener() {

                  public void windowActivated(WindowEvent e) {

                  }

                  public void windowClosed(WindowEvent e) {
                        System.exit(0);

                  }

                  public void windowClosing(WindowEvent e) {

                  }

                  public void windowDeactivated(WindowEvent e) {

                  }

                  public void windowDeiconified(WindowEvent e) {

                  }

                  public void windowIconified(WindowEvent e) {

                  }

                  public void windowOpened(WindowEvent e) {
                  }

            });
      }
}
```

1.8.5 StatisticGraphic.Java

```
package visao.gui.telas.graficos.classes;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;
import controle.main.App;

public class StatisticGraphic {

	private Thinlet thinlet;

	private ThinletDialog dialog;

	private Object lblGrafico;

	private JFreeChart freeChart = null;

	private CategoryDataset dataset = new DefaultCategoryDataset();

	private Timer timer = new Timer(true);

	private TimerTask task = new AtualizaGrafico();

	public StatisticGraphic(Dialog owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public StatisticGraphic(Dialog owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
	}

	public StatisticGraphic(Frame owner) {
		this.dialog = new ThinletDialog(owner, false);
		this.init();
	}

	public StatisticGraphic(Frame owner, String title) {
		this.dialog = new ThinletDialog(owner, title, false);
		this.init();
```

```
	}

	private void init() {
		this.thinlet = new AntiAliasedThinlet();
		Object panel = null;

		try {
			panel = thinlet.parse(

	"/visao/gui/telas/graficos/StatisticGraphic.xml", this);
		} catch (IOException e1) {
			e1.printStackTrace();
		}

		thinlet.add(panel);
		dialog.setContent(thinlet);
		createChart();
		timer.scheduleAtFixedRate(task, 1000, 1000);
	}

	private void createImage() {

		int height = dialog.getHeight() - 30;
		if (height <= 0)
			height = 1;
		synchronized (App.getLockGraphicStatistic()) {

		thinlet.setIcon(lblGrafico, "icon", (Image) freeChart
				.createBufferedImage(thinlet.getWidth() - 10,
height));
	}
	}
	public void show() {
		dialog.pack();
		dialog.setSize(800, 400);
		dialog.setAlwaysOnTop(true);
		dialog.setLocationRelativeTo(dialog.getOwner());
		dialog.setVisible(true);
		createImage();
	}

	public void initParameters(Object labelImagem) {
		this.lblGrafico = labelImagem;
	}

	private void createChart() {
		dataset = App.getDatasetpool().getDatasetEstatistica();
	  synchronized (App.getLockGraphicStatistic()) {

		// create the chart...
		freeChart = ChartFactory.createLineChart("Evolução de Erros",
// chart
				// title
				"Época", // domain axis label
				"Porcentagem de Acerto", // range axis label
				dataset, // data
				PlotOrientation.VERTICAL, // orientation
```

```
                        true, // include legend
                        true, // tooltips
                        false // urls
                        );
            }

      }

      public static void main(String[] args) {
            final StatisticGraphic sg = new StatisticGraphic(new Frame(),
                        "Gráfico de Estatísticas");
            sg.show();
            sg.dialog.addWindowListener(new WindowListener() {

                  public void windowActivated(WindowEvent e) {

                  }

                  public void windowClosed(WindowEvent e) {
                        System.exit(0);

                  }

                  public void windowClosing(WindowEvent e) {

                  }

                  public void windowDeactivated(WindowEvent e) {

                  }

                  public void windowDeiconified(WindowEvent e) {

                  }

                  public void windowIconified(WindowEvent e) {

                  }

                  public void windowOpened(WindowEvent e) {

                  }

            });
      }

      private class AtualizaGrafico extends TimerTask {

            public void run() {
                  createChart();
                  createImage();
            }

      }
}
```

1.9 PACKAGE VISAO.GUI.TELAS.UTILS;

1.9.1 CarregarTabela.Java

```
package visao.gui.telas.utils;

import java.io.IOException;
import java.lang.reflect.Array;
import java.util.Arrays;

import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;


public class CarregarTabela extends AntiAliasedThinlet {

	private static final long serialVersionUID = 1L;

	private int pageSize = 200;

	private int page = 1;

	private String[][] matrix;

	private int maxPages;

	private Object table, labelNowPage, labelMaxPage;

	private Thinlet thinlet;

	public CarregarTabela(Thinlet thinlet, Object tabela,
			Object labelPaginaAtual, Object labelPaginaMax) throws
IOException {
			this.labelMaxPage = labelPaginaMax;
			this.labelNowPage = labelPaginaAtual;
			this.table = tabela;
			this.thinlet = thinlet;
	}

	public void carregaDados(String[][] vetor) {
			addRows(vetor);
	}

	public void addRow(final String... strings) {
			Object header = Thinlet.create("header");

			int qtdCaracteristicas = 0;
			for (int coluna = 0; coluna < strings.length - 1; coluna++) {
				qtdCaracteristicas ++;
				Object column = Thinlet.create("column");
				String carac = "Caracteristica ";
				carac += coluna + 1;
				thinlet.setString(column, "text", carac);
				thinlet.setInteger(column, "width", 150);
				thinlet.add(header, column);
			}
```

```
      thinlet.setString(thinlet.find("caracTsN"),"text",Integer.toString(
qtdCaracteristicas)) ;
            Object column = Thinlet.create("column");
            thinlet.setString(column, "text", "Classe");
            thinlet.add(header, column);
            thinlet.add(table, header);

            String[] lin = strings;
            int col = lin.length;

            System.out.println(col);
            Object row = Thinlet.create("row");
            Object cell;


            for (String s : strings) {
                  cell = Thinlet.create("cell");
                  thinlet.setString(cell, "text", s);
                  thinlet.add(row, cell);
            thinlet.add(table, row);

          }
         }

      public void addRows(final String[][] table) {
            if (matrix ==  null) {
                  this.matrix = table;
            }
            else {
                  int tamanho = matrix.length + table.length;
                  String[][] novaMatrix = new
String[tamanho][matrix[0].length];
                  int fim1 = matrix.length;
                  int fim2 = matrix[0].length;
                  int k = 0;
                  for (int i = 0; i < fim1; i++ ) {
                        for (int j = 0; j < fim2; j++ ) {
                              novaMatrix[k][j] = matrix[i][j];
                        }
                        k++;
                  }
                  fim1 = table.length;
                  fim2 = table[0].length;
                  for (int i = 0; i < fim1; i++ ) {
                        for (int j = 0; j < fim2; j++ ) {
                              novaMatrix[k][j] = table[i][j];
                        }
                        k++;
                  }
                  this.matrix = novaMatrix;
            }
            this.page = 1;
            calculateMaxPages();
            draw();
      }
```

```
	public void clearTable() {
		removeAll(table);
		matrix = null;
	}

	private void calculateMaxPages() {
		this.maxPages = (this.matrix.length / this.pageSize);
		if ((this.matrix.length % this.pageSize) > 0)
			this.maxPages++;
	}

	public void draw() {
		thinlet
				.setString(labelMaxPage, "text", Integer
						.toString(this.maxPages));
		thinlet.setString(labelNowPage, "text",
Integer.toString(this.page));
		int i = (this.page - 1) * this.pageSize;
		int fim1 = (this.page) * this.pageSize;
		int fim2 = this.matrix.length;
		removeAll(table);
		int qtdPadroes = 0;
		for (; i < fim1 && i < fim2; i++) {
			addRow(this.matrix[i]);
			qtdPadroes++;
		}

	thinlet.setString(thinlet.find("padroesTsN"),"text",Integer.toStrin
g(qtdPadroes)) ;
	}

	public int getPage() {
		return this.page;
	}

	public void setPage(int page) {
		this.page = page;
		draw();
	}

	public int getPageSize() {
		return this.pageSize;
	}

	public void setPageSize(int pageSize) {
		this.pageSize = pageSize;
		calculateMaxPages();
		draw();
	}

	public void nextPage() {
		if (page < maxPages)
			setPage(++this.page);
	}

	public void beforePage(){
		if (page > 1)
```

```
                setPage(--this.page);
        }
}
```

1.9.2 DataView.Java

```
package visao.gui.telas.utils;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;

import thinlet.Thinlet;
import thinletcommons.AntiAliasedThinlet;
import thinletcommons.ThinletDialog;

public class DataView {

      private Thinlet thinlet;
      private ThinletDialog dialog;
      private Object table;
      private Object titleLabel;
      private Object numPatterns;
      private Object numClasses;
      private Object labelPage, labelMaxPage;
      private int pageSize = 200;
      private int page = 1;
      private String[][] matrix;
      private int maxPages;

      public DataView(Dialog owner) {
            this.dialog = new ThinletDialog(owner,false);
            this.init();
      }

      public DataView(Dialog owner, String title) {
            this.dialog = new ThinletDialog(owner,title,false);
            this.init();
      }

      public DataView(Frame owner) {
            this.dialog = new ThinletDialog(owner,false);
            this.init();
      }

      public DataView(Frame owner, String title) {
            this.dialog = new ThinletDialog(owner,title,false);
            this.init();
      }

      private void init() {
            this.thinlet = new AntiAliasedThinlet();
            Object panel = null;
            try {
                  panel =
thinlet.parse("/visao/gui/telas/utils/dataview.xml", this);
```

```
            } catch (IOException e) {
                  e.printStackTrace();

            }
        thinlet.add(panel);
        dialog.setContent(thinlet);
      }

      public void show() {
            dialog.pack();
            dialog.setAlwaysOnTop(true);
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);
        labelMaxPage = thinlet.find("pageMax");
        labelPage = thinlet.find("page");
        if (this.matrix != null) draw();
      }

      public void initParameters(Object table, Object titleLabel, Object
numPatterns, Object numClasses) {
            this.table = table;
            this.titleLabel = titleLabel;
            this.numPatterns = numPatterns;
            this.numClasses = numClasses;
      }

      public void setTitleLabel(String text) {
            thinlet.setString(this.titleLabel, "text", text);
      }

      public void setNumClasses(int num) {
            thinlet.setString(this.numClasses, "text",
Integer.toString(num));
      }

      public void updatePatternsCount() {
            thinlet.setString(this.numPatterns, "text",
Integer.toString(thinlet.getCount(table)));
      }

      public void addRow(final String... strings) {
            Object header = Thinlet.create("header");
                int qtdClasses = 0;

                for (int coluna = 0; coluna < strings.length -1 ;
coluna++) {
                  qtdClasses ++;
                  Object column = Thinlet.create("column");
                        String carac = "Caracteristica ";
                        carac += coluna+1;
                        thinlet.setString(column,"text", carac );
                        thinlet.setInteger(column,"width",150);
                        thinlet.add(header, column);
                  }

thinlet.setString(thinlet.find("caracTrN"),"text",Integer.toString(qtdCla
sses)) ;
```

```
                Object column = Thinlet.create("column");
                  thinlet.setString(column,"text", "Classe" );
                  thinlet.add(header, column);
                  thinlet.add(table, header);
                Object row = Thinlet.create("row");
               Object cell;
                  for (String s : strings ) {
                       cell = Thinlet.create("cell");
                       thinlet.setString(cell,"text", s);
                       thinlet.add(row, cell);
                  }
                  thinlet.add(table, row);
            }

      public void addRows(final String[][] table) {
            this.matrix = table;
            this.page = 1;
            calculateMaxPages();
            draw();
      }

      public void draw() {
            thinlet.setString(labelMaxPage, "text",
Integer.toString(this.maxPages));
        thinlet.setString(labelPage, "text",
Integer.toString(this.page));
            int i = (this.page-1) * this.pageSize;
            int fim1 = (this.page) * this.pageSize;
            int fim2 = this.matrix.length;
            this.clearTable();
            int qtdPadroes = 0;
            for(; i < fim1 && i < fim2;i++) {
                  addRow(this.matrix[i]);
                  qtdPadroes++;
            }

      thinlet.setString(thinlet.find("padroesTrN"),"text",Integer.toStrin
g(qtdPadroes)) ;
      }

      public void clearTable() {
            thinlet.removeAll(table);
      }
      private static final long serialVersionUID = 5195907069388087662L;

      public static void main(String[] args) {
            DataView dw = new DataView(new Frame(),"Teste");
            dw.addRow(new String[]{"re","123","1we1123", "as"});
            dw.addRow("123","1sadsad23","1we1123", "a333");
            dw.addRows(new String[][]{{"re","123","1we1123",
"as"},{"123","1sadsad23","1we1123", "a333"}});
            dw.setTitleLabel("Um Teste");
            dw.setNumClasses(3);
            dw.show();
            dw.addRows(new String[][]{{"re","123","1we1123",
"as"},{"123","1sadsad23","1we1123", "a333"}});
            dw.updatePatternsCount();
```

```
            while (dw.dialog.isShowing()) {}
            dw.dialog.dispose();

            System.exit(0);

      }

      public int getPage() {
            return this.page;
      }

      public void setPage(int page) {
            this.page = page;
            draw();
      }

      public int getPageSize() {
            return this.pageSize;
      }

      public void setPageSize(int pageSize) {
            this.pageSize = pageSize;
            calculateMaxPages();
            draw();
      }

      public void nextPage() {
            if (page < maxPages)
                  setPage(++this.page);
      }

      public void beforePage(){
            if (page > 1)
                  setPage(--this.page);
      }

      private void calculateMaxPages() {
            this.maxPages = (this.matrix.length / this.pageSize);
            if ((this.matrix.length % this.pageSize) > 0)
this.maxPages++;
      }

}
```

1.9.3 DataView.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--DOCTYPE panel SYSTEM "../telas/resources/dtd/thinlet.dtd" >-->
<panel border="true" bottom="4" columns="14" gap="4" left="4" right="4"
scrollable="true" top="4"    init="initParameters(tabela, tituloLabel,
padroesTrN, classeTrN)">
                        <label weightx="1" name="tituloLabel" text="
"/>
                        <label text="  "/>
                        <label text="N° de Padrões da Página: "/>
```

```
                        <label name="padroesTrN" text="             "/>
                        <label text="   "/>
                        <label text="N° de Caracteristicas : "/>
                        <label name="caracTrN" text="     "/>
                        <label text="  "/>
                        <label text="Pagina: "/>
                        <label name="page" text=" "/>
                        <label text="/"/>
                        <label name="pageMax" text=" "/>
                        <button enabled="true"
icon="/visao/gui/telas/icones/anteriorPagina.png" text="Anterior"
name="pagAnt" action="beforePage()"/>
                        <button enabled="true"
icon="/visao/gui/telas/icones/proximaPagina.png" text="Proxima"
name="proxPag" action="nextPage()"/>
                        <panel colspan="14" columns="10" rowspan="20"
scrollable="true" weightx="1" weighty="1">
                            <table name="tabela"  colspan="10"
weightx="1" weighty="1">
                            </table>
                        </panel>
</panel>
```

1.10 PACKAGE VISAO.GUI.WIZARD;

1.10.1 Wizard.Java

```
package visao.gui.wizard;

import java.awt.Dialog;
import java.awt.Frame;
import java.io.IOException;
import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

import thinletcommons.AntiAliasedThinlet;
import thinletcommons.MessageDialog;
import thinletcommons.ThinletDialog;
import controle.eventos.EvtAbrirArquivo;
import controle.eventos.EvtLimparConjunto;
import controle.main.App;
import thinlet.Thinlet;
import visao.gui.telas.classes.viewHelp;
import visao.gui.telas.utils.CarregarTabela;
import visao.gui.telas.utils.DataView;

public class Wizard extends AntiAliasedThinlet{
      private Frame frame;
      private Thinlet thinlet;
      private Thinlet thinletExtension;
      private ThinletDialog dialog;
      private ThinletDialog novoConjunto;
      private ThinletDialog explExtensao;
      private viewHelp vHelp ;
      private String[][] array = null;
```

```
        private String[][] arrayTeste = null;
        private CarregarTabela carregarConjunto = null;
        private CarregarTabela carregarTeste = null;
        private EvtAbrirArquivo evtAbrirArquivo = new EvtAbrirArquivo();
        private int index = 0;
        private Object intro = null;
        private Object passo1 = null;
        private Object passo2 = null;
        private Object passo3 = null;
        private Object passo4 = null;
        private Object passo5 = null;
        private Object passoConjTeste = null;
        private Object cargaEpd = null;
        private Object cargaXls = null;
        private Object cargaTxt = null;
        private Object cargaDat = null;

        //abrir wizard apos chamada do easyfan
         public void AbrirWizard(Frame ower, String title) {
                    frame = ower;
                    this.dialog = new ThinletDialog(ower,title,false);
                    String Intro = null;
                    this.initProximasPaginas(Intro);
         }

         // init das proximas paginas
         private void initProximasPaginas(String Passo) {
                   vHelp = new viewHelp(dialog);
                     String passo = null;
                   if (Passo != null)
                           passo = Passo;
                     this.thinlet = new AntiAliasedThinlet();
                     try {
                           if (passo != null){
                              if (passo.equals("Introducao")){
                                   if (passo1 == null){
                                     passo1 =
thinlet.parse("/visao/gui/wizard/passo1.xml", this);
                                   }
                                     thinlet.add(passo1);
                                    dialog.setContent(thinlet);
                                    dialog.setModal(true);
                              }else
                              if (passo.equals("Passo1")){
                                    if (passo2 == null){
                                           passo2 =
thinlet.parse("/visao/gui/wizard/passo2.xml", this);
                                    }
                                           thinlet.add(passo2);
                                           dialog.setContent(thinlet);
                                         dialog.setModal(true);
                     }else
                           if (passo.equals("Passo2")){
                                 if (passo3 == null){
                                           passo3 =
thinlet.parse("/visao/gui/wizard/passo3.xml", this);
                                 }
```

```
                                        thinlet.add(passo3);
                                    dialog.setContent(thinlet);
                                    dialog.setModal(true);
                              }else
                              if (passo.equals("Passo3")){
                                    if (passo4 == null){
                                          passo4 =
thinlet.parse("/visao/gui/wizard/passo4.xml", this);
                                    }
                                          thinlet.add(passo4);
                                    dialog.setContent(thinlet);
                                    dialog.setModal(true);

                              }else
                              if (passo.equals("Passo4")){
                                    if (passo5 == null){
                                          passo5 =
thinlet.parse("/visao/gui/wizard/passo5.xml", this);
                                    }
                                          thinlet.add(passo5);
                                    dialog.setContent(thinlet);
                                    dialog.setModal(true);
                              }

                             }else{
                                   Object panel =
thinlet.parse("/visao/gui/wizard/introducaoWizard.xml", this);
                                   thinlet.add(panel);
                               dialog.setContent(thinlet);
                               dialog.setModal(true);
                             }
                       } catch (IOException e) {
                             e.printStackTrace();
                       }

                }

        //navegacao das paginas anteriores
        private void initAnterior(String Passo) {
                    String passo = null;
                  if (Passo != null)
                          passo = Passo;
                    this.thinlet = new AntiAliasedThinlet();
                    try {
                          if (passo != null){
                             if (passo.equals("Passo1")){
                                    intro =
thinlet.parse("/visao/gui/wizard/introducaoWizard.xml", this);
                                      thinlet.add(intro);
                                    dialog.setContent(thinlet);
                                    dialog.setModal(true);
                              }else
                              if (passo.equals("Passo2")){
                                   if (passo2 == null){
                                         passo1 =
thinlet.parse("/visao/gui/wizard/passo1.xml", this);
                                    }
```

```
                                        thinlet.add(passo1);
                                        dialog.setContent(thinlet);
                                        dialog.setModal(true);
                                }else
                                if (passo.equals("Passo3")){
                                     if (passo3 == null){
                                              passo2 =
thinlet.parse("/visao/gui/wizard/passo2.xml", this);
                                     }
                                              thinlet.add(passo2);
                                              dialog.setContent(thinlet);
                                              dialog.setModal(true);
                                }else
                                if (passo.equals("Passo4")){
                                     if (passo4 == null){
                                              passo4 =
thinlet.parse("/visao/gui/wizard/passo3.xml", this);
                                     }
                                              thinlet.add(passo3);
                                              dialog.setContent(thinlet);
                                              dialog.setModal(true);
                                }else
                                if (passo.equals("Passo5")){
                                   if (passo5 == null){
                                              passo4 =
thinlet.parse("/visao/gui/wizard/passo4.xml", this);
                                   }
                                              thinlet.add(passo4);
                                              dialog.setContent(thinlet);
                                              dialog.setModal(true);
                                }
                           }
                     } catch (IOException e) {
                          e.printStackTrace();
                     }
          }

     public void show() {
                dialog.pack();
                dialog.setSize(830,500);
                dialog.setAlwaysOnTop(false);
          dialog.setLocationRelativeTo(dialog.getOwner());
          dialog.setVisible(true);
          }


   public void paginaProxima(Object passo){
              String ps  = this.getString(passo, "text");
                initProximasPaginas(ps);
                if(ps.equalsIgnoreCase("Introducao")){
                      show();
                }else
                if(ps.equalsIgnoreCase("Passo1")){
                      show();
                }else
                if(ps.equalsIgnoreCase("Passo2")){
                      show();
```

```
                }else
                if(ps.equalsIgnoreCase("Passo3")){
                      show();
                }
                if(ps.equalsIgnoreCase("Passo4")){
                      if (getIndex() == 0){
                            try {
                                  Object Screen =
App.getTelaEasyFAN().getSreen();

      App.getTelaEasyFAN().mudaAba(thinlet.find(Screen,"geral"),thinlet.f
ind(Screen,"btAbaTreino"));

      App.getGerenciadorPadroes().setConjuntoTeste(App.getGerenciadorPadr
oes().getConjuntoTreinamento());
                                  if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0)

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTreinamento();

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTeste();
                            } catch (Exception e) {
                                  e.printStackTrace();
                            }
                      }else if (getIndex() == 1){
                            Object Screen =
App.getTelaEasyFAN().getSreen();

App.getTelaEasyFAN().mudaAba(thinlet.find(Screen,"geral"),thinlet.find(Sc
reen,"btAbaTreino"));
                            Object porcTreino =
thinlet.find(passo4,"txtPorcentagemTreino");
                            int pTreino =
Integer.parseInt(getString(porcTreino,"text"));
                            try {

App.getGerenciadorPadroes().setConjuntoTreinamentoComPorcentagemDe(pTrein
o);

      App.getGerenciadorPadroes().setConjuntoTeste(App.getGerenciadorPadr
oes().getConjuntoTreinamento());
                                  if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0)

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());
                            } catch (Exception e) {
                                  e.printStackTrace();
                            }

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTreinamento();

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTeste();
                      }
```

```
                        else if (getIndex() == 2 ){
                              try{
                                    Object Screen =
App.getTelaEasyFAN().getSreen();

      App.getTelaEasyFAN().mudaAba(thinlet.find(Screen,"geral"),thinlet.f
ind(Screen,"btAbaTreino"));

      App.getGerenciadorPadroes().getConjuntoTesteRede().addAll(App.getGe
renciadorPadroes().getConjuntoTesteRede());

      App.getGerenciadorPadroes().setConjuntoTeste(App.getGerenciadorPadr
oes().getConjuntoTeste());
                                    if
(App.getGerenciadorPadroes().getConjuntoTreinamentoRede().size() > 0)

      App.getMonitorFAN().setClasses(App.getGerenciadorPadroes().getClass
esConjuntoTreinamento());

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTreinamento();

      App.getTelaEasyFAN().configurarHabilitacaoConjuntoTeste();
                              }catch (Exception e) {
                                    e.printStackTrace();
                              }
                        }
                        show();

                  }

       }

       public void paginaAnterior(Object passo){
                String ps  = this.getString(passo, "text");
                  initAnterior(ps);
                  if(ps.equalsIgnoreCase("Passo1")){
                              show();
                  }else
                  if(ps.equalsIgnoreCase("Passo2")){
                              show();
                  }else
                  if(ps.equalsIgnoreCase("Passo3")){
                              show();
                  }else
                  if(ps.equalsIgnoreCase("Passo4")){
                              show();
                }else
                  if(ps.equalsIgnoreCase("Passo5")){
                              show();
                  }
       }

     //     adicionar arquivo passo 2 utilizando o filechooser
          public void adicionarArquivo(Object table, Object button1,Object
pagAtual, Object maxPage,Object button2,Object button3,Object
btProximoPasso){
                array = evtAbrirArquivo.abrirTreinamento();
```

```
                if (array != null){
                      try {
                      if (carregarConjunto == null)
                            carregarConjunto = new
CarregarTabela(this.thinlet,table,pagAtual,maxPage);
                      carregarConjunto.carregaDados(array);
                          this.repaint();
                      } catch (Exception e) {
                            e.printStackTrace();
                      }
                      setBoolean(button1, "enabled", true);
                      setBoolean(button2, "enabled", true);
                      setBoolean(button3, "enabled", true);
                      setBoolean(btProximoPasso, "enabled", true);
                }
                thinlet.repaint();
                dialog.repaint();
          }

        //limpar conjunto de arquivos
         public void limparDados(Object table, Object button1, Object
label1,
                      Object label2, Object button2, Object button3,
Object btProximoPasso, Object lbNumPadroes,Object lbNumCarac) {
                this.array = null;
                carregarConjunto.clearTable();
                EvtLimparConjunto.limparConjuntoTreinamento();
                setBoolean(button1, "enabled", false);
                setString(lbNumCarac, "text","");
                setString(lbNumPadroes, "text","");
                setString(label1, "text","");
                setString(label2, "text","");
                setBoolean(button2, "enabled", false);
                setBoolean(button3, "enabled", false);
                setBoolean(btProximoPasso, "enabled", false);
                thinlet.repaint();
                dialog.repaint();
            }

      //limpar conjunto de teste
         public void limparDadosTeste(Object table, Object button1,
Object label1,
                      Object label2, Object button2, Object button3,
Object lbNumPadroes,Object lbNumCarac) {
                this.arrayTeste = null;
                Object btProximoPasso =
thinlet.find(passo4,"btProximoPasso");
                DataView dv = new DataView(new Frame());
                if(carregarTeste != null){
                      carregarTeste.clearTable();
                }
                EvtLimparConjunto.limparConjuntoTeste();
                setBoolean(button1, "enabled", false);
                setString(label1, "text","");
                setString(label2, "text","");
                setString(lbNumCarac, "text","");
                setString(lbNumPadroes, "text","");
```

```
                setBoolean(button2, "enabled", false);
                setBoolean(button3, "enabled", false);
                setBoolean(btProximoPasso,"enabled",false);
                thinlet.repaint();
                dialog.repaint();
            }

          //proxima pagina conjunto de arquivos
          public void nextPage(){
                carregarConjunto.nextPage();

          }

         //pagina anterior conjunto de arquivos
          public void beforePage(){
                carregarConjunto.beforePage();
          }

       // adicionar novo conjunto para teste passo 4
        public void adicionarNovoConjuntoPasso4(){
                initNovoConjuntoPasso4();
                showNovoConjuntoPasso4();
        }

          private void initNovoConjuntoPasso4() {
                  this.thinlet = new AntiAliasedThinlet();
                  try {
                        if (passoConjTeste == null){
                           passoConjTeste =
thinlet.parse("/visao/gui/wizard/passo4AdicionarConjunto.xml", this);
                        }
                  } catch (IOException e) {
                        new MessageDialog(dialog, "Erro ao abrir a
janela", "Erro ao tentar abrir a configuracao de tempera").show();
                        e.printStackTrace();

                  }
              thinlet.add(passoConjTeste);
              this.novoConjunto = new ThinletDialog(new Frame(),"Novo
Conjunto",false);
              novoConjunto.setContent(thinlet);
       }

       public void showNovoConjuntoPasso4() {
                  novoConjunto.pack();
                  novoConjunto.setSize(723,425);
                  novoConjunto.setAlwaysOnTop(false);
               novoConjunto.setLocationRelativeTo(dialog.getOwner());
               novoConjunto.setModal(true);
               novoConjunto.setVisible(true);
            }

    // adicionar novo conunto de testes passo 4
       public void adicionarArquivoTeste(Object table, Object
button1,Object pagAtual, Object maxPage,Object button2,Object button3){
              arrayTeste = evtAbrirArquivo.abrirTeste();
```

```
              Object btProximoPasso =
thinlet.find(passo4,"btProximoPasso");
                  if (arrayTeste != null){
                        try {
                        if (carregarTeste == null)
                              carregarTeste = new
CarregarTabela(this.thinlet,table,pagAtual,maxPage);
                        carregarTeste.carregaDados(arrayTeste);
                            this.repaint();
                        } catch (Exception e) {
                              e.printStackTrace();
                        }
                        setBoolean(button1, "enabled", true);
                        setBoolean(button2, "enabled", true);
                        setBoolean(button3, "enabled", true);
                        setBoolean(btProximoPasso,"enabled",true);
                  }
                  thinlet.repaint();
                  dialog.repaint();
          }

        //proxima pagina conjunto de teste
          public void nextPageTeste(){
                carregarTeste.nextPage();

          }

         //pagina anterior conjunto de teste
          public void beforePageTeste(){
                carregarTeste.beforePage();
          }

      //confirmar novo conjunto para teste passo 4
      public void confirmarNovoConjunto(){
                novoConjunto.setVisible(false);
                thinlet.repaint();
        }

      //cancelar novo conjunto para teste passo 4
      public void cancelarNovoConjunto(Object tabelaWZ,Object
btLimparConjunto,Object pagAtual,Object pagMax,Object
btPaginaAnterior,Object btProximaPagina,Object lbCaracteristicas,Object
lbPadroes){

limparDadosTeste(tabelaWZ,btLimparConjunto,pagAtual,pagMax,btPaginaAnteri
or,btProximaPagina,lbCaracteristicas,lbPadroes);
                novoConjunto.setVisible(false);
                thinlet.repaint();
      }

      //chamar help
      public void chamarHelp(){
              vHelp.visualizarHelp();
      }

   // validacao radiogroup passo4
```

```
   public void selectedRadio(Object rGroup,Object Lb1, Object Txt1,
Object Lb2, Object Lb3,Object Txt2, Object Lb4, Object Bt1,Object
btProximoPasso){
         String Selected = this.getString(rGroup,"name");
       if (Selected.equalsIgnoreCase("cbPorcentagem")){
          String porcTreino = getString(Txt1, "text");
          String porcTeste = getString(Txt2, "text");
                 if (porcTreino.equals("") || porcTeste.equals("")){
                       setBoolean(btProximoPasso,"enabled",false);
                 }else{
                       setBoolean(btProximoPasso,"enabled",true);
                 }
          setBoolean(Lb1,"enabled" ,true);
                 setBoolean(Txt1,"enabled" ,true);
                 setBoolean(Lb2,"enabled" ,true);
                 setBoolean(Lb3,"enabled" ,true);
                 setBoolean(Txt2,"enabled" ,true);
                 setBoolean(Lb4,"enabled" ,true);
                 setBoolean(Bt1,"enabled" ,false);
                 setIndex(1);
           } else if (Selected.equalsIgnoreCase("cbMesmoConjunto")){
                 setBoolean(btProximoPasso,"enabled",true);
                 setBoolean(Lb1,"enabled" ,false);
                 setBoolean(Txt1,"enabled" ,false);
                 setBoolean(Lb2,"enabled" ,false);
                 setBoolean(Lb3,"enabled" ,false);
                 setBoolean(Txt2,"enabled" ,false);
                 setBoolean(Lb4,"enabled" ,false);
                 setBoolean(Bt1,"enabled" ,false);
                 setIndex(0);
        }else if (Selected.equalsIgnoreCase("cbNovoConjunto")){
                thinlet.repaint();
                if (this.arrayTeste == null){
                  setBoolean(btProximoPasso,"enabled",false);
                }else{
                  setBoolean(btProximoPasso,"enabled",true);
                }
                  setBoolean(Lb1,"enabled" ,false);
                  setBoolean(Txt1,"enabled" ,false);
                  setBoolean(Lb2,"enabled" ,false);
                  setBoolean(Lb3,"enabled" ,false);
                  setBoolean(Txt2,"enabled" ,false);
                  setBoolean(Lb4,"enabled" ,false);
                  setBoolean(Bt1,"enabled" ,true);
                  setIndex(2);
       }
    thinlet.repaint();
    dialog.repaint();
   }

   public int getIndex() {
      return index;
   }

   public void setIndex(int index) {
         this.index = index;
   }
```

```
   // iniciar treinamento
   public void iniciarTreinamento(){
        dialog.setVisible(false);
        App.getTelaEasyFAN().clickBotaoTreinarParar();

   }

   public void calculaPorcentagemTreino(Object textTreino, Object
textTeste, Object btProximaPagina) {
            String pTreino = getString(textTreino, "text");
            String pTeste = getString(textTeste, "text");
            int porcentagem = 0;
            if (pTreino != null && !pTreino.equals("")) {
                     try {
                              porcentagem = Integer.parseInt(pTreino);
                              if (porcentagem > 99) {
                                    pTreino = pTreino.substring(0,
pTreino.length() - 1);
                              }
                                    pTeste = Integer.toString(100 -
Integer.parseInt(pTreino));
                                    setString(textTreino, "text",
pTreino);
                                    setString(textTeste, "text", pTeste);

      setBoolean(btProximaPagina,"enabled",true);
                        } catch (NumberFormatException e) {
                              StringBuilder sb = new
StringBuilder(pTreino);
                              int j = 0;
                              while(j < sb.length()) {
                                    try {
                                          int z =
Integer.parseInt(Character.toString(sb.charAt(j)));
                                          j++;
                                    }
                                    catch (NumberFormatException ex) {
                                          sb.deleteCharAt(j);
                                    }
                              }
                              setString(textTreino,"text",sb.toString());
                              pTreino = pTreino.substring(0,
pTreino.length() - 1);
                              if (pTreino != null && !pTreino.equals("")
){
                                    pTeste = Integer.toString(100 -
Integer.parseInt(sb.toString()));
                              }
                  }
            }else{
                              setString(textTreino, "text", "");
                              setString(textTeste, "text", "");
                              setBoolean(btProximaPagina,"enabled",false);

      }
        thinlet.repaint();
```

```
    }

   public void calculaPorcentagemTeste(Object textTreino, Object
textTeste, Object btProximaPagina) {
            String pTreino = getString(textTreino, "text");
            String pTeste = getString(textTeste, "text");
            int porcentagem = 0;
            if (pTeste != null && !pTeste.equals("")) {
                  try {
                        porcentagem = Integer.parseInt(pTeste);
                        if (porcentagem > 99) {
                              pTeste = pTeste.substring(0, pTeste.length()
- 1);
                        }
                              pTreino = Integer.toString(100 -
Integer.parseInt(pTeste));
                              setString(textTreino, "text", pTreino);
                              setString(textTeste, "text", pTeste);
                              setBoolean(btProximaPagina,"enabled",true);
                  } catch (NumberFormatException e) {
                        StringBuilder sb = new StringBuilder(pTeste);
                        int j = 0;
                        while(j < sb.length()) {
                              try {
                                    int z =
Integer.parseInt(Character.toString(sb.charAt(j)));
                                    j++;
                              }
                              catch (NumberFormatException ex) {
                                    sb.deleteCharAt(j);
                              }
                        }
                        setString(textTeste,"text",sb.toString());
                        pTeste = pTeste.substring(0, pTeste.length() - 1);
                        if (pTeste != null && !pTeste.equals("")){
                              pTreino = Integer.toString(100 -
Integer.parseInt(sb.toString()));
                        }
                  }
            }else{
                  setString(textTreino, "text", "");
                  setString(textTeste, "text", "");
                  setBoolean(btProximaPagina,"enabled",false);
            }
       thinlet.repaint();
   }

   // cancelar wizard
   public void close(){
         Object screenEasyFan  = App.getTelaEasyFAN().getSreen();
         Object btLimparTesteEFAN =
thinlet.find(screenEasyFan,"btDescarregarTs");
         App.getTelaEasyFAN().limparDados(btLimparTesteEFAN);
         App.getTelaEasyFAN().descarregarArquivo();
         if (array != null){
               array = null;
               carregarConjunto.clearTable();
```

```
                EvtLimparConjunto.limparConjuntoTreinamento();
          }
          if (arrayTeste != null){
                arrayTeste = null;
                carregarTeste.clearTable();
                EvtLimparConjunto.limparConjuntoTeste();
          }
             intro = null;
             passo1 = null;
             passo2 = null;
             passo3 = null;
             passo4 = null;
             passo5 = null;
             passoConjTeste = null;
             thinlet.repaint();
             dialog.setVisible(false);
     }

    // fechar wizard
    public void fechar(){
             dialog.setVisible(false);
    }

    public String[][] getArrayTeste(){
          return arrayTeste;
    }

    public String[][] getArrayTreino(){
          return array;
    }

       public void showExplicacaoArquivos(int x, int y) {
             explExtensao.pack();
             explExtensao.setSize(x,y);
             explExtensao.setAlwaysOnTop(false);
             explExtensao.setLocationRelativeTo(dialog.getOwner());
             explExtensao.setModal(true);
             explExtensao.setVisible(true);
       }

       public void cancelarExplExtensao(){
             explExtensao.setVisible(false);
             explExtensao = null;
             dialog.repaint();
       }

    public void explicacaoEpd(){
          this.thinletExtension = new AntiAliasedThinlet();
             try {
                   if (cargaEpd == null){
                      cargaEpd =
thinletExtension.parse("/visao/gui/wizard/cargaEpd.xml", this);
                   }
             } catch (IOException e) {
                   new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuração de tempera").show();
                   e.printStackTrace();
```

```
            }
      thinletExtension.add(cargaEpd);
      this.explExtensao = new ThinletDialog(new Frame(),"Arquivos
Epd",false);
      explExtensao.setContent(thinletExtension);
        showExplicacaoArquivos(610,240);
   }

   public void explicacaoDat(){
         this.thinletExtension = new AntiAliasedThinlet();
            try {
                  if (cargaDat == null){
                     cargaDat =
thinletExtension.parse("/visao/gui/wizard/cargaDat.xml", this);
                  }
            } catch (IOException e) {
                  new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuração de tempera ").show();
                  e.printStackTrace();

            }
      thinletExtension.add(cargaDat);
      this.explExtensao = new ThinletDialog(new Frame(),"Arquivos
DAT",false);
      explExtensao.setContent(thinletExtension);
        showExplicacaoArquivos(530,180);
   }

   public void explicacaoTxt(){
         this.thinletExtension = new AntiAliasedThinlet();
            try {
                  if (cargaTxt == null){
                     cargaTxt =
thinletExtension.parse("/visao/gui/wizard/cargaTxt.xml", this);
                  }
            } catch (IOException e) {
                  new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuração da tempera ").show();
                  e.printStackTrace();

            }
      thinletExtension.add(cargaTxt);
      this.explExtensao = new ThinletDialog(new Frame(),"Arquivos
TXT",false);
      explExtensao.setContent(thinletExtension);
        showExplicacaoArquivos(555,300);
   }

   public void explicacaoXls(){
         this.thinletExtension = new AntiAliasedThinlet();
            try {
                  if (cargaXls == null){
                     cargaXls =
thinletExtension.parse("/visao/gui/wizard/cargaXls.xml", this);
                  }
            } catch (IOException e) {
```

```
                    new MessageDialog(dialog, "Erro ao abrir a janela",
"Erro ao tentar abrir a configuraï¿½ï¿½o de tï¿½mpera").show();
                    e.printStackTrace();

            }
      thinletExtension.add(cargaXls);
      this.explExtensao = new ThinletDialog(new Frame(),"Arquivos
XLS",false);
      explExtensao.setContent(thinletExtension);
        showExplicacaoArquivos(555,360);
   }

   public void msgDivisaoConjuntos(Object tipoDivisao){
         String s = this.getProperty(tipoDivisao, "indice").toString();
        if (s.equalsIgnoreCase("mesmoConjunto")){
             MessageDialog msgDialog = new MessageDialog(dialog,"Mesmo
Conjunto","Selecionando a opção de 'Mesmo Conjunto', o sistema irá fazer
uma duplicação dos arquivos \n escolhidos anteriormente e o mesmo
conjunto será carregado tanto para treino quanto para teste.");
             msgDialog.show();
        }else if (s.equalsIgnoreCase("porcentagem")){
              MessageDialog msgDialog = new
MessageDialog(dialog,"Divisão por Porcentagem","Selecionando a opção
'Divisão por Porcentagem', você pode especificar qual o número
percentual(Quantidade) \n do conjunto de arquivos escolhido anteriormente
que será carregado para treino e teste.");
             msgDialog.show();
        }else if (s.equalsIgnoreCase("novoConjunto")){
              MessageDialog msgDialog = new MessageDialog(dialog,"Novo
Conjunto para Teste","Selecionando a opção 'Novo Conjunto de Teste', o
sistema irá definir o conjunto escolhido anteriormente \n como conjunto
de treino e o usuário terá de escolher um novo arquivo para fazer ´parte
do conjunto de teste.");
             msgDialog.show();
        }
   }
}
```

1.10.2 CargaDat.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" left="5" rowspan="0" width="1">
    <label font="18 bold" text="Carga de Arquivos *.dat"/>
    <label text=" "/>
    <label text="Tipo de arquivo padrão do LabFAN. Os padrões são
separados por lihas e as características"/>
    <label text="são separadas por um espaço duplo."/>
    <label text=" "/>
    <label font="15" text=" "/>
    <panel halign="right">
        <button action = "cancelarExplExtensao()" name="btFechar"
text="Fechar" icon="/visao/gui/wizard/icones/cancelar.png" />
    </panel>
</panel>
```

1.10.3 CargaEpd.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" left="5" rowspan="0" width="1">
    <label font="18 bold" text="Carga de Arquivos *.epd"/>
    <label text=" "/>
    <label text="EPD (EasyFAN Pattern Data) Arquivo de Dados Padrão do
EasyFAN, elaborado na sintaxe de Arquivos XML "/>
    <label text="XML (eXtensible Markup Language) é uma recomendação da
W3C para gerar linguagens de marcação para"/>
    <label text="necessidades especiais."/>
    <label/>
    <label text="Capaz de descrever diversos tipos de dados, seu
propósito principal é a facilidade de compartilhamento"/>
    <label text="de informações."/>
    <label text=" "/>
    <label font="15" text=" "/>
    <panel halign="right">
        <button action="cancelarExplExtensao()"
icon="/visao/gui/wizard/icones/cancelar.png" name="btFechar"
text="Fechar"/>
    </panel>
</panel>
```

1.10.4 CargaTxt.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" left="5" rowspan="0" width="1">
    <label font="18 bold" text="Carga de Arquivos *.txt"/>
    <label text=" "/>
    <label text="Arquivo de texto com pouca formatação, mas pode ser
aberto com qualquer editor de texto."/>
    <label text=" "/>
    <label text="No EasyFAN os arquivos txt podem possuir qualquer tipo
de separador entre as características"/>
    <label text="e os padrões devem ser divididos em linhas."/>
    <label text=" "/>
    <label text="O usuário não precisa necessáriamente carregar todos os
dados do arquivo,o sistema permite"/>
    <label text="que sejam escolhidos quais colunas e linhas que se
deseje carregar e qual  a coluna do "/>
    <label text="arquivo que servirá de classe."/>
    <label/>
   <label text=" "/>
   <label font="15" text=" "/>
    <panel halign="right">
        <button action = "cancelarExplExtensao()" name="btFechar"
text="Fechar" icon="/visao/gui/wizard/icones/cancelar.png"/>
    </panel>
</panel>
```

1.10.5 CargaXls.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" left="5" rowspan="0" width="1">
    <label font="18 bold" text="Carga de Arquivos *.xls"/>
    <label text=" "/>
    <label text="A extensão xls é associada com os arquivos criados no
Microsoft excel."/>
    <label text=" "/>
    <label text="Se você tiver ou estiver usando um arquivo com extensão
xls (por exemplo, pessoa_ts.xls), "/>
    <label text="é provável que ele tenha sido criada no Microsoft
excel."/>
    <label text=" "/>
    <label text="Microsoft Excel é um programa onde você pode criar
planilhas eletrônicas e os dados são "/>
    <label text="dispostos em forma de tabela (linha/coluna) onde podem
ser manipulados"/>
    <label text=" "/>
    <label text="O EasyFAN reconhece os padrões dos arquivos xls
separados por linhas e as características "/>
    <label text="separadas por colunas, mas o usuário pode escolher quais
linhas e colunas que deseje "/>
    <label text="carregar e qual será a coluna que será definida como
coluna da classe."/>
    <label text=" "/>
    <label text=" "/>
    <label font="15" text=" "/>
    <panel halign="right">
        <button action = "cancelarExplExtensao()" name="btFechar"
text="Fechar" icon="/visao/gui/wizard/icones/cancelar.png"/>
    </panel>
</panel>
```

1.10.6 IntroducaoWizard.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="1" gap="4" left="4" right="4" top="4">
        <panel colspan="0" columns="2">
            <panel>
                <label colspan="0"
icon="/visao/gui/wizard/icones/introWizard.png"/>
            </panel>
            <label font="26 bold" name="lbWelcome" text="
Bem vindo ao Wizard do EasyFAN"/>
        </panel>
        <panel bottom="4" columns="2" gap="4" left="14" right="4"
top="4">
            <label font="50" name="tamanho1" text="  "/>
            <label name="Introducao" text="Introducao" visible="false"/>
            <label colspan="2" font="40" text=" "/>
```

```
            <label/>
            <label colspan="2"/>
            <label name="exWizard" text=" Através do Wizard do EasyFAN
você podera em 5 passos realizar as operações básicas do programa que é
treinar uma rede.                    "/>
            <label colspan="3"/>
            <label name="lbPassosAjudar" rowspan="3" text=" Os seguintes
passos irão o ajudar a ter um melhor entendimento sobre o programa:"/>
            <label colspan="2"/>
            <label name="lbPasso1" text="              1.  Breve
Explicação Sobre Reconhecimento de Padões e o EasYFAN"/>
            <label text="     "/>
            <label name="lbPasso2" text="              2. Carregamento de
Arquivos"/>
            <label/>
            <label name="lbPasso3" text="              3. O que é o
Treinamento e o que é Teste"/>
            <label/>
            <label name="lbPasso4" text="              4. Divisão dos
Conjuntos (Treino/Teste)"/>
            <label/>
            <label name="lbPasso5" text="              5. Iniciando o
Treinamento"/>
            <label/>
        </panel>
        <label/>
        <label font="85" name="tamanho2" text="        "/>
        <panel/>
        <panel bottom="4" columns="5" gap="4" halign="right" left="4"
right="4" top="4">
            <button action="paginaProxima(Introducao)" alignment="left"
icon="/visao/gui/wizard/icones/proximaPagina.png" text="Próximo"/>
            <label text="     "/>
            <button action = "fechar()"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()" alignment="left"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.10.7 Passo1.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" halign="right" left="4"
name="dialog" right="4" top="4">
    <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4" top="4">
        <panel bottom="4" columns="2" gap="4" halign="right" left="4"
right="4" top="4">
            <label name="Passo1" text="Passo1" visible="false"/>
            <label font="20 bold" name="lbPasso1intro" text="1º Passo :
Reconhecimento de Padrões e o EasyFAN *"/>
            <label rowspan="24"/>
            <label/>
```

```
            <label font="20" name="tamanho1" text=" "/>
            <label font="bold" name="lbReconhecimentoPadroes" text="O Que
é Reconhecimento de Padrões?"/>
            <label/>
            <label name="lbOqueE" text="O Reconhecimento de Padrões (RP)
é a ciência que trata da classificação e descrição de objetos"/>
            <label/>
            <label name="lbTecnicas" text="As técnicas de RP são usadas
para classificar ou descrever padrões ou objetos através de um conjunto
de propriedades ou características."/>
            <label/>
            <label name="lbObjetivos1" text="O RP tem por objetivo
atribuir um padrão a um conjunto desconhecido de classes de padrões ou
identificar um padrão como membro de um "/>
            <label/>
            <label name="lbObjetivos2" text="conjunto conhecido de
classes "/>
            <label/>
            <label/>
            <label/>
            <label font="bold" name="lbEasyFan" text="O que é EasyFAN?"/>
            <label/>
            <label text="É um software para reconhecimento de padrões que
utiliza da técnica de redes neurais do tipo FAN. "/>
            <label/>
            <label text="O software é multiplataforma e amigável ao
usuário e rode em qualquer plataforma que suporte uma máquina virtual
java"/>
            <label font="16" name="tamanho2" text="    "/>
        </panel>
        <label text="  "/>
        <panel border="true" columns="3">
            <label name="lbEasyHelp" text="* Maiores Informações sobre
Reconhecimento de Padrões e o EasyFAN consulte o EasyHELP clicando no
botão ao lado :         "/>
            <button action="chamarHelp()"
icon="/visao/gui/wizard/icones/easyHelp.png" name="btEasyHelp"
text="EasyHELP"/>
            <label text="  "/>
        </panel>
        <label/>
        <label/>
        <label font="55" name="tamanho3" text="  "/>
        <panel bottom="4" columns="5" gap="4" halign="right" left="4"
right="4" top="4" weightx="1">
            <button action="paginaAnterior(Passo1)" alignment="left"
halign="right" icon="/visao/gui/wizard/icones/anteriorPagina.png"
text="Anterior"/>
            <button action="paginaProxima(Passo1)" alignment="left"
halign="right" icon="/visao/gui/wizard/icones/proximaPagina.png"
text="Próximo"/>
            <label halign="right" text="     "/>
            <button action = "fechar()" halign="right"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()" halign="right"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
        </panel>
```

```
    </panel>
</panel>
```

1.10.8 Passo2.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label name="Passo2" text="Passo2" visible="false"/>
            <label font="20 bold" name="lbPasso2intro" text="2º Passo :
Carga de Arquivos
"/>
            <label/>
            <label/>
            <label/>
            <label name="lbTiposCarregamento" text="   Para treinar a
rede, primeiro você deve carregar  o(s) arquivo(s) de padrões e suas
respectivas característcas e classes"/>
            <label/>
            <label name="lbOqueE" text="   Abaixo estão os tipos de
extensões que podem ser carregadas:"/>
            <label/>
            <panel columns="11" halign="center">
                <label name="lbDat" text="*.dat "/>
                <button action = "explicacaoDat()"
icon="/visao/gui/wizard/icones/ajuda.png" name="btDat"/>
                <label text="          "/>
                <label name="lbTxt" text="*.txt "/>
                <button action="explicacaoTxt()"
icon="/visao/gui/wizard/icones/ajuda.png" name="btTxt"/>
                <label text="          "/>
                <label name="lbXls" text="*.xls "/>
                <button action="explicacaoXls()"
icon="/visao/gui/wizard/icones/ajuda.png" name="btXls"/>
                <label text="          "/>
                <label name="lbEpd" text="*.epd "/>
                <button action = "explicacaoEpd()"
icon="/visao/gui/wizard/icones/ajuda.png" name="btEpd"/>
            </panel>
            <label/>
            <label name="lb2Carregamentos" text="Podem ser carregados
arquivos de diferentes tipos de extensões desde que possuam o mesmo
número de características"/>
            <label/>
            <label/>
            <label/>
            <label name="lbIntroAberturaArquivos" text="Carregue abaixo o
seu conjunto de arquivos:"/>
            <label/>
            <panel border="true" columns="2" gap="4" left="3" right="2"
top="4">
                <label font="bold" text="Opções:                  "/>
                <panel columns="3" gap="3" top="4">
```

```
                    <button
action="adicionarArquivo(tabelaWZ,btLimparConjunto,pagAtual,pagMax,btPagi
naAnterior,btProximaPagina,btProximoPasso)" alignment="left"
icon="/visao/gui/wizard/icones/carregar.png" name="btCarregarArquivo"
text="Adicionar Arquivo" weightx="1"/>
                    <button
action="limparDados(tabelaWZ,btLimparConjunto,pagAtual,pagMax,btPaginaAnt
erior,btProximaPagina,btProximoPasso,padroesTsN,caracTsN)"
alignment="left" enabled="false"
icon="/visao/gui/wizard/icones/descarregar.png" name="btLimparConjunto"
text="Limpar Conjunto" weightx="1"/>
                </panel>
            </panel>
            <label/>
            <panel background="#cfcfcf" border="true" bottom="4"
colspan="10" columns="10" gap="4" left="4" right="4" scrollable="true"
top="4">
                <label text="Conjunto de Arquivos"/>
                <label text="            "/>
                <label text="N° de Padrões da Página: "/>
                <label name="padroesTsN" text="             "/>
                <label text="     "/>
                <label text="N° de Característcas:  "/>
                <label name="caracTsN" text="       "/>
                <label name="tamTabela" text="
"/>
                <label/>
                <label text="
"/>
                <table colspan="10" name="tabelaWZ" rowspan="9"
weighty="1"/>
                <panel bottom="4" colspan="10" columns="10" gap="4"
halign="right" left="4" right="4" top="4">
                    <label weightx="1"/>
                    <label text="Pagina: "/>
                    <label name="pagAtual" text="  "/>
                    <label text="/"/>
                    <label name="pagMax" text="  "/>
                    <button action="beforePage()" enabled="false"
halign="left" icon="/visao/gui/wizard/icones/anteriorPagina.png"
name="btPaginaAnterior" text="Página Anterior"/>
                    <button action="nextPage()" enabled="false"
halign="left" icon="/visao/gui/wizard/icones/proximaPagina.png"
name="btProximaPagina" text="Próxima Página"/>
                    <label text="  " weightx="1"/>
                </panel>
                <label/>
            </panel>
            <label/>
            <label/>
            <label/>
        </panel>
        <label/>
        <label/>
        <label/>
        <panel bottom="4" columns="5" gap="4" halign="right" left="4"
right="4" top="4">
```

```
            <button action="paginaAnterior(Passo2)" alignment="left"
icon="/visao/gui/wizard/icones/anteriorPagina.png" text="Anterior"/>
            <button action="paginaProxima(Passo2)" alignment="left"
enabled="false" icon="/visao/gui/wizard/icones/proximaPagina.png"
name="btProximoPasso" text="Próximo"/>
            <label text="     "/>
            <button action="fechar()"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.10.9 Passo3.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label name="Passo3" text="Passo3" visible="false"/>
            <label font="20 bold" name="lbPasso3intro" text="3º Passo : O
que é o Treinamento e o que é Teste?"/>
            <label text=" "/>
            <label font="21" name="tamanho0" text="    "/>
            <label name="tamanho1" rowspan="52"/>
            <label font="bold" name="lbTreinamento" text="O Que é
Treinamento?"/>
            <label/>
            <label name="lbOqueE" text="É a sistemática necessária para
executar adequadamente o processamento desejado dos dados fornecidos."/>
            <label/>
            <label name="lbTecnicas" text="Em outras palavras treinamento
é o processo pelo qual a rede adquire conhecimento para classifivcar os
padrões posteriormente                          "/>
            <label/>
            <label name="lbObjetivos1" text="Durante o Treinamento do
EasyFAN você acompanha estatistica e graficamente o desempenho da rede"/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label font="bold" name="lbTeste" text="O que é Teste?"/>
            <label/>
            <label text="Quando o sistema já está treinado, a capacidade
de classificação é feita por um algoritmo de teste"/>
            <label/>
            <label text="Para o teste, os padrões são do mesmo tipo dos
de treinamento, com classes conhecidas, mas que não foram usados no
treinamento."/>
            <label/>
            <label text="A taxa de acerto no conjunto de testes diz qual
foi a performance do sistema."/>
```

```
        </panel>
        <label/>
        <label/>
        <label/>
        <label/>
        <label/>
        <panel bottom="4" columns="5" gap="4" halign="right" left="4"
right="4" top="4">
            <button action="paginaAnterior(Passo3)" alignment="left"
icon="/visao/gui/wizard/icones/anteriorPagina.png" text="Anterior"/>
            <button action="paginaProxima(Passo3)" alignment="left"
icon="/visao/gui/wizard/icones/proximaPagina.png" text="Próximo"/>
            <label text="     "/>
            <button action = "fechar()"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

1.10.10 Passo4.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label name="Passo4" text="Passo4" visible="false"/>
            <label font="20 bold" name="lbPasso4intro" text="4° Passo :
Divisão do conjunto de arquivos
"/>
            <label/>
            <label font="20" name="tamanho1" text=" "/>
            <label name="tamanho2" rowspan="44"/>
            <label name="lbposCarregamento" text="Após o carregamento
do(s) arquivo(s) você deve esolher a divisão entre eles"/>
            <label/>
            <label name="lbDivisao" text="Os arquivos devem ser divididos
em 2 conjuntos (Treinamento e Teste)"/>
            <label/>
            <label name="lbTiposDivisao" text="Você pode escolher os
seguintes tipos de divisões:"/>
            <label font="15" text=" "/>
            <panel border="false" bottom="4" columns="3" gap="4"
halign="center" left="4" right="4" top="4">
                <button action = "msgDivisaoConjuntos(this)"
icon="/visao/gui/wizard/icones/ajuda.png" name="btDVMesmoConjunto"
property="indice=mesmoConjunto" />
                <checkbox
action="selectedRadio(this,lbCjTreino,txtPorcentagemTreino,lbPorc,lbCjTes
te,txtPorcentagemTeste,lbPorc2,btAdiciConjunto,btProximoPasso)"
group="gp" name="cbMesmoConjunto" selected="true" text="Mesmo Conjunto"/>
                <label/>
```

```
                <button action = "msgDivisaoConjuntos(this)"
icon="/visao/gui/wizard/icones/ajuda.png" name="btDVPorventagem"
property="indice=porcentagem"/>
                <checkbox
action="selectedRadio(this,lbCjTreino,txtPorcentagemTreino,lbPorc,lbCjTes
te,txtPorcentagemTeste,lbPorc2,btAdiciConjunto,btProximoPasso)"
group="gp" name="cbPorcentagem" text="Por porcentagem: "/>
                <panel>
                    <label enabled="false" name="lbCjTreino"
text="Conjunto Treino: "/>
                    <textfield
action="calculaPorcentagemTreino(this,txtPorcentagemTeste,btProximoPasso)
" enabled="false" name="txtPorcentagemTreino" text="50"/>
                    <label enabled="false" name="lbPorc" text="%      "/>
                    <label/>
                    <label enabled="false" name="lbCjTeste"
text="Conjunto de Teste: "/>
                    <textfield
action="calculaPorcentagemTeste(txtPorcentagemTreino,this,btProximoPasso)
" enabled="false" name="txtPorcentagemTeste" text="50"/>
                    <label enabled="false" name="lbPorc2" text="%"/>
                </panel>
                <button action = "msgDivisaoConjuntos(this)"
icon="/visao/gui/wizard/icones/ajuda.png" name="btDVNovoConjunto"
property="indice=novoConjunto"/>
                <checkbox
action="selectedRadio(this,lbCjTreino,txtPorcentagemTreino,lbPorc,lbCjTes
te,txtPorcentagemTeste,lbPorc2,btAdiciConjunto,btProximoPasso)"
group="gp" name="cbNovoConjunto" text="Novo Conjunto para Teste"/>
                <panel>
                    <button action="adicionarNovoConjuntoPasso4()"
enabled="false" icon="/visao/gui/wizard/icones/novo.png"
name="btAdiciConjunto" text="Adicionar Conjunto"/>
                </panel>
            </panel>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <panel/>
        </panel>
        <label/>
        <label/>
        <label/>
        <panel bottom="4" columns="5" gap="4" halign="right" left="4"
right="4" top="4">
            <button action="paginaAnterior(Passo4)" alignment="left"
icon="/visao/gui/wizard/icones/anteriorPagina.png" text="Anterior"/>
            <button name = "btProximoPasso"
action="paginaProxima(Passo4)" alignment="left"
icon="/visao/gui/wizard/icones/proximaPagina.png" text="Próximo"/>
            <label text="     "/>
            <button action = "fechar()"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
```

```
        </panel>
    </panel>
</panel>
```

1.10.11 Passo4AdicionarConjunto.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<dialog bottom="4" columns="1" gap="4" left="4" modal="true"
name="dialog" right="4" top="4">
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
        <panel border="true" bottom="4" columns="2" gap="4" left="4"
right="4" top="4">
            <label/>
            <label font="14 bold" name="lbCarregarArqTeste"
text="Carregar novo Arquivo para o Conjunto de Teste"/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label name="lbConjTeste" text="Carregue abaixo o seu
conjunto de teste:"/>
            <label/>
            <panel border="true" columns="2" gap="4" left="3" right="2"
top="4">
                <label font="bold" text="Opções:                  "/>
                <panel columns="3" gap="3" top="4">
                    <button
action="adicionarArquivoTeste(tabelaWZ,btLimparConjunto,pagAtual,pagMax,b
tPaginaAnterior,btProximaPagina)" alignment="left"
icon="/visao/gui/wizard/icones/carregar.png" name="btCarregarArquivo"
text="Adicionar Arquivo" weightx="1"/>
                    <button
action="limparDadosTeste(tabelaWZ,btLimparConjunto,pagAtual,pagMax,btPagi
naAnterior,btProximaPagina,padroesTsN,caracTsN)" alignment="left"
enabled="false" icon="/visao/gui/wizard/icones/descarregar.png"
name="btLimparConjunto" text="Limpar Conjunto" weightx="1"/>
                </panel>
            </panel>
            <label/>
            <panel background="#cfcfcf" border="true" bottom="4"
colspan="10" columns="10" gap="4" left="4" right="4" scrollable="true"
top="4">
                <label text="Conjunto de Arquivos"/>
                <label text="            "/>
                <label text="N° de Padrões da Página: "/>
                <label name="padroesTsN" text="             "/>
                <label text="     "/>
                <label text="N° de Característcas:  "/>
                <label name="caracTsN" text="       "/>
                <label text="    "/>
                <label/>
```

```
                <label text="
"/>
                <table colspan="10" name="tabelaWZ" rowspan="9"
weighty="1"/>
                <panel bottom="4" colspan="10" columns="10" gap="4"
left="4" right="4" top="4">
                    <label weightx="1"/>
                    <label text="Pagina: "/>
                    <label name="pagAtual" text="  "/>
                    <label text="/"/>
                    <label name="pagMax" text="  "/>
                    <button action="beforePageTeste()" enabled="false"
halign="left" icon="/visao/gui/wizard/icones/anteriorPagina.png"
name="btPaginaAnterior" text="Página Anterior"/>
                    <button action="nextPageTeste()" enabled="false"
halign="left" icon="/visao/gui/wizard/icones/proximaPagina.png"
name="btProximaPagina" text="Próxima Página"/>
                    <label text="  " weightx="1"/>
                </panel>
                <label/>
            </panel>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label/>
        </panel>
        <label/>
        <label/>
        <label/>
        <label/>
        <label/>
        <panel bottom="4" columns="4" gap="4" halign="right" left="4"
right="4" top="4">
            <label/>
            <button action="confirmarNovoConjunto()" alignment="left"
icon="/visao/gui/wizard/icones/confirmar.png" name="btConfirmar"
text="Confirmar"/>
            <button
action="cancelarNovoConjunto(tabelaWZ,btLimparConjunto,pagAtual,pagMax,bt
PaginaAnterior,btProximaPagina,padroesTsN,caracTsN)" alignment="left"
icon="/visao/gui/wizard/icones/cancelar.png" name="btCancelar"
text="Cancelar"/>
        </panel>
    </panel>
</dialog>
```

1.10.12 Passo5.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel bottom="4" columns="1" gap="4" left="4" name="dialog" right="4"
top="4">
    <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
```

```
        <panel bottom="4" columns="2" gap="4" left="4" right="4" top="4">
            <label name="Passo5" text="Passo5" visible="false"/>
            <label font="20 bold" name="lbPasso5intro" text="5º Passo :
Iniciando o Treinamento
"/>
            <label/>
            <label font="22" name="tamanho1" text=" "/>
            <label name="tamanho2" rowspan="23"/>
            <label text="Após passar pelos 4 passos anteriores agora
pode-se iniciar o treinamento"/>
            <label/>
            <label text="Para isso basta clicar no botão abaixo Treinar /
Finalizar"/>
            <label/>
            <label/>
            <label/>
            <label/>
            <label name="tamanho3" text=" "/>
            <panel columns="1" halign="center">
                <button action="iniciarTreinamento()" halign="center"
icon="/visao/gui/wizard/icones/treinar2.png" text="Treinar / Finalizar"/>
            </panel>
            <label/>
            <label/>
            <label/>
            <label/>
        </panel>
        <label/>
        <panel border="true" columns="3" halign="center">
            <label name="lbEasyHelp" text="* Maiores Informações sobre
Treinamento e o EasyFAN consulte o EasyHELP clicando no botão ao lado:
"/>
            <button action="chamarHelp()" halign="left"
icon="/visao/gui/wizard/icones/easyHelp.png" name="btEasyHelp"
text="EasyHELP"/>
            <label name="tamanho4" text="                          "/>
        </panel>
        <label/>
        <label font="115" name="tamanho5" text="     "/>
        <label/>
        <panel bottom="4" columns="4" gap="4" halign="right" left="4"
right="4" top="4">
            <button action="paginaAnterior(Passo5)" alignment="left"
icon="/visao/gui/wizard/icones/anteriorPagina.png" name="btAnterior"
text="Anterior"/>
            <label text="     "/>
            <button action = "fechar()"
icon="/visao/gui/wizard/icones/fechar.png" text="Fechar"/>
            <button action="close()"
icon="/visao/gui/wizard/icones/cancelar.png" text="Cancelar"/>
        </panel>
    </panel>
</panel>
```

2. CADERNO FONTE – JFAN

2.1 PACKAGE FAN.UTILITARIAS;

2.1.1 FastFAN.Java

```
package fan.utilitarias;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

public class fastFAN
{
	public double [][] test;
	public double [][] trainning;
	public double [][] testNorm;
	public double [][] trainningNorm;
	public double [][][] testConjDifuso;
	public double [][][] trainningConjDifuso;
	public double [] testSomaConjDifuso;
	public double [] trainningSomaConjDifuso;
	private String messageStatus;
	public static int MAX_CLA =  7;
	public static int MAX_RAN = 110;
	public static int MAX_CAR =  3;
	public static int MAX_PAD = 10000;
	private int n_entr;
	private int n_neur;
	private int n_pad;
	private int max_range;
	private int [] d;
	private double [] nx;
	private double [] peso_neu;
	private double [][][] c1;
	private int [] classe_real;
	private int [] classe_fan;
	private int [] classe_certo;
	private int [] classe_erro;
	private double max_sem;
	public double minval_max;
	public double minval_old;
	public double s_harm;
	public double max_tot;
	private Random rand;
	private String fileTreino;
	private String fileTeste;
	private int m_aprendizagem;

	public double [][] m_vMatconf;
	public double [][] m_pMatconf;
	private boolean [] usar_car;
	private int [] pesos;
```

```
public double m_certos;
public double m_total;
public double m_errados;
public double m_taxaerrados;
private boolean normalized;

protected fastFAN()
{
	normalized = false;

	try
	{
		test = new double [MAX_PAD][MAX_CAR+1];
		trainning = new double [MAX_PAD][MAX_CAR+1];

		testNorm = new double [MAX_PAD][MAX_CAR];
		trainningNorm = new double [MAX_PAD][MAX_CAR];

		testSomaConjDifuso = new double [MAX_PAD];
		trainningSomaConjDifuso = new double [MAX_PAD];

		c1 = new double [MAX_CLA] [MAX_CAR] [MAX_RAN];
		d = new int [MAX_CAR];
		nx = new double [MAX_CAR + 1];
			//ultima posicao contem o total da normal
		usar_car = new boolean [MAX_CAR];
		peso_neu = new double [MAX_CLA];//[MAX_CAR];
		classe_real = new int [MAX_CLA];
		classe_fan = new int [MAX_CLA];
		classe_certo = new int [MAX_CLA];
		classe_erro = new int [MAX_CLA];
		rand = new Random();
		pesos = new int [MAX_CLA];
		m_vMatconf = new double [MAX_CLA] [MAX_CLA];
		m_pMatconf = new double [MAX_CLA] [MAX_CLA];
	}
	catch (Exception ex)
	{
		messageStatus += ex.getMessage();
	}

	m_aprendizagem = 45;
	n_entr = 3;//35;!!!
	n_pad = 4096;
	n_neur = 7;
	max_range = 101;
	max_sem = 0.10;
	minval_max = 0;
	minval_old = 0;
	s_harm = 0;
	max_tot = 0;
	m_certos = 0;
	m_total = 0;
	for (int i = 0; i < MAX_CAR; i++)
		d[i] = 6;

	for (int i = 0; i < MAX_CLA; i++)
```

```
                pesos[i] = 500;

            double maiorRaio = 0;
            for (double dd : d) {
                if (dd > maiorRaio) {
                    maiorRaio = dd;
                }
            }


            testConjDifuso = new double
[MAX_PAD][n_entr][(int)(maiorRaio*2)+1];
            trainningConjDifuso = new double
[MAX_PAD][n_entr][(int)(maiorRaio*2)+1];

        }
        public int random(int maximo)
        {
            int intRet = rand.nextInt(maximo+1);// Next(0,1000);
            return intRet;
        }
        private boolean num_car(int num_cars)
        {
            try {
            File fso = new File(fileTreino);

            if (!fso.exists())
            {
                messageStatus += "Err000 - Arquivo de treinamento não
encontrado";
                return false;
            }
            BufferedReader sr;
            String line;

                sr = new BufferedReader(new FileReader(fileTreino));
                line = sr.readLine();

            String [] values = line.split("\t");
            sr.close();


            num_cars = 0;
            for (int i = 0; i < values.length; i++)
            {
                if (values[i].toString() != "" || values[i].toString()
!= null)
                    num_cars = i - 1;
                else
                    break;
            }

            return true;
            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
```

```
	}

	private double triang ( double x, double centro, double d)
	{
		double y, d1;
		d1 = d + 1;
		if (x<centro)
			y = (double) (1.0f/d1)*(x+d1-centro);
		else
			y = (double) (1.0f/d1)*(-x+d1+centro);

		return (y);
	}

	private boolean normal (String file)
	{
		try {
		int j;

		//verifica se o arquivo existe
		File fso = new File(file);
		if (!fso.exists())
		{
			return false;
		}

		BufferedReader sr = new BufferedReader(new FileReader(file));

		//limpa todas as normais
		for(j=0; j<n_entr; j++)
			nx[j] = 0;


			//inicia o calculo da normal
			String line = sr.readLine();
			String [] values = line.split("  ");
			for(j=0; j<n_entr; j++)
				nx [j] = Double.parseDouble(values[1+j]);
			int count = 2;
			//verifica se existe alguma maior que a atual maior
			line = sr.readLine();
			while (line != null && line != "")
			{
				values = line.split("  ");
				for(j=0; j<n_entr; j++)
					if (nx[j] < Double.parseDouble(values[j+1]))
{
						nx[j] =
Double.parseDouble(values[j+1]);
						if(nx[j] > 2.0f) {
							@SuppressWarnings("unused")
							int as = 0;
						}
					}
				line = sr.readLine();
				count++;
```

```
                }

            nx[n_entr] = 0;
            for(j=0; j<n_entr; j++)
                nx [n_entr] += nx[j];

            sr.close();
            return true;
        }catch(Exception ex)
            {
                messageStatus += ex.getMessage();
                return false;
            }
        }

    private void store (double [][][] c, int neu, double [] arg,
double[][] idx, double[] x)
    {
        int i, j, count;
        int iip;
        double u, s;
        double mais;


        for (j=0; j<n_entr; j++)
        {
            count = 0;
            for (i=-d[j]; i<=d[j]; i++)
            {
                iip = (int) x [j] + i;
                if (iip<0)
                    iip = 0;
                if (iip>max_range-1) iip = max_range-1;
                //idx = triang(iip,x[j],d[j]);
                u = c [neu][j][iip];
                s = c [neu][j][max_range];

                mais = idx[j][count];
                u += mais;
                s += mais;
                if (random(100)<100*idx[j][count])
                {
                    c [neu][j][iip] = u;
                    c [neu][j][max_range] = s;
                }
                count++;
            }
        }
    }
    private void unstore (double [][][] c, int neu, double [] arg,
double[][] idx, double[] x)
    {
        int i, j, count;
        int iip;
        double u;

        for (j=0; j<n_entr; j++) {
```

```
                  count = 0;
                  if (random(10000)>peso_neu[neu]*10) {
                        for (i=-d[j]; i<=d[j]; i++)
                        {
                              iip = (int) x[j] + i;
                              if (iip<0) iip = 0;
                              if (iip>max_range-1) iip = max_range-1;
                              u = c [neu][j][iip];
                              if (random(100)>1)
                                    u *= Math.pow(1-
idx[j][count],0.001)*0.999;
                              else
                                    u -= idx[j][count];
                              if (u<0) u = 0;
                              if (random(100)<100*idx[j][count])
                                    c [neu][j][iip] = u;

                        }
                        count++;
                  }
            }
      }

      private double tests (double [][][]c, int neu, double [] arg,
double somaCD, double[][] idx, double[] x)
      {
            double pi;
            int i,j;
            int iip;
            double u, s;
            pi = 1;

            for (j=0; j<n_entr; j++)
            {
                  int count = 0;
                  for (i=-d[j]; i<=d[j]; i++)
                  {
                        iip = (int) x[j] + i;
                        if (iip<0) iip = 0;
                        if (iip>max_range-1) iip = max_range-1;
                        u   = c [neu][j][iip];
                        s   = c [neu][j][max_range];
                        if (s > 0)
                        {
                              if (usar_car[j])
                                    pi *= (1-
Math.sqrt(idx[j][count]*u/(s*somaCD)));                     }
                        count++;
                  }
            }
            return (1-Math.pow(pi, 1/(double)n_entr));
      }

      protected void inicializa()
      {
            double r;
            int i, j, k;
```

```
        for (k=0; k<n_neur; k++)
              for (i=0; i<n_entr; i++)
                    for (j=0; j<max_range; j++)
                    {
                          //r = random(1000)/1000.0;
                          r = 0;
                          c1 [k][i][j] = r;
                          c1 [k][i][max_range] += r;
                    }
}

public void renormaliza()
{
        int i, j, k;

        for (k=0; k<n_neur; k++)
              for (i=0; i<n_entr; i++)
              {
                    for (j=0; j<max_range; j++)
                    {
                          c1 [k][i][j] /= c1 [k][i][max_range];
                    }
                    c1 [k][i][max_range] = 1;
              }
}

protected boolean cromo()
{
        double [] e = new double [MAX_CAR];
        double [] xin = new double [MAX_CAR];
        double [] bxin = new double [MAX_CAR];
        double [] a  = new double [MAX_CLA];
        double [] q = new double [MAX_CLA];
        double ma;
        double prop;
        double certost = 0;  double certosm = 0;
        double errosm  = 0;  double errost  = 0;
        int maior, maior2;
        int ce = 0;
        int i,k, padrao;

        padrao = 0;
        m_total = 0;
        m_certos = 0;

        //inicializacao dos contadores de classe, acertos e erros
        for (i=0; i<n_neur; i++)
        {
              classe_real [i] = 0;
              classe_fan  [i] = 0;
              classe_certo[i] = 0;
              classe_erro [i] = 0;
        }

        while (padrao < n_pad)
        {
              //leitura de todas as caracteristicas deste padrao
```

```
                        for(k=0; k<n_entr; k++)
                        {
                              e[k] = trainning[padrao][k];
                        }
                        //leitura da saida correspondente
                        ce = (int)trainning[padrao] [n_entr] - 1;

                  //normalizacao do padrao
                  for(k=0; k<n_entr; k++)
                        if (nx[k] != 0)
                              xin[k] = (/*0.9**/e[k]/nx[k]);
                        else
                              xin[k] = 0;

                  //eliminacao do sinal (calculo do modulo) dos dados
                  for (i=0; i<n_entr; i++)
                        bxin [i] = Math.abs(xin[i]);

                  ma = maior = maior2 = 0;

                  //consulta da pertinencia em relacao a cada classe para
                  //o padrao sendo testado
                  for (i=0; i<n_neur; i++)
                  {
                        a[i] = tests(c1,i,bxin,
trainningSomaConjDifuso[padrao], trainningConjDifuso[padrao],
trainningNorm[padrao]);
                        ma += a [i];
                  }

                  //verificacao da classe com maior pertinencia
                  for (i=0; i<n_neur; i++)
                  {
                        q[i] = a[i];
                        if (q[i]>q[maior])
                        {
                              maior2 = maior;
                              maior  = i;
                        }
                  }
                  if (q[maior] > 0)
                        prop = (q[maior]-q[maior2])/q[maior];
                  else
                        prop = 1;

                  m_total +=1;

                  /// Armazenamento
                  classe_real [ce] += 1;
                  classe_fan  [maior] += 1;

                  //aprendizagem
                  if (random(100) > m_aprendizagem)
                  {
                        store(c1,ce,bxin, trainningConjDifuso[padrao],
trainningNorm[padrao]);
                        //se a opiniao da FAN estiver errada,
```

```
                        //realizar penalizacao
                        if (maior != ce && q[maior] > q[ce])
                        {
                              unstore(c1,maior,bxin,
trainningConjDifuso[padrao], trainningNorm[padrao]);

                              for (int j = 0; j < n_neur; j++)
                              {
                                    peso_neu[j] = pesos[j] + random(1000-
pesos[j]);
                              }

                        }
                  }
                  /// Fim Armazenamento

                  //contagem do numero de acertos/erros
                  if (maior == ce)
                  {
                        classe_certo [ce] += 1;
                        if (prop > max_sem)
                              certost += 1;
                        else
                              certosm += 1;
                  }
                  else
                  {
                        classe_erro [maior] += 1;
                        if (prop > max_sem)
                              errost +=1;
                        else
                              errosm +=1;
                  }

                  padrao++;
            }
            m_certos = certost + certosm;
            m_errados = errost + errosm;
            return true;
      }

      protected boolean cromot(boolean saida, int rede)
      {
            double [][] conf = new double [MAX_CLA] [MAX_CLA];
            double [] e = new double [MAX_CAR];
            double [] xin = new double [MAX_CAR];
            double [] bxin = new double [MAX_CAR];
            double [] a = new double [MAX_CLA];
            //double [] b = new double [MAX_CLA];
            double [] q = new double [MAX_CLA];
            double ma;
            double prop;
            double certost = 0;  double certosm = 0;
            double errosm  = 0;  double errost  = 0;
            double maior, maior2;
            int ce,i,j,k, padrao;
            BufferedWriter swFC = null;
```

```
            double [][][] c = new double[1][1][1];

            if (rede == 1)
            {
                  c = new double [MAX_CLA] [MAX_CAR] [MAX_RAN];
                  if (!load_net("amax.fan", c))
                  {
                        return false;
                  }
            }
            else
                  if (rede == 2)
            {
                  c = new double [MAX_CLA] [MAX_CAR] [MAX_RAN];
                  if (!load_net("amaxg.fan", c))
                  {
                        return false;
                  }
            }

            padrao = 0;
            m_certos = 0;
            m_total = 0;

            if (saida)
            {
                  try
                  {
                        //abre arquivo de saida
                        swFC = new BufferedWriter(new
FileWriter("fc.fsy"));

                  }
                  catch (Exception ex)
                  {
                        messageStatus += "Err000 - Não foi possivel criar
arquivo de saída";
                        return false;
                  }
            }

            //inicializacao dos contadores de classe, acertos e erros
            for (i=0; i<n_neur; i++)
            {
                  classe_real [i] = 0;
                  classe_fan  [i] = 0;
                  classe_certo[i] = 0;
                  classe_erro [i] = 0;
                  for (j=0; j<n_neur; j++)
                        conf [i][j] = 0;
            }

            while (padrao < n_pad)
            {
                        //leitura de todas as caracteristicas deste padrao
                        for(k=0; k<n_entr; k++)
```

```
                        {
                              e[k] = test[padrao][k];
                        }
                        //leitura da saida correspondente
                        ce = (int)test[padrao][n_entr] - 1;
                  //}

                  //normalizacao dos dados a serem testados
                  for(k=0; k<n_entr; k++)
                  {
                        if (nx[k] != 0)
                              xin[k] = (/*0.9**/e[k]/nx[k]);
                        else
                              xin[k] = 0;
                  }
                  for (i=1; i<n_entr; i++)
                  {
                        if (xin[i] > 1)
                              xin[i] = 1;
                  }

                  //eliminacao do sinal
                  for (i=0; i<n_entr; i++)
                  {
                        bxin [i] = Math.abs(xin[i]);
                  }

                  ma = maior = maior2 = 0;

                  for (i=0; i<n_neur; i++)
                  {
                        if (rede == 1 || rede == 2)
                        {
                              a[i] = tests(c,i,bxin,
testSomaConjDifuso[padrao], testConjDifuso[padrao], testNorm[padrao]);
                        }
                        else
                        {
                              a[i] = tests(c1,i,bxin,
testSomaConjDifuso[padrao], testConjDifuso[padrao], testNorm[padrao]);
                        }
                        ma += a [i];
                  }

                  //verificacao da classe com maior pertinencia
                  for (i=0; i<n_neur; i++)
                  {
                        q[i] = a[i];
                        if (q[i] > q[(int)maior])
                        {
                              maior2 = maior;
                              maior  = i;
                        }
                  }
                  if (q[(int)maior] > 0)
                        prop = (q[(int)maior]-
q[(int)maior2])/q[(int)maior];
```

```
                else
                      prop = 1;

                m_total +=1;

                classe_real [ce] += 1;
                classe_fan [(int)maior] += 1;
                conf [ce][(int)maior] +=1;

                //atualizacao da quantidade de acertos e erros
                if (maior == ce)
                {
                      classe_certo [ce] += 1;
                      if (prop > max_sem)
                            certost+=1;
                      else
                            certosm+=1;
                }
                else
                {
                      classe_erro [(int)maior] += 1;
                      if (prop > max_sem)
                            errost +=1;
                      else
                            errosm +=1;
                }
                try{
                //salva as pertinencias de cada classe para cada padrao
testado
                if(saida)
                {
                      for(i=0; i<n_neur; i++)
                      {
                            double dvalue = q[i];
                            swFC.write("\t" + Double.toString(dvalue));
                      }
                      int iValue = ce + 1;
                      swFC.write("\t" + Integer.toString(iValue)+ "\n");
                }
                }
                catch(Exception ex) {
                      ex.printStackTrace();
                      return false;
                }
                padrao++;

          }

          m_certos = certost + certosm;
          m_errados = errost + errosm;
          try {
          if (saida)
           swFC.close();
          }
          catch(Exception ex) {
                ex.printStackTrace();
          }
```

```
            //calculo da media harmonica
            if (rede == 0)
            {
                  s_harm = 0;
                  for (i=0; i<n_neur; i++)
                  {
                        if (conf[i] [i] > 0)
                        {
                              s_harm += classe_real[i]/conf[i][i]; //
tentar 0 no final
                        }
                        else
                        {
                              s_harm = 100;
                              break;
                        }
                  }
                  s_harm = n_neur/s_harm;

                  minval_old=0;
                  if (classe_real[0] > 0 && classe_real[1] > 0)
                        minval_old =
Math.min(conf[0][0]/classe_real[0],conf[1][1]/classe_real[1]);

                  //salva o maior valor minimo da taxa de acertos
                  if (minval_old > minval_max)
                  {
                        minval_max = minval_old;
                        save_net("amaxg.fan");
                  }

                  //salvar a melhor media absoluta
                  if (max_tot < s_harm)
                  {
                        max_tot = s_harm;
                        save_net("amax.fan");
                  }

            }
            //calcula e salva matriz de confusao
            m_taxaerrados = 100-(double)(100*m_certos/m_total);

            for (i=0; i<n_neur; i++)
                  for(j=0; j<n_neur; j++)
                        m_vMatconf[i][j] = conf [i][j];

            for (i=0; i<n_neur; i++)
                  for(j=0; j<n_neur; j++)
                        m_pMatconf[i][j] = conf [i][j]/classe_real[i];
            return true;
      }

      private boolean load_net(String file, double [][][] c)
      {
            int i,j,l;
            String [] values;
```

```
            File fso = new File(file);
            if(!fso.exists())
            {
                  messageStatus += "Err000 - Arquivo não pode ser
localizado";
            }
            else
            {
                  try {
                  BufferedReader sr = new BufferedReader(new
FileReader(file));
                  for (i=0; i<n_neur; i++)
                        for (l=0; l<=max_range; l++)
                        {
                              String line = sr.readLine();
                              values = line.split(" ");
                              for (j=0; j<n_entr; j++)
                              {
                                    c [i][j][l] =
Double.parseDouble(values[j]);
                              }
                        }
                  sr.close();
                  }
                  catch (Exception ex) {
                        ex.printStackTrace();
                  }
            }
            return true;
      }

      private boolean load_data(String file,double[][] dat)
      {
            int j, padroes;
            String [] values;

            File fso = new File(file);
            if(!fso.exists())
            {
                  messageStatus += "Err000 - Arquivo não pode ser
localizado";
            }
            else
            {
                  try {
                  padroes = 0;
                  BufferedReader sr = new BufferedReader(new
FileReader(file));
                  String line = sr.readLine();
                  while (line != null && line != "")
                  {
                        values = line.split("  ");
                        for (j=0; j<= values.length - 2; j++)    //!!!!
Era menor ou igual antes
                        {
```

```
                              dat[ padroes] [j] =
Double.parseDouble(values[j+1]);
                        }
                        padroes++;
                        line = sr.readLine();
                  }
                  sr.close();
                  n_pad = padroes;
                  }
                  catch(Exception ex) {
                        ex.printStackTrace();
                        return false;
                  }
            }
            return true;
      }

      protected boolean load_net(String file)
      {
            int i,j,l;
            String [] values;

            File fso = new File(file);
            if(!fso.exists())
            {
                  messageStatus += "Err000 - Arquivo não pode ser
localizado";
            }
            else
            {
                  try{
                  BufferedReader sr = new BufferedReader(new
FileReader(file));
                  for (i=0; i<n_neur; i++)
                        for (l=0; l<=max_range; l++)
                        {
                              String line = sr.readLine();
                              values = line.split(" ");
                              for (j=0; j<n_entr; j++)
                              {
                                    c1 [i][j][l] =
Double.parseDouble(values[j]);
                              }
                        }
                  sr.close();
                  }
                  catch(Exception ex) {
                        ex.printStackTrace();
                        return false;
                  }
            }
            return true;
      }

      protected boolean save_net(String file)
      {
            int i,j,l;
```

```
            double cell;
            BufferedWriter sw = null;
            try
            {
                  sw = new BufferedWriter(new FileWriter(file));
            }
            catch(Exception err)
            {
                  messageStatus += "Err000 - " + err.getMessage();
            }
            for (i=0; i<n_neur; i++)
                  for (l=0; l<=max_range; l++)
                  {
                        String line = "";
                        for (j=0; j<n_entr; j++)
                        {
                              cell = c1 [i][j][l];
                              line += Double.toString(cell) + " ";
                        }
                        try {
                              sw.write(line+"\n");
                        } catch (IOException e) {
                              e.printStackTrace();
                  }
            try {
                  sw.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
            return true;
      }

      protected boolean fanInit()
      {
            //verifica a existencia dos arquivos de treino, teste e
configuraçao
            if(fileTreino == null || fileTeste == null || fileTreino ==
"" || fileTeste == "")
            {
                  messageStatus += "Err000 - Nome de arquivo inválido";
                  return false;
            }

            File fso = new File(fileTreino);
            if(!fso.exists())
            {
                  messageStatus += "Err000 - Erro ao localizar arquivo de
treino";
                  return false;
            }
            fso = new File(fileTeste);

            if(!fso.exists())
            {
                  messageStatus += "Err000 - Erro ao localizar arquivo de
teste";
                  return false;
```

```
            }

            if (!num_car(n_entr))
            {
                  return false;
            }
            //inicializa a rede randomicamente
            inicializa();

            if (!load_data(fileTreino, trainning))
            {
                  return false;
            }
            if (!load_data(fileTeste, test))
            {
                  return false;
            }

            //Calcula nx[]
            if (!normalized)
            {
                  if (!normal(fileTreino))
                  {
                        return false;
                  }
                  normalized = true;
            }

            //preenche os arrays com  os padroes normalizados
            normalizarPadroes(test, testNorm);
            normalizarPadroes(trainning, trainningNorm);

            calcularConjuntosDifusos(testNorm, testConjDifuso,
testSomaConjDifuso);
            calcularConjuntosDifusos(trainningNorm, trainningConjDifuso,
trainningSomaConjDifuso);

            for (int i = 0; i < MAX_CAR; i++)
            {
                  usar_car [i] = true;
            }
            for (int i = 0; i < MAX_CLA; i++)
            {
                  peso_neu [i] = 500.0;
            }
            return true;
      }

      private void calcularConjuntosDifusos(double[][] padroesNorm,
double[][][] padroesConjDifuso, double[] padroesSomaConjDifuso) {
            int padrao = 0;
            int iip;
            while (padrao < n_pad){
                  for (int j=0; j<n_entr; j++)
                  {
                        int k = 0;
                        double sidx = 0;
```

```
                        for (int i=-d[j]; i<=d[j]; i++)
                        {
                              iip = (int) padroesNorm[padrao][j] + i;
                              if (iip<0) iip = 0;
                              if (iip>max_range-1) iip = max_range-1;
                              padroesConjDifuso[padrao][j][k] =
triang(iip,padroesNorm[padrao][j],d[j]);
                              sidx += padroesConjDifuso[padrao][j][k];
                              k++;
                        }
                        padroesSomaConjDifuso[padrao] = sidx;
                  }
                  padrao++;
            }

      }
      private void normalizarPadroes(double[][] padroes, double[][]
padroesNorm) {
            int padrao = 0;
            while (padrao < n_pad)
            {
                  for(int k=0; k<n_entr; k++)   {
                        if (nx[k] != 0)
                              padroesNorm[padrao][k] =
(Math.abs(padroes[padrao][k]/nx[k]))*(max_range-1);
                        else
                              padroesNorm[padrao][k] = 0;
                  }
                  padrao++;
            }
      }

      public static void main(String[] args) {
            fastFAN rede = new fastFAN();
            rede.fileTeste = ".//src//data//cromo_ts.dat";
            rede.fileTreino = ".//src//data//cromo_tr.dat";
            rede.fanInit();
            int epoca = 1;
            while(/*epoca <= 100*/true) {
                  rede.cromot(false,0);
                  rede.cromo();
                  System.out.println(epoca + "\t" + rede.s_harm*100+
"\t"+rede.max_tot *100+"\t"+rede.m_certos+ "\t"+ rede.m_errados);
                  epoca++;
            }

      }
}
```

2.2 PACKAGE JFAN.EXCEPTIONS;

2.2.1 JfanFormatoArquivoException.Java

```
package jfan.exceptions;
```

```
import java.io.IOException;

public class JfanFormatoArquivoException extends IOException{
	private static final long serialVersionUID = 1L;
	public JfanFormatoArquivoException(){}
	public JfanFormatoArquivoException(String umaMensagem){
		super(umaMensagem);
	}
	public String getMessage(){
		return "Formato de arquivo não reconhecido, as extensões de
arquivo válidas são apenas [*.txt],[*.dat],[*.xls] e [*.efn]";
	}
}
```

2.2.2 JfanPorcentagemException.Java

```
package jfan.exceptions;

public class JfanPorcentagemException extends Exception {
	private static final long serialVersionUID = -8146973284868245415L;
	private String mensagem = "A porcentagem deve estar entre 0
(inclusive) e 100(inclusive)";

	public JfanPorcentagemException(){}

	public JfanPorcentagemException(String message) {
		mensagem = message;
	}

	@Override
	public String getMessage() {
		return this.mensagem;
	}
}
```

2.3 PACKAGE JFAN.FAN;

2.3.1 IMonitor.Java

```
package jfan.fan;

public interface IMonitor {

	/**
	 * Deve iniciar a execução da rede, em geral esse
	 * método deve iniciar uma thread para manter a
	 * execução em paralelo.
	 */
	public void execute();

	/**
	 * Deve indicar que o monitor deve parar a execução chamada
	 * pelo método execute.
	 */
```

```
	public void stop();

	/**
	 * Deve retornar se o monitor está em execução (true) ou não
	 * (false).
	 * @return True se o monitor está em execução, false se não
	 * estiver executando.
	 */
	public boolean isExecuting();
}
```

2.3.2 ITreinador.Java

```
package jfan.fan;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;

public interface ITreinador {
	public void treinar(ArrayList<IPadrao> p);
	public void setNeuronios(ArrayList<NeuronioFAN> neurons);
	public void setTemperaSimulada(ITemperaSimulada tempera);
}
```

2.3.3 Main.Java

```
package jfan.fan;

public class Main {
	public static void main(String[] args) {
		System.out.println("jfan -- alpha version ");
	}

}
```

2.3.4 MonitorFAN.Java

```
package jfan.fan;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.EventListener;
import java.util.Random;

import javax.swing.event.EventListenerList;

import jfan.exceptions.JfanPorcentagemException;
import jfan.fan.beans.ErrorBean;
import jfan.fan.beans.StatisticBean;
```

```
import jfan.fan.events.FaseRedeListener;
import jfan.fan.events.MelhorAritimeticaEvent;
import jfan.fan.events.MelhorAritimeticaListener;
import jfan.fan.events.MelhorHarmonicaEvent;
import jfan.fan.events.MelhorHarmonicaListener;
import jfan.fan.events.MelhorMaximoMinimoEvent;
import jfan.fan.events.MelhorMaximoMinimoListener;
import jfan.fan.events.RedeParouEvent;
import jfan.fan.events.TrocaEpocaEvent;
import jfan.fan.events.TrocaEpocaListener;
import jfan.fan.events.ValidacaoEvent;
import jfan.fan.events.ValidacaoListener;
import jfan.fan.normalizadores.INormalizator;
import jfan.fan.normalizadores.NormalizatorConfiguration;
import jfan.fan.normalizadores.NormalizatorMax;
import jfan.fan.normalizadores.NormalizatorMaxMean;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.temperas.TemperaAleatoria;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.arquivos.CarregaNeuronioXML;
import jfan.io.arquivos.GerenciadorPadroes;
import jfan.io.arquivos.PersistorDados;

public class MonitorFAN implements IMonitor {

	private Object lockNumPadroesTreinamento = new Object();

	private EventListenerList listenerList = new EventListenerList();

	private RedeFAN rede = null;

	private GerenciadorPadroes gerenciadorPadroes = null;

	private RunTreinamento runTreinamento = new RunTreinamento();

	private Thread threadTreinamento = null;

	private int epoca = 0;

	private int numEpocasTreinar = -1;

	private int cadaEpocaValidar = 0;

	private StoredNetworkType tipoRedeValidacao =
StoredNetworkType.BestHarmonicMean;


	private int numClasses = -1;

	private final ArrayList<NeuronioFAN> neuroniosMelhorHarmonica = new
ArrayList<NeuronioFAN>();
```

```
	private final ArrayList<NeuronioFAN> neuroniosMelhorMaximoMinimo =
new ArrayList<NeuronioFAN>();

	private final ArrayList<NeuronioFAN> neuroniosMelhorMediaAritmetica
= new ArrayList<NeuronioFAN>();

	private float melhorMediaHarmonica = 0;

	private float melhorMaximoDoMinimo = 0.f;

	private float melhorMediaAritimetica = 0;

	private float mediaAritimetica = 0;
	private float mediaHarmonica;

	private float maximoDoMinimo;

	int erros;

	int[][] errosClasses;

	float[] mediaAritimeticaClasses;

	public ArrayList<IPadrao> conjuntoTreinamento = null;

	public ArrayList<IPadrao> conjuntoTeste = null;

	public ArrayList<IPadrao> conjuntoValidacao = null;

	public ArrayList<IPadrao> conjuntoClassificacao = null;
	private int chanceRenormalizar = 0;

	private int cadaEpocaRenormalizar = 0;

	private int cadaEpocaStepTempora = 0;
	private int cadaEpocaReiniciarTempora = 0;

	private ITemperaSimulada tempera = null;
	private int numPadroesTreinamento = 0;

	private ArrayList<IPadrao> PadroesTemp = null;

	private int[] PadroesJaSaiu = null;
	private int cadaEpocaEmbaralharPadroes = 0;

	private Random random = new Random();

	private int[][] matrizConfusao;

	private class RunTreinamento implements Runnable {
		private boolean keepGoing = true;
		private int contatorEpocas = 0;
		public void run() {
			while (keepGoing) {
				if (numEpocasTreinar < 1 || contatorEpocas <
numEpocasTreinar) {
					treinar();
```

```
                        contatorEpocas++;
                    }
                    else {
                        stop();
                    }
                }
            }

            public void stop() {
                keepGoing = false;
                contatorEpocas = 0;
                fireRedeParouEvent(new RedeParouEvent(epoca));
            }
        }
        public MonitorFAN (RedeFAN rede, GerenciadorPadroes gerenciador,
ITreinador trainer) throws Exception {
            this.rede = rede;
            this.gerenciadorPadroes = gerenciador;
            rede.setTreinador(trainer);
            this.setarConjuntos();
        }
        public MonitorFAN (RedeFAN rede, GerenciadorPadroes gerenciador)
throws Exception {
            this.rede = rede;
            this.gerenciadorPadroes = gerenciador;
            ITreinador trein = new TreinoFAN();
            trein.setNeuronios(rede.getNeuroniosAtuais());
            rede.setTreinador(trein);
            this.setarConjuntos();
        }

        public MonitorFAN(GerenciadorPadroes gerenciador) throws Exception
{
            this(new RedeFAN(6,100), gerenciador,null);
            ITreinador trein = new TreinoFAN();
            trein.setNeuronios(rede.getNeuroniosAtuais());
            rede.setTreinador(trein);
        }
        public void execute() {
            runTreinamento.keepGoing = true;
            threadTreinamento = new Thread(runTreinamento);
            threadTreinamento.start();
        }

        public void stop() {
            runTreinamento.stop();
        }

        public boolean isExecuting() {
            return threadTreinamento.isAlive();
        }

        public void treinar() {
            epoca++;
            if (this.numPadroesTreinamento <= 0) {
                rede.treinar();
            }
```

```
            else {
                  synchronized (lockNumPadroesTreinamento) {
                        int indice;
                        boolean continuar = false;

                        this.PadroesTemp.clear();
                        Arrays.fill(this.PadroesJaSaiu, -1);

                        for(int i = 0; i < this.numPadroesTreinamento;
i++) {
                              loopInterno:
                              do {
                                    indice =
random.nextInt(this.numPadroesTreinamento);
                                    continuar = false;
                                    for(int ii : this.PadroesJaSaiu) {

                                          if(ii == -1) {
                                                break loopInterno;
                                          }
                                          if(ii == indice) {
                                                continuar = true;
                                                continue loopInterno;
                                          }
                                    }
                              }while(continuar);

      this.PadroesTemp.add(this.conjuntoTreinamento.get(indice));
                              this.PadroesJaSaiu[i] = indice;
                        }
                  }
                  rede.treinar(PadroesTemp);

            }
            //testar
            ErrorBean bean = rede.testar();
            calcularEstatisticas(bean, true);

            fireTrocaEpocaEvent(new TrocaEpocaEvent(epoca,
mediaHarmonica, mediaAritimetica,
maximoDoMinimo,bean.getMatrizConfusao()));

            if (cadaEpocaValidar > 0 && epoca % cadaEpocaValidar == 0 &&
conjuntoValidacao.size() != 0) {
                  StatisticBean sb = validar(tipoRedeValidacao);
                  fireValidacaoEvent(new
ValidacaoEvent(epoca,tipoRedeValidacao,sb) );
            }
            if(this.chanceRenormalizar > 0 && this.chanceRenormalizar <=
100 && random.nextInt(101) <= this.chanceRenormalizar) {
                  for (NeuronioFAN n : this.rede.getNeuroniosAtuais())
                        n.normalizar();
            }
            if(this.cadaEpocaRenormalizar > 0 && this.epoca %
this.cadaEpocaRenormalizar == 0) {
                  for (NeuronioFAN n : this.rede.getNeuroniosAtuais())
                        n.normalizar();
```

```
            }
            if(this.cadaEpocaEmbaralharPadroes > 0 && this.epoca %
this.cadaEpocaEmbaralharPadroes == 0) {
                  Collections.shuffle(this.conjuntoTreinamento, random);
            }
            if(this.cadaEpocaStepTempora > 0 && this.tempera != null &&
this.epoca % this.cadaEpocaStepTempora == 0) {
                  tempera.step();
            }
            if(this.cadaEpocaReiniciarTempora > 0 && this.tempera != null
&& this.epoca % this.cadaEpocaReiniciarTempora == 0) {
                  tempera.reset();
            }
      }

      public void setClasses(int[] classes) {
            numClasses = classes.length;
            mediaAritimeticaClasses = new float[numClasses];
            rede.setDominioClasses(classes);
      }

      private void calcularEstatisticas(ErrorBean bean, boolean
gravarMelhores) {
                  calcularMediaAritmeticaGeral(bean, gravarMelhores);
                  calcularMediaAritmeticaClasses(bean);
                  calcularMaximoDoMinimo(bean, gravarMelhores);
                  calcularMediasHarmonicas(bean, gravarMelhores);
                  this.matrizConfusao = bean.getMatrizConfusao();
      }

      private StatisticBean calcularEstatisticaToStatisticBean(ErrorBean
errorBean) {
            StatisticBean b = new StatisticBean();
            b.setAritmetica(calcularMediaAritmetica(errorBean));
            b.setHarmonica(calcularMediasHarmonicasToFloat(errorBean));
            b.setMaxmin(calcularMaximoDoMinimoToFloat(errorBean));
            b.setMatrizConfusao(errorBean.getMatrizConfusao());
            b.calcularAcertosErrosApartirMatrizNeural();
            return b;
      }

      private void calcularMediaAritmeticaGeral(ErrorBean bean, boolean
gravarMelhor) {
            mediaAritimetica = calcularMediaAritmetica(bean);

            if (gravarMelhor && mediaAritimetica >
melhorMediaAritimetica) {
                  melhorMediaAritimetica = mediaAritimetica;
                  neuroniosMelhorMediaAritmetica.clear();
                  for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
                        neuroniosMelhorMediaAritmetica.add(n.clone());
                  }
                  //disparar evento de media aritmetica superada
                  fireMelhorAritimeticaEvent(new
MelhorAritimeticaEvent(mediaAritimetica,bean.getMatrizConfusao()));
            }
      }
```

```
	private float calcularMediaAritmetica(ErrorBean bean) {
		float mediaAri = 0;

		if (bean.getTipo() != null &&
bean.getTipo().equals("validacao")) {
			mediaAri =
((float)(bean.getErros())/((float)this.conjuntoValidacao.size()))*100;
		}
		else {
			mediaAri =
((float)(bean.getErros())/((float)this.conjuntoTeste.size()))*100;
		}

		mediaAri = 100 - mediaAri;
		return mediaAri;
	}

	private float[] calcularMediaAritmeticaClassesToArray(ErrorBean
bean) {
		int fim = numClasses;
		float[] mediaAritimeticaClasses = new float[fim];
		for (int i = 0; i < fim;i++) {
			mediaAritimeticaClasses[i] = (((float)
bean.getErrosClasses()[i])/(float)bean.getContagemClasses()[i])*100;
			mediaAritimeticaClasses[i] = 100.0f -
mediaAritimeticaClasses[i];
		}
		return mediaAritimeticaClasses;
	}

	private void calcularMediaAritmeticaClasses(ErrorBean bean) {
		int fim = numClasses;
		for (int i = 0; i < fim;i++) {
			mediaAritimeticaClasses[i] = (((float)
bean.getErrosClasses()[i])/(float)bean.getContagemClasses()[i])*100;
			mediaAritimeticaClasses[i] = 100.0f -
mediaAritimeticaClasses[i];
		}
	}

	private void calcularMaximoDoMinimo(ErrorBean bean, boolean
gravarMelhor){
	  //Seleciona o menor acerto
		float menor = 100;
		for(float f : mediaAritimeticaClasses) {
			if (f < menor) {
				menor = f;
			}
		}
		maximoDoMinimo = menor;
		if(gravarMelhor && melhorMaximoDoMinimo < menor) {
			melhorMaximoDoMinimo = menor;
			neuroniosMelhorMaximoMinimo.clear();
			for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
				neuroniosMelhorMaximoMinimo.add(n.clone());
			}
```

```
                this.fireMelhorMaximoMinimoEvent(new
MelhorMaximoMinimoEvent(melhorMaximoDoMinimo,bean.getMatrizConfusao()));
            }
      }

      private float calcularMaximoDoMinimoToFloat(ErrorBean bean){
        //Seleciona o menor acerto
            float menor = 100;
            float[] mediaAritimeticaClasses =
calcularMediaAritmeticaClassesToArray(bean);
            for(float f : mediaAritimeticaClasses) {
                  if (f < menor) {
                        menor = f;
                  }
            }
            return menor;
      }

      private void calcularMediasHarmonicas(ErrorBean bean, boolean
gravarMelhor) {
            mediaHarmonica = 0;
            int fim = numClasses;
            for (int count = 0; count < fim ;count++) {
                  mediaHarmonica += mediaAritimeticaClasses[count];
            }
            mediaHarmonica /= mediaAritimeticaClasses.length;
            if (gravarMelhor && this.mediaHarmonica >
this.melhorMediaHarmonica) {
                  this.melhorMediaHarmonica = this.mediaHarmonica;
                  neuroniosMelhorHarmonica.clear();
                  for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
                        neuroniosMelhorHarmonica.add(n.clone());
                  }
                  this.fireMelhorHarmonicaEvent(new
MelhorHarmonicaEvent(melhorMediaHarmonica,bean.getMatrizConfusao()));
            }
      }

      private float calcularMediasHarmonicasToFloat(ErrorBean bean) {
            float mediaHarmonica = 0;
            float[] mediaAritimeticaClasses =
calcularMediaAritmeticaClassesToArray(bean);
            int fim = numClasses;
            for (int count = 0; count < fim ;count++) {
                  mediaHarmonica += mediaAritimeticaClasses[count];
            }
            mediaHarmonica /= mediaAritimeticaClasses.length;

            return mediaHarmonica;

      }

      public float getMelhorMediaHarmonica() {
            return this.melhorMediaHarmonica;
      }

      public float getMelhorMaximoDoMinimo() {
```

```
		return this.melhorMaximoDoMinimo;
	}

	public float getMaximoDoMinimo() {
		return this.maximoDoMinimo;
	}

	public float getMediaHarmonica() {
		return this.mediaHarmonica;
	}

	public float getMediaAritmetica() {
		return this.mediaAritimetica;
	}


	public float getMelhorMediaAritimetica() {
		return melhorMediaAritimetica;
	}

	public long getEpoca(){
		return this.epoca;
	}
	public void setPrioridadeThreadTreinamento(int priority) {
		threadTreinamento.setPriority(priority);
	}
	public GerenciadorPadroes getGerenciadorPadroes() {
		return this.gerenciadorPadroes;
	}

	public void ativarPesosAleatorios(int pesoBase) {
		rede.onRandomizarPesos(pesoBase);
	}

	public void desativarPesosAleatorios() {
		rede.offRandomizarPesos();
	}

	public void setChanceRenormalizacao(int i) throws
JfanPorcentagemException {
		if (i > 100 || i < 0) {
			throw new JfanPorcentagemException();
		}
		this.chanceRenormalizar = i;
	}

	public void setCadaEpocaRenormalizar(int i) {
		this.cadaEpocaRenormalizar = i;
	}

	public void setCadaEpocaEmbaralharPadroes(int i) {
		this.cadaEpocaEmbaralharPadroes = i;
	}

	public void setNumeroPadroesTreinamento(int i) {
		synchronized (lockNumPadroesTreinamento) {
			this.numPadroesTreinamento = i;
```

```
                if (i > 0) {
                        this.PadroesTemp = new ArrayList<IPadrao>(i);
                        this.PadroesJaSaiu = new int[i];
                }
        }
    }

    public void setCadaEpocaStepTempera(int i) {
        this.cadaEpocaStepTempora = i;
    }

    public void setCadaEpocaReiniciarTempera(int i) {
        this.cadaEpocaReiniciarTempora = i;
    }

    public void setTemperaSimulada(ITemperaSimulada t) {
        synchronized (rede.getTreinador()) {
            this.tempera = t;
            rede.setTemperaSimulada(t);
            rede.getTreinador().setTemperaSimulada(t);
        }
    }

    public void removerTemperaSimulada() {
        synchronized (rede.getTreinador()) {
            this.tempera = null;
            rede.setTemperaSimulada(null);
            rede.getTreinador().setTemperaSimulada(this.tempera);
        }
    }
    private void setarConjuntos() throws Exception {

    this.rede.setConjuntoTreinamento(this.gerenciadorPadroes.getConjunt
oTreinamentoRede());

    this.rede.setConjuntoTeste(this.gerenciadorPadroes.getConjuntoTeste
Rede());

    this.rede.setConjuntoValidacao(this.gerenciadorPadroes.getConjuntoV
alidacaoRede());

    this.rede.setConjuntoClassificacao(this.gerenciadorPadroes.getConju
ntoClassificacaoRede());
            this.conjuntoTreinamento =
this.gerenciadorPadroes.getConjuntoTreinamentoRede();
            this.conjuntoTeste =
this.gerenciadorPadroes.getConjuntoTesteRede();
            this.conjuntoClassificacao =
this.gerenciadorPadroes.getConjuntoClassificacaoRede();
            this.conjuntoValidacao =
this.gerenciadorPadroes.getConjuntoValidacaoRede();
        }
        public void addTrocaEpocaListener(TrocaEpocaListener listener)
        {
           listenerList.add(TrocaEpocaListener.class, listener);
        }
```

```
      public void removeTrocaEpocaListener(TrocaEpocaListener listener)
      {
         listenerList.remove(TrocaEpocaListener.class, listener);
      }

      public void fireTrocaEpocaEvent(TrocaEpocaEvent event)
      {
         EventListener[] listeners =
listenerList.getListeners(TrocaEpocaListener.class);
         for (EventListener l : listeners)
            ((TrocaEpocaListener) l).trocouEpoca(event);
      }
      public void addMelhorHarmonicaListener(MelhorHarmonicaListener
listener)
      {
         listenerList.add(MelhorHarmonicaListener.class, listener);
      }

      public void removeMelhorHarmonicaListener(MelhorHarmonicaListener
listener)
      {
         listenerList.remove(MelhorHarmonicaListener.class, listener);
      }

      public void fireMelhorHarmonicaEvent(MelhorHarmonicaEvent event)
      {
         EventListener[] listeners =
listenerList.getListeners(MelhorHarmonicaListener.class);
         for (EventListener l : listeners)
            ((MelhorHarmonicaListener)
l).trocouMelhorMediaHarmonica(event);
      }
      public void addMaximoMinimoListener(MelhorMaximoMinimoListener
listener)
      {
         listenerList.add(MelhorMaximoMinimoListener.class, listener);
      }

      public void
removeMelhorMaximoMinimoListener(MelhorMaximoMinimoListener listener)
      {
         listenerList.remove(MelhorMaximoMinimoListener.class, listener);
      }

      public void fireMelhorMaximoMinimoEvent(MelhorMaximoMinimoEvent
event)
      {
         EventListener[] listeners =
listenerList.getListeners(MelhorMaximoMinimoListener.class);
         for (EventListener l : listeners)
            ((MelhorMaximoMinimoListener)
l).trocouMelhorMaximoMinimo(event);
      }
      public void addMelhorAritimeticaListener(MelhorAritimeticaListener
listener)
      {
         listenerList.add(MelhorAritimeticaListener.class, listener);
```

```
	}

	public void
removeMelhorAritimeticaListener(MelhorAritimeticaListener listener)
	{
	   listenerList.remove(MelhorAritimeticaListener.class, listener);
	}

	public void fireMelhorAritimeticaEvent(MelhorAritimeticaEvent
event)
	{
	   EventListener[] listeners =
listenerList.getListeners(MelhorAritimeticaListener.class);
	   for (EventListener l : listeners)
	      ((MelhorAritimeticaListener)
l).trocouMelhorMediaAritimetica(event);
	}
	public void addValidacaoListener(ValidacaoListener listener)
	{
	   listenerList.add(ValidacaoListener.class, listener);
	}

	public void removeValidacaoListener(ValidacaoListener listener)
	{
	   listenerList.remove(ValidacaoListener.class, listener);
	}

	public void fireValidacaoEvent(ValidacaoEvent event)
	{
	   EventListener[] listeners =
listenerList.getListeners(ValidacaoListener.class);
	   for (EventListener l : listeners)
	      ((ValidacaoListener) l).validou(event);
	}
	public void addFaseRedeListener(FaseRedeListener listener)
	{
	   listenerList.add(FaseRedeListener.class, listener);
	}

	public void removeFaseRedeListener(FaseRedeListener listener)
	{
	   listenerList.remove(FaseRedeListener.class, listener);
	}

	public void fireRedeParouEvent(RedeParouEvent event)
	{
	   EventListener[] listeners =
listenerList.getListeners(FaseRedeListener.class);
	   for (EventListener l : listeners)
	      ((FaseRedeListener) l).redeParou(event);
	}
	public int getNumeroPadroesPorcentagem(int porcentagem) throws
Exception {
	    if (porcentagem <= 0 || porcentagem > 100) {
	          throw new JfanPorcentagemException("A porcentagem deve
estar entre 1 e 100");
	    }
```

```
            int padroes =
gerenciadorPadroes.getConjuntoTreinamentoRede().size();
            padroes *= porcentagem;
            padroes /= 100;
            return padroes;
      }
      public ArrayList<NeuronioFAN> getNeuroniosAtuais(){
            return rede.getNeuroniosAtuais();
      }

      public ArrayList<NeuronioFAN> getNeuroniosMelhorHarmonica(){
            return neuroniosMelhorHarmonica;
      }

      public ArrayList<NeuronioFAN> getNeuroniosMelhorAritmetica(){
            return neuroniosMelhorMediaAritmetica;
      }

      public ArrayList<NeuronioFAN> getNeuroniosMelhorMaximoMinimo(){
            return neuroniosMelhorMaximoMinimo;
      }

      public void setNeuronios(ArrayList<NeuronioFAN> neuronios) {
            rede.setNeuronios(neuronios);
      }
      public StatisticBean testar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes) {
            ErrorBean bean = rede.testar(neuronios,padroes);
            return calcularEstatisticaToStatisticBean(bean);
      }

      public StatisticBean testar(ArrayList<NeuronioFAN> neuronios) {
            return testar(neuronios, this.conjuntoTeste);
      }


      public StatisticBean testar() {
            return testar(this.getNeuroniosAtuais(), conjuntoTeste);
      }
      public StatisticBean validar(ArrayList<NeuronioFAN> neuronios) {
            return validar(neuronios, this.conjuntoValidacao);
      }

      public StatisticBean validar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> conjunto) {
            ErrorBean bean = rede.validar(neuronios,conjunto);
            return calcularEstatisticaToStatisticBean(bean);
      }

      public StatisticBean validar(StoredNetworkType tipoRede) {
            StatisticBean b = null;
            switch (tipoRede) {
            case Actual:
                  b = validar(this.getNeuroniosAtuais(),
conjuntoValidacao);
                  break;
```

```
            case BestAritmethicMean:
                  b = validar(this.getNeuroniosMelhorAritmetica(),
conjuntoValidacao);
                  break;
            case BestHarmonicMean:
                  b = validar(this.getNeuroniosMelhorHarmonica(),
conjuntoValidacao);
                  break;
            case BestMaxMin:
                  b = validar(this.getNeuroniosMelhorMaximoMinimo(),
conjuntoValidacao);
                  break;
            }
            return b;
      }

      public void classificar(StoredNetworkType tipoRede) {
            switch (tipoRede) {
            case Actual:
                  classificar(getNeuroniosAtuais());
                  break;
            case BestAritmethicMean:
                  classificar(neuroniosMelhorMediaAritmetica);
                  break;
            case BestHarmonicMean:
                  classificar(neuroniosMelhorHarmonica);
                  break;
            case BestMaxMin:
                  classificar(neuroniosMelhorMaximoMinimo);
                  break;
            }
      }

      public void classificar(ArrayList<NeuronioFAN> neuronios) {
            rede.classificar(neuronios);
      }

      public void classificar(ArrayList<NeuronioFAN> neuronios,
CarregaNeuronioXML cnx) throws Exception  {
            ArrayList<NeuronioFAN> arr = cnx.getNeuroniosFAN();
            int fim = arr.size();
            int[] classes = new int[fim];
            for(int i = 0; i < fim; i++) {
                  classes[i] = arr.get(i).getClasseAssociada();
            }
            RecalculadorClasses.indexarNeuronios(classes, arr);

        //Normalizar os dados
        NormalizatorConfiguration config = new
NormalizatorConfiguration(cnx.getMaximos(),cnx.getMinimos(),
cnx.getMedias());

        NormalizatorTypes tipo = cnx.getTipoNormalizacao();

        INormalizator norm;
            if (tipo == NormalizatorTypes.NormalizatorMaxMin) {
                  norm = new NormalizatorMaxMin(config);
```

```
            }
            else if (tipo == NormalizatorTypes.NormalizatorMaxMean) {
                  norm = new NormalizatorMaxMean(config);
            }
            else{
                  norm = new NormalizatorMax(config);
            }

      norm.normalizar(gerenciadorPadroes.getConjuntoClassificacaoRede());

            CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(cnx.getRaioDifuso(),
cnx.getNumeroSuporteConjuntoDifuso());

      calc.calcularCaracteristicasFAN(gerenciadorPadroes.getConjuntoClass
ificacaoRede());
            rede.classificar(neuronios);
            gerenciadorPadroes.atualizarConjuntoClassificacao();
      }

      @SuppressWarnings("unchecked")
      public static void main(String[] args) {
            try {
                  GerenciadorPadroes gp = new GerenciadorPadroes();
                  File f = new
File("//home//filipe//Desktop//treinoFcontrol.dat");
                  gp.adicionarConjuntoTreinamento(f.getAbsolutePath()," 
", 0, 0, 0, 0, 0);

      gp.getConjuntoTesteRede().addAll(gp.getConjuntoTreinamentoRede());
                  double[] max =
DetectorMaxMinMean.procurarMaximos(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
                  double[] min =
DetectorMaxMinMean.procurarMinimos(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
                  double[] mean =
DetectorMaxMinMean.procurarMedias(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
                  NormalizatorMaxMin np = new NormalizatorMaxMin(max,
min);
                  np.normalizarThreaded(gp.getConjuntoTreinamentoRede());
                  np.normalizarThreaded(gp.getConjuntoTesteRede());
                  CalcularCaracteristicasFANBatch c = new
CalcularCaracteristicasFANBatch(6,100);

      c.calcularCaracteristicasFAN(gp.getConjuntoTreinamentoRede());

      c.calcularCaracteristicasFAN(gp.getConjuntoTesteRede());

                  MonitorFAN monitor = new MonitorFAN(gp);

                  monitor.setClasses(gp.getClassesConjuntoTreinamento());

                  monitor.rede.setTreinador(new TreinoQueue());

                  TemperaAleatoria t = new TemperaAleatoria();
```

```
                monitor.setTemperaSimulada(t);
                monitor.setCadaEpocaStepTempera(100);
                monitor.setCadaEpocaReiniciarTempera(1300);
                monitor.setCadaEpocaEmbaralharPadroes(20);
                monitor.setCadaEpocaRenormalizar(2400);
                Thread.currentThread().setPriority(10);

                int j = 0;
                while (j<100) {
                      monitor.treinar();
                      System.out.print(monitor.getEpoca() + "\t");

System.out.print(monitor.getMediaHarmonica()+"\t");

System.out.print(monitor.getMaximoDoMinimo()+"\t");

System.out.print(monitor.getMelhorMaximoDoMinimo()+"\t");

System.out.print(monitor.getMelhorMediaHarmonica()+"\t");

                      System.out.println();
                      j++;
                }

                PersistorDados p = new PersistorDados();

                monitor.melhorMediaAritimetica = 0;
                monitor.mediaAritimetica = 0;
                monitor.epoca = 0;
                monitor.testar(monitor.neuroniosMelhorMaximoMinimo);
                System.out.print(monitor.getEpoca() + "\t");
                System.out.println(monitor.melhorMaximoDoMinimo+"\t");


          } catch (Exception e) {
                e.printStackTrace();
          }

}

public int[][] getMatrizConfusao() {
      return matrizConfusao.clone();
}

public void setNumeroEpocasTreinar(int i) {
      numEpocasTreinar = i;
}

public void setCadaEpocaValidar(int i) {
      cadaEpocaValidar = i;
}

public void setTipoRedeValidar(StoredNetworkType networkType) {
      tipoRedeValidacao = networkType;
}

public StoredNetworkType getTipoRedeValidacao() {
```

```
            return tipoRedeValidacao;
      }

      public int[] getDominioClasses() {
            return rede.getDominioClasses();
      }

}
```

2.3.5 MonitorTeste.Java

```
package jfan.fan;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.EventListener;
import java.util.Random;

import javax.swing.event.EventListenerList;

import jfan.exceptions.JfanPorcentagemException;
import jfan.fan.beans.ErrorBean;
import jfan.fan.events.MelhorAritimeticaEvent;
import jfan.fan.events.MelhorAritimeticaListener;
import jfan.fan.events.MelhorHarmonicaEvent;
import jfan.fan.events.MelhorHarmonicaListener;
import jfan.fan.events.MelhorMaximoMinimoEvent;
import jfan.fan.events.MelhorMaximoMinimoListener;
import jfan.fan.events.TrocaEpocaEvent;
import jfan.fan.events.TrocaEpocaListener;
import jfan.fan.normalizadores.NormalizatorMaxMin;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.temperas.TemperaAleatoria;
import jfan.fan.utilitarias.CalcularCaracteristicasFANBatch;
import jfan.fan.utilitarias.DetectorMaxMinMean;
import jfan.io.arquivos.CarregaNeuronioXML;
import jfan.io.arquivos.GerenciadorPadroes;

public class MonitorTeste implements IMonitor {

      private Object lockNumPadroesTreinamento = new Object();

      private EventListenerList listenerList = new EventListenerList();
      private RedeFAN rede = null;

      private GerenciadorPadroes gerenciadorPadroes = null;

      private ITreinador treinador = null;


      private RunTreinamento runTreinamento = new RunTreinamento();

      private Thread threadTreinamento = null;
```

```
	private int epoca = 0;

	private int numClasses;

	private final ArrayList<NeuronioFAN> neuroniosMelhorHarmonica = new
ArrayList<NeuronioFAN>();

	private final ArrayList<NeuronioFAN> neuroniosMelhorMaximoMinimo =
new ArrayList<NeuronioFAN>();

	private final ArrayList<NeuronioFAN> neuroniosMelhorMediaAritmetica
= new ArrayList<NeuronioFAN>();

	private float melhorMediaHarmonica = 0;

	private float melhorMaximoDoMinimo = 0;

	private float melhorMediaAritimetica = 0;

	private float mediaAritimetica = 0;

	private float mediaHarmonica;

	private float maximoDoMinimo;

	int erros;


	int[][] errosClasses;

	float[] mediaAritimeticaClasses;

	public ArrayList<IPadrao> conjuntoTreinamento = null;

	public ArrayList<IPadrao> conjuntoTeste = null;

	public ArrayList<IPadrao> conjuntoValidacao = null;

	public ArrayList<IPadrao> conjuntoClassificacao = null;
	private int chanceRenormalizar = 0;
	private int cadaEpocaRenormalizar = 0;
	private int cadaEpocaStepTempora = 0;
	private int cadaEpocaReiniciarTempora = 0;
	private ITemperaSimulada tempera = null;
	private int numPadroesTreinamento = 0;
	private ArrayList<IPadrao> PadroesTemp = null;
	private int[] PadroesJaSaiu = null;
	private int cadaEpocaEmbaralharPadroes = 0;

	private Random random = new Random();

	private class RunTreinamento implements Runnable {
		private boolean keepGoing = true;

		public void run() {
			while (keepGoing) {
```

```
                        treinar();
                  }
            }

            public void stop() {
                  keepGoing = false;
            }
      }
      public MonitorTeste (RedeFAN rede, GerenciadorPadroes gerenciador,
ITreinador trainer) throws Exception {
            this.rede = rede;
            this.gerenciadorPadroes = gerenciador;
            this.treinador = trainer;
            rede.setTreinador(trainer);
            this.setarConjuntos();
      }

      public MonitorTeste (RedeFAN rede, GerenciadorPadroes gerenciador)
throws Exception {
            this.rede = rede;
            this.gerenciadorPadroes = gerenciador;
            this.treinador = new TreinoFAN();
            treinador.setNeuronios(rede.getNeuroniosAtuais());
            rede.setTreinador(this.treinador);
            this.setarConjuntos();
      }

      public MonitorTeste(GerenciadorPadroes gerenciador) throws
Exception {
            this(new RedeFAN(6,100), gerenciador,null);
            this.treinador = new TreinoFAN();
            treinador.setNeuronios(rede.getNeuroniosAtuais());
            rede.setTreinador(treinador);
      }

      public void execute() {
            runTreinamento.keepGoing = true;
            rede.setTreinador(treinador);
            threadTreinamento = new Thread(runTreinamento);
            threadTreinamento.start();
      }

      public void stop() {
            runTreinamento.stop();
      }

      public boolean isExecuting() {
            return threadTreinamento.isAlive();
      }

      public void treinar() {
            epoca++;
            if (this.numPadroesTreinamento <= 0) {
                  rede.treinar();
            }
            else {
                  synchronized (lockNumPadroesTreinamento) {
```

```
                        int indice;
                        boolean continuar = false;

                        this.PadroesTemp.clear();
                        Arrays.fill(this.PadroesJaSaiu, -1);

                        for(int i = 0; i < this.numPadroesTreinamento;
i++) {
                              loopInterno:
                              do {
                                    indice =
random.nextInt(this.numPadroesTreinamento);
      continuar = false;
                                    for(int ii : this.PadroesJaSaiu) {

                                          if(ii == -1) {
                                                break loopInterno;
                                          }
                                          if(ii == indice) {
                                                continuar = true;
                                                continue loopInterno;
                                          }
                                    }
                              }while(continuar);

      this.PadroesTemp.add(this.conjuntoTreinamento.get(indice));
                              this.PadroesJaSaiu[i] = indice;
                        }
                  }
                  rede.treinar(PadroesTemp);

            }
            ErrorBean bean = rede.testar();
            calcularEstatisticas(bean);

            fireTrocaEpocaEvent(new TrocaEpocaEvent(epoca,
mediaHarmonica, mediaAritimetica,
maximoDoMinimo,bean.getMatrizConfusao()));

            if(this.chanceRenormalizar > 0 && this.chanceRenormalizar <=
100 && random.nextInt(101) <= this.chanceRenormalizar) {
                  for (NeuronioFAN n : this.rede.getNeuroniosAtuais())
                        n.normalizar();
            }
            if(this.cadaEpocaRenormalizar > 0 && this.epoca %
this.cadaEpocaRenormalizar == 0) {
                  for (NeuronioFAN n : this.rede.getNeuroniosAtuais())
                        n.normalizar();
            }
            if(this.cadaEpocaEmbaralharPadroes > 0 && this.epoca %
this.cadaEpocaEmbaralharPadroes == 0) {
                  Collections.shuffle(this.conjuntoTreinamento, random);
            }
            if(this.cadaEpocaStepTempora > 0 && this.tempera != null &&
this.epoca % this.cadaEpocaStepTempora == 0) {
                  tempera.step();
            }
```

```
            if(this.cadaEpocaReiniciarTempora > 0 && this.tempera != null
&& this.epoca % this.cadaEpocaReiniciarTempora == 0) {
                  tempera.reset();
            }
      }
      public void setClasses(int[] classes) {
            numClasses = classes.length;
            mediaAritimeticaClasses = new float[numClasses];
            rede.setDominioClasses(classes);
      }

      private void calcularEstatisticas(ErrorBean bean) {
            calcularMediaAritmeticaGeral(bean);
            calcularMediaAritmeticaClasses(bean);
            calcularMaximoDoMinimo(bean);
            calcularMediasHarmonicas(bean);
            //fireMatrizConfusaoEvent(new
MatrizConfusaoEvent(epoca,bean.getMatrizConfusao()));
}

private void calcularMediaAritmeticaGeral(ErrorBean bean) {
      mediaAritimetica =
((float)(bean.getErros())/((float)this.conjuntoTeste.size()))*100;
      mediaAritimetica = 100 - this.mediaAritimetica;

      if (mediaAritimetica > melhorMediaAritimetica) {
            melhorMediaAritimetica = mediaAritimetica;
            neuroniosMelhorMediaAritmetica.clear();
            for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
                  neuroniosMelhorMediaAritmetica.add(n.clone());
            }
            fireMelhorAritimeticaEvent(new
MelhorAritimeticaEvent(mediaAritimetica,bean.getMatrizConfusao()));
      }
}

private void calcularMediaAritmeticaClasses(ErrorBean bean) {
      int fim = numClasses;
      for (int i = 0; i < fim;i++) {
            mediaAritimeticaClasses[i] = (((float)
bean.getErrosClasses()[i])/(float)bean.getContagemClasses()[i])*100;
            mediaAritimeticaClasses[i] = 100 -mediaAritimeticaClasses[i];
      }
}

private void calcularMaximoDoMinimo(ErrorBean bean){
    //Seleciona o menor acerto
      float menor = 100;
      for(float f : mediaAritimeticaClasses) {
            if (f < menor) {
                  menor = f;
            }
      }
      maximoDoMinimo = menor;
      if(melhorMaximoDoMinimo < menor) {
            melhorMaximoDoMinimo = menor;
            neuroniosMelhorMaximoMinimo.clear();
```

```
            for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
                  neuroniosMelhorMaximoMinimo.add(n.clone());
            }
            this.fireMelhorMaximoMinimoEvent(new
MelhorMaximoMinimoEvent(melhorMaximoDoMinimo,bean.getMatrizConfusao()));
      }
}

private void calcularMediasHarmonicas(ErrorBean bean) {
      mediaHarmonica = 0;
      int fim = numClasses;
      for (int count = 0; count < fim ;count++) {
            mediaHarmonica += mediaAritimeticaClasses[count];
      }
      mediaHarmonica /= mediaAritimeticaClasses.length;
      if (this.mediaHarmonica > this.melhorMediaHarmonica) {
            this.melhorMediaHarmonica = this.mediaHarmonica;
            neuroniosMelhorHarmonica.clear();
            for(NeuronioFAN n : rede.getNeuroniosAtuais()) {
                  neuroniosMelhorHarmonica.add(n.clone());
            }
            this.fireMelhorHarmonicaEvent(new
MelhorHarmonicaEvent(melhorMediaHarmonica,bean.getMatrizConfusao()));
      }
}


      public float getMelhorMediaHarmonica() {
            return this.melhorMediaHarmonica;
      }

      public float getMelhorMaximoDoMinimo() {
            return this.melhorMaximoDoMinimo;
      }

      public float getMaximoDoMinimo() {
            return this.maximoDoMinimo;
      }

      public float getMediaHarmonica() {
            return this.mediaHarmonica;
      }

      public float getMelhorMediaAritimetica() {
            return melhorMediaAritimetica;
      }

      public long getEpoca(){
            return this.epoca;
      }
      public void setPrioridadeThreadTreinamento(int priority) {
            threadTreinamento.setPriority(priority);
      }
      public GerenciadorPadroes getGerenciadorPadroes() {
            return this.gerenciadorPadroes;
      }
```

```
	public void ativarPesosAleatorios(int pesoBase) {
		rede.onRandomizarPesos(pesoBase);
	}

	public void desativarPesosAleatorios() {
		rede.offRandomizarPesos();
	}

	public void setChanceRenormalizacao(int i) throws
JfanPorcentagemException {
		if (i > 100 || i < 0) {
			throw new JfanPorcentagemException();
		}
		this.chanceRenormalizar = i;
	}

	public void setCadaEpocaRenormalizar(int i) {
		this.cadaEpocaRenormalizar = i;
	}

	public void setCadaEpocaEmbaralharPadroes(int i) {
		this.cadaEpocaEmbaralharPadroes = i;
	}

	public void setNumeroPadroesTreinamento(int i) {
		synchronized (lockNumPadroesTreinamento) {
			this.numPadroesTreinamento = i;
			if (i > 0) {
				this.PadroesTemp = new ArrayList<IPadrao>(i);
				this.PadroesJaSaiu = new int[i];
			}
		}
	}

	public void setCadaEpocaStepTempera(int i) {
		this.cadaEpocaStepTempora = i;
	}

	public void setCadaEpocaReiniciarTempera(int i) {
		this.cadaEpocaReiniciarTempora = i;
	}

	public void setTemperaSimulada(ITemperaSimulada t) {
		synchronized (this.treinador) {
			this.tempera = t;
			rede.setTemperaSimulada(t);
			this.treinador.setTemperaSimulada(t);
		}
	}

	public void removerTemperaSimulada() {
		synchronized (this.treinador) {
			this.tempera = null;
			rede.setTemperaSimulada(null);
			this.treinador.setTemperaSimulada(this.tempera);
		}
	}
```

```
      private void setarConjuntos() throws Exception {

      this.rede.setConjuntoTreinamento(this.gerenciadorPadroes.getConjunt
oTreinamentoRede());

      this.rede.setConjuntoTeste(this.gerenciadorPadroes.getConjuntoTeste
Rede());

      this.rede.setConjuntoValidacao(this.gerenciadorPadroes.getConjuntoV
alidacaoRede());

      this.rede.setConjuntoClassificacao(this.gerenciadorPadroes.getConju
ntoClassificacaoRede());
            this.conjuntoTreinamento =
this.gerenciadorPadroes.getConjuntoTreinamentoRede();
            this.conjuntoTeste =
this.gerenciadorPadroes.getConjuntoTesteRede();
            this.conjuntoClassificacao =
this.gerenciadorPadroes.getConjuntoClassificacaoRede();
            this.conjuntoValidacao =
this.gerenciadorPadroes.getConjuntoValidacaoRede();
      }
      public void addTrocaEpocaListener(TrocaEpocaListener listener)
      {
         listenerList.add(TrocaEpocaListener.class, listener);
      }

      public void removeTrocaEpocaListener(TrocaEpocaListener listener)
      {
         listenerList.remove(TrocaEpocaListener.class, listener);
      }

      public void fireTrocaEpocaEvent(TrocaEpocaEvent event)
      {
         EventListener[] listeners =
listenerList.getListeners(TrocaEpocaListener.class);
         for (EventListener l : listeners)
            ((TrocaEpocaListener) l).trocouEpoca(event);
      }
      public void addMelhorHarmonicaListener(MelhorHarmonicaListener
listener)
      {
         listenerList.add(MelhorHarmonicaListener.class, listener);
      }

      public void removeMelhorHarmonicaListener(MelhorHarmonicaListener
listener)
      {
         listenerList.remove(MelhorHarmonicaListener.class, listener);
      }

      public void fireMelhorHarmonicaEvent(MelhorHarmonicaEvent event)
      {
         EventListener[] listeners =
listenerList.getListeners(MelhorHarmonicaListener.class);
         for (EventListener l : listeners)
```

```
            ((MelhorHarmonicaListener)
l).trocouMelhorMediaHarmonica(event);
      }
      public void addMaximoMinimoListener(MelhorMaximoMinimoListener
listener)
      {
         listenerList.add(MelhorMaximoMinimoListener.class, listener);
      }

      public void
removeMelhorMaximoMinimoListener(MelhorMaximoMinimoListener listener)
      {
         listenerList.remove(MelhorMaximoMinimoListener.class, listener);
      }

      public void fireMelhorMaximoMinimoEvent(MelhorMaximoMinimoEvent
event)
      {
         EventListener[] listeners =
listenerList.getListeners(MelhorMaximoMinimoListener.class);
         for (EventListener l : listeners)
            ((MelhorMaximoMinimoListener)
l).trocouMelhorMaximoMinimo(event);
      }
      public void addMelhorAritimeticaListener(MelhorAritimeticaListener
listener)
      {
         listenerList.add(MelhorAritimeticaListener.class, listener);
      }

      public void
removeMelhorAritimeticaListener(MelhorAritimeticaListener listener)
      {
         listenerList.remove(MelhorAritimeticaListener.class, listener);
      }

      public void fireMelhorAritimeticaEvent(MelhorAritimeticaEvent
event)
      {
         EventListener[] listeners =
listenerList.getListeners(MelhorAritimeticaListener.class);
         for (EventListener l : listeners)
            ((MelhorAritimeticaListener)
l).trocouMelhorMediaAritimetica(event);
      }
      public int getNumeroPadroesPorcentagem(int porcentagem) throws
Exception {
            if (porcentagem <= 0 || porcentagem > 100) {
                  throw new JfanPorcentagemException("A porcentagem deve
estar entre 1 e 100");
            }

            int padroes =
gerenciadorPadroes.getConjuntoTreinamentoRede().size();
            padroes *= porcentagem;
            padroes /= 100;
            return padroes;
```

```
	}
	public ArrayList<NeuronioFAN> getNeuroniosAtuais(){
		return rede.getNeuroniosAtuais();
	}

	public ArrayList<NeuronioFAN> getNeuroniosMelhorHarmonica(){
		return neuroniosMelhorHarmonica;
	}

	public ArrayList<NeuronioFAN> getNeuroniosMelhorAritmetica(){
		return neuroniosMelhorMediaAritmetica;
	}

	public ArrayList<NeuronioFAN> getNeuroniosMelhorMaximoMinimo(){
		return neuroniosMelhorMaximoMinimo;
	}

	public void setNeuronios(ArrayList<NeuronioFAN> neuronios) {
		rede.setNeuronios(neuronios);
	}
	public void testar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes) {
		ErrorBean bean = rede.testar(neuronios,padroes);
		calcularEstatisticas(bean);
	}

	public void testar(ArrayList<NeuronioFAN> neuronios) {
		testar(neuronios, this.conjuntoTeste);
	}

	public void testar() {
		testar(this.getNeuroniosAtuais(), conjuntoTeste);
	}

	@SuppressWarnings("unchecked")
	public static void main(String[] args) {
		try {
			GerenciadorPadroes gp = new GerenciadorPadroes();
			File f = new File(".//src//data//cromo_tr.dat");
			File f2 = new File(".//src//data//cromo_tr.dat");
			gp.adicionarConjuntoTreinamento(f.getAbsolutePath()," 
", 0, 0, 0, 0, 0);
			gp.adicionarConjuntoTeste(f2.getAbsolutePath()," ", 0,
0, 0, 0, 0);
			double[] max =
DetectorMaxMinMean.procurarMaximos(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
			double[] min =
DetectorMaxMinMean.procurarMinimos(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
			double[] mean =
DetectorMaxMinMean.procurarMedias(gp.getConjuntoTreinamentoRede(),
gp.getConjuntoTesteRede());
			NormalizatorMaxMin np = new NormalizatorMaxMin(max,
min);
			np.normalizarThreaded(gp.getConjuntoTreinamentoRede());
			np.normalizarThreaded(gp.getConjuntoTesteRede());
```

```
                    CarregaNeuronioXML cxml = new
CarregaNeuronioXML("testeRede.xml",gp);
                    CalcularCaracteristicasFANBatch c = new
CalcularCaracteristicasFANBatch(6,100);

      c.calcularCaracteristicasFAN(gp.getConjuntoTreinamentoRede());

      c.calcularCaracteristicasFAN(gp.getConjuntoTesteRede());

                    MonitorTeste monitor = new MonitorTeste(gp);
                    monitor.setClasses(gp.getClassesConjuntoTreinamento());

                  monitor.setNeuronios(cxml.getNeuroniosFAN());

                    monitor.treinador = new TreinoQueue();

      monitor.treinador.setNeuronios(monitor.rede.getNeuroniosAtuais());

                    monitor.rede.setTreinador(monitor.treinador);
                    TemperaAleatoria t = new TemperaAleatoria();
                    t.setLimiteMaximo(1.0);
                    t.setLimiteMinimo(0.1);
                    t.setStep(0.1);
                    monitor.setTemperaSimulada(t);
                    monitor.setCadaEpocaStepTempera(100);
                    monitor.setCadaEpocaReiniciarTempera(1300);
                    monitor.setCadaEpocaEmbaralharPadroes(50);
                    monitor.setCadaEpocaRenormalizar(2400);
                    Thread.currentThread().setPriority(10);

            monitor.calcularEstatisticas(monitor.rede.testar());
                        System.out.print(monitor.getEpoca() + "\t");

      System.out.println(monitor.melhorMaximoDoMinimo+"\t");

             } catch (Exception e) {
                    e.printStackTrace();
            }

      }
}
```

2.3.6 NeuronioFAN.Java

```
package jfan.fan;

import java.util.Random;

import jfan.fan.padroes.CaracteristicaFAN;
import jfan.fan.padroes.IPadrao;

public class NeuronioFAN {
      private double[][] matrizNeural;

      private double[] somatorios;
```

```
	private int numCaractertisticas;

	private int J, raioDifuso;

	private int tamanhoArray;

	private int classe;
	private int pesoPenalizacao = 1000;

	public NeuronioFAN(int numeroCaracteristicas, int
suporteConjuntosDifusos,
			int raioDifuso, int classeRepresentada) {
		this.numCaractertisticas = numeroCaracteristicas;
		this.J = suporteConjuntosDifusos;
		this.somatorios = new double[this.numCaractertisticas];
		int inicio = raioDifuso;
		int fim = raioDifuso;
		this.raioDifuso = raioDifuso;
		this.tamanhoArray = this.J + inicio + fim + 1;
	this.matrizNeural = new
double[this.numCaractertisticas][this.tamanhoArray];
		this.classe = classeRepresentada;
		this.inicializar();
	}

	private NeuronioFAN(int numeroCaracteristicas, int
suporteConjuntosDifusos,
			int raioDifuso, int classeRepresentada, boolean
inicializar) {
		this.numCaractertisticas = numeroCaracteristicas;
		this.J = suporteConjuntosDifusos;
		this.somatorios = new double[this.numCaractertisticas];
		int inicio = raioDifuso;
		int fim = raioDifuso;
		this.raioDifuso = raioDifuso;
		this.tamanhoArray = this.J + inicio + fim + 1;
	this.matrizNeural = new
double[this.numCaractertisticas][this.tamanhoArray];
		this.classe = classeRepresentada;
		if (inicializar)
			this.inicializar();
	}

	public void inicializar() {

		Random rand;
		for (int i = 0; i < numCaractertisticas; i++) {
			rand = new Random();
			this.somatorios[i] = 0;
			for (int j = 0; j < this.tamanhoArray; j++) {
				this.matrizNeural[i][j] = rand.nextDouble();
				this.somatorios[i] += this.matrizNeural[i][j];
			}
		}
	}
```

```
	public void inicializarZerado() {
		for (int i = 0; i < numCaractertisticas; i++) {
			this.somatorios[i] = 0;
			for (int j = 0; j < this.tamanhoArray; j++) {
				this.matrizNeural[i][j] = 0;
			}
		}
	}

	public void normalizar() {
		for (int i = 0; i < numCaractertisticas; i++) {
			for (int j = 0; j < this.tamanhoArray; j++) {
				this.matrizNeural[i][j] /= somatorios[i];
			}
		}
	}

	public double determinarForca(IPadrao p) {
		int num_carac;
		double retorno = 1.0;
		double prodSomas;
		CaracteristicaFAN mi;
		double[] conjuntoMi;
		num_carac = p.getQuantasCaracteristicas();
		for (int ii = 1; ii <= num_carac; ii++) {
			mi = p.getCaracteristicaFAN(ii);
			conjuntoMi = mi.getConjuntoDifuso();
			prodSomas = mi.getSomatorio() * this.getSomatorio(ii);
			int k = 0;
			double num1 = 0;
			if (this.getSomatorio(ii) > 0) {
				int fim = mi.getFim();
				for (int j = mi.getInicio(); j <= fim; j++) {
					num1 = conjuntoMi[k] * this.getValor(ii, j)
/ prodSomas;
					num1 = Math.sqrt(num1);
					retorno *= (1 - num1);
					k++;
				}
			}
		}
		double expoente = 1 / (double)num_carac;
		double num2 = Math.pow(retorno, expoente);
		retorno = (1 - num2);
		return retorno;
	}

	public void setValor(int caracteristica, int posicao, double valor)
{
		this.matrizNeural[caracteristica - 1][posicao +
this.raioDifuso] = valor;
	}

	public int getClasseAssociada() {
		return this.classe;
	}
```

```
	public double getValor(int caracteristica, int posicao) {
		return this.matrizNeural[caracteristica - 1][posicao +
this.raioDifuso];
	}

	public int quantasCaracteristicas() {
		return this.matrizNeural.length;
	}
	public int getPesoPenalizacao() {
		return pesoPenalizacao;
	}

	public void setPesoPenalizacao(int pesoPenalizacao) {
		this.pesoPenalizacao = pesoPenalizacao;
	}

	public void adicionaValor(int caracteristica, int posicao, double
valor) {
		this.matrizNeural[caracteristica - 1][posicao +
this.raioDifuso] += valor;
		this.somatorios[caracteristica - 1] += valor;
	}
	public final double[][] getMatrizNeural() {
		return this.matrizNeural;
	}
	public double getSomatorio(int caracteristica) {
		return this.somatorios[caracteristica - 1];
	}

	public NeuronioFAN clone() {
		NeuronioFAN n = new
NeuronioFAN(this.numCaractertisticas,this.J,this.raioDifuso,this.classe,f
alse);
		n.tamanhoArray = this.tamanhoArray;
		int fim1 = this.matrizNeural.length;
		int fim2 = this.matrizNeural[0].length;
		for (int i = 0; i < fim1; i++) {
			for (int j = 0; j < fim2; j++) {
				n.matrizNeural[i][j] = this.matrizNeural[i][j];
			}
			n.somatorios[i] = this.somatorios[i];
		}
		return n;
	}

	public void setClasseAssociada(int novaClasse) {
		this.classe = novaClasse;
	}

	public void setSomatorio(int indiceCaracteristica, double
somatorioCaracteristica){

	this.somatorios[indiceCaracteristica]=somatorioCaracteristica;
	}
	public int getRaioDifuso() {
		return raioDifuso;
	}
```

```
	public int getSuporteConjuntoDifuso(){
		return this.J;
	}
}
```

2.3.7 RedeFAN.Java

```
package jfan.fan;


import java.util.ArrayList;
import java.util.Random;

import jfan.fan.beans.ErrorBean;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
import jfan.fan.utilitarias.RecalculadorClasses;

public class RedeFAN {

	private ErrorBean errorBean = null;

	private int[][] matrizConfusao = null;

	private int[] dominioClasses = null;

	private int[] contagemClasses = null;

	private int[] contagemClassesValidacao = null;

	private int pesoBase = 0;

	private boolean randomizarPesos = false;
	private ITemperaSimulada tempera = null;

	private Random random = new Random();

	private ITreinador treinador;

	private TesteFAN testador = new TesteFAN();

	private final ArrayList<NeuronioFAN> neuronioAtual = new
ArrayList<NeuronioFAN>(2);


	private int raioDifuso;

	private int suporteConjuntosDifusos;

	private ArrayList<IPadrao> conjuntoTreinamento = null;

	private ArrayList<IPadrao> conjuntoTeste = null;

	private ArrayList<IPadrao> conjuntoValidacao = null;
```

```
	private ArrayList<IPadrao> conjuntoClassificacao = null;

	int erros;

	int[] errosClasses;

	int[] errosClassesValidacao;


    public RedeFAN(int raioDifuso, int suporteConjuntosDifusos) {
		this.raioDifuso = raioDifuso;
		this.suporteConjuntosDifusos = suporteConjuntosDifusos;
	}

	public void adicionarNeuronio(int classeAssociada, int peso)
			throws NullPointerException {
		if (this.conjuntoTreinamento == null)
			throw new NullPointerException(
					"é preciso ser definido um conjunto de
treinamento antes de adicionar os neuronios");
		NeuronioFAN neuron = new
NeuronioFAN(this.conjuntoTreinamento.get(0)
				.getQuantasCaracteristicas(),
this.suporteConjuntosDifusos,
				this.raioDifuso, classeAssociada);
		neuron.setPesoPenalizacao(peso);
		this.neuronioAtual.add(neuron);
	}

	public void adicionarNeuronio(NeuronioFAN neuronio) {
		this.neuronioAtual.add(neuronio);
	}

	public void adicionarNeuronios(ArrayList<NeuronioFAN> neuronios) {
		for (NeuronioFAN neuronio : neuronios) {
			this.neuronioAtual.add(neuronio);
		}
	}

	public void adicionarNeuronios(int[] classesAssociadas)
			throws NullPointerException {
		if (this.conjuntoTreinamento == null) {
			throw new NullPointerException(
					"é preciso ser definido um conjunto de
treinamento antes de adicionar os neuronios");
		}
		for (int classe : classesAssociadas) {
			IPadrao padrao = null;
			if (conjuntoTreinamento.size() > 0) {
				padrao = this.conjuntoTreinamento.get(0);
			}
			else if (conjuntoTeste.size() > 0) {
				padrao = this.conjuntoTeste.get(0);
			}
			else if (conjuntoValidacao.size() > 0) {
				padrao = this.conjuntoValidacao.get(0);
```

```
                }
                else if (conjuntoClassificacao.size() > 0) {
                      padrao = this.conjuntoClassificacao.get(0);
                }


                NeuronioFAN neuron = new
NeuronioFAN(padrao.getQuantasCaracteristicas(),
                            this.suporteConjuntosDifusos,
this.raioDifuso, classe);
                this.neuronioAtual.add(neuron);
          }
      }

      public void setConjuntoTreinamento(ArrayList<IPadrao> c) {
            this.conjuntoTreinamento = c;
      }

      public void setConjuntoTeste(ArrayList<IPadrao> c) {
            this.conjuntoTeste = c;
      }

      public void setConjuntoValidacao(ArrayList<IPadrao> c) {
            this.conjuntoValidacao = c;
      }

      public void setConjuntoClassificacao(ArrayList<IPadrao> c) {
            this.conjuntoClassificacao = c;
      }

      public int getQuantosPadroes(int classe) {
            int temp = 0;
            for (IPadrao p : this.conjuntoTeste) {
                  if (p.getClasse() == classe)
                        temp++;
            }
            return temp;
      }

      public int getQuantosPadroesValidacao(int classe) {
            int temp = 0;
            for (IPadrao p : this.conjuntoValidacao) {
                  if (p.getClasse() == classe)
                        temp++;
            }
            return temp;
      }

      public int getPesoBase() {
            return pesoBase;
      }

      public void treinar(ArrayList<IPadrao> conjTreinamento) {
            this.treinador.treinar(conjTreinamento);
      }

      public void treinar() {
```

```
            treinador.treinar(conjuntoTreinamento);
            if(this.randomizarPesos)
                  if (this.pesoBase < 1000)
                        for (NeuronioFAN n : this.neuronioAtual)
                                    n.setPesoPenalizacao(this.pesoBase +
random.nextInt(1000-this.pesoBase));

      }

      public ErrorBean testar() {
            return testar(neuronioAtual, conjuntoTeste);
      }

      public ErrorBean testar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes) {
            return testar(neuronios, padroes,contagemClasses);
      }

      public ErrorBean validar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes) {
            ErrorBean eb = testar(neuronios,
padroes,contagemClassesValidacao);
            eb.setTipo("validacao");
            return eb;
      }

      private ErrorBean testar(ArrayList<NeuronioFAN> neuronios,
ArrayList<IPadrao> padroes, int[] contagemClasses) {
            testador.setNeuronios(neuronios);
            erros = 0;
            int fim = neuronios.size();
            for (int i = 0;i<fim;i++) {
                  errosClasses[i] = 0;
            }
            matrizConfusao = new int[fim][fim];
            for (int i = 0; i < fim; i++)
                  for (int j = 0; j < fim; j++)
                        matrizConfusao[i][j] = 0;
            for (IPadrao p : padroes) {
                  int classeSugerida =
testador.classificar(p).getClasseAssociada();
                  matrizConfusao[p.getClasse()][classeSugerida]++;
                  if (classeSugerida != p.getClasse()) {
                        erros++;
                        errosClasses[p.getClasse()]++;
                  }
            }
            errorBean = new ErrorBean();
            errorBean.setErros(erros);
            errorBean.setErrosClasses(errosClasses);
            errorBean.setMatrizConfusao(matrizConfusao);
            errorBean.setContagemClasses(contagemClasses);

            return errorBean;
      }

      private NeuronioFAN determinaNeuronio(int classe) {
```

```
		NeuronioFAN neuronio = null;
		loop:
		for (NeuronioFAN n : this.neuronioAtual) {
			if (n.getClasseAssociada() == classe) {
				neuronio = n;
				break loop;
			}
		}
		return neuronio;
	}

	public void setPesoNeuronio(int classeNeuronio,int peso) {
		determinaNeuronio(classeNeuronio).setPesoPenalizacao(peso);
	}

	public void setTemperaSimulada(ITemperaSimulada t) {
			this.tempera = t;
			this.treinador.setTemperaSimulada(this.tempera);
	}

	public void removerTemperaSimulada() {
			this.tempera = null;
			this.treinador.setTemperaSimulada(this.tempera);
	}

	public ErrorBean validar() {
		ErrorBean eb = testar(neuronioAtual,conjuntoValidacao);
		eb.setTipo("validacao");
		return eb;
	}

	public void classificar(ArrayList<NeuronioFAN> neuronios) {
		testador.setNeuronios(neuronios);
		for (IPadrao p : conjuntoClassificacao) {

	p.setClasse(testador.classificar(p).getClasseAssociada());
		}
	}

	public ArrayList<IPadrao> getConjuntoTreinamento() {
		return this.conjuntoTreinamento;
	}

	public ArrayList<IPadrao> getConjuntoTeste() {
		return this.conjuntoTeste;
	}

	public ArrayList<IPadrao> getConjuntoValidacao() {
		return this.conjuntoValidacao;
	}

	public ArrayList<IPadrao> getConjuntoClassificacao() {
		return this.conjuntoClassificacao;
	}

	public void onRandomizarPesos(int pesoBase) {
		this.randomizarPesos = true;
```

```
            this.pesoBase = pesoBase;
      }

      public void offRandomizarPesos() {
            this.randomizarPesos = false;
            this.pesoBase = 0;
      }

      public ArrayList<NeuronioFAN> getNeuroniosAtuais() {
            return neuronioAtual;
      }

      public void setDominioClasses(int[] classes) {
            this.dominioClasses = classes;
            this.neuronioAtual.clear();
            this.neuronioAtual.trimToSize();
            this.adicionarNeuronios(classes);
            testador = new TesteFAN(this.neuronioAtual);
            contagemClasses = new int[this.neuronioAtual.size()];
            contagemClassesValidacao = new
int[this.neuronioAtual.size()];
            this.errosClasses = new int[classes.length];
            errosClassesValidacao = new int[classes.length];
            RecalculadorClasses.indexarNeuronios(this.dominioClasses,
this.neuronioAtual);
            int fim = this.neuronioAtual.size();
            for (int i = 0;i<fim;i++) {
                  int classe = neuronioAtual.get(i).getClasseAssociada();
                  errosClasses[classe] = 0;
                  contagemClasses[classe] =
this.getQuantosPadroes(classe);
            }
            if (conjuntoValidacao.size() > 0) {
                  for (int i = 0;i<fim;i++) {
                        int classe =
neuronioAtual.get(i).getClasseAssociada();
                        errosClassesValidacao[classe] = 0;
                        contagemClassesValidacao[classe] =
this.getQuantosPadroesValidacao(classe);
                  }
            }
      }

      public void setPesoBase(int pesoBase) throws Exception {
            if (pesoBase < 0 || pesoBase > 1000)
                  throw new Exception("O valor do peso deve estar entre 0
e 1000.");
            this.pesoBase = pesoBase;
      }

      public void setTreinador(ITreinador treinador) {
            this.treinador = treinador;
      }

      public void setNeuronios(ArrayList<NeuronioFAN> neuronios) {
            int[] classes = new int[neuronios.size()];
            for (int i = 0; i < neuronios.size();i++) {
```

```
                classes[i] = neuronios.get(i).getClasseAssociada();
            }

            RecalculadorClasses.indexarNeuronios(classes, neuronios);
            neuronioAtual.clear();
            neuronioAtual.addAll(neuronios);
      }

      public ITreinador getTreinador(){
            return treinador;
      }

      public int[] getDominioClasses() {
            return dominioClasses;
      }

}
```

2.3.8 StoredNetworkType.Java

```
package jfan.fan;

public enum StoredNetworkType {
      BestHarmonicMean ("Melhor Média Harmonica "),
      BestAritmethicMean("Melhor Média Aritmética"),
      BestMaxMin("Melhor Máximo do Mínimo"),
      Actual("Atual");

      private String texto;

      private StoredNetworkType(String texto){
            this.texto = texto;
      }

      public String toString() {
            return texto;
      }
}
```

2.3.9 TesteFAN.Java

```
package jfan.fan;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public class TesteFAN {
      private ArrayList<NeuronioFAN> neuronios = null;

      public TesteFAN() {}

      public TesteFAN(ArrayList<NeuronioFAN> neuronios) {
            this.neuronios = neuronios;
```

```
	}

	public NeuronioFAN classificar(IPadrao p) {
		NeuronioFAN melhorNeuronio = this.neuronios.get(0);
		double maiorForca = melhorNeuronio.determinarForca(p);
		double forcaAtual;
		for (int i= 1;i<this.neuronios.size();i++) {
			forcaAtual = this.neuronios.get(i).determinarForca(p);
			if (maiorForca < forcaAtual) {
				melhorNeuronio = this.neuronios.get(i);
				maiorForca = forcaAtual;
			}
		}
		return melhorNeuronio;
	}

	public void setNeuronios(ArrayList<NeuronioFAN> neurons) {
		this.neuronios = neurons;
	}
}
```

2.3.10 TreinoFAN.Java

```
package jfan.fan;

import java.util.ArrayList;
import java.util.Random;

import jfan.fan.padroes.CaracteristicaFAN;
import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;
public class TreinoFAN implements ITreinador {

	private ITemperaSimulada tempera = null;
	protected ArrayList<NeuronioFAN> neuronios;

	protected TesteFAN testeFan;
	protected Random random = new Random();
	public TreinoFAN() {
		testeFan = new TesteFAN();
	}

	public void treinar(ArrayList<IPadrao> padroes) {
		for (IPadrao p : padroes) {
			treinar(p);
		}
	}
	public void treinar(IPadrao p) {
		NeuronioFAN n = testeFan.classificar(p);
			if (n.getClasseAssociada() != p.getClasse()) {
				if (n.getPesoPenalizacao() >= 1000 ||
random.nextInt(1001) > n.getPesoPenalizacao()) {
					penalizar(n, p);
				}
			}
```

```
                        reforcar(p);
        }

        protected void penalizar(NeuronioFAN neuronio, IPadrao p) {
                double novoValor;
                int fim, k;
                CaracteristicaFAN caracFan;
                double[] conjDifuso;
                int carateristicas = neuronio.quantasCaracteristicas();

                for (int i = 1; i <= carateristicas; i++) {
                        k = 0;
                        caracFan = p.getCaracteristicaFAN(i);
                        conjDifuso = caracFan.getConjuntoDifuso();
                        fim = caracFan.getFim();
                        fim++;
                        for(int j = caracFan.getInicio(); j < fim; j++) {
                                novoValor = neuronio.getValor(i,j);
                                if (random.nextInt(100)>10) {
                                        novoValor *=
Math.pow(conjDifuso[k],0.00001);

                                }
                                else {
                                        novoValor -=  conjDifuso[k];
                                }
                                synchronized (this) {
                                        if(this.tempera != null) {
                                                novoValor *= tempera.getValor();
                                        }
                                }
                                if (novoValor<0) novoValor = 0;
                                        neuronio.setValor(i,j,novoValor);
                                k++;
                        }
                }
        }

        protected void reforcar(IPadrao p) {
                CaracteristicaFAN caracFan;
                int k, fim;
                double[] conjDifuso;
                NeuronioFAN neuronio = this.neuronios.get(p.getClasse());
                int carateristicas = neuronio.quantasCaracteristicas();
                for (int i = 1; i <= carateristicas; i++) {
                        k = 0;
                        caracFan = p.getCaracteristicaFAN(i);
                        conjDifuso = caracFan.getConjuntoDifuso();
                        fim = caracFan.getFim();
                        fim++;
                        for(int j = caracFan.getInicio(); j < fim; j++) {
                                        neuronio.adicionaValor(i,j,(conjDifuso[k]));
                                k++;
                        }
                }

        }
```

```
	public void setTemperaSimulada(ITemperaSimulada t) {
		synchronized (this) {
			this.tempera = t;
		}
	}

	public void setNeuronios(ArrayList<NeuronioFAN> neurons) {
		this.neuronios = neurons;
		this.testeFan.setNeuronios(neurons);
	}
}
```

2.3.11 TreinoQueue.Java

```
package jfan.fan;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

import jfan.fan.padroes.IPadrao;
import jfan.fan.temperas.ITemperaSimulada;

public class TreinoQueue extends TreinoFAN implements ITreinador {

	LinkedBlockingQueue<Tarefa>[] filas;

	TreinaFila[] treinaFila;

	Thread[] threadsTreinaFila;

	public void treinar(ArrayList<IPadrao> padroes) {
		for (int i = 0; i < treinaFila.length; i++) {
			treinaFila[i] = new TreinaFila();
			treinaFila[i].fila = filas[i];
			threadsTreinaFila[i] = new Thread(treinaFila[i]);
			threadsTreinaFila[i].start();
		}

		for(IPadrao p : padroes ) {
			enfileirar(p);
			Thread.yield();
		}

			Thread.yield();

		for (TreinaFila t : treinaFila) {
			t.keepGoing = false;
		}

		while(temThreadViva()) {
			Thread.yield();
		}
	}
```

```
	private boolean temThreadViva(){
		for (Thread t : threadsTreinaFila) {
			if( t.isAlive() ) return true;
		}

		return false;
	}

	private void enfileirar(IPadrao p){

		NeuronioFAN n = testeFan.classificar(p);
		if (n.getClasseAssociada() != p.getClasse()) {
			if (n.getPesoPenalizacao() >= 1000 ||
random.nextInt(1001) > n.getPesoPenalizacao()) {
				Tarefa task = new Tarefa();
				task.alvo = n;
				task.padrao = p;
				task.tipoTarefa = TipoTarefa.penalizar;
				try {
					filas[n.getClasseAssociada()].put(task);
				} catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		}
		Tarefa task = new Tarefa();
		task.alvo = null;
		task.padrao = p;
		task.tipoTarefa = TipoTarefa.reforcar;
		try {
			filas[p.getClasse()].put(task);
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
	}

	@SuppressWarnings("unchecked")
	public void setNeuronios(ArrayList<NeuronioFAN> neurons) {
		super.setNeuronios(neurons);
		filas = (LinkedBlockingQueue<Tarefa>[])
Array.newInstance(LinkedBlockingQueue.class, neurons.size());
		treinaFila = new TreinaFila[neurons.size()];
		threadsTreinaFila = new Thread[neurons.size()];
		for (int i = 0; i < filas.length; i++) {
			filas[i] = new LinkedBlockingQueue<Tarefa>();
		}
	}

	private class TreinaFila implements Runnable {

		private LinkedBlockingQueue<Tarefa> fila;

		private boolean keepGoing = true;

		public void run() {
			while (keepGoing || fila.size() > 0) {
```

```
                        if (fila.size() > 0){
                              try {
                                    Tarefa task = fila.poll(200,
TimeUnit.MILLISECONDS);
                                    if (task.tipoTarefa ==
TipoTarefa.reforcar) {
                                          reforcar(task.padrao);
                                    }
                                    else if (task.tipoTarefa ==
TipoTarefa.penalizar) {
                                          penalizar(task.alvo,
task.padrao);
                                    }
                              } catch (InterruptedException e) {
                                    System.out.println("Ocorreu um erro de
interrupção inesperada.");
                                    e.printStackTrace();
                              }
                        }
                        else {
                              Thread.yield();
                        }
                  }
            }

      }

      private enum TipoTarefa {
            reforcar, penalizar
      }

      private class Tarefa {
            private TipoTarefa tipoTarefa;
            private IPadrao padrao;
            private NeuronioFAN alvo;
      }
}
```

2.4 PACKAGE JFAN.FAN.BEANS;

2.4.1 ErrorBean.Java

```
package jfan.fan.beans;

public class ErrorBean {

      private int erros = 0;

      private int[] errosClasses = null;

      private int[][] matrizConfusao = null;

      private int[] contagemClasses = null;

      private String tipo;
```

```
	public int getErros() {
		return this.erros;
	}

	public void setErros(int erros) {
		this.erros = erros;
	}

	public int[] getErrosClasses() {
		return this.errosClasses;
	}

	public void setErrosClasses(int[] errosClasses) {
		this.errosClasses = errosClasses;
	}

	public int[][] getMatrizConfusao() {
		return this.matrizConfusao;
	}

	public void setMatrizConfusao(int[][] matrizConfusao) {
		this.matrizConfusao = matrizConfusao;
	}

	public int[] getContagemClasses() {
		return this.contagemClasses;
	}

	public void setContagemClasses(int[] contagemClasses) {
		this.contagemClasses = contagemClasses;
	}

	public String getTipo() {
		return tipo;
	}

	public void setTipo(String tipo) {
		this.tipo = tipo;
	}

}
```

2.4.2 StatisticBean.Java

```
package jfan.fan.beans;

public class StatisticBean {

	private float harmonica,aritmetica,maxmin;
	private int[][] matrizConfusao;
	private int acertos,erros;

	public int getAcertos() {
		return acertos;
```

```
	}
	public void setAcertos(int acertos) {
		this.acertos = acertos;
	}
	public float getAritmetica() {
		return aritmetica;
	}
	public void setAritmetica(float aritmetica) {
		this.aritmetica = aritmetica;
	}
	public int getErros() {
		return erros;
	}
	public void setErros(int erros) {
		this.erros = erros;
	}
	public float getHarmonica() {
		return harmonica;
	}
	public void setHarmonica(float harmonica) {
		this.harmonica = harmonica;
	}
	public int[][] getMatrizConfusao() {
		return matrizConfusao;
	}
	public void setMatrizConfusao(int[][] matrizConfusao) {
		this.matrizConfusao = matrizConfusao;
	}
	public float getMaxmin() {
		return maxmin;
	}
	public void setMaxmin(float maxmin) {
		this.maxmin = maxmin;
	}
	public void calcularAcertosErrosApartirMatrizNeural() {
		if (matrizConfusao != null) {
			int[][] matrix = matrizConfusao;
			for (int i = 0;i < matrix.length;i++) {
				for (int j = 0;j < matrix[i].length;j++) {
					if (i != j) {
						erros += matrix[i][j];
					}
					else {
						acertos += matrix[i][j];
					}
				}
			}
		}

	}

}
```

2.5 PACKAGE JFAN.FAN.EVENTS;

2.5.1 FaseRedeEvent.Java

```
ackage jfan.fan.events;

public class FaseRedeEvent {

	private String faseRede;

	public FaseRedeEvent(String faseRede) {
		this.faseRede = faseRede;
	}

	public String getFaseRede() {
		return this.faseRede;
	}


}
```

2.5.2 FaseRedeListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface FaseRedeListener extends EventListener {

	public void mudouFaseRede(FaseRedeEvent event);

	public void redeParou(RedeParouEvent event);
}
```

2.5.3 MelhorAritmeticaEvent.Java

```
package jfan.fan.events;

public class MelhorAritimeticaEvent {

	private float mediaAritimetica;
	private final int[][] matrixConfusao;

	public MelhorAritimeticaEvent(float mediaAritmetica, int[][]
matrizConfusao) {
		this.mediaAritimetica = mediaAritmetica;
		this.matrixConfusao = matrizConfusao;
	}

	public float getMediaAritimetica() {
		return this.mediaAritimetica;
	}

	public int[][] getMatrizConfusao() {
		return matrixConfusao;
	}
```

```
}
```

2.5.4 MelhorAritmeticaListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface MelhorAritimeticaListener extends EventListener {
      public void trocouMelhorMediaAritimetica(MelhorAritimeticaEvent
event);
}
```

2.5.5 MelhorHarmonicaEvent.Java

```
package jfan.fan.events;

import jfan.fan.RedeFAN;

public class MelhorHarmonicaEvent{

      private final float medHarmonica;
      private final int[][] matrixConfusao;

      public MelhorHarmonicaEvent(float mediaHarmonica, int[][]
matrizConfusao) {
            medHarmonica = mediaHarmonica;
            this.matrixConfusao = matrizConfusao;
      }

      public float getMediaHarmonica() {
            return this.medHarmonica;
      }

      public int[][] getMatrizConfusao() {
            return matrixConfusao;
      }

}
```

2.5.6 MelhorHarmonicaListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface MelhorHarmonicaListener extends EventListener {
      public void trocouMelhorMediaHarmonica(MelhorHarmonicaEvent event);
}
```

2.5.7 MelhorMaximoMinimoEvent.Java

```
package jfan.fan.events;


public class MelhorMaximoMinimoEvent  {

	private final float maxMin;
	private final int[][] matrixConfusao;

	public MelhorMaximoMinimoEvent(float maximoMinimo, int[][]
matrizConfusao) {
		maxMin = maximoMinimo;
		this.matrixConfusao = matrizConfusao;
	}

	public float getMaximoDoMinimo() {
		return this.maxMin;
	}

	public int[][] getMatrizConfusao() {
		return matrixConfusao;
	}
}
```

2.5.8 MelhorMaximoMinimoListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface MelhorMaximoMinimoListener extends EventListener {
	public void trocouMelhorMaximoMinimo(MelhorMaximoMinimoEvent
event);
}
```

2.5.9 RedeParouEvent.Java

```
package jfan.fan.events;

public class RedeParouEvent {
	private int epoca;

	public RedeParouEvent(int epoca) {
		this.epoca = epoca;
	}

	public int getEpocaDeParada() {
		return epoca;
	}
}
```

2.5.10 TrocaEpocaEvent.Java

```
package jfan.fan.events;

public class TrocaEpocaEvent {

	private float mediaHarmonica;
	private float mediaAritmetica;
	private float maxDoMin;
	private int[][] matrizConfusao;
	private int epoca;


	public TrocaEpocaEvent(int epoc, float medHarmonica, float
medAritmetica, float maxDoMin, int[][] matrixConfusao) {
		this.mediaAritmetica = medAritmetica;
		this.mediaHarmonica = medHarmonica;
		this.maxDoMin = maxDoMin;
		this.epoca = epoc;
		this.matrizConfusao = matrixConfusao;
	}

	public int getEpoca() {
		return this.epoca;
	}

	public float getMaximoDoMinimo() {
		return this.maxDoMin;
	}

	public float getMediaAritmetica() {
		return this.mediaAritmetica;
	}

	public float getMediaHarmonica() {
		return this.mediaHarmonica;
	}

	public int[][] getMatrizConfusao() {
		return matrizConfusao;
	}
}
```

2.5.11 TrocaEpocaListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface TrocaEpocaListener extends EventListener {
	public void trocouEpoca(TrocaEpocaEvent event);
}
```

2.5.12 TrocaValorTemperaEvent.Java

```
package jfan.fan.events;

public class TrocaValorTemperaEvent {
	private double valor;

	public TrocaValorTemperaEvent(double valor) {
		this.valor = valor;
	}

	public double getValor() {
		return valor;
	}
}
```

2.5.13 TrocaValorTemperaListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface TrocaValorTemperaListener extends EventListener {
	public void trocouValorTempera(TrocaValorTemperaEvent event);
}
```

2.5.14 ValidacaoEvent.Java

```
package jfan.fan.events;

import jfan.fan.StoredNetworkType;
import jfan.fan.beans.StatisticBean;

public class ValidacaoEvent {

	private int epocaValidacao;

	private StoredNetworkType tipoRedeValidacao;

	private StatisticBean statisticas;

	public ValidacaoEvent(int epoca, StoredNetworkType
tipoRedeValidacao, StatisticBean statisticas) {
		this.epocaValidacao = epoca;
		this.tipoRedeValidacao = tipoRedeValidacao;
		this.statisticas = statisticas;
	}

	public int getEpocaValidacao() {
		return epocaValidacao;
	}

	public StatisticBean getStatisticas() {
```

```
            return statisticas;
      }


      public StoredNetworkType getTipoRedeValidacao() {
            return tipoRedeValidacao;
      }

}
```

2.5.15 ValidacaoListener.Java

```
package jfan.fan.events;

import java.util.EventListener;

public interface ValidacaoListener extends EventListener {

      public void validou(ValidacaoEvent event);

}
```

2.6 PACKAGE JFAN.FAN.NORMALIZADORES;

2.6.1 INormalizator.Java

```
package jfan.fan.normalizadores;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public interface INormalizator {

      public void normalizar(ArrayList<IPadrao> padroes);

      public void normalizar(ArrayList<IPadrao> padroes, int
caracteristica);

      public void normalizar(IPadrao p);

      public void normalizar(IPadrao p, int caracteristica);

      public void normalizarThreaded(ArrayList<IPadrao> padroes);

      public void setValorMaximoCaracteristica(int caracteristica, double
valor);

      public void setValorMinimoCaracteristica(int caracteristica, double
valor);

      public void setValorMedioCaracteristica(int caracteristica, double
valor);
```

```
	public void setValoresMaximosCaracteristicas(double valores[]);

	public void setValoresMinimosCaracteristicas(double valores[]);

	public void setValoresMediasCaracteristicas(double valores[]);
}
```

2.6.2 NormalizatorConfiguration.Java

```
package jfan.fan.normalizadores;

public class NormalizatorConfiguration {
	private double[] maxs;
	private double[] mins;
	private double[] means;

	public NormalizatorConfiguration() {}

	public NormalizatorConfiguration(double[] max, double[] min,
double[] mean) {
		maxs = max;
		mins = min;
		means = mean;
	}

	public double[] getMaximos() {
		return this.maxs;
	}

	public void setMaximos(double[] maxs) {
		this.maxs = maxs;
	}

	public double[] getMedias() {
		return this.means;
	}

	public void setMedias(double[] means) {
		this.means = means;
	}

	public double[] getMinimos() {
		return this.mins;
	}

	public void setMinimos(double[] mins) {
		this.mins = mins;
	}
}
```

2.6.3 NormalizatorMax.Java

```
package jfan.fan.normalizadores;
```

```
import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public class NormalizatorMax implements INormalizator, Runnable {

	private static double[] max = null;
	private int caracEscolhida;
	private ArrayList<IPadrao> conjuntoArmazenado;

	public NormalizatorMax(){}

	public NormalizatorMax(NormalizatorConfiguration cfg){
		max = cfg.getMaximos();
	}

	public void normalizar(ArrayList<IPadrao> padroes) {
		for (IPadrao p : padroes) normalizar(p);
	}

	public void normalizar(ArrayList<IPadrao> padroes, int
caracteristica) {
		for (IPadrao p : padroes) normalizar(p, caracteristica);
	}

	public void normalizar(IPadrao p) {
		int quantas = p.getQuantasCaracteristicas();
		for (int i = 1; i <= quantas; i++) normalizar(p,i);
	}

	public void normalizar(IPadrao p, int caracteristica) {
		double caracNorm = p.getCaracteristica(caracteristica);
		caracNorm /= max[caracteristica-1];
		if (caracNorm > 1.0) caracNorm = 1.0;
		if (caracNorm < 0) caracNorm = 0.0;
		p.setCaracteristicaNormalizada(caracteristica, caracNorm);
	}

	public void normalizarThreaded(ArrayList<IPadrao> padroes) {
		if (padroes.size() <= 0) return;
		int quantCarac = padroes.get(0).getQuantasCaracteristicas();
		Thread[] threads = new Thread[quantCarac];
		NormalizatorMax nm;
		for (int i = 1; i<= quantCarac; i++){
			nm = new NormalizatorMax();
			nm.conjuntoArmazenado = padroes;
			nm.caracEscolhida = i;
			nm.setValoresMaximosCaracteristicas(max);
			threads[i-1] = new Thread(nm);
			threads[i-1].start();
		}
		boolean loop;
		do {
			try {
				Thread.sleep(100);
			} catch (InterruptedException e) {
				e.printStackTrace();
```

```
                }
                loop = false;
                for (Thread t : threads) {
                        loop = loop || t.isAlive();
                }
        }
        while(loop);
    }

    public void setValorMaximoCaracteristica(int caracteristica, double
valor) {
        setValorCaracteristica(caracteristica, valor, max);
    }

    public void setValorMinimoCaracteristica(int caracteristica, double
valor) {
    }

    public void setValoresMaximosCaracteristicas(double[] valores) {
        max = valores;
    }

    public void setValoresMinimosCaracteristicas(double[] valores) {
    }

    private void setValorCaracteristica(int caracteristica, double
valor, double[] array) {
        if(array == null) {
                array = new double[caracteristica];
        }
        else if (array.length < caracteristica) {
                ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
                for(Double d : array) dtemp.add(d);
                for (int i = dtemp.size(); i < caracteristica-1; i++)
dtemp.add(0.0);
                dtemp.add(caracteristica-1, valor);
                array = new double[dtemp.size()];
                for (int i = 0; i < array.length; i++) {
                        array[i] = dtemp.get(i);
                }
        }
        else {
                ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
                for(Double d : array) dtemp.add(d);
                if (dtemp.get(caracteristica-1)==null) {
                        dtemp.add(caracteristica-1, valor);
                }
                else {
                        dtemp.set(caracteristica-1, valor);
                }
                array = new double[dtemp.size()];
                for (int i = 0; i < array.length; i++) {
                        array[i] = dtemp.get(i);
                }
        }
```

```
	}

	public void run() {
		normalizar(conjuntoArmazenado, caracEscolhida);
	}

	public void setValorMedioCaracteristica(int caracteristica, double
valor) {

	}

	public void setValoresMediasCaracteristicas(double[] valores) {

	}
}
```

2.6.4 NormalizatorMaxMean.Java

```
package jfan.fan.normalizadores;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public class NormalizatorMaxMean implements INormalizator, Runnable {

	private double[] max;
	private double[] mean;
	private int caracEscolhida;
	private ArrayList<IPadrao> conjuntoArmazenado;

	public NormalizatorMaxMean(NormalizatorConfiguration config) {
		max = config.getMaximos();
		mean = config.getMedias();
	}

	public NormalizatorMaxMean(){}

	public void normalizar(ArrayList<IPadrao> padroes) {
		for (IPadrao p : padroes) normalizar(p);
	}

	public void normalizar(ArrayList<IPadrao> padroes, int
caracteristica) {
		for (IPadrao p : padroes) normalizar(p, caracteristica);
	}

	public void normalizar(IPadrao p) {
		int quantas = p.getQuantasCaracteristicas();
		for (int i = 1; i <= quantas; i++) normalizar(p,i);
	}

	public void normalizar(IPadrao p, int caracteristica) {
		double caracNorm = p.getCaracteristica(caracteristica) -
mean[caracteristica-1];
```

```
            caracNorm /= (max[caracteristica-1] - mean[caracteristica-
1]);
            caracNorm++;
            caracNorm /= 2;
            p.setCaracteristicaNormalizada(caracteristica, caracNorm);
      }

      public void normalizarThreaded(ArrayList<IPadrao> padroes) {
            if (padroes.size() <= 0) return;
            int quantCarac = padroes.get(0).getQuantasCaracteristicas();
            Thread[] threads = new Thread[quantCarac];
            NormalizatorMaxMean nm;
            for (int i = 1; i<= quantCarac; i++){
                  nm = new NormalizatorMaxMean();
                  nm.conjuntoArmazenado = padroes;
                  nm.caracEscolhida = i;
                  nm.setValoresMaximosCaracteristicas(max);
                  nm.setValoresMediasCaracteristicas(mean);
                  threads[i-1] = new Thread(nm);
                  threads[i-1].start();
            }
            boolean loop;
            do {
                  try {
                        Thread.sleep(100);
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }
                  loop = false;
                  for (Thread t : threads) {
                        loop = loop || t.isAlive();
                  }
            }
            while(loop);
      }

      public void setValorMaximoCaracteristica(int caracteristica, double
valor) {
            setValorCaracteristica(caracteristica, valor, max);
      }

      public void setValorMinimoCaracteristica(int caracteristica, double
valor) {

      }

      public void setValoresMaximosCaracteristicas(double[] valores) {
            max = valores;
      }

      public void setValoresMinimosCaracteristicas(double[] valores) {
      }

      public void run() {
            normalizar(conjuntoArmazenado, caracEscolhida);
      }
```

```
	private void setValorCaracteristica(int caracteristica, double
valor, double[] array) {
		if(array == null) {
			array = new double[caracteristica];
		}
		else if (array.length < caracteristica) {
			ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
			for(Double d : array) dtemp.add(d);
			for (int i = dtemp.size(); i < caracteristica-1; i++)
dtemp.add(0.0);
			dtemp.add(caracteristica-1, valor);
			array = new double[dtemp.size()];
			for (int i = 0; i < array.length; i++) {
				array[i] = dtemp.get(i);
			}
		}
		else {
			ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
			for(Double d : array) dtemp.add(d);
			if (dtemp.get(caracteristica-1)==null) {
				dtemp.add(caracteristica-1, valor);
			}
			else {
				dtemp.set(caracteristica-1, valor);
			}
			array = new double[dtemp.size()];
			for (int i = 0; i < array.length; i++) {
				array[i] = dtemp.get(i);
			}
		}
	}

	public void setValorMedioCaracteristica(int caracteristica, double
valor) {
		setValorCaracteristica(caracteristica, valor, mean);
	}

	public void setValoresMediasCaracteristicas(double[] valores) {
		mean = valores;
	}

}
```

2.6.5 NormalizatorMaxMin.Java

```
package jfan.fan.normalizadores;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public class NormalizatorMaxMin implements Runnable, INormalizator {

	double[] max = null;
```

```
	double[] min = null;

	double[] maxMinusMin;

	ArrayList<IPadrao> conjuntoArmazenado = null;
	int caracEscolhida;

	public NormalizatorMaxMin(){}

	public NormalizatorMaxMin(NormalizatorConfiguration cfg){
		max = cfg.getMaximos();
		min = cfg.getMinimos();
		calcularMaxMinusMin();
	}

	private void calcularMaxMinusMin() {
		if (min != null && max != null && min.length == max.length) {
			maxMinusMin = new double[min.length];
			for (int i = 0; i < maxMinusMin.length; i++) {
				maxMinusMin[i] = max[i] - min[i];
			}
		}
	}

	public void setValoresMaximosCaracteristicas(double[] maxs) {
		max = maxs;
		calcularMaxMinusMin();
	}

	public void setValoresMinimosCaracteristicas(double[] mins) {
		min = mins;
		calcularMaxMinusMin();
	}

	public NormalizatorMaxMin(double[] max, double[] min) {
		this.max = max;
		this.min = min;
		calcularMaxMinusMin();
	}

	public void normalizar(ArrayList<IPadrao> padroes) {
		for (IPadrao p : padroes) {
			normalizar(p);
		}
	}

	public void normalizar(ArrayList<IPadrao> padroes, int
caracteristica) {
		for (IPadrao p : padroes) {
			normalizar(p,caracteristica);
		}
	}

	public void normalizar(IPadrao p) {
		int fim = p.getQuantasCaracteristicas();
		double caracNorm;
		for(int i = 1; i <= fim; i++) {
```

```
                caracNorm = p.getCaracteristica(i) - min[i-1];
                caracNorm /= maxMinusMin[i-1];
                if (caracNorm > 1.0) caracNorm = 1.0;
                if (caracNorm < 0) caracNorm = 0.0;
                p.setCaracteristicaNormalizada(i, caracNorm);
          }
    }

    public void normalizar(IPadrao p, int caracteristica) {
          double caracNorm;
          caracNorm = p.getCaracteristica(caracteristica) -
min[caracteristica-1];
          caracNorm /= maxMinusMin[caracteristica-1];
          if (caracNorm > 1.0) caracNorm = 1.0;
          if (caracNorm < 0) caracNorm = 0.0;
          p.setCaracteristicaNormalizada(caracteristica, caracNorm);
    }

    public void normalizarThreaded(ArrayList<IPadrao> padroes) {
          if (padroes.size() <= 0) return;
          int quantCarac = padroes.get(0).getQuantasCaracteristicas();
          Thread[] threads = new Thread[quantCarac];
          NormalizatorMaxMin np;
          for (int i = 1; i<= quantCarac; i++){
                np = new NormalizatorMaxMin();
                np.conjuntoArmazenado = padroes;
                np.caracEscolhida = i;
                np.setValoresMaximosCaracteristicas(max);
                np.setValoresMinimosCaracteristicas(min);
                threads[i-1] = new Thread(np);
                threads[i-1].start();
          }
          boolean loop;
          do {
                try {
                      Thread.sleep(100);
                } catch (InterruptedException e) {
                      e.printStackTrace();
                }
                loop = false;
                for (Thread t : threads) {
                      loop = loop || t.isAlive();
                }
          }
          while(loop);
    }

    public void run() {
          normalizar(conjuntoArmazenado, caracEscolhida);
    }

    private void setValorCaracteristica(int caracteristica, double
valor, double[] array) {
          if(array == null) {
                array = new double[caracteristica];
          }
          else if (array.length < caracteristica) {
```

```
                  ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
                  for(Double d : array) dtemp.add(d);
                  for (int i = dtemp.size(); i < caracteristica-1; i++)
dtemp.add(0.0);
                  dtemp.add(caracteristica-1, valor);
                  array = new double[dtemp.size()];
                  for (int i = 0; i < array.length; i++) {
                        array[i] = dtemp.get(i);
                  }
            }
            else {
                  ArrayList<Double> dtemp = new
ArrayList<Double>(caracteristica);
                  for(Double d : array) dtemp.add(d);
                  if (dtemp.get(caracteristica-1)==null) {
                        dtemp.add(caracteristica-1, valor);
                  }
                  else {
                        dtemp.set(caracteristica-1, valor);
                  }
                  array = new double[dtemp.size()];
                  for (int i = 0; i < array.length; i++) {
                        array[i] = dtemp.get(i);
                  }
            }
            this.calcularMaxMinusMin();
      }

      public void setValorMaximoCaracteristica(int caracteristica, double
valor) {
            setValorCaracteristica(caracteristica, valor, max);
      }

      public void setValorMinimoCaracteristica(int caracteristica, double
valor) {
            setValorCaracteristica(caracteristica, valor, min);
      }

      public void setValorMedioCaracteristica(int caracteristica, double
valor) {

      }

      public void setValoresMediasCaracteristicas(double[] valores) {

      }

}
```

2.6.6 NormalizatorTypes.Java

```
package jfan.fan.normalizadores;

public enum NormalizatorTypes {
      NormalizatorMax,
```

```
	NormalizatorMaxMin,
	NormalizatorMaxMean,
	NormalizatorArcSinMaxMin
}
```

2.7 PACKAGE JFAN.FAN.PADROES;

2.7.1 CaracteristicaFAN.Java

```
package jfan.fan.padroes;

public class CaracteristicaFAN {

	private double[] conjuntoDifuso;
	private int inicio,fim;
	private double somaConjuntoDifuso;

	public CaracteristicaFAN(double[] conjuntoDifuso, int a, int b) {
		this.conjuntoDifuso = conjuntoDifuso;
		double soma = 0;
		for(double d: this.conjuntoDifuso) {
			soma += d;
		}
		this.somaConjuntoDifuso = soma;
		this.inicio = a + 1;
		this.fim = b - 1; }

	final public double[] getConjuntoDifuso() {
		return this.conjuntoDifuso;
	}

	final public int getInicio() {
		return this.inicio;
	}
	final public int getFim() {
		return this.fim;
	}

	final public double getSomatorio() {
		return this.somaConjuntoDifuso;
	}

	public CaracteristicaFAN clone() {
		double[] arr = new double[this.conjuntoDifuso.length];
		for (int i = 0; i < arr.length; i++) {
			arr[i] = conjuntoDifuso[i];
		}
		CaracteristicaFAN c = new CaracteristicaFAN(arr,0,0);
		c.fim = this.fim;
		c.inicio = this.inicio;
		c.somaConjuntoDifuso = this.somaConjuntoDifuso;
		return c;
	}
}
```

2.7.2 IFornecedorPadroes.Java

```
package jfan.fan.padroes;

import java.util.ArrayList;

public interface IFornecedorPadroes {
	public int numeroCaracteristicas();
	public int numeroClasses();
	public int numeroPadroes();
	public IPadrao getPadrao(int indice);
	public ArrayList<IPadrao> getPadroes();
	public ArrayList<IPadrao> getConjuntoTreinamento();
	public ArrayList<IPadrao> getConjuntoTeste();
	public ArrayList<IPadrao> getConjuntoValidacao();
	public ArrayList<IPadrao> getConjuntoClassificacao();
}
```

2.7.3 IPadrao.Java

```
package jfan.fan.padroes;


public interface IPadrao {
	public double getCaracteristica(int numCaracteristica);
	public double getCaracteristicaNormalizada(int numCaracteristica);
	public int getClasse();
	public double[] getCaracteristicas();
	public int getQuantasCaracteristicas();
	public CaracteristicaFAN getCaracteristicaFAN(int
numCaracteristica);
	public CaracteristicaFAN[] getCaracteristicasFAN();
	public void setCaracteristicaFAN(int numCaracteristica,
CaracteristicaFAN caracteristicaFAN);
	public void setCaracteristicaNormalizada(int numCaracteristica,
double valor);
	public void setClasse(int classe);
	public IPadrao clone();
}
```

2.7.4 Padrao.Java

```
package jfan.fan.padroes;

public class Padrao extends PadraoNormalizado implements IPadrao {
	private double[] caracteristicaNormalizada;

	public Padrao(double[] caracteristicas, int classe) {
		super(caracteristicas,classe);
		this.caracteristicaNormalizada = new
double[caracteristicas.length];
	}
```

```
	public void setCaracteristicaNormalizada(int numCaracteristica,
double valor) {
		this.caracteristicaNormalizada[numCaracteristica-1] = valor;
	}

	public double getCaracteristicaNormalizada(int numCaracteristica) {
		return this.caracteristicaNormalizada[numCaracteristica-1];
	}

	public IPadrao clone() {
		double[] carac = new double[caracteristica.length];
		double[] caracNorm = new
double[caracteristicaNormalizada.length];
		CaracteristicaFAN[] caracFan = new
CaracteristicaFAN[caracteristicasFAN.length];
		for (int i = 0; i < caracteristica.length; i++) {
			carac[i] = caracteristica[i];
		}
		for (int i = 0; i < caracteristicaNormalizada.length; i++) {
			caracNorm[i] = caracteristicaNormalizada[i];
		}
		for (int i = 0; i < caracteristicasFAN.length; i++) {
			if (caracteristicasFAN[i] != null){
				caracFan[i] = caracteristicasFAN[i].clone();
			}
		}
		IPadrao p = new Padrao(carac,this.classe);
		((Padrao)p).caracteristicasFAN = caracFan;
		((Padrao)p).caracteristicaNormalizada = caracNorm;

		return p;
	}

}
```

2.7.5 PadraoNormalizado.Java

```
package jfan.fan.padroes;

public class PadraoNormalizado implements IPadrao {
	double[] caracteristica;

	int classe;

	CaracteristicaFAN[] caracteristicasFAN;

	public PadraoNormalizado(double[] caracteristicas, int classe) {
		this.caracteristica = caracteristicas;
		this.classe = classe;
		this.caracteristicasFAN = new
CaracteristicaFAN[caracteristicas.length];
	}

	public double getCaracteristica(int numCaracteristica) {
```

```
            return this.caracteristica[numCaracteristica-1];
      }

      public double getCaracteristicaNormalizada(int numCaracteristica) {
            return this.caracteristica[numCaracteristica-1];
      }

      public void setCaracteristicaNormalizada (int numCaracteristica,
double valor) {
            this.caracteristica[numCaracteristica-1] = valor;
      }

      public int getClasse() {
            return this.classe;
      }

      public double[] getCaracteristicas() {
            return this.caracteristica;
      }

      public int getQuantasCaracteristicas() {
            return this.caracteristica.length;
      }
      public CaracteristicaFAN getCaracteristicaFAN(int
numCaracteristica) {
            return caracteristicasFAN[numCaracteristica-1];
      }

      public void setCaracteristicaFAN(int numCaracteristica,
CaracteristicaFAN caracteristicaFAN) {
            this.caracteristicasFAN[numCaracteristica-1] =
caracteristicaFAN;
      }

      public CaracteristicaFAN[] getCaracteristicasFAN() {
            return caracteristicasFAN;
      }

      public void setClasse(int classe) {
            this.classe = classe;
      }

      public IPadrao clone() {
            double[] carac = new double[caracteristica.length];
            CaracteristicaFAN[] caracFan = new
CaracteristicaFAN[caracteristicasFAN.length];
            for (int i = 0; i < caracteristica.length; i++) {
                  carac[i] = caracteristica[i];
            }
            for (int i = 0; i < caracteristicasFAN.length; i++) {
                  caracFan[i] = caracteristicasFAN[i].clone();
            }
            IPadrao p = new PadraoNormalizado(carac,this.classe);
            ((PadraoNormalizado)p).caracteristicasFAN = caracFan;

            return p;
      }
```

```
}
```

2.8 PACKAGE JFAN.FAN.TEMPERAS;

2.8.1 ITemperaSimulada.Java

```
package jfan.fan.temperas;

import jfan.fan.events.TrocaValorTemperaEvent;
import jfan.fan.events.TrocaValorTemperaListener;

public interface ITemperaSimulada {

	public double getValor();

	public void setLimiteMinimo(double d);

	public void setLimiteMaximo(double d);

	public void setStep(double d);

	public void step();

	public void reset();

	public double getLimiteMaximo();

	public double getLimiteMinimo();

	public double getStepValue();

	public void addTrocaValorTemperaListener(TrocaValorTemperaListener
listener);

	public void
removeTrocaValorTemperaListener(TrocaValorTemperaListener listener);

	public void fireTrocaValorTemperaEvent(TrocaValorTemperaEvent
event);

}
```

2.8.2 TemperaAleatoria.Java

```
package jfan.fan.temperas;

import java.util.EventListener;
import java.util.Random;

import javax.swing.event.EventListenerList;

import jfan.fan.events.TrocaValorTemperaEvent;
import jfan.fan.events.TrocaValorTemperaListener;

public class TemperaAleatoria implements ITemperaSimulada {
```

```
	private Random random = new Random();
	private double valor, valorMinDefinido = 0.7943;
	private double valorMax = 1.0, valorMin = 0.7943;
	private double step = 0.0;
	private EventListenerList listenerList = new EventListenerList();


	public double getValor() {
		if (valorMin >= valorMax) return valorMax;
		valor = ((valorMax*10000.)-(valorMin*10000.));
		if (valor < 1)
			valor = 1;
		valor = random.nextInt((int)valor);

		valor /= 10000;
		valor += this.valorMin;
		fireTrocaValorTemperaEvent(new
TrocaValorTemperaEvent(valor));
		return valor;
	}

	public void reset() {
		random = new Random();
		valorMin = valorMinDefinido;
	}

	public void setLimiteMaximo(double d) {
		this.valorMax = d;
	}

	public void setLimiteMinimo(double d) {
		valorMin = d;
		valorMinDefinido = d;
	}

	public void setStep(double d) {
		this.step = d;
	}

	public void step() {
		this.valorMin += step;
		if (valorMin >= valorMax) valorMin = valorMax;
	}

	public double getLimiteMaximo() {
		return valorMax;
	}

	public double getLimiteMinimo() {
		return valorMinDefinido;
	}

	public double getStepValue() {
		return step;
	}
```

```
//    ---------Para eventos de troca de valor atual-----------
      public void addTrocaValorTemperaListener(TrocaValorTemperaListener
listener)
      {
         listenerList.add(TrocaValorTemperaListener.class, listener);
      }

      public void
removeTrocaValorTemperaListener(TrocaValorTemperaListener listener)
      {
         listenerList.remove(TrocaValorTemperaListener.class, listener);
      }

      public void fireTrocaValorTemperaEvent(TrocaValorTemperaEvent
event)
      {
         EventListener[] listeners =
listenerList.getListeners(TrocaValorTemperaListener.class);
         for (EventListener l : listeners)
            ((TrocaValorTemperaListener) l).trocouValorTempera(event);
      }

}
```

2.8.3 TemperaSimuladaFAN.Java

```
package jfan.fan.temperas;

import java.util.EventListener;

import javax.swing.event.EventListenerList;

import jfan.fan.events.TrocaValorTemperaEvent;
import jfan.fan.events.TrocaValorTemperaListener;

public class TemperaSimuladaFAN implements ITemperaSimulada {

      private double valorAtual = 1.0;
      private double valorStep = 0.1;
      private double limiteMaximo =  1.0;
      private double limiteMinimo =  0.0;
      private EventListenerList listenerList = new EventListenerList();

      public double getValor() {
            return this.valorAtual;
      }

      public void reset() {
            this.valorAtual = this.limiteMinimo;
      }

      public void setLimiteMaximo(double d) {
            this.limiteMaximo = d;
      }
```

```
	public void setLimiteMinimo(double d) {
		this.limiteMinimo = d;
	}

	public void setStep(double d) {
		this.valorStep = d;

	}

	public void setValorAtual(double d) {
		this.valorAtual = d;
	}

	public void step() {
		this.valorAtual += this.valorStep;
		if (this.valorAtual > this.limiteMaximo) {
			this.valorAtual = this.limiteMaximo;
		}
		else if(this.valorAtual < this.limiteMinimo) {
			this.valorAtual = this.limiteMinimo;
		}
		fireTrocaValorTemperaEvent(new
TrocaValorTemperaEvent(valorAtual));
	}

	public double getLimiteMaximo() {
		return limiteMaximo;
	}

	public double getLimiteMinimo() {
		return limiteMinimo;
	}

	public double getStepValue() {
		return valorStep;
	}

//	---------Para eventos de troca de valor atual----------
	public void addTrocaValorTemperaListener(TrocaValorTemperaListener
listener)
	{
	   listenerList.add(TrocaValorTemperaListener.class, listener);
	}

	public void
removeTrocaValorTemperaListener(TrocaValorTemperaListener listener)
	{
	   listenerList.remove(TrocaValorTemperaListener.class, listener);
	}

	public void fireTrocaValorTemperaEvent(TrocaValorTemperaEvent
event)
	{
	   EventListener[] listeners =
listenerList.getListeners(TrocaValorTemperaListener.class);
	   for (EventListener l : listeners)
	      ((TrocaValorTemperaListener) l).trocouValorTempera(event);
```

```
	}

}
```

2.8.4 TipoTempera.Java

```
package jfan.fan.temperas;

public enum TipoTempera {
	TemperaAleatoria("TemperaAleatoria"),
	TemperaSimuladaFAN("TemperaSimulada");

	private String texto;

	private TipoTempera(String texto){
		this.texto = texto;
	}

	public String toString() {
		return texto;
	}

}
```

2.9 PACKAGE JFAN.FAN.UTILITARIAS;

2.9.1 CalcularCaracteristicaFANBatch.Java

```
package jfan.fan.utilitarias;

import java.util.ArrayList;

import jfan.fan.padroes.CaracteristicaFAN;
import jfan.fan.padroes.IPadrao;
import jfan.fan.utilitarias.fuzzy.FuncaoPertinencia;

public class CalcularCaracteristicasFANBatch {
	/**
	 * O raio difuso da rede.
	 */
	private int raioDifuso;
	private int J;
	public CalcularCaracteristicasFANBatch(int raioDifuso, int
suporteConjuntoDifuso) {

		this.raioDifuso = raioDifuso;
		this.J = suporteConjuntoDifuso;
	}

	void calcularCaracteristicaFAN(IPadrao p, int caracteristica) {
		double[] conjuntoDifuso;

		// Calcula o tamanho do array para gerar o conjunto difuso
		int tamanhoConjuntoDifuso = (2 *this.raioDifuso)+1;
```

```
            // Para cada padrao calcula o cojunto da caracteristica
selecionada
                  //o conjunto difuso que representa a caracteristicaFAN
                  conjuntoDifuso = new double[tamanhoConjuntoDifuso];
                  // Calcula o meio do array
                  int meio = (int) ((tamanhoConjuntoDifuso / 2))/* - 1*/;
                  //Calcula os suportes a e b
                  double centro =
p.getCaracteristicaNormalizada(caracteristica) * J;
                  }*/
                  int a = ((int)centro) - this.raioDifuso -1 /*+ 1*/;
                  int b = ((int)centro) + this.raioDifuso + 1;

                  // Atribui a pertinencia do centro
                  conjuntoDifuso[meio] =
FuncaoPertinencia.Triangular.calcularPertinencia((int)centro,centro,raioD
ifuso);
                  // Preenche o array de conjunto difusos
                  int valor = 0;
                  for (int i = 1; i <= this.raioDifuso; i++) {
                        valor = (int) (centro-i);
                        conjuntoDifuso[meio - (i)] =
FuncaoPertinencia.Triangular
                                    .calcularPertinencia(
                                                valor,
                                                centro,
                                                raioDifuso);
                  }
                  for (int i = 1; i <= this.raioDifuso; i++) {
                        valor = (int) (centro + i);
                        conjuntoDifuso[meio + (i)] =
FuncaoPertinencia.Triangular
                                    .calcularPertinencia(
                                                valor,
                                                centro,
                                                raioDifuso);
                  }

                  p.setCaracteristicaFAN(caracteristica,
                              new CaracteristicaFAN(conjuntoDifuso, a,
b));
      }

      public void calcularCaracteristicasFAN(ArrayList<IPadrao> padroes)
{
            if (padroes.size() == 0) return;
            int numCaracteristicas =
padroes.get(0).getQuantasCaracteristicas();

            for (int i = 1; i <= numCaracteristicas; i++) {
                  for (IPadrao p: padroes) {
                        calcularCaracteristicaFAN(p,i);
                  }
            }
            int k = 0;
            k++;
```

```
	}
}
```

2.9.2 CalcularCaracteristicaFANThread.Java

```
package jfan.fan.utilitarias;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public class CalcularCaracteristicasFANThread implements Runnable {
	/**
	 * A caracterísca selecionada para ser calculada.
	 */
	private int caracteristicaSelecionada;

	/**
	 * O conjunto de padrões que terá a a característica calculada.
	 */
	private ArrayList<IPadrao> padroes = null;

	/**
	 * O raio difuso da rede.
	 */
	private int raioDifuso;

	/**
	 * O parâmetro de suporte a conjuntos difusos da rede.
	 */
	private int J;

	/**
	 * Instancia a classe para pronta execução do método run.
	 */
	public CalcularCaracteristicasFANThread(ArrayList<IPadrao> padroes,
int numCaracteristica, int raioDifuso, int suporteConjuntoDifuso) {
		this.caracteristicaSelecionada = numCaracteristica;
		this.padroes = padroes;
		this.raioDifuso = raioDifuso;
		this.J = suporteConjuntoDifuso;
	}

	/**
	 * Retorna a característica que foi selecionada.
	 * @return a característica que foi selecionada.
	 */
	public int getCaracteristica() {
		return this.caracteristicaSelecionada;
	}

	/**
	 * Retorna o conjunto de padrões que foi ou será utilizado.
	 * @return o conjunto de padrões que foi ou será utilizado.
	 */
```

```
	public ArrayList<IPadrao> getPadroes() {
		return this.padroes;
	}

	/**
	 * Executa o calculo das caracteristicasFAN de forma paralela.
	 */
	public void run() {
		CalcularCaracteristicasFANBatch calc = new
CalcularCaracteristicasFANBatch(this.raioDifuso,this.J);;
		for (IPadrao p : this.padroes)

	calc.calcularCaracteristicaFAN(p,this.caracteristicaSelecionada);
	}
}
```

2.9.3 DetectorMaxMinMeans.Java

```
package jfan.fan.utilitarias;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;

public abstract class DetectorMaxMinMean {

	public static double[] procurarMaximos (ArrayList<IPadrao>...
padroes) {
		double[][] maxs = null;
		double[] max = null;
		for (int i = 0; i < padroes.length; i++) {
			if (padroes[i].size() == 0) {
				continue;
			}
			else {
				maxs = new
double[padroes.length][padroes[i].get(0).getQuantasCaracteristicas()];
				max = new
double[padroes[i].get(0).getQuantasCaracteristicas()];
			}

		}
		for (int i = 0; i < padroes.length; i++) {
			maxs[i] = procurarMaximos(padroes[i]);
		}

		for(int i = 0; i < max.length; i++) {
			int index = 0;
			for (int j = 0; j < maxs.length; j++) {
				if (maxs[j] == null) {
					continue;
				}
				else {
					index = j;
					break;
```

```
                        }
                  }


                  max[i] = maxs[index][i];
                  for (int j = 1; j < padroes.length; j++) {
                        if (maxs[j] == null) continue;
                        if (max[i]<maxs[j][i]) max[i] = maxs[j][i];
                  }
            }
            return max;
      }

      public static double[] procurarMedias (ArrayList<IPadrao>...
padroes) {
            double[] mean = null;

            for (int i = 0; i < padroes.length; i++) {
                  if (padroes[i].size() == 0) {
                        continue;
                  }
                  else {
                        mean = new
double[padroes[i].get(0).getQuantasCaracteristicas()];
                  }

            }
            int count;
            for (int i = 0; i < mean.length; i++)
                  mean[i] = 0.0;
            for (int i = 0; i < mean.length; i++) {
                  count = 0;
                  for(ArrayList<IPadrao> arrayListPadroes : padroes){
                        for(IPadrao p : arrayListPadroes) {
                              mean[i] += p.getCaracteristica(i+1);
                              count++;
                        }
                  }
                  mean[i] /= count;
            }
            return mean;
      }

      public static double[] procurarMaximos(ArrayList<IPadrao> padroes)
{
            if (padroes.size() == 0) return null;
            double[] max = new
double[padroes.get(0).getQuantasCaracteristicas()];
            for (int i = 0; i < max.length; i++)
                  max[i] = Double.MIN_VALUE;
            for (int i = 0; i < max.length; i++) {
                  for(IPadrao p : padroes) {
                        if (max[i] < p.getCaracteristica(i+1)) {
                              max[i] = p.getCaracteristica(i+1);
                        }
                  }
            }
```

```
            return max;
      }

      public static double[] procurarMedias(ArrayList<IPadrao> padroes) {
            if (padroes.size() == 0) return null;
            double[] means = new
double[padroes.get(0).getQuantasCaracteristicas()];
            int count;
            for (int i = 0; i < means.length; i++)
                  means[i] = 0.0;
            for (int i = 0; i < means.length; i++) {
                  count = 0;
                  for(IPadrao p : padroes) {
                        means[i] += p.getCaracteristica(i+1);
                        count++;
                  }
                  means[i] /= count;
            }
            return means;
      }


      public static double procurarMaximo(ArrayList<IPadrao> padroes, int
caracteristica) {
            double max = Double.MIN_VALUE;
            for(IPadrao p : padroes) {
                        if (max < p.getCaracteristica(caracteristica)) {
                              max = p.getCaracteristica(caracteristica);
                        }
                  }
            return max;
      }

      public static double procurarMedia(ArrayList<IPadrao> padroes, int
caracteristica) {
            double media = 0.0;
            double count = 0;
            for(IPadrao p : padroes) {
                        media += p.getCaracteristica(caracteristica);
                        count++;
            }
            media /= count;
            return media;
      }

      public static double[] procurarMinimos (ArrayList<IPadrao>...
padroes) {

            double[][] mins = null;
            double[] min = null;
            for (int i = 0; i < padroes.length; i++) {
                  if (padroes[i].size() == 0) {
                        continue;
                  }
                  else {
                        mins = new
double[padroes.length][padroes[i].get(0).getQuantasCaracteristicas()];
```

```
                        min = new
double[padroes[i].get(0).getQuantasCaracteristicas()];
                  }

            }

            for (int i = 0; i < padroes.length; i++) {
                  mins[i] = procurarMinimos(padroes[i]);
            }
            for(int i = 0; i < min.length; i++) {
                  int index = 0;
                  for (int j = 0; j < mins.length; j++) {
                        if (mins[j] == null) {
                              continue;
                        }
                        else {
                              index = j;
                              break;
                        }
                  }
                  min[i] = mins[index][i];
                  for (int j = 1; j < padroes.length; j++) {
                        if (mins[j] == null) continue;
                        if (min[i]>mins[j][i]) min[i] = mins[j][i];
                  }
            }
            return min;
      }

      public static double procurarMinimo(ArrayList<IPadrao> padroes, int
caracteristica) {
            double max = Double.MAX_VALUE;
            for(IPadrao p : padroes) {
                        if (max > p.getCaracteristica(caracteristica)) {
                              max = p.getCaracteristica(caracteristica);
                        }
                  }
            return max;
      }

      public static double[] procurarMinimos(ArrayList<IPadrao> padroes)
{
            if (padroes.size() == 0) return null;
            double[] max = new
double[padroes.get(0).getQuantasCaracteristicas()];
            for (int i = 0; i < max.length; i++)
                  max[i] = Double.MIN_VALUE;
            for (int i = 0; i < max.length; i++){
                  max[i] =procurarMinimo(padroes, i+1);
            }
            return max;
      }

}
```

2.9.4 RecalculadorClasses.Java

```
package jfan.fan.utilitarias;

import java.util.ArrayList;
import java.util.HashMap;

import jfan.fan.NeuronioFAN;
import jfan.fan.padroes.IPadrao;

public class RecalculadorClasses {

	private static HashMap<Integer, Integer> classesMapRealIndex;
	private static HashMap<Integer, Integer> classesMapIndexReal;
	public static void indexarClasses(int[] classes, ArrayList<IPadrao>
padroes) {
		try {
		//Mapeamento usando hash para agilizar o processo
		if (classesMapIndexReal == null || classesMapRealIndex ==
null) {
			classesMapRealIndex = new HashMap<Integer,
Integer>(classes.length);
			classesMapIndexReal = new HashMap<Integer,
Integer>(classes.length);
		}

		for (int i = 0; i < classes.length; i++) {
			if (!classesMapRealIndex.containsKey(classes[i])) {
				classesMapRealIndex.put(classes[i], i);
			}
			if (!classesMapIndexReal.containsKey(i)) {
				classesMapIndexReal.put(i,classes[i]);
			}
		}
		//Substitui as classes originais pelos indices delas.
		Integer novaClasse = null;
		for (IPadrao p : padroes) {
			novaClasse = classesMapRealIndex.get(p.getClasse());
			if (novaClasse == null) {
				int i = p.getClasse();
				System.out.println(i);
			}
			p.setClasse(novaClasse);

		}
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}

	public static void indexarNeuronios (int[] classes,
ArrayList<NeuronioFAN> neuronios) {
		try {
			if (classesMapIndexReal == null || classesMapRealIndex
== null) {
```

```
                        classesMapRealIndex = new HashMap<Integer,
Integer>(classes.length);
                        classesMapIndexReal = new HashMap<Integer,
Integer>(classes.length);
                  }

                  for (int i = 0; i < classes.length; i++) {
                        if (!classesMapRealIndex.containsKey(classes[i]))
{
                              classesMapRealIndex.put(classes[i], i);
                        }

                        if (!classesMapIndexReal.containsKey(i)) {
                              classesMapIndexReal.put(i,classes[i]);
                        }
                  }
                  Integer novaClasse = null;
                  for (NeuronioFAN n : neuronios) {
                        novaClasse =
classesMapRealIndex.get(n.getClasseAssociada());
                        n.setClasseAssociada(novaClasse);
                  }
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public static void reverterIndexacaoPadroes(ArrayList<IPadrao>
padroes) {
            if (padroes != null && classesMapIndexReal != null) {
                  for (IPadrao p : padroes) {
                        int i = classesMapIndexReal.get(p.getClasse());
                        p.setClasse(i);
                  }
            }
      }

      public static HashMap<Integer, Integer> getClassesMapIndexReal() {
            return classesMapIndexReal;
      }

      public static HashMap<Integer, Integer> getNeuroniosMapIndexReal()
{
            return classesMapIndexReal;
      }

      public static HashMap<Integer, Integer> getClassesMapRealIndex() {
            return classesMapRealIndex;
      }

      public static HashMap<Integer, Integer> getNeuroniosMapRealIndex()
{
            return classesMapRealIndex;
      }

}
```

2.10 PACKAGE JFAN.FAN.UTILITARIAS.FUZZY;

2.10.1 FuncaoPertinencia.Java

```
package jfan.fan.utilitarias.fuzzy;

public class FuncaoPertinencia {

	/**
	 * Classe que mantém os métodos para calcular a pertinência
	 * usando uma função triangular.
	 */
	public static class Triangular {
		/**
		 * Calcula a pertinência de um valor.
		 */
		public static double  calcularPertinencia(double valor,
double inicio, double centro, double fim) {
			if (valor< inicio || fim <= valor)
				return 0;
			else {
				if (inicio <= valor && valor < centro) {
					return ((valor-inicio)/(centro-inicio));
				}
				else {
					return ((fim-valor)/(fim-centro));
				}
			}
		}

		/**
		 * Calcula a pertinência de um valor.
		 */
		public static double  calcularPertinencia(double valor,
double centro, int raioDecomposicao) {
			raioDecomposicao++;
			if (valor<centro)
				return (float) (valor+raioDecomposicao-
centro)/(raioDecomposicao);
			else
				return (float) (-
valor+raioDecomposicao+centro)/(raioDecomposicao);

		}
	}
	/**
	 * Classe que mantém os métodos para calcular a pertinência
	 * usando uma função trapezoidal
	 */
	public static class Trapezoidal {
		/**
		 * Calcula a pertinência de um valor.
		 */
```

```
            public static double calcularPertinencia(double valor, double
inicio, double canto1, double canto2, double fim) {
                  if (valor < inicio || fim <= valor) {
                        return 0f;
                  } else if (canto1 <= valor && valor < canto2){
                        return 1f;
                  } else if (inicio <= valor && valor < canto1){
                        return ((valor-inicio)/(canto1-inicio));
                  } else {
                        return ((fim-valor)/(fim-canto2));
                  }
            }
      }
      /**
       * Classe que mantém os métodos para calcular a pertinência
       * usando uma função gaussiana.
       */
      public static class Gaussiana {
            /**
             * Calcula a pertinência de um valor.
             */
            public static double calcularPertinencia(double valor, double
centro, double largura) {
                  return (Math.exp(-0.5*Math.pow(((valor-
centro)/largura),2)));
            }
      }

      /**
       * Classe que mantém os métodos para calcular a pertinência
       * usando uma função sino generalizada.
       */
      public static class Sino {
            /**
             * Calcula a pertinência de um valor.
             */
            public static double calcularPertinencia(double valor, double
centro, double largura, double declive) throws Exception {
                  if (largura == 0) throw new Exception("O valor de
largura da função de pertinência sino não pode ser 0.");
                  return (1/(1+Math.pow((Math.abs((valor-
centro)/largura)),(2*declive))));
            }
      }
      /**
       * Classe que mantém os métodos para calcular a pertinência
       * usando uma função senoidal.
       */
      public static class Senoidal {
            /**
             * Calcula a pertinência de um valor.
             */
            public static double calcularPertinencia(double valor, double
centro, double declive) {
                  return (double)(1/(1+(Math.exp(-declive*(valor-
centro)))));
            }
```

```
	}
	public static class SimetricaFechada {
		public static double calcularPertinencia(double valor, double
centro1, double declive1, double centro2, double declive2) {
			if ((declive1 < 0 && 0 <= declive2) || (declive1 >= 0
&& 0 > declive2)) {
				return (1/((1+Math.exp(-declive1*(valor-
centro1)))*(1+Math.exp(-declive2*(valor-centro2)))));
			} else {
				return (((1+Math.exp(-declive2*(valor-centro2)))-
(1+Math.exp(-declive1*(valor-centro1))))/((1+Math.exp(-declive1*(valor-
centro1)))*(1+Math.exp(-declive2*(valor-centro2)))));
			}
		}
	}
	//Fim da classe para função simétrica fechada
}
```

2.11 PACKAGE JFAN.IO;

2.11.1 IGerenciadorPadroes.Java

```
package jfan.io;

import java.util.ArrayList;
import java.util.HashMap;
import jfan.fan.padroes.IPadrao;

public interface IGerenciadorPadroes {
	public void adicionarConjuntoTreinamento(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception;
	public String[][] getConjuntoTreinamento() throws Exception;
	public ArrayList<IPadrao> getConjuntoTreinamentoRede() throws
Exception;
	public boolean getTextualConjuntoTreinamentoRede(int
umIndice)throws Exception;

	public void adicionarConjuntoTeste(String umNomeDeArquivo, String
umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception;
	public String[][] getConjuntoTeste() throws Exception;
	public ArrayList<IPadrao> getConjuntoTesteRede() throws Exception;
	public boolean getTextualConjuntoTesteRede(int umIndice)throws
Exception;

	public void adicionarConjuntoValidacao(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception;
	public String[][] getConjuntoValidacao() throws Exception;
	public ArrayList<IPadrao> getConjuntoValidacaoRede() throws
Exception;
	public boolean getTextualConjuntoValidacaoRede(int umIndice)throws
Exception;
```

```
	public void adicionarConjuntoClassificacao(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception;
	public String[][] getConjuntoClassificacao() throws Exception;
	public ArrayList<IPadrao> getConjuntoClassificacaoRede() throws
Exception;
	public boolean getTextualConjuntoClassificacaoRede(int
umIndice)throws Exception;

	public HashMap<Integer,String> getMapaConjunto()throws Exception;

}
```

2.11.2 ILeitura.Java

```
package jfan.io;

import java.util.ArrayList;

import jfan.fan.padroes.IPadrao;


public interface ILeitura {
	public int getNumClasses() throws Exception;
	public int getNumCaracteristicas() throws Exception;
	public int getNumPadroes() throws Exception;
	public int[] getClasses() throws Exception;
	public ArrayList<IPadrao> getPadroes() throws Exception;
	public String[][] getPadroesOriginais() throws Exception;
}
```

2.11.3 IPersistorDados.Java

```
package jfan.io;

import java.util.ArrayList;
import jfan.fan.NeuronioFAN;
import jfan.fan.padroes.IPadrao;

public interface IPersistorDados {
	public void gravaPadroesDat(ArrayList<IPadrao> listaComPadroes,
String caminhoAbsolutoNomeArquivo) throws Exception;
	public void gravaNeuroniosFanDat(ArrayList<NeuronioFAN>
listaComNeuroniosFAN, String caminhoAbsolutoNomeArquivo) throws
Exception;
}
```

2.12 PACKAGE JFAN.IO.ARQUIVOS;

2.12.1 CarregaDAT.Java

```
package jfan.io.arquivos;
```

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.Vector;

import jfan.fan.padroes.IPadrao;
import jfan.fan.padroes.PadraoNormalizado;
import jfan.io.ILeitura;

public class CarregaDat implements ILeitura {

	/**
	 * Contém os padrões normalizados
	 */
	public ArrayList<IPadrao> padroes = null;

	/**
	 * Define o tipo de separador de colunas no arquivo de padrões
	 */
	private String separadorColuna = "  ";

	/**
	 * Define o nome de arquivo de padrões
	 */
	private String nomeArquivo = null;

	public CarregaDat(String umNomeDeArquivo, String
umSeparadorDeColuna) {
		this.nomeArquivo = umNomeDeArquivo;
		this.separadorColuna = umSeparadorDeColuna;
	}

	public int getNumClasses() throws Exception {
		BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
		ArrayList<Double> asClasses = new ArrayList<Double>();
		String linha = "";
		int contadorLinhas = 0;
		while ((linha = arqDados.readLine()) != null) {
			StringTokenizer t = new StringTokenizer(linha,
separadorColuna);
			int contadorTokens = 0;
			while (t.hasMoreTokens()) {
				if (t.nextToken().length() != 0
						&& contadorTokens == t.countTokens())

	asClasses.add(Double.parseDouble(t.nextToken()));
					else
						t.nextToken();
					contadorTokens += 1;
			}
			contadorLinhas += 1;
		}
		ArrayList<Double> achados = new ArrayList<Double>();
		for (int i = 0; i < asClasses.size(); i++)
			if (!achados.contains(asClasses.get(i)))
```

```
                        achados.add(asClasses.get(i));
            arqDados.close();
            return achados.size();
      }

      public int[] getClasses() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            ArrayList<Double> asClasses = new ArrayList<Double>();
            String linha = "";
            int contadorLinhas = 0;
            while ((linha = arqDados.readLine()) != null) {
                  StringTokenizer t = new StringTokenizer(linha,
separadorColuna);
                  int contadorTokens = 0;
                  while (t.hasMoreTokens()) {
                        if (t.nextToken().length() != 0
                                    && contadorTokens == t.countTokens())

      asClasses.add(Double.parseDouble(t.nextToken()));
                        else
                              t.nextToken();
                        contadorTokens += 1;
                  }
                  contadorLinhas += 1;
            }
            ArrayList<Double> achados = new ArrayList<Double>();
            for (int i = 0; i < asClasses.size(); i++)
                  if (!achados.contains(asClasses.get(i)))
                        achados.add(asClasses.get(i));
            arqDados.close();
        int[] classes = new int[achados.size()];
        for (int i=0;i<achados.size();i++)
            classes[i]=achados.get(i).intValue();
            return classes;
      }

      public int getNumCaracteristicas() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            String linha = "";
            linha = arqDados.readLine();
            StringTokenizer t = new StringTokenizer(linha,
separadorColuna);
            arqDados.close();
            return t.countTokens() - 1;
      }

      public int getNumPadroes() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            int c = 0;
            while ((arqDados.readLine()) != null)
                  c += 1;
            arqDados.close();
            return c;
      }
```

```
	public ArrayList<IPadrao> getPadroes() throws Exception {
		BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
		String linha = "";
		Vector<Double> asCaracteristicas = new Vector<Double>();
		int classe = 0;
		padroes = new ArrayList<IPadrao>();
		padroes.clear();
		while ((linha = arqDados.readLine()) != null) {
			StringTokenizer t = new StringTokenizer(linha,
separadorColuna);
			while (t.hasMoreTokens())

	asCaracteristicas.add(Double.parseDouble(t.nextToken()));
			classe = asCaracteristicas.lastElement().intValue();

	asCaracteristicas.removeElementAt(asCaracteristicas.size() - 1);
			asCaracteristicas.trimToSize();

			double[] copiaDasCaracteristicas = new
double[asCaracteristicas.size()];
			for (int j = 0; j < asCaracteristicas.size(); j++)
				copiaDasCaracteristicas[j] =
asCaracteristicas.elementAt(j);

			padroes.add(new
PadraoNormalizado(copiaDasCaracteristicas, classe));
			asCaracteristicas.removeAllElements();
			padroes.trimToSize();
		}
		arqDados.close();
		return padroes;
	}

	public void setArquivo(String umArquivo) throws Exception {
		this.nomeArquivo = umArquivo;
	}

	public void setSeparadorColuna(String umSeparadorColuna) throws
Exception {
		this.separadorColuna = umSeparadorColuna;
	}

	public int getNumLinhasArquivo() throws Exception {
		BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
		int contadorLinhas = 0;
	  while ((arqDados.readLine())!=null)
			contadorLinhas+=1;
	  arqDados.close();
	  return contadorLinhas;
	}

	public int getNumColunasArquivo() throws Exception {
```

```
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            int contadorColunas = 0;
            String linha = "";
        while ((linha = arqDados.readLine())!=null){
            StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
            contadorColunas=t.countTokens();
            break;
        }
        arqDados.close();
        return contadorColunas;
      }

      public String[][] getPadroesOriginais() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            String linha = "";
            Vector<String> umPadrao = new Vector<String>();
            String[][] arrayPadroes = new
String[getNumLinhasArquivo()][getNumColunasArquivo()];
            int contadorLinhas = 0;
            while ((linha = arqDados.readLine())!=null){
                  contadorLinhas+=1;
                  StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
                  while (t.hasMoreTokens())
                        umPadrao.add(t.nextToken());
                  for (int i=0;i<umPadrao.size();i++)
                        arrayPadroes[contadorLinhas-
1][i]=umPadrao.elementAt(i);
                  umPadrao.removeAllElements();
            }
            arqDados.close();
            return arrayPadroes;
      }

}
```

2.12.2 CarregaNeuronioFAN.Java

```
package jfan.io.arquivos;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.Vector;

import jfan.fan.NeuronioFAN;

public class CarregaNeuronioFAN {

      private String nomeArquivo = null;

      private String separadorColuna = null;
```

```
	int numSuporteConjuntoDifuso;

	int raioDifuso;

	public CarregaNeuronioFAN(String umArquivoDeNeuronios, String
umSeparadorDeColuna, int raioDifuso, int suporteConjuntosDifusos) throws
Exception{
		setNomeArquivo(umArquivoDeNeuronios);
		setSeparadorColuna(umSeparadorDeColuna);
		this.raioDifuso = raioDifuso;
		this.numSuporteConjuntoDifuso = suporteConjuntosDifusos;
	}

	public void setNomeArquivo(String umArquivo) throws Exception {
		this.nomeArquivo = umArquivo;
	}

	public String getNomeArquivo() throws Exception {
		return this.nomeArquivo;
	}

	public void setSeparadorColuna(String umSeparadorDeColuna) throws
Exception {
		this.separadorColuna = umSeparadorDeColuna;
	}

	public String getSeparadorColuna(){
		return this.separadorColuna;
	}

	public Double[][] getDadosArquivoNeuronio() throws Exception {
		BufferedReader arqDados = new BufferedReader(new
FileReader(getNomeArquivo()));
		String linha = "";
		Vector<Double> umaTupla = new Vector<Double>();
		Double[][] arrayNeuronios = new
Double[getNumLinhasArquivo()][getNumColunasArquivo()];
		int contadorLinhas = 0;
		while ((linha = arqDados.readLine())!=null){
			contadorLinhas+=1;
			StringTokenizer t = new
StringTokenizer(linha,getSeparadorColuna());
			while (t.hasMoreTokens()) {
				@SuppressWarnings("unused")
				final String s = t.nextToken();
				umaTupla.add(Double.parseDouble(t.nextToken()));
			}

			for (int i=0;i<umaTupla.size();i++)
				arrayNeuronios[contadorLinhas-
1][i]=umaTupla.elementAt(i);
			umaTupla.removeAllElements();
		}
		arqDados.close();
		return arrayNeuronios;
	}
```

```
      public int getNumLinhasArquivo() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(getNomeArquivo()));
            int contadorLinhas = 0;
        while ((arqDados.readLine())!=null)
                  contadorLinhas+=1;
        arqDados.close();
        return contadorLinhas;
      }

      public int getNumColunasArquivo() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(getNomeArquivo()));
            int contadorColunas = 0;
            String linha = "";
        while ((linha = arqDados.readLine())!=null){
            StringTokenizer t = new
StringTokenizer(linha,getSeparadorColuna());
            contadorColunas=t.countTokens();
            break;
        }
        arqDados.close();
        return contadorColunas;
      }

      public ArrayList<NeuronioFAN> getNeuroniosFAN() throws Exception {
            Double[][] arrayNeuronios = getDadosArquivoNeuronio();
            int numDeCaracteristicas = arrayNeuronios[0].length;
            int numDeClasses = (arrayNeuronios.length+1)/100;
            ArrayList<NeuronioFAN> listaComNeuronios  = new
ArrayList<NeuronioFAN>();
            listaComNeuronios.clear();

            int aLinhaNoArray = 0;
            for (int classe=1;classe<=numDeClasses;classe++){
                  NeuronioFAN neu = new
NeuronioFAN(numDeCaracteristicas,numSuporteConjuntoDifuso,raioDifuso,clas
se);
                  neu.inicializarZerado();
                  for (int
caracteristica=0;caracteristica<numDeCaracteristicas;caracteristica++){
                        int i = 0;
                        int fim = aLinhaNoArray+numSuporteConjuntoDifuso;
                        for (int posicao = aLinhaNoArray; posicao < fim;
posicao++){
                              neu.setValor(caracteristica+1, i ,
arrayNeuronios[posicao][caracteristica]);
                              i+=1;
                        }
                        neu.setSomatorio(caracteristica,
arrayNeuronios[aLinhaNoArray+numSuporteConjuntoDifuso][caracteristica]);
                  }
                  listaComNeuronios.add(neu);
                  aLinhaNoArray+=100;
            }
            return listaComNeuronios;
```

```
	}

	public static void main(String[] args) {
		try {

		} catch (Exception e) {
			e.printStackTrace();
		}

	}

}
```

2.12.3 CarregaNeuronioXML.Java

```
package jfan.io.arquivos;

import java.io.IOException;
import java.util.ArrayList;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import jfan.fan.NeuronioFAN;
import jfan.fan.normalizadores.NormalizatorTypes;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class CarregaNeuronioXML extends DefaultHandler{

	String tipoNormalizacao = null;
	NormalizatorTypes tipoNormal;

	int i = -1;

	String dado = null;
	String tag = null;
	Attributes atributo = null;

	private StringBuffer sb = new StringBuffer();

	NeuronioFAN neuronio;

	ArrayList<NeuronioFAN> listaComNeuronios = null;

	private final GerenciadorPadroes gerenciadorPadroes;

	double[] minimos;

	double[] maximos;
```

```
	double[] medias;

	/**
	 * Nome do projeto no arquivo xml
	 */
	String nomeDoProjeto = null;

	int numCaracteristicas = 0;

	/**
	 * Raio difuso
	 */
	int raioDifuso = 0;

	int numSuporteConjuntoDifuso = 0;

	int pesoNeuronio = 0;

	int classeAssocida = 0;

	/**
	 * Indica o indice na matriz neural com a caracteristica respectiva
no arquivo xml
	 */
	int indiceCaracteristica = 0;

	double somatorioCaracteristica = 0;


	private int proximoI(){
		this.i++;
		return this.i;
	}

	private void inicializaI(){
		this.i=-1;
	}

	private int atualI(){
		return this.i;
	}


	private int getIndiceCaracteristica(){
		return this.indiceCaracteristica;
	}

	private void setIndiceCaracteristica(int oIndice){
		this.indiceCaracteristica=oIndice;
	}

	private double getSomatorioCaracteristica(){
		return this.somatorioCaracteristica;
	}

	private void setSomatorioCaracteristica(double oSomatorio){
		this.somatorioCaracteristica=oSomatorio;
```

```
	}


	/** comeca um documento novo */
	public void startDocument() {
	}

	/** comeca uma tag nova */
	public void startElement(String uri,String localName,String
tag,Attributes atributos){
		setTag(tag);
		setAtributo(atributos);
	}

	/** recebe os dados de uma tag */
	/**Faz todo o processamento do documento**/
	public void characters(char[] ch, int start, int length) throws
SAXException{
		if (getTag().equals("projectname")){
			setNomeProjeto(new String (ch,start,length));
		}

		else if (getTag().equals("quantityfeatures")){
			setNumeroCaracteristicas(Integer.parseInt(new String
(ch,start,length)));
			try {
				minimos = new double[getNumeroCaracteristicas()];
				maximos = new double[getNumeroCaracteristicas()];
				medias = new double[getNumeroCaracteristicas()];
			} catch (Exception e) {
				e.printStackTrace();
			}
		}

		else if (getTag().equals("difuseradius")){
			setRaioDifuso(Integer.parseInt(new String
(ch,start,length)));
		}

		else if (getTag().equals("setdifusesupport")){
			setNumeroSuporteConjuntoDifuso(Integer.parseInt(new
String (ch,start,length)));
		}

		else if (getTag().equals("normalization")){

			if
(getAtributo(0).equalsIgnoreCase("NormalizatorArcSinMaxMin"))

	setTipoNormalizacao(NormalizatorTypes.NormalizatorArcSinMaxMin);

			else if
(getAtributo(0).equalsIgnoreCase("NormalizatorMaxMean"))

	setTipoNormalizacao(NormalizatorTypes.NormalizatorMaxMean);
```

```
                  else if
(getAtributo(0).equalsIgnoreCase("NormalizatorMaxMin"))

      setTipoNormalizacao(NormalizatorTypes.NormalizatorMaxMin);

                  else if
(getAtributo(0).equalsIgnoreCase("NormalizatorMax"))

      setTipoNormalizacao(NormalizatorTypes.NormalizatorMax);

            }

            else if (getTag().equals("/featuremin")){
                  minimos[Integer.parseInt(getAtributo(0))-
1]=Double.parseDouble(getAtributo(1));
            }
            else if (getTag().equals("/featuremax")){
                  maximos[Integer.parseInt(getAtributo(0))-
1]=Double.parseDouble(getAtributo(1));
            }
            else if (getTag().equals("/featuremean")){
                  medias[Integer.parseInt(getAtributo(0))-
1]=Double.parseDouble(getAtributo(1));
            }
            else if (getTag().equals("neurons")){
                  this.listaComNeuronios = new ArrayList<NeuronioFAN>();
            }

            else if (getTag().equals("neurone")){
                  setPesoNeuronio(Integer.parseInt(getAtributo(0)));
                  int classeInt;
                  try {
                        classeInt = Integer.parseInt(getAtributo(1));
                  }
                  catch (NumberFormatException e) {
                        classeInt =
gerenciadorPadroes.adicionarClasseParaMapeamento(getAtributo(1));
                  }

                  setClasseAssociada(classeInt);
                  try {
                        this.neuronio = new
NeuronioFAN(getNumeroCaracteristicas(),getNumeroSuporteConjuntoDifuso(),g
etRaioDifuso(),getClasseAssociada());
                        this.neuronio.inicializarZerado();

      this.neuronio.setPesoPenalizacao(getPesoNeuronio());
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }

            else if (getTag().equals("feature")){

      setIndiceCaracteristica(Integer.parseInt(getAtributo(0)));

      setSomatorioCaracteristica(Double.parseDouble(getAtributo(1)));
```

```
                  this.neuronio.setSomatorio(getIndiceCaracteristica()-1,
getSomatorioCaracteristica());
                  inicializaI();
            }

            else if (getTag().equals("/feature")){
                  inicializaI();
            }

            else if (getTag().equals("value")){
                  sb.append(ch,start,length);
            }
            else if (getTag().equals("/neurone")){
                  listaComNeuronios.add(neuronio);
                  inicializaI();
            }
            else if (getTag().equals("/value")) {
                  try {
                        int tam =
(getNumeroSuporteConjuntoDifuso())+(getRaioDifuso()*2)+1;
                        if ((atualI()+1)<=tam)

      this.neuronio.setValor(getIndiceCaracteristica(), proximoI()-
getRaioDifuso(), Double.parseDouble(sb.toString()));


                        sb.setLength(0);
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
      }

      public void endElement(String uri, String localName, String tag){
            setTag("/"+tag);
      }

      public void endDocument() {
            listaComNeuronios.trimToSize();
      }

      private String getTag() {
            return this.tag;
      }

      private void setTag(String umaTag){
            this.tag=umaTag;
      }

      private String getAtributo(int umIndice) {
            return this.atributo.getValue(umIndice);
      }

      private void setAtributo(Attributes atributos){
            this.atributo=atributos;
      }
```

```
private void setNomeProjeto(String oNomeDoProjeto){
      this.nomeDoProjeto=oNomeDoProjeto;
}

public String getNomeProjeto() throws Exception {
      return this.nomeDoProjeto;
}

private void setNumeroCaracteristicas(int quantasCaracteristicas){
      this.numCaracteristicas=quantasCaracteristicas;
}

public int getNumeroCaracteristicas() throws Exception {
      return this.numCaracteristicas;
}

private void setRaioDifuso(int oRaioDifuso){
      this.raioDifuso=oRaioDifuso;
}

public int getRaioDifuso() throws Exception {
      return this.raioDifuso;
}

private void setNumeroSuporteConjuntoDifuso(int oNumeroDeSuporte){
      this.numSuporteConjuntoDifuso=oNumeroDeSuporte;
}

public int getNumeroSuporteConjuntoDifuso() throws Exception {
      return this.numSuporteConjuntoDifuso;
}

private void setPesoNeuronio(int oPeso){
      this.pesoNeuronio=oPeso;
}

private int getPesoNeuronio() throws Exception {
      return this.pesoNeuronio;
}

private void setClasseAssociada(int aClasse){
      this.classeAssocida=aClasse;
}

private int getClasseAssociada() throws Exception {
      return this.classeAssocida;
}

public double[] getMinimos() throws Exception {
      return this.minimos;
}

public double[] getMaximos() throws Exception {
      return this.maximos;
}

public double[] getMedias() throws Exception {
```

```
            return this.medias;
      }

      public ArrayList<NeuronioFAN> getNeuroniosFAN() throws Exception {
            return listaComNeuronios;
      }

      public NormalizatorTypes getTipoNormalizacao() {
            return this.tipoNormal;
      }
      private void setTipoNormalizacao(NormalizatorTypes
tipoNormalizacao) {
            this.tipoNormal = tipoNormalizacao;
      }

      public CarregaNeuronioXML(String umArquivoComNeuronios,
GerenciadorPadroes gp) {
            gerenciadorPadroes = gp;
            try {

                  SAXParser parser =
SAXParserFactory.newInstance().newSAXParser();

                  InputSource input = new
InputSource(umArquivoComNeuronios);

                  parser.parse(input, this);

            }
            catch (ParserConfigurationException ex) {
                  ex.printStackTrace();
            }
            catch (SAXException ex) {
                  ex.printStackTrace();
            }
            catch (IOException ex) {
                  ex.printStackTrace();
            }
      }

      public static void main(String[] args) {
            try {
                  CarregaNeuronioXML cnx = new
CarregaNeuronioXML("c:\\rede.xml",null);
                  System.out.println(cnx.getTipoNormalizacao());
                  System.out.println(cnx.getMedias());
            }
            catch (Exception ex) {
                  ex.printStackTrace();

            }
      }

}
```

2.12.4 CarregaTxt.Java

```
package jfan.io.arquivos;


import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.Vector;

import jfan.fan.padroes.IPadrao;
import jfan.fan.padroes.Padrao;
import jfan.io.ILeitura;

public class CarregaTxt implements ILeitura {
      public ArrayList<IPadrao> padroes = null;

      private String separadorColuna = "  ";

      private String nomeArquivo = "";

      private int colunaInicial = 0;

      private int colunaFinal = 1;

      private int linhaInicial = 1;

      private int linhaFinal = 0;

      private int colunaClasse = 0;

      public CarregaTxt(String umNomeDeArquivo, String
umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal){
            this.nomeArquivo = umNomeDeArquivo;
            this.separadorColuna = umSeparadorDeColuna;
            this.colunaClasse = aColunaDaClasse;
            this.colunaInicial = aColunaInicial;
            this.colunaFinal = aColunaFinal;
            this.linhaInicial = aLinhaInicial;
            this.linhaFinal = aLinhaFinal;
      }

      public int[] getClasses() throws Exception {
        BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
        ArrayList<Double> asClasses = new ArrayList<Double>();
        String linha = "";
        int contadorLinhas = 0;
        while ((linha = arqDados.readLine())!=null){
            contadorLinhas+=1;
            if(getLinhaInicial()<=contadorLinhas &&
contadorLinhas<=getLinhaFinal()){
                  StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
                  int contadorTokens = 0;
```

```
                while (t.hasMoreTokens()){
                      if (contadorTokens==getColunaClasse())

      asClasses.add(Double.parseDouble(t.nextToken()));
                      else
                            t.nextToken();
                      contadorTokens+=1;
                }
            }
        }
        ArrayList<Double> achados = new ArrayList<Double>();
        for (int i=0;i<asClasses.size();i++)
            if (!achados.contains(asClasses.get(i)))
                  achados.add(asClasses.get(i));

        int[] classes = new int[achados.size()];
        for (int i=0;i<achados.size();i++)
            classes[i]=achados.get(i).intValue();
        arqDados.close();
        return classes;
      }

      public int getNumClasses() throws Exception {
        BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
        ArrayList<Double> asClasses = new ArrayList<Double>();
        String linha = "";
        int contadorLinhas = 0;
        while ((linha = arqDados.readLine())!=null){
            contadorLinhas+=1;
            if(getLinhaInicial()<=contadorLinhas &&
contadorLinhas<=getLinhaFinal()){
                  StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
                  int contadorTokens = 0;
                  while (t.hasMoreTokens()){
                      if (contadorTokens==getColunaClasse())

      asClasses.add(Double.parseDouble(t.nextToken()));
                      else
                            t.nextToken();
                      contadorTokens+=1;
                }
            }
        }
        ArrayList<Double> achados = new ArrayList<Double>();
        for (int i=0;i<asClasses.size();i++)
            if (!achados.contains(asClasses.get(i)))
                  achados.add(asClasses.get(i));
        arqDados.close();
        return achados.size();
      }

      public int getNumCaracteristicas() throws Exception {
        BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            String linha = "";
```

```
        linha = arqDados.readLine();
        StringTokenizer t = new StringTokenizer(linha,separadorColuna);
            if (t.countTokens() < getColunaFinal())
                  setColunaFinal(t.countTokens());
        arqDados.close();
        return getColunaFinal()-getColunaInicial()+1;
      }

      public int getNumPadroes() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            int contadorLinhas = getLinhaInicial();
        while ((arqDados.readLine())!=null)
            if (getLinhaInicial()<=contadorLinhas &&
contadorLinhas<=getLinhaFinal())
                  contadorLinhas+=1;
        arqDados.close();
        return (contadorLinhas-getLinhaInicial());
      }

      public int getNumLinhasArquivo() throws Exception {
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            int contadorLinhas = 0;
        while ((arqDados.readLine())!=null)
                  contadorLinhas+=1;
        arqDados.close();
        return contadorLinhas;
      }


      public ArrayList<IPadrao> getPadroes() throws Exception{
                  BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
                  String linha = "";
                  Vector<Double> asCaracteristicas = new
Vector<Double>();
                  int classe = 0;
                  padroes = new ArrayList<IPadrao>();
                  padroes.clear();
                  int contadorLinhas = 0;
                  while ((linha = arqDados.readLine())!=null){
                        contadorLinhas+=1;
                        if (getLinhaInicial()<=contadorLinhas &&
contadorLinhas<=getLinhaFinal()){
                              StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
                              if (t.countTokens() < getColunaFinal())
                                    setColunaFinal(t.countTokens()-1);
                              while (t.hasMoreTokens())

      asCaracteristicas.add(Double.parseDouble(t.nextToken()));

                              double[] copiaDasCaracteristicas = new
double[getColunaFinal()-getColunaInicial()+1];
                              for (int j=0;j<getColunaFinal()-
getColunaInicial()+1;j++)
```

```
      copiaDasCaracteristicas[j]=asCaracteristicas.elementAt(getColunaIni
cial()+j);

                              classe =
asCaracteristicas.elementAt(getColunaClasse()).intValue();


      asCaracteristicas.removeElementAt(getColunaClasse());
                              asCaracteristicas.trimToSize();

                              for (int
i=0;i<copiaDasCaracteristicas.length;i++)

      System.out.print(copiaDasCaracteristicas[i]+" ");
                              System.out.println("Caracteríticas ");
                              System.out.println("Classe "+classe);

                              padroes.add(new
Padrao(copiaDasCaracteristicas,classe));
                              asCaracteristicas.removeAllElements();
                              padroes.trimToSize();
                        }
                  }
                  arqDados.close();
                  return padroes;
      }

      public String[][] getPadroesOriginais() throws Exception{
            BufferedReader arqDados = new BufferedReader(new
FileReader(nomeArquivo));
            String linha = "";
            Vector<String> umPadrao = new Vector<String>();
            String[][] arrayPadroes = new
String[getLinhaFinal()][getColunaFinal()-getColunaInicial()+2];
            int contadorLinhas = 0;
            int contadorColunas = 0;
            while ((linha = arqDados.readLine())!=null){
                  contadorLinhas+=1;
                  if (getLinhaInicial()<=contadorLinhas &&
contadorLinhas<=getLinhaFinal()){
                        StringTokenizer t = new
StringTokenizer(linha,separadorColuna);
                        if (t.countTokens() < getColunaFinal())
                              setColunaFinal(t.countTokens()-1);
                        contadorColunas = 0;
                        while (t.hasMoreTokens()){
                              if (contadorColunas>=getColunaInicial() &&
contadorColunas<=getColunaFinal())
                                    umPadrao.add(t.nextToken());
                              else if (contadorColunas==getColunaClasse())
                                    umPadrao.add(t.nextToken());
                              else
                                    t.nextToken();
                              contadorColunas+=1;
                        }
                        for (int i=0;i<umPadrao.size();i++){
```

```
                                    arrayPadroes[contadorLinhas-
1][i]=umPadrao.elementAt(i);
                              }
                              umPadrao.removeAllElements();
                        }
                  }
                  arqDados.close();
                  return arrayPadroes;
            }

            public void setArquivo(String umArquivo) throws Exception {
                  this.nomeArquivo = umArquivo;
            }

            public void setSeparadorColuna(String umSeparadorColuna) throws
Exception {
                  this.separadorColuna = umSeparadorColuna;
            }

            public void setColunaInicial(int umaColunaInicial) throws Exception
{
                  if (umaColunaInicial>0)
                        this.colunaInicial = umaColunaInicial-1;
                  else
                        this.colunaInicial = 0;
            }

            public int getColunaInicial() throws Exception {
                  return this.colunaInicial;
            }

            public void setColunaFinal(int umaColunaFinal) throws Exception {
                  this.colunaFinal = umaColunaFinal-1;
            }

            public int getColunaFinal() throws Exception {
                  return this.colunaFinal;
            }

            public void setLinhaInicial(int umaLinhaInicial) throws Exception {
                  if (umaLinhaInicial>0)
                        this.linhaInicial = umaLinhaInicial;
                  else
                        this.linhaInicial = 0;
            }

            public int getLinhaInicial() throws Exception {
                  return this.linhaInicial;
            }

            public int getLinhaFinal() throws Exception {
                  return this.linhaFinal;
            }

            public void setLinhaFinal(int umaLinhaFinal) throws Exception {
                  if (umaLinhaFinal<1)
                        this.linhaFinal=getNumLinhasArquivo();
```

```
            else
                  this.linhaFinal = umaLinhaFinal;
      }

      public void setColunaClasse(int umaColunaClasse) throws Exception {
            if (umaColunaClasse>0)
                  this.colunaClasse = umaColunaClasse-1;
            else
                  this.colunaClasse = 0;
      }

      public int getColunaClasse() throws Exception {
            return this.colunaClasse;
      }
}
```

2.12.5 CarregaXls.Java

```
package jfan.io.arquivos;

import java.io.File;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Vector;
import jfan.fan.padroes.IPadrao;
import jfan.fan.padroes.Padrao;
import jfan.io.ILeitura;
import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;

public class CarregaXls implements ILeitura {

      public ArrayList<IPadrao> padroes = null;

      private String nomeArquivo = "";

      private int colunaInicial = 0;

      private int colunaFinal = 1;

      private int linhaInicial = 1;

      private int linhaFinal = 0;

      private int colunaClasse = 0;

      public CarregaXls(String umNomeDeArquivo, int aColunaDaClasse, int
aColunaInicial, int aColunaFinal, int aLinhaInicial, int aLinhaFinal){
            this.nomeArquivo = umNomeDeArquivo;
            this.colunaClasse = aColunaDaClasse;
            this.colunaInicial = aColunaInicial;
            this.colunaFinal = aColunaFinal;
            this.linhaInicial = aLinhaInicial;
            this.linhaFinal = aLinhaFinal;
      }
```

```
	public int[] getClasses() throws Exception {
		ArrayList<String> asClasses = new ArrayList<String>();
		asClasses.clear();
		String[][] osPadroes = getPadroesOriginais();
		for (int i=0;i<osPadroes.length;i++)
		if (!asClasses.contains(osPadroes[i][osPadroes[0].length-1]))
			asClasses.add(osPadroes[i][osPadroes[0].length-1]);

		int[] arrayClasses = new int[asClasses.size()];
		int intClasse =0;
		Double doubleClasse;
		for (int i=0;i<asClasses.size();i++){
			doubleClasse = Double.parseDouble(asClasses.get(i));
			intClasse = doubleClasse.intValue();
			arrayClasses[i]=intClasse;
		}
		Arrays.sort(arrayClasses);
		return arrayClasses;
	}

	public int getNumCaracteristicas() throws Exception {
		Workbook workbook = Workbook.getWorkbook(new
File(getArquivo()));
	    Sheet sheet = workbook.getSheet(0);
	    int numColunas = sheet.getColumns();
	    workbook.close();
		return numColunas-1;
	}

	public int getNumClasses() throws Exception {
		ArrayList<String> asClasses = new ArrayList<String>();
		asClasses.clear();
		String[][] osPadroes = getPadroesOriginais();
		for (int i=0;i<osPadroes.length;i++)
		if (!asClasses.contains(osPadroes[i][osPadroes[0].length-1]))
			asClasses.add(osPadroes[i][osPadroes[0].length-1]);
		return asClasses.size();
	}

	public int getNumPadroes() throws Exception {
	    Workbook workbook = Workbook.getWorkbook(new
File(getArquivo()));
	    Sheet sheet = workbook.getSheet(0);
	    int numLinhas = sheet.getRows();
	    workbook.close();
		return numLinhas;
	}

	public ArrayList<IPadrao> getPadroes() throws Exception {
		padroes = new ArrayList<IPadrao>();
		padroes.clear();
		String[][] osPadroes = getPadroesOriginais();
		Vector<Double> umPadrao = new Vector<Double>();
		double[] arrayCaracteristicas = new
double[osPadroes[0].length-1];
		int classe = 0;
```

```
            for (int i=0;i<osPadroes.length;i++){
                  for (int j=0;j<osPadroes[0].length;j++)
                        umPadrao.add(Double.parseDouble(osPadroes[i][j]));

                  for(int k=0;k<umPadrao.size()-1;k++)
                        arrayCaracteristicas[k]=umPadrao.get(k);
                  classe=umPadrao.lastElement().intValue();
                  padroes.add(new Padrao(arrayCaracteristicas,classe));
                  umPadrao.clear();
            }
            return padroes;
      }

      public String[][] getPadroesOriginais() throws Exception {
          Workbook workbook = Workbook.getWorkbook(new
File(getArquivo()));
          Sheet sheet = workbook.getSheet(0);
          int numLinhas = sheet.getRows();
          int numColunas = sheet.getColumns();
          //obtem todos os dados da planilha excel
          String[][] arrayPadroes = new String[numLinhas][numColunas];
          for (int i=0;i<numLinhas;i++)
            for (int j=0;j<numColunas;j++){
                  Cell valorCelula = sheet.getCell(j,i);
                  if (valorCelula.getContents().length()==0)
                        arrayPadroes[i][j]="0.0";
                      else
                        arrayPadroes[i][j] =
valorCelula.getContents().replaceAll(",",".");
            }
          workbook.close();

          //faz verificação se número de linhas especificado é maior do
que contém no arquivo
          if (this.linhaFinal>numLinhas)
            this.linhaFinal=numLinhas;

          //faz verificação se número de colunas especificado é maior do
que contém no arquivo
          if (this.colunaFinal>numColunas)
            this.colunaFinal=numColunas;

          //monta array delimitando dados
          numLinhas = (this.linhaFinal-this.linhaInicial)+1;
          numColunas = (this.colunaFinal-this.colunaInicial)+2;
          String[][] arrayPadroesDelimitados = new
String[numLinhas][numColunas];
          int l = 0;
          int c = 0;
          for (int i=linhaInicial-1;i<linhaFinal;i++){
                c = 0;
            for (int j=colunaInicial-1;j<colunaFinal;j++){
                  arrayPadroesDelimitados[l][c]=arrayPadroes[i][j];
                  c++;
            }
            arrayPadroesDelimitados[l][arrayPadroesDelimitados[0].length-
1]=arrayPadroes[i][colunaClasse-1];
```

```
                l++;
            }

            return arrayPadroesDelimitados;
        }

        public String getArquivo() throws Exception {
              return this.nomeArquivo;
        }

        public static void main(String[] args) {
              try {
                    CarregaXls xls = new
CarregaXls("c:\\fan\\arquivo.xls",1,1,3,4000,20000);
                    System.out.println(xls.getPadroesOriginais());
              } catch (Exception e) {
                    e.printStackTrace();
              }
        }

}
```

2.12.6 CarregaXml.Java

```
package jfan.io.arquivos;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Vector;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import jfan.fan.padroes.IPadrao;
import jfan.fan.padroes.Padrao;
import jfan.io.ILeitura;

import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class CarregaXml extends DefaultHandler implements ILeitura{

        int i = -1;
        int j = 0;

        String dado = null;
        String tag = null;
        String atributo = null;

        private StringBuffer sb = new StringBuffer();

        String nomeDoProjeto = null;
```

```
	String tipoConjunto = null;

	int numPadroes = 0;

	int numClasses = 0;

	int numCaracteristicas = 0;

	int[] classes = null;

	public String[][] arrayPadroes = null;

	public String[] nomeCaracteristicas = null;

	private GerenciadorPadroes gerenciadorPadroes;

	private int proximoI(){
		this.i+=1;
		return this.i;
	}

	private void inicializaI(){
		this.i=-1;
	}

	private int atualI(){
		return this.i;
	}

	private int proximoJ(){
		this.j+=1;
		return this.j;
	}

	private int atualJ(){
		return this.j;
	}

	public void startDocument() {
	}

	public void startElement(String uri,String localName,String
tag,Attributes atributos){
		setTag(tag);
	}

	public void characters(char[] ch, int start, int length) throws
SAXException{
		if (getTag().equals("projectname")){
			setNomeProjeto(new String (ch,start,length));
		}
		else if (getTag().equals("quantitypatterns")){
			setNumPadroes(Integer.parseInt(new String
(ch,start,length)));
		}
		else if (getTag().equals("typeset")){
			setTipoConjunto(new String (ch,start,length));
```

```
            }
            else if (getTag().equals("quantityclass")){
                  setNumClasses(Integer.parseInt(new String
(ch,start,length)));
            }
            else if (getTag().equals("quantityfeatures")){
                  setNumCaracteristicas(Integer.parseInt(new String
(ch,start,length)));
                  try {
                        nomeCaracteristicas = new
String[getNumCaracteristicas()];
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            else if (getTag().equals("name")){
                  try {
                        nomeCaracteristicas[proximoI()] = new
String(ch,start,length);
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            else if (getTag().equals("classes")){
                  try {
                        classes = new int[getNumClasses()];
                        inicializaI();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }
            else if (getTag().equals("value")){
                  int classeInt;
                  String sClasse = new String (ch,start,length);
                  try {
                        classeInt = Integer.parseInt(sClasse);
                  }
                  catch (NumberFormatException e) {
                        classeInt =
gerenciadorPadroes.adicionarClasseParaMapeamento(sClasse);
                  }
                  setClasses(proximoI(),classeInt);
            }
            else if (getTag().equals("patterns")){
                  try {
                        this.arrayPadroes = new
String[getNumPadroes()][getNumCaracteristicas()+1];
                        inicializaI();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
            }



            else if (getTag().equals("feature")){
                        sb.append(ch,start,length);/**adiï¿½ï¿½o***/
```

```
            }

            else if (getTag().equals("/feature")){
                  try {
                        sb.append(ch,start,length);/**adiï¿½ï¿½o***/
                        if (atualI()<getNumCaracteristicas())
                              this.arrayPadroes[atualJ()][proximoI()] =
sb.toString();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
                  sb.setLength(0);
            }

            else if (getTag().equals("memberclass")){
                  sb.append(ch,start,length);/**adiï¿½ï¿½o***/
            }


            else if (getTag().equals("/memberclass")){
                  try {
                        if (atualI()<getNumCaracteristicas())
                            this.arrayPadroes[atualJ()][proximoI()] =
sb.toString();
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
                  sb.setLength(0);
            }

            else if (getTag().equals("/pattern")){
                        inicializaI();
                        proximoJ();
            }
      }

      public void endElement(String uri, String localName, String tag){
            setTag("/"+tag);
      }

      public void endDocument() {
      }


      private String getTag() {
            return this.tag;
      }

      private void setTag(String umaTag){
            this.tag=umaTag;
      }

      private void setNomeProjeto(String oNomeDoProjeto){
            this.nomeDoProjeto=oNomeDoProjeto;
      }

      public String getNomeProjeto() throws Exception {
```

```
            return this.nomeDoProjeto;
      }

      private void setTipoConjunto(String qualTipoConjunto){
            this.tipoConjunto=qualTipoConjunto;
      }

      public String getTipoConjunto() throws Exception {
            return this.tipoConjunto;
      }

      private void setNumPadroes(int numPadroes){
            this.numPadroes=numPadroes;
      }

      public int getNumPadroes() throws Exception {
            return this.numPadroes;
      }

      private void setNumCaracteristicas(int numCaracteristicas){
            this.numCaracteristicas=numCaracteristicas;
      }

      public int getNumCaracteristicas() throws Exception {
            return this.numCaracteristicas;
      }

      private void setClasses(int indiceNoArray, int valorClasse){
            this.classes[indiceNoArray]=valorClasse;
      }

      public int[] getClasses() throws Exception {
            return this.classes;
      }

      private void setNumClasses(int numeroDeClasses){
            this.numClasses=numeroDeClasses;
      }

      public int getNumClasses() throws Exception {
            return this.numClasses;
      }

      public String[] getNomeCaracteristicas() throws Exception {
            return this.nomeCaracteristicas;
      }

      public ArrayList<IPadrao> getPadroes() throws Exception {
            ArrayList<IPadrao> padroes = new ArrayList<IPadrao>();
            Vector<Double> umPadrao = new Vector<Double>();
            double[] arrayCaracteristicas = new
double[this.arrayPadroes[0].length-1];
            int classe = 0;
            for (int i=0;i<this.arrayPadroes.length;i++){
                  for (int j=0;j<this.arrayPadroes[0].length;j++)

      umPadrao.add(Double.parseDouble(this.arrayPadroes[i][j]));
```

```
                    for(int k=0;k<umPadrao.size()-1;k++)
                          arrayCaracteristicas[k]=umPadrao.get(k);
                    classe=umPadrao.lastElement().intValue();
                    padroes.add(new Padrao(arrayCaracteristicas,classe));
                    umPadrao.clear();
              }
              return padroes;
        }

        public String[][] getPadroesOriginais() throws Exception {
              return this.arrayPadroes;
        }

        public CarregaXml(String umNomeDeArquivo, GerenciadorPadroes gp){
              gerenciadorPadroes = gp;
              try {
                    SAXParser parser =
SAXParserFactory.newInstance().newSAXParser();

                    InputSource input = new InputSource(umNomeDeArquivo);

                    parser.parse(input, this);

              }
              catch (ParserConfigurationException ex) {
                    ex.printStackTrace();
              }
              catch (SAXException ex) {
                    ex.printStackTrace();
              }
              catch (IOException ex) {
                    ex.printStackTrace();
              }
        }


        public static void main(String[] args) {
              try {
                    CarregaXml txml = new CarregaXml("c:\\teste.xml",null);
                    System.out.println(txml.getNumClasses());
                    System.out.println(txml.getNumPadroes());
                    System.out.println(txml.getNumCaracteristicas());
                    System.out.println(txml.getNomeProjeto());
                    System.out.println(txml.getClasses());
                    System.out.println(txml.getTipoConjunto());
                    ArrayList<IPadrao> x = new ArrayList<IPadrao>();
                    x = txml.getPadroes();
                    System.out.println(x.get(0).getClasse());
                    System.out.println(x.get(1).getClasse());
                    System.out.println(x.get(4059).getClasse());
                    System.out.println();
              } catch (Exception ex) {
                    ex.printStackTrace();

              }
        }
```

```
}
```

2.12.7 GerenciadorPadroes.Java

```
package jfan.io.arquivos;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Vector;

import sun.reflect.generics.visitor.Reifier;
import jfan.exceptions.JfanFormatoArquivoException;
import jfan.fan.padroes.*;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.*;

public class GerenciadorPadroes implements IGerenciadorPadroes{

      ILeitura leitor;

      private String[] nomeCaracteristicas = null;

      private ArrayList<String> arrayMapa = new ArrayList<String>();

      private ArrayList<String> arrayMapaClasses = new
ArrayList<String>();

      private HashMap<Integer,String> mapaConjunto = null;
      private HashMap<Integer,String> mapaClasses = null;

      private HashMap<String, Integer> mapaConjuntoInverso = null;
      private HashMap<String,Integer> mapaClassesInverso = null;

      private final ArrayList<IPadrao> conjuntoTreinamentoRede = new
ArrayList<IPadrao>();

      private boolean[] colunasTextoTreinamento = null;

      private String[][] conjuntoTreinamento = null;

      private final ArrayList<IPadrao> conjuntoTesteRede = new
ArrayList<IPadrao>();
      private boolean[] colunasTextoTeste = null;
      private String[][] conjuntoTeste = null;

      private final ArrayList<IPadrao> conjuntoValidacaoRede = new
ArrayList<IPadrao>();
      private boolean[] colunasTextoValidacao = null;
      private String[][] conjuntoValidacao = null;

      private final ArrayList<IPadrao> conjuntoClassificacaoRede = new
ArrayList<IPadrao>();
      private boolean[] colunasTextoClassificacao = null;
```

```
	private String[][] conjuntoClassificacao = null;


	public GerenciadorPadroes(){
	}

	private boolean[] getColunasTexto(String[][] umArrayString){
		boolean[] arrayColunaTexto = new
boolean[umArrayString[0].length];
		for (int l=0;l<umArrayString.length;l++){
			for (int c=0;c<umArrayString[l].length;c++){
				try{
					Double.parseDouble(umArrayString[l][c]);
				}
				catch (Exception e) {
					arrayColunaTexto[c]=true;
					continue;
				}
			}
		}
		return arrayColunaTexto;
	}

	public String[] getNomeCaracteristicasTreinamento() throws
Exception {
		return this.nomeCaracteristicas;
	}

	public void adicionarConjuntoTreinamento(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception {
		try{
			String formatoArquivo =
umNomeDeArquivo.substring(umNomeDeArquivo.length()-
3,umNomeDeArquivo.length()).toUpperCase();
			if (formatoArquivo.equals("DAT")){
				leitor = new
CarregaDat(umNomeDeArquivo,umSeparadorDeColuna);
				((CarregaDat)leitor).setArquivo(umNomeDeArquivo);

	((CarregaDat)leitor).setSeparadorColuna(umSeparadorDeColuna);
				String[][] tempArray =
((CarregaDat)leitor).getPadroesOriginais();
				if (this.conjuntoTreinamento != null) {
					List<String[]> temp =
Arrays.asList(this.conjuntoTreinamento);
					ArrayList<String[]> tempArrList = new
ArrayList<String[]>(temp.size());
					for(String[] linha : temp) {
						tempArrList.add(linha);
					}
					for(String[] linha : tempArray) {
						tempArrList.add(linha);
					}
					this.conjuntoTreinamento =
tempArrList.toArray(new
String[tempArrList.size()][tempArrList.get(0).length]);
```

```
                        }
                        else {
                              this.conjuntoTreinamento = tempArray;
                        }
                  }
                  else if (formatoArquivo.equals("TXT")){
                        leitor = new
CarregaTxt(umNomeDeArquivo,umSeparadorDeColuna,aColunaDaClasse,aColunaIni
cial,aColunaFinal,aLinhaInicial,aLinhaFinal);
                        ((CarregaTxt)leitor).setArquivo(umNomeDeArquivo);

      ((CarregaTxt)leitor).setSeparadorColuna(umSeparadorDeColuna);

      ((CarregaTxt)leitor).setColunaClasse(aColunaDaClasse);

      ((CarregaTxt)leitor).setColunaInicial(aColunaInicial);
                        ((CarregaTxt)leitor).setColunaFinal(aColunaFinal);

      ((CarregaTxt)leitor).setLinhaInicial(aLinhaInicial);
                        ((CarregaTxt)leitor).setLinhaFinal(aLinhaFinal);

      this.conjuntoTreinamento=((CarregaTxt)leitor).getPadroesOriginais()
;
                  }
                  else if (formatoArquivo.equals("XLS")){
                        leitor = new
CarregaXls(umNomeDeArquivo,aColunaDaClasse,aColunaInicial,aColunaFinal,aL
inhaInicial,aLinhaFinal);

      this.conjuntoTreinamento=((CarregaXls)leitor).getPadroesOriginais()
;
                  }
                  else if (formatoArquivo.equals("XML")){
                        leitor = new CarregaXml(umNomeDeArquivo,this);

      this.conjuntoTreinamento=((CarregaXml)leitor).getPadroesOriginais()
;

      this.nomeCaracteristicas=((CarregaXml)leitor).getNomeCaracteristica
s();
                  }
                  else
                        throw new JfanFormatoArquivoException();

                  criaConjuntoTreinamentoRede();
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public String[][] getConjuntoTreinamento() throws Exception {
            return this.conjuntoTreinamento;
      }

      public ArrayList<IPadrao> getConjuntoTreinamentoRede() throws
Exception {
            return this.conjuntoTreinamentoRede;
```

```
	}

	private ArrayList<IPadrao> criaConjuntoTreinamentoRede() throws
Exception {
		this.conjuntoTreinamentoRede.clear();
		Vector<Double> umPadrao = new Vector<Double>();
		umPadrao.clear();
		this.colunasTextoTreinamento =
getColunasTexto(this.conjuntoTreinamento);
		for (int l=0;l<this.conjuntoTreinamento.length;l++){
			for (int c=0;c<this.conjuntoTreinamento[l].length;c++){
				try{
					if (this.colunasTextoTreinamento[c]){
						if (c ==
this.conjuntoTreinamento[l].length-1) {
							if
(!this.arrayMapaClasses.contains(this.conjuntoTreinamento[l][c])){

	this.arrayMapaClasses.add(this.conjuntoTreinamento[l][c]);

	umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoTreinamento[l][c])));
							}
							else

	umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoTreinamento[l][c])));
						}
						else if
(!this.arrayMapa.contains(this.conjuntoTreinamento[l][c])){

	this.arrayMapa.add(this.conjuntoTreinamento[l][c]);

	umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoTre
inamento[l][c])));
						}
						else
	umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoTre
inamento[l][c])));
					}
					else{
						double valorCelula =
Double.parseDouble(this.conjuntoTreinamento[l][c]);
						umPadrao.add(valorCelula);
					}
				}
				catch (Exception e) {
					e.printStackTrace();
					continue;
				}
			}
			double[] asCaracteristicas = new
double[umPadrao.size()-1];
			for (int j=0;j<umPadrao.size()-1;j++)
				asCaracteristicas[j]=umPadrao.elementAt(j);
			int valorClasse=umPadrao.lastElement().intValue();
```

```
                  this.conjuntoTreinamentoRede.add(new
Padrao(asCaracteristicas,valorClasse));
                  umPadrao.removeAllElements();
            }
            this.gerarMapeamentos();

            this.conjuntoTreinamentoRede.trimToSize();


      RecalculadorClasses.indexarClasses(getClassesConjuntoTreinamento(),
conjuntoTreinamentoRede);

            return this.conjuntoTreinamentoRede;
      }

      public boolean getTextualConjuntoTreinamentoRede(int umIndice){
            return this.colunasTextoTreinamento[umIndice];
      }

      public HashMap<Integer,String> getMapaConjunto(){
            return this.mapaConjunto;
      }

      public int[] getClassesConjuntoTreinamento(){

            return getClassesConjunto(conjuntoTreinamento,
colunasTextoTreinamento);
      }

      public int[] getClassesConjuntoTeste(){
            return getClassesConjunto(conjuntoTeste, colunasTextoTeste);
      }


      public String[] getClassesConjuntoTreinamentoReais(){
            return getClassesReais(conjuntoTreinamento);
      }

      public String[] getClassesConjuntoValidacaoReais(){
            return getClassesReais(conjuntoValidacao);
      }

      public String[] getClassesReais(String[][] conjunto){
            ArrayList<String> listaClasses = new ArrayList<String>();
            for(String[] arr : conjunto) {
                  if (!listaClasses.contains(arr[arr.length-1]))
                        listaClasses.add(arr[arr.length-1]);
            }
            String[] arrayClasses = new String[listaClasses.size()];
            for (int i=0;i<listaClasses.size();i++ )
                  arrayClasses[i]=listaClasses.get(i);
            Arrays.sort(arrayClasses);
            return arrayClasses;
      }

      private int[] getClassesConjunto(String[][] conjunto, boolean[]
colunasTexto){
```

```
            ArrayList<Integer> listaClasses = new ArrayList<Integer>();
            for(String[] arr : conjunto) {
                  if (colunasTexto[arr.length-1]) {
                        int i = mapaClassesInverso.get(arr[arr.length-1]);
                        if (!listaClasses.contains(i))
                              listaClasses.add(i);
                  }
                  else if
(!listaClasses.contains((int)Double.parseDouble(arr[arr.length-1])))

      listaClasses.add((int)Double.parseDouble(arr[arr.length-1]));
            }
            int[] arrayClasses = new int[listaClasses.size()];
            for (int i=0;i<listaClasses.size();i++ )
                  arrayClasses[i]=listaClasses.get(i);
            Arrays.sort(arrayClasses);
            return arrayClasses;
      }

      public void adicionarConjuntoTeste(String umNomeDeArquivo, String
umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception {
            try{
                  String formatoArquivo =
umNomeDeArquivo.substring(umNomeDeArquivo.length()-
3,umNomeDeArquivo.length()).toUpperCase();
                  if (formatoArquivo.equals("DAT")){
                        leitor = new
CarregaDat(umNomeDeArquivo,umSeparadorDeColuna);
                        ((CarregaDat)leitor).setArquivo(umNomeDeArquivo);

      ((CarregaDat)leitor).setSeparadorColuna(umSeparadorDeColuna);

      this.conjuntoTeste=((CarregaDat)leitor).getPadroesOriginais();

                  }
                  else if (formatoArquivo.equals("TXT")){
                        leitor = new
CarregaTxt(umNomeDeArquivo,umSeparadorDeColuna,aColunaDaClasse,aColunaIni
cial,aColunaFinal,aLinhaInicial,aLinhaFinal);
                        ((CarregaTxt)leitor).setArquivo(umNomeDeArquivo);

      ((CarregaTxt)leitor).setSeparadorColuna(umSeparadorDeColuna);

      ((CarregaTxt)leitor).setColunaClasse(aColunaDaClasse);

      ((CarregaTxt)leitor).setColunaInicial(aColunaInicial);
                        ((CarregaTxt)leitor).setColunaFinal(aColunaFinal);

      ((CarregaTxt)leitor).setLinhaInicial(aLinhaInicial);
                        ((CarregaTxt)leitor).setLinhaFinal(aLinhaFinal);

      this.conjuntoTeste=((CarregaTxt)leitor).getPadroesOriginais();
                  }
                  else if (formatoArquivo.equals("XLS")){
```

```
                        leitor = new
CarregaXls(umNomeDeArquivo,aColunaDaClasse,aColunaInicial,aColunaFinal,aL
inhaInicial,aLinhaFinal);

      this.conjuntoTeste=((CarregaXls)leitor).getPadroesOriginais();

                  }
                  else if (formatoArquivo.equals("XML")){
                        leitor = new CarregaXml(umNomeDeArquivo,this);

      this.conjuntoTeste=((CarregaXml)leitor).getPadroesOriginais();

                  }
                  else
                        throw new JfanFormatoArquivoException();

                  criaConjuntoTesteRede();
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public String[][] getConjuntoTeste() throws Exception {
            return this.conjuntoTeste;
      }

      public ArrayList<IPadrao> getConjuntoTesteRede() throws Exception {
            return this.conjuntoTesteRede;
      }

      private ArrayList<IPadrao> criaConjuntoTesteRede() throws Exception
{
            this.conjuntoTesteRede.clear();
            Vector<Double> umPadrao = new Vector<Double>();
            umPadrao.clear();
            this.colunasTextoTeste = getColunasTexto(this.conjuntoTeste);
            for (int l=0;l<this.conjuntoTeste.length;l++){
                  for (int c=0;c<this.conjuntoTeste[l].length;c++){
                        try{
                              if (this.colunasTextoTeste[c]){
                                    if (c == this.conjuntoTeste[l].length-
1) {
                                          if
(!this.arrayMapaClasses.contains(this.conjuntoTeste[l][c])){

      this.arrayMapaClasses.add(this.conjuntoTeste[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoTeste[l][c])));
                                          }
                                          else

      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoTeste[l][c])));
                                    }
                                    else if
(!this.arrayMapa.contains(this.conjuntoTeste[l][c])){
```

```
      this.arrayMapa.add(this.conjuntoTeste[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoTes
te[l][c])));
                                    }
                                    else

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoTes
te[l][c])));
                              }
                              else{
                                    double valorCelula =
Double.parseDouble(this.conjuntoTeste[l][c]);
                                    umPadrao.add(valorCelula);
                              }
                        }
                        catch (Exception e) {
                              e.printStackTrace();
                              continue;
                        }
                  }
                  double[] asCaracteristicas = new
double[umPadrao.size()-1];
                  for (int j=0;j<umPadrao.size()-1;j++)
                        asCaracteristicas[j]=umPadrao.elementAt(j);
                  int valorClasse=umPadrao.lastElement().intValue();
                  this.conjuntoTesteRede.add(new
Padrao(asCaracteristicas,valorClasse));
                  umPadrao.removeAllElements();
            }
            this.gerarMapeamentos();

            this.conjuntoTesteRede.trimToSize();


      RecalculadorClasses.indexarClasses(getClassesConjunto(conjuntoTeste
, colunasTextoTeste), conjuntoTesteRede);

            return this.conjuntoTesteRede;
      }

      public boolean getTextualConjuntoTesteRede(int umIndice){
            return this.colunasTextoTeste[umIndice];
      }

      public void adicionarConjuntoValidacao(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception {
            try{
                  String formatoArquivo =
umNomeDeArquivo.substring(umNomeDeArquivo.length()-
3,umNomeDeArquivo.length()).toUpperCase();
                  if (formatoArquivo.equals("DAT")){
                        leitor = new
CarregaDat(umNomeDeArquivo,umSeparadorDeColuna);
                        ((CarregaDat)leitor).setArquivo(umNomeDeArquivo);
```

```
      ((CarregaDat)leitor).setSeparadorColuna(umSeparadorDeColuna);

      this.conjuntoValidacao=((CarregaDat)leitor).getPadroesOriginais();

                  }
                  else if (formatoArquivo.equals("TXT")){
                        leitor = new
CarregaTxt(umNomeDeArquivo,umSeparadorDeColuna,aColunaDaClasse,aColunaIni
cial,aColunaFinal,aLinhaInicial,aLinhaFinal);
                        ((CarregaTxt)leitor).setArquivo(umNomeDeArquivo);

      ((CarregaTxt)leitor).setSeparadorColuna(umSeparadorDeColuna);

      ((CarregaTxt)leitor).setColunaClasse(aColunaDaClasse);

      ((CarregaTxt)leitor).setColunaInicial(aColunaInicial);
                        ((CarregaTxt)leitor).setColunaFinal(aColunaFinal);

      ((CarregaTxt)leitor).setLinhaInicial(aLinhaInicial);
                        ((CarregaTxt)leitor).setLinhaFinal(aLinhaFinal);

      this.conjuntoValidacao=((CarregaTxt)leitor).getPadroesOriginais();
                  }
                  else if (formatoArquivo.equals("XLS")){
                        leitor = new
CarregaXls(umNomeDeArquivo,aColunaDaClasse,aColunaInicial,aColunaFinal,aL
inhaInicial,aLinhaFinal);

      this.conjuntoValidacao=((CarregaXls)leitor).getPadroesOriginais();

                  }
                  else if (formatoArquivo.equals("XML")){
                        leitor = new CarregaXml(umNomeDeArquivo,this);

      this.conjuntoValidacao=((CarregaXml)leitor).getPadroesOriginais();

                  }
                  else
                        throw new JfanFormatoArquivoException();

                  criaConjuntoValidacaoRede();
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public String[][] getConjuntoValidacao() throws Exception {
            return this.conjuntoValidacao;
      }

      public ArrayList<IPadrao> getConjuntoValidacaoRede() throws
Exception {
            return this.conjuntoValidacaoRede;
      }
```

```
      private ArrayList<IPadrao> criaConjuntoValidacaoRede() throws
Exception {
            this.conjuntoValidacaoRede.clear();
            Vector<Double> umPadrao = new Vector<Double>();
            umPadrao.clear();
            this.colunasTextoValidacao =
getColunasTexto(this.conjuntoValidacao);
            for (int l=0;l<this.conjuntoValidacao.length;l++){
                  for (int c=0;c<this.conjuntoValidacao[l].length;c++){
                        try{
                              if (this.colunasTextoValidacao[c]){
                                    if (c ==
this.conjuntoValidacao[l].length-1) {
                                          if
(!this.arrayMapaClasses.contains(this.conjuntoValidacao[l][c])){

      this.arrayMapaClasses.add(this.conjuntoValidacao[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoValidacao[l][c])));
                                          }
                                          else

      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoValidacao[l][c])));
                                    }
                                    else if
(!this.arrayMapa.contains(this.conjuntoValidacao[l][c])){

      this.arrayMapa.add(this.conjuntoValidacao[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoVal
idacao[l][c])));
                                    }
                                    else

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoVal
idacao[l][c])));
                              }
                              else{
                                    double valorCelula =
Double.parseDouble(this.conjuntoValidacao[l][c]);
                                    umPadrao.add(valorCelula);
                              }
                        }
                        catch (Exception e) {
                              e.printStackTrace();
                              continue;
                        }
                  }
                  double[] asCaracteristicas = new
double[umPadrao.size()-1];
                  for (int j=0;j<umPadrao.size()-1;j++)
                        asCaracteristicas[j]=umPadrao.elementAt(j);
                  int valorClasse=umPadrao.lastElement().intValue();
                  this.conjuntoValidacaoRede.add(new
Padrao(asCaracteristicas,valorClasse));
```

```
                        umPadrao.removeAllElements();
                }
                this.gerarMapeamentos();

                this.conjuntoValidacaoRede.trimToSize();

        RecalculadorClasses.indexarClasses(getClassesConjunto(conjuntoValid
acao, colunasTextoValidacao), conjuntoValidacaoRede);

                return this.conjuntoValidacaoRede;
        }

        public boolean getTextualConjuntoValidacaoRede(int umIndice){
                return this.colunasTextoValidacao[umIndice];
        }

        public void adicionarConjuntoClassificacao(String umNomeDeArquivo,
String umSeparadorDeColuna, int aColunaDaClasse, int aColunaInicial, int
aColunaFinal, int aLinhaInicial, int aLinhaFinal) throws Exception {
                try{
                        String formatoArquivo =
umNomeDeArquivo.substring(umNomeDeArquivo.length()-
3,umNomeDeArquivo.length()).toUpperCase();
                        if (formatoArquivo.equals("DAT")){
                                leitor = new
CarregaDat(umNomeDeArquivo,umSeparadorDeColuna);
                                ((CarregaDat)leitor).setArquivo(umNomeDeArquivo);

        ((CarregaDat)leitor).setSeparadorColuna(umSeparadorDeColuna);

        this.conjuntoClassificacao=((CarregaDat)leitor).getPadroesOriginais
();
                        }
                        else if (formatoArquivo.equals("TXT")){
                                leitor = new
CarregaTxt(umNomeDeArquivo,umSeparadorDeColuna,aColunaDaClasse,aColunaIni
cial,aColunaFinal,aLinhaInicial,aLinhaFinal);
                                ((CarregaTxt)leitor).setArquivo(umNomeDeArquivo);

        ((CarregaTxt)leitor).setSeparadorColuna(umSeparadorDeColuna);

        ((CarregaTxt)leitor).setColunaClasse(aColunaDaClasse);

        ((CarregaTxt)leitor).setColunaInicial(aColunaInicial);
                                ((CarregaTxt)leitor).setColunaFinal(aColunaFinal);

        ((CarregaTxt)leitor).setLinhaInicial(aLinhaInicial);
                                ((CarregaTxt)leitor).setLinhaFinal(aLinhaFinal);

        this.conjuntoClassificacao=((CarregaTxt)leitor).getPadroesOriginais
();
                        }
                        else if (formatoArquivo.equals("XLS")){
                                leitor = new
CarregaXls(umNomeDeArquivo,aColunaDaClasse,aColunaInicial,aColunaFinal,aL
inhaInicial,aLinhaFinal);
```

```
      this.conjuntoClassificacao=((CarregaXls)leitor).getPadroesOriginais
();
                  }
                  else if (formatoArquivo.equals("XML")){
                        leitor = new CarregaXml(umNomeDeArquivo,this);

      this.conjuntoClassificacao=((CarregaXml)leitor).getPadroesOriginais
();
                  }
                  else
                        throw new JfanFormatoArquivoException();

                  criaConjuntoClassificacaoRede();
            }catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public String[][] getConjuntoClassificacao() throws Exception {
            return this.conjuntoClassificacao;
      }

      public ArrayList<IPadrao> getConjuntoClassificacaoRede() throws
Exception {
            return this.conjuntoClassificacaoRede;
      }

      private ArrayList<IPadrao> criaConjuntoClassificacaoRede() throws
Exception {
            this.conjuntoClassificacaoRede.clear();
            for (int l=0;l<this.conjuntoClassificacao.length;l++)

      this.conjuntoClassificacao[l][this.conjuntoClassificacao[l].length-
1]="Desconhecida";

            Vector<Double> umPadrao = new Vector<Double>();
            umPadrao.clear();
            this.colunasTextoClassificacao =
getColunasTexto(this.conjuntoClassificacao);
            for (int l=0;l<this.conjuntoClassificacao.length;l++){
                  for (int
c=0;c<this.conjuntoClassificacao[l].length;c++){
                        try{
                              if (this.colunasTextoClassificacao[c]){
                                    if (c ==
this.conjuntoClassificacao[l].length-1) {
                                          if
(!this.arrayMapaClasses.contains(this.conjuntoClassificacao[l][c])){

      this.arrayMapaClasses.add(this.conjuntoClassificacao[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoClassificacao[l][c])));
                                          }
                                          else
```

```
      umPadrao.add(Double.valueOf(this.arrayMapaClasses.indexOf(this.conj
untoClassificacao[l][c])));
                                    }
                                    else if
(!this.arrayMapa.contains(this.conjuntoClassificacao[l][c])){

      this.arrayMapa.add(this.conjuntoClassificacao[l][c]);

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoCla
ssificacao[l][c])));
                                    }
                                    else

      umPadrao.add(Double.valueOf(this.arrayMapa.indexOf(this.conjuntoCla
ssificacao[l][c])));
                              }
                              else{
                                    double valorCelula =
Double.parseDouble(this.conjuntoClassificacao[l][c]);
                                    umPadrao.add(valorCelula);
                              }
                        }
                        catch (Exception e) {
                              e.printStackTrace();
                              continue;
                        }
                  }

                  double[] asCaracteristicas = new
double[umPadrao.size()-1];
                  for (int j=0;j<umPadrao.size()-1;j++)
                        asCaracteristicas[j]=umPadrao.elementAt(j);


                  this.conjuntoClassificacaoRede.add(new
Padrao(asCaracteristicas,-1));
                  umPadrao.removeAllElements();
            }
            this.gerarMapeamentos();

            this.conjuntoClassificacaoRede.trimToSize();

            return this.conjuntoClassificacaoRede;
      }

      public boolean getTextualConjuntoClassificacaoRede(int umIndice){
            return this.colunasTextoClassificacao[umIndice];
      }

      public void limparConjuntoTreinamento() {
            conjuntoTreinamento = null;
            conjuntoTreinamentoRede.clear();
      }

      public void limparConjuntoTeste() {
            conjuntoTeste = null;
```

```
            conjuntoTesteRede.clear();
      }

      public void limparConjuntoValidacao() {
            conjuntoValidacao = null;
            conjuntoValidacaoRede.clear();
      }

      public void limparConjuntoClassificacao() {
            conjuntoClassificacao = null;
            conjuntoClassificacaoRede.clear();
      }

      public HashMap<Integer, String> getMapaClasses() {
            gerarMapeamentos();
            return mapaClasses;
      }

      public HashMap<String,Integer> getMapaClassesReaisMapeadas() {
            return mapaClassesInverso;
      }

      public HashMap<Integer, String> getMapaConjuntos() {
            return mapaConjunto;
      }

      public ArrayList<String> getArrayAjudaConjuntos() {
            return arrayMapa;
      }

      public ArrayList<String> getArrayAjudaClasses() {
            return arrayMapaClasses;
      }

      public int adicionarClasseParaMapeamento(String classe) {
            if (!arrayMapaClasses.contains(classe)) {
                  arrayMapaClasses.add(classe);
            }
            gerarMapeamentos();
            return mapaClassesInverso.get(classe);
      }

      public void adicionarCaracteristicaParaMapeamento(String
caracteristica) {
            if (!arrayMapa.contains(caracteristica)) {
                  arrayMapa.add(caracteristica);
            }
      }

      public void gerarMapeamentos(){
            if (mapaConjunto == null || mapaConjuntoInverso == null) {
                  mapaConjunto = new HashMap<Integer,String>();
                  mapaConjuntoInverso = new HashMap<String,Integer>();
            }
            for(int i=0;i<this.arrayMapa.size();i++) {
                  if (!mapaConjunto.containsKey(i)) {
                        mapaConjunto.put(i,this.arrayMapa.get(i));
```

```
                        mapaConjuntoInverso.put(this.arrayMapa.get(i),i);
                  }
            }
            if (mapaClasses == null || mapaClassesInverso == null) {
                  mapaClasses = new HashMap<Integer,String>();
                  mapaClassesInverso = new HashMap<String,Integer>();
            }

            for(int i=0;i<this.arrayMapaClasses.size();i++) {
                  if (!mapaClasses.containsKey(i)) {
                        mapaClasses.put(i,this.arrayMapaClasses.get(i));

      mapaClassesInverso.put(this.arrayMapaClasses.get(i),i);
                  }
            }

      }

      public ArrayList<IPadrao> throws Exception {
            if (aPorcentagem>0 && aPorcentagem<101){
                  ArrayList<ArrayList<IPadrao>> listaComListaDePadroes =
new ArrayList<ArrayList<IPadrao>>();
                  int[] arrayClasses = getClassesIndexadasTreinamento();
                  for (int i=0;i<arrayClasses.length;i++){
                        ArrayList<IPadrao> listaTemporaria = new
ArrayList<IPadrao>();
                        for (IPadrao p: conjuntoTreinamentoRede)
                              if (arrayClasses[i]==p.getClasse())
                                    listaTemporaria.add(p);
                        listaComListaDePadroes.add(listaTemporaria);
                  }

                  listaComListaDePadroes.trimToSize();
                  ArrayList<IPadrao> listaFinal = new
ArrayList<IPadrao>();
                  for (ArrayList<IPadrao> lista: listaComListaDePadroes){
                        lista.trimToSize();
                        if (lista.size()>0){
                              int qtdAmostra = (int) Math.ceil(((double)
aPorcentagem / 100.0 * (double) lista.size()));
                              if (qtdAmostra==0)
                                    qtdAmostra=1;
                              for (int k = 0; k<qtdAmostra;k++)
                                    listaFinal.add(lista.get(k));
                        }
                        else
                              System.out.println("erro");
                  }
                  conjuntoTreinamentoRede.clear();
                  conjuntoTreinamentoRede.addAll(listaFinal);
                  conjuntoTesteRede.clear();
                  conjuntoTesteRede.addAll(listaFinal);
            }
            else
                  throw new Exception("Valor invalido para porcentagem,
deve variar de 1 ate 100%");
```

```
            return this.conjuntoTreinamentoRede;
      }

      public int[] getClassesIndexadasTreinamento(){
            ArrayList<Integer> listaClasses = new ArrayList<Integer>();
            for (IPadrao p:   this.conjuntoTreinamentoRede){
                  if (!listaClasses.contains(p.getClasse()))
                        listaClasses.add(p.getClasse());
            }
            int[] arrayClasses = new int[listaClasses.size()];
            for (int i=0;i<listaClasses.size();i++ )
                  arrayClasses[i]=listaClasses.get(i);
            Arrays.sort(arrayClasses);
            return arrayClasses;
      }

      public void setConjuntoTreinamentoComPorcentagemDe(int
aPorcentagem) throws Exception {
            if (aPorcentagem>0 && aPorcentagem<101){
                  Vector<Vector<IPadrao>> listaComListaDePadroes = new
Vector<Vector<IPadrao>>();
                  int[] arrayClasses = getClassesIndexadasTreinamento();
                  for (int i=0;i<arrayClasses.length;i++){
                        Vector<IPadrao> listaTemporaria = new
Vector<IPadrao>();
                        for (IPadrao p: conjuntoTreinamentoRede)
                              if (arrayClasses[i]==p.getClasse())
                                    listaTemporaria.add(p);
                        listaComListaDePadroes.add(listaTemporaria);
                  }

                  listaComListaDePadroes.trimToSize();
                  Vector<IPadrao> listaComPorcTreino = new
Vector<IPadrao>();
                  Vector<IPadrao> listaComPorcTeste = new
Vector<IPadrao>();
                  for (Vector<IPadrao> lista: listaComListaDePadroes){
                        lista.trimToSize();
                        if (lista.size()>0){
                              int qtdAmostra = (int) Math.ceil(((double)
aPorcentagem / 100.0 * (double) lista.size()));
                              if (qtdAmostra==0)
                                    qtdAmostra=1;
                              for (int k = 0; k<lista.size();k++){
                                    if (k<qtdAmostra)

      listaComPorcTreino.add(lista.get(k));
                                    else if (qtdAmostra==1){

      listaComPorcTreino.add(lista.get(k));

      listaComPorcTeste.add(lista.get(k));
                                    }
                                    else

      listaComPorcTeste.add(lista.get(k));
                              }
```

```
                        }
                  }
                  conjuntoTreinamentoRede.clear();
                  conjuntoTreinamentoRede.addAll(listaComPorcTreino);
                  conjuntoTreinamentoRede.trimToSize();
                  conjuntoTesteRede.clear();
                  conjuntoTesteRede.addAll(listaComPorcTeste);
                  conjuntoTesteRede.trimToSize();

            }
            else
                  throw new Exception("Valor invalido para porcentagem,
deve variar de 1 ate 100%");
      }

      public static void main(String[] args) {
            try {
                  GerenciadorPadroes ga = new GerenciadorPadroes();

      ga.adicionarConjuntoTreinamento("d:\\arquivos\\patterns\\arquivo.da
t","  ",0,0,0,0,0);
                  ga.getMapaClasses();
                  ga.setConjuntoTreinamentoComPorcentagemDe(10);

            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }


      public void setConjuntoTeste(String[][] conjuntoTeste) {
            this.conjuntoTeste = conjuntoTeste;
            try {
                  criaConjuntoTesteRede();
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

      public void limparMapeamentos() {
            arrayMapa.clear();
            arrayMapaClasses.clear();
            mapaClasses.clear();
            mapaClassesInverso.clear();
            mapaConjunto.clear();
            mapaConjuntoInverso.clear();
      }

      public void atualizarConjuntoClassificacao() {
            int fim = conjuntoClassificacao[0].length - 1;
            int fim2 = conjuntoClassificacao.length;
            for (int i = 0; i < fim2; i++) {
                  IPadrao p = conjuntoClassificacaoRede.get(i);
                  if (p.getClasse() > -1) {
                        int cl1 =
RecalculadorClasses.getNeuroniosMapIndexReal().get(p.getClasse());
```

```
                        String classe = mapaClasses.get(cl1);
                        if (classe == null) {
                              classe = Integer.toString(cl1);
                        }
                        conjuntoClassificacao[i][fim] = classe;
                  }
                  else {
                        conjuntoClassificacao[i][fim] = "Desconhecida";
                  }
            }
      }

      public int[] getClassesConjuntoClassificacao() {
            return getClassesConjunto(conjuntoClassificacao,
colunasTextoClassificacao);
      }

      public int[] getClassesConjuntoValidacao() {
            return getClassesConjunto(conjuntoValidacao,
colunasTextoValidacao);
      }

      public String[] getClassesConjuntoTesteReais() {
            return getClassesReais(conjuntoTeste);
      }

}
```

2.12.8 PersistorDados.java

```
package jfan.io.arquivos;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.ArrayList;
import jfan.fan.NeuronioFAN;
import jfan.fan.normalizadores.NormalizatorTypes;
import jfan.fan.padroes.IPadrao;
import jfan.fan.utilitarias.RecalculadorClasses;
import jfan.io.IPersistorDados;

public class PersistorDados implements IPersistorDados {

      public void gravaNeuroniosFanDat(ArrayList<NeuronioFAN>
listaComNeuroniosFAN, String caminhoAbsolutoNomeArquivo) throws Exception
{
            String oSeparadorDeColunas = " ";
            FileWriter fw = new FileWriter(caminhoAbsolutoNomeArquivo);
            PrintWriter pw = new PrintWriter(fw);
            String umaLinha = " ";
            for (NeuronioFAN nf: listaComNeuroniosFAN){
                  double[][] matrizNeural = nf.getMatrizNeural();
                  int raioDisuso = nf.getRaioDifuso();
                  for (int i = raioDisuso;i<(matrizNeural[0].length-
raioDisuso-1); i++){
                        umaLinha = "";
```

```
                        for (int j = 0;j<matrizNeural.length; j++)

      umaLinha+=String.valueOf(matrizNeural[j][i])+oSeparadorDeColunas;

                        pw.println(umaLinha);
                  }
                  int qtdCarateristicas = nf.quantasCaracteristicas();
                  umaLinha = "";
                  for (int i = 0;i<qtdCarateristicas; i++)

      umaLinha+=nf.getSomatorio(i+1)+oSeparadorDeColunas;
                  pw.println(umaLinha);
            }
            pw.close();
      }

      public void gravaPadroesDat(ArrayList<IPadrao> listaComPadroes,
String caminhoAbsolutoNomeArquivo) throws Exception {
            String oSeparadorDeColunas = "  ";
            FileWriter fw = new FileWriter(caminhoAbsolutoNomeArquivo);
            PrintWriter pw = new PrintWriter(fw);
            for (IPadrao p: listaComPadroes){
                  String umPadrao = "  ";
                  for (int i = 0;i<p.getCaracteristicas().length; i++)

      umPadrao+=String.valueOf(p.getCaracteristicas()[i])+oSeparadorDeCol
unas;

      umPadrao+=String.valueOf(p.getClasse())+oSeparadorDeColunas;
                  pw.println(umPadrao);
            }
            pw.close();
      }

      public void gravaNeuroniosFanXml(ArrayList<NeuronioFAN>
listaComNeuroniosFAN, String caminhoAbsolutoNomeArquivo, String
nomeProjeto, NormalizatorTypes tipoNormalizacao, double[] minimos,
double[] maximos, double[] medias,GerenciadorPadroes gp) throws Exception
{
            FileWriter fw = new FileWriter(caminhoAbsolutoNomeArquivo);
            PrintWriter pw = new PrintWriter(fw);
            int raioDifuso = listaComNeuroniosFAN.get(0).getRaioDifuso();
            int suporteConjunto =
listaComNeuroniosFAN.get(0).getSuporteConjuntoDifuso();
            int numCaracteristicas =
listaComNeuroniosFAN.get(0).quantasCaracteristicas();
            pw.println("<?xml version=\"1.0\"?>");
            pw.println("<net>");
            pw.println("<netinformation>");
            pw.println("<projectname>"+nomeProjeto+"</projectname>");

      pw.println("<quantityfeatures>"+numCaracteristicas+"</quantityfeatu
res>");
            pw.println("<difuseradius>"+raioDifuso+"</difuseradius>");

      pw.println("<setdifusesupport>"+suporteConjunto+"</setdifusesupport
>");
```

```
            pw.println("</netinformation>");

            pw.println("<normalization
type=\""+tipoNormalizacao.toString()+"\">");
            pw.println("<mins>");
            for (int i = 0;i<minimos.length; i++)
                  pw.println("<featuremin index=\""+(i+1)+"\"
value=\""+minimos[i]+"\"/>");
            pw.println("</mins>");

            pw.println("<maxs>");
            for (int i = 0;i<maximos.length; i++)
                  pw.println("<featuremax index=\""+(i+1)+"\"
value=\""+maximos[i]+"\"/>");
            pw.println("</maxs>");

            pw.println("<means>");
            for (int i = 0;i<medias.length; i++)
                  pw.println("<featuremean index=\""+(i+1)+"\"
value=\""+medias[i]+"\"/>");
            pw.println("</means>");

            pw.println("</normalization>");

            pw.println("<neurons>");
            for (NeuronioFAN nf: listaComNeuroniosFAN){
                  int cl1 =
RecalculadorClasses.getNeuroniosMapIndexReal().get(nf.getClasseAssociada(
));
                  String classe = gp.getMapaClasses().get(cl1);
                  if (classe == null) {
                        classe = Integer.toString(cl1);
                  }

                  pw.println("<neurone
weight=\""+nf.getPesoPenalizacao()+"\" class=\""+classe+"\">");
                  double[][] matrizNeural = nf.getMatrizNeural();
                  for (int i = 0;i<matrizNeural.length; i++){
                        pw.println("<feature index=\""+(i+1)+"\"
sum=\""+nf.getSomatorio(i+1)+"\">");
                        for (int j = 0;j<matrizNeural[0].length; j++)

      pw.println("<value>"+String.valueOf(matrizNeural[i][j])+"</value>")
;
                        pw.println("</feature>");
                  }
                  pw.println("</neurone>");
            }
            pw.println("</neurons>");
            pw.println("</net>");
            pw.close();
      }

      public void gravaPadroesXml(ArrayList<IPadrao> listaComPadroes,
String caminhoAbsolutoNomeArquivo, String nomeProjeto, String
tipoConjunto, String[] nomeCaracteristicas) throws Exception {
            FileWriter fw = new FileWriter(caminhoAbsolutoNomeArquivo);
```

```
            PrintWriter pw = new PrintWriter(fw);
            ArrayList<String> asClasses = new ArrayList<String>();
            asClasses.clear();
            for (IPadrao p: listaComPadroes){
                  int classe = p.getClasse();
                  if (!asClasses.contains(String.valueOf(classe)))
                        asClasses.add(String.valueOf(classe));
            }
            int numCaracteristicas =
listaComPadroes.get(0).getQuantasCaracteristicas();
            int numClasses = asClasses.size();
            int numPadroes = listaComPadroes.size();
            pw.println("<?xml version=\"1.0\"?>");
            pw.println("<set>");
            pw.println("<fileinformation>");
            pw.println("<projectname>"+nomeProjeto+"</projectname>");

      pw.println("<quantitypatterns>"+numPadroes+"</quantitypatterns>");
            pw.println("<typeset>"+tipoConjunto+"</typeset>");
            pw.println("<quantityclass>"+numClasses+"</quantityclass>");

      pw.println("<quantityfeatures>"+numCaracteristicas+"</quantityfeatu
res>");
            pw.println("<featuresname>");
            for(int i = 0;i<nomeCaracteristicas.length;i++ )
                  pw.println("<name>"+nomeCaracteristicas[i]+"</name>");
            pw.println("</featuresname>");
            pw.println("<classes>");
            for (String valorClasses: asClasses)
                  pw.println("<value>"+valorClasses+"</value>");
            pw.println("</classes>");
            pw.println("</fileinformation>");
            pw.println("<patterns>");
            for (IPadrao p: listaComPadroes){
                  pw.println("<pattern>");
                  for (int i = 0;i<p.getCaracteristicas().length; i++)

      pw.println("<feature>"+String.valueOf(p.getCaracteristicas()[i])+"<
/feature>");

      pw.println("<memberclass>"+String.valueOf(p.getClasse())+"</memberc
lass>");
                  pw.println("</pattern>");
            }
            pw.println("</patterns>");
            pw.println("</set>");
            pw.close();
      }
      public void gravaPadroesXml(GerenciadorPadroes gp,
ArrayList<IPadrao> listaComPadroes, String caminhoAbsolutoNomeArquivo,
String nomeProjeto, String tipoConjunto, String[] nomeCaracteristicas)
throws Exception {
            FileWriter fw = new FileWriter(caminhoAbsolutoNomeArquivo);
            PrintWriter pw = new PrintWriter(fw);
            int[] classes = gp.getClassesIndexadasTreinamento();
            int numCaracteristicas =
listaComPadroes.get(0).getQuantasCaracteristicas();
```

```
            int numPadroes = listaComPadroes.size();
            pw.println("<?xml version=\"1.0\"?>");
            pw.println("<set>");
            pw.println("<fileinformation>");
            pw.println("<projectname>"+nomeProjeto+"</projectname>");

      pw.println("<quantitypatterns>"+numPadroes+"</quantitypatterns>");
            pw.println("<typeset>"+tipoConjunto+"</typeset>");

      pw.println("<quantityclass>"+classes.length+"</quantityclass>");

      pw.println("<quantityfeatures>"+numCaracteristicas+"</quantityfeatu
res>");
            pw.println("<featuresname>");
            for(int i = 0;i<nomeCaracteristicas.length;i++ )
                  pw.println("<name>"+nomeCaracteristicas[i]+"</name>");
            pw.println("</featuresname>");
            pw.println("<classes>");
            for (int i =0; i<classes.length; i++){
                  int cl1 =
RecalculadorClasses.getClassesMapIndexReal().get(classes[i]);
                  String cl = gp.getMapaClasses().get(cl1);
                  if (cl == null)
                        cl = Integer.toString(cl1);
                  pw.println("<value>"+cl+"</value>");
            }

            pw.println("</classes>");
            pw.println("</fileinformation>");
            pw.println("<patterns>");
            for (IPadrao p: listaComPadroes){
                  pw.println("<pattern>");
                  for (int i = 0;i<p.getCaracteristicas().length; i++)

      pw.println("<feature>"+String.valueOf(p.getCaracteristicas()[i])+"<
/feature>");
                  int cl1 =
RecalculadorClasses.getClassesMapIndexReal().get(p.getClasse());
                  String cl = gp.getMapaClasses().get(cl1);
                  if (cl == null)
                        cl = Integer.toString(cl1);
                  pw.println("<memberclass>"+cl+"</memberclass>");
                  pw.println("</pattern>");
            }
            pw.println("</patterns>");
            pw.println("</set>");
            pw.close();
      }


      public static void main(String[] args) {
            try {
                  CarregaDat cd = new
CarregaDat("C:\\patterns\\arquivo.dat","  ");
                  PersistorDados p = new PersistorDados();
                  String[] nomes = {"Centromero","Altura","Raio"};
```

```
                    p.gravaPadroesXml(cd.getPadroes(), "C:\\teste.xml",
"ProjetoCromo", "Treinamento", nomes);

            } catch (Exception e) {
            }
            System.out.println("d");

      }
}
```

3. CADERNO FONTE – THINGEASYFAN

3.1 PACKAGE THING;

3.1.1 ThinletDTD.Java

```
package thing;


import java.util.Arrays;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeSet;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ThinletDTD
{
    private static final Logger log = Logger.getLogger("thing");
      private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    private static final Object[] dtd;
    static
    {
        Integer integer_1 = new Integer(-1);
        Integer integer0 = new Integer(0);
        Integer integer1 = new Integer(1);
        String[] orientation = { "horizontal", "vertical" };
        String[] leftcenterright = { "left", "center", "right" };
        String[] selections = { "single", "interval", "multiple" };
        dtd = new Object[] {
            "component", null, new Object[][] {
                { "string", "name", null, null },
                { "boolean", "enabled", "paint", Boolean.TRUE },
                { "boolean", "visible", "parent", Boolean.TRUE },
                { "boolean", "i18n", "validate", Boolean.FALSE },
                { "string", "tooltip", null, null },
                { "font", "font", "validate", null },
                { "color", "foreground", "paint", null },
                { "color", "background", "paint", null },
                { "integer", "width", "validate", integer0 },
                { "integer", "height", "validate", integer0 },
                { "integer", "colspan", "validate", integer1 },
                { "integer", "rowspan", "validate", integer1 },
                { "integer", "weightx", "validate", integer0 },
                { "integer", "weighty", "validate", integer0 },
                { "choice", "halign", "validate",
                    new String[] { "fill", "center", "left", "right" } },
                { "choice", "valign", "validate",
                new String[] { "fill", "center", "top", "bottom" } },
                { "property", "property", null, null },
                { "method", "init" },
                { "method", "focuslost" },
                { "method", "focusgained" } },
            "label", "component", new Object[][] {
```

```
                { "string", "text", "validate", null },
                { "icon", "icon", "validate", null },
                { "choice", "alignment", "validate", leftcenterright },
                { "integer", "mnemonic", "paint", integer_1 },
                { "component", "for", null, null } },
            "button", "label", new Object[][] {
                { "choice", "alignment", "validate", new String[] {
"center", "left", "right" } },
                { "method", "action" },
                { "choice", "type", "paint", new String[] { "normal",
"default", "cancel", "link" } } },
            "checkbox", "label", new Object[][] {
                { "boolean", "selected", "paint", Boolean.FALSE },
                { "string", "group", "paint", null },
                { "method", "action" } },
            "togglebutton", "checkbox", null,
            "combobox", "textfield", new Object[][] {
                { "icon", "icon", "validate", null },
                { "integer", "selected", "layout", integer_1 } },
            "choice", null, new Object[][] {
                { "string", "name", null, null },
                { "boolean", "enabled", "paint", Boolean.TRUE },
                { "boolean", "i18n", "validate", Boolean.FALSE },
                { "string", "text", "parent", null },
                { "icon", "icon", "parent", null },
                { "choice", "alignment", "parent", leftcenterright },
                { "string", "tooltip", null, null },
                { "font", "font", "validate", null },
                { "color", "foreground", "paint", null },
                { "color", "background", "paint", null },
                { "property", "property", null, null } },
            "textfield", "component", new Object[][] {
                { "string", "text", "layout", "" },
                { "integer", "columns", "validate", integer0 },
                { "boolean", "editable", "paint", Boolean.TRUE },
                { "integer", "start", "layout", integer0 },
                { "integer", "end", "layout", integer0 },
                { "method", "action" },
                { "method", "insert" },
                { "method", "remove" },
                { "method", "caret" },
                { "method", "perform" } },
            "passwordfield", "textfield", null,
            "textarea", "textfield", new Object[][] {
                { "integer", "rows", "validate", integer0 },
                { "boolean", "border", "validate", Boolean.TRUE },
                { "boolean", "wrap", "layout", Boolean.FALSE } },
            "tabbedpane", "component", new Object[][] {
                { "choice", "placement", "validate",
                    new String[] { "top", "left", "bottom", "right",
"stacked" } },
                { "integer", "selected", "paint", integer0 },
                { "method", "action" } }, //...focus
            "tab", "choice", new Object[][] {
                { "integer", "mnemonic", "paint", integer_1 } },
            "panel", "component", new Object[][] {
                { "integer", "columns", "validate", integer0 },
```

```
                { "integer", "top", "validate", integer0 },
                { "integer", "left", "validate", integer0 },
                { "integer", "bottom", "validate", integer0 },
                { "integer", "right", "validate", integer0 },
                { "integer", "gap", "validate", integer0 },
                { "string", "text", "validate", null },
                { "icon", "icon", "validate", null },
                { "boolean", "border", "validate", Boolean.FALSE },
                { "boolean", "scrollable", "validate", Boolean.FALSE } },
            "desktop", "component", null,
            "dialog", "panel", new Object[][] {
                { "boolean", "modal", null, Boolean.FALSE },
                { "boolean", "resizable", null, Boolean.FALSE },
                { "method", "close" },
                { "boolean", "maximizable", "paint", Boolean.FALSE },
                { "boolean", "iconifiable", "paint", Boolean.FALSE },
                { "boolean", "closable", null, Boolean.FALSE } },
            "spinbox", "textfield", new Object[][] {
                { "integer", "minimum", null, new
Integer(Integer.MIN_VALUE) },
                { "integer", "maximum", null, new
Integer(Integer.MAX_VALUE) },
                { "integer", "step", null, integer1 },
                { "integer", "value", null, integer0 } },
            "progressbar", "component", new Object[][] {
                { "choice", "orientation", "validate", orientation },
                { "integer", "minimum", "paint", integer0 },
                { "integer", "maximum", "paint", new Integer(100) },
                { "integer", "value", "paint", integer0 } },
            "slider", "progressbar", new Object[][] {
                { "integer", "unit", null, new Integer(5) },
                { "integer", "block", null, new Integer(25) },
                { "method", "action" } },
            "splitpane", "component", new Object[][] {
                { "choice", "orientation", "validate", orientation },
                { "integer", "divider", "layout", integer_1 } },
            "list", "component", new Object[][] {
                { "choice", "selection", "paint", selections },
                { "method", "action" },
                { "method", "perform" },
                { "boolean", "line", "validate", Boolean.TRUE } },
            "item", "choice", new Object[][] {
                { "boolean", "selected", null, Boolean.FALSE } },
            "table", "list", new Object[][] {
                    },
            "header", null, new Object[][] {
                { "method", "action" },
                { "boolean", "resizable", null, Boolean.TRUE } },
            "column", "choice", new Object[][] {
                { "integer", "width", null, new Integer(80) },
                { "choice", "sort", null, new String[] { "none",
"ascent", "descent" } },
                { "boolean", "selected", null, Boolean.FALSE } },
            "row", null, new Object[][] {
                { "boolean", "selected", null, Boolean.FALSE } },
            "cell", "choice", null,
            "tree", "list", new Object[][] {
```

```
                { "boolean", "angle", null, Boolean.FALSE },
                { "method", "expand" },
                { "method", "collapse" } },
            "node", "choice", new Object[][] {
                { "boolean", "selected", null, Boolean.FALSE },
                { "boolean", "expanded", null, Boolean.TRUE } },
            "separator", "component", null,
            "menubar", "component", new Object[][] {
                { "choice", "placement", "validate", new String[] {
"top", "bottom" } } },
            "menu", "choice", new Object[][] {
                { "integer", "mnemonic", "paint", integer_1 } },
            "menuitem", "choice", new Object[][] {
                { "keystroke", "accelerator", null, null },
                { "method", "action" },
                { "integer", "mnemonic", "paint", integer_1 } },
            "checkboxmenuitem", "menuitem", new Object[][] {
                { "boolean", "selected", "paint", Boolean.FALSE },
                { "string", "group", "paint", null } },
"popupmenu", "component", new Object[][] {
                { "method", "menushown" } },
            "bean", "component", new Object[][] {
                { "bean", "bean", null, null } }
        };
    }
    private static final String[] components =
    {
        "bean",  "button",  "checkbox",  "combobox",  "desktop",
"dialog",
        "label",  "list",  "menubar",  "panel",  "passwordfield",
"popupmenu",
        "progressbar",  "separator",  "slider",  "spinbox",  "splitpane",
        "tabbedpane",  "table",  "textarea",  "textfield",
"togglebutton",
        "tree"
    };

    private static final Object[] allowedSubWidgets =
    {
        "bean",             new String[] { "popupmenu" },
        "button",           new String[] { "popupmenu" },
        "cell",             new String[] { },
        "checkbox",         new String[] { "popupmenu" },
        "checkboxmenuitem", new String[] { },
        "choice",           new String[] { },
        "column",           new String[] { },
        "combobox",         new String[] { "choice", "popupmenu" },
        "desktop",          components,
        "dialog",           components,
        "header",           new String[] { "column" },
        "item",             new String[] { },
        "label",            new String[] { "popupmenu" },
        "list",             new String[] { "item", "popupmenu" },
        "menu",             new String[] { "menu", "menuitem",
"checkboxmenuitem", "separator" },
        "menubar",          new String[] { "menu" },
        "menuitem",         new String[] { },
```

```
        "node",             new String[] { "node" },
        "panel",            components,
        "passwordfield",    new String[] { "popupmenu" },
        "popupmenu",        new String[] { "menu", "menuitem",
"checkboxmenuitem", "separator" },
        "progressbar",      new String[] { "popupmenu" },
        "row",              new String[] { "cell" },
        "separator",        new String[] { "popupmenu" },
        "slider",           new String[] { "popupmenu" },
        "spinbox",          new String[] { "popupmenu" },
        "splitpane",        components,
        "tab",              components,
        "tabbedpane",       new String[] { "tab", "popupmenu" },
        "table",            new String[] { "row", "header", "popupmenu"
},
        "textarea",         new String[] { "popupmenu" },
        "textfield",        new String[] { "popupmenu" },
        "togglebutton",     new String[] { "popupmenu" },
        "tree",             new String[] { "node", "popupmenu" }
    };


    private static final HashMap<String,Widget> widgets;

    static
    {
        widgets = new HashMap<String,Widget>();

        for(int i = 0; i < dtd.length; i += 3)
        {
            String classname = (String) dtd[i];
            Widget widget = new Widget(classname);
            widgets.put(classname, widget);
            Object[][] propDefs = (Object[][]) dtd[i + 2];
            Property[] properties = new Property[0];
            if(propDefs != null)
            {
                TreeSet<Property> props = new TreeSet<Property>();
                for(int j = 0; j < propDefs.length; ++j)
                {
                    String type = (String) propDefs[j][0];
                    String name = (String) propDefs[j][1];
                    Object defaultValue = (propDefs[j].length > 3) ?
propDefs[j][3] : null;
                    props.add(new Property(widget, name, type,
defaultValue));
                }
                properties = (Property[]) props.toArray(new
Property[props.size()]);
            }
            widget.setProperties(properties);
        }

        for(int i = 0; i < dtd.length; i += 3)
        {
            String classname = (String) dtd[i];
            String parentclassname = (String) dtd[i + 1];
```

```
            Widget widget = (Widget) widgets.get(classname);
            Widget parentWidget = (Widget) widgets.get(parentclassname);
            widget.setParent(parentWidget);
        }

        for(int i = 0; i < allowedSubWidgets.length; i += 2)
        {
            String classname = (String) allowedSubWidgets[i];
            String[] allowed = (String[]) allowedSubWidgets[i + 1];
            if(allowed.length > 0)
            {
                Widget widget = (Widget) widgets.get(classname);
                for(int j = 0; j < allowed.length; ++j)
                {
                    Widget allowedWidget = (Widget)
widgets.get(allowed[j]);
                    widget.addAllowedSubWidget(allowedWidget);
                }
            }
        }
    }

    public static Widget getWidget(String classname)
    {
        Widget widget = (Widget) widgets.get(classname);
        if(widget == null)
            throw new IllegalArgumentException("unknown classname: " +
classname);
        else
            return widget;
    }

    public static Property[] getProperties(String classname)
    {
        return getWidget(classname).getProperties();
    }

    public static Property getProperty(String classname, String propName)
    {
        Property property = getWidget(classname).getProperty(propName);
        if(property != null)
            return property;
        else
            throw new IllegalArgumentException("unknown property: " +
propName + " in class: " + classname);
    }

    public static class Widget implements Comparable
    {
        private String classname;
        private Widget parent;
        private Property[] properties;
        private ArrayList<Widget> allowedSubWidgets = new
ArrayList<Widget>();
        private boolean allPropertiesCalculated = false;
        private HashMap<String,Property> propMap = new
HashMap<String,Property>();
```

```
        Widget(String classname)
        {
            this.classname = classname;
        }

        void setProperties(Property[] properties)
        {
            this.properties = properties;
        }

        void setParent(Widget parent)
        {
            this.parent = parent;
        }

        void addAllowedSubWidget(Widget w)
        {
            this.allowedSubWidgets.add(w);
        }

        public String getClassname()
        {
            return classname;
        }

        public Widget getParent()
        {
            return parent;
        }

        public String getParentClassname()
        {
            return getParent() != null ? getParent().getClassname() :
null;
        }

        public Property getProperty(String propName)
        {
            calculateAllProperties();
            return (Property) this.propMap.get(propName);
        }

        public boolean hasProperty(String propName)
        {
            return getProperty(propName) != null;
        }

        public Property[] getProperties()
        {
            calculateAllProperties();
            return properties;
        }

        public Widget[] getAllowedSubWidgets()
        {
```

```
            return (Widget[]) allowedSubWidgets.toArray(new
Widget[allowedSubWidgets.size()]);
        }

        public boolean isSubWidgetAllowed(Widget subWidget)
        {
            return isSubWidgetAllowed(subWidget.getClassname());
        }

        public boolean isSubWidgetAllowed(String classname)
        {
            for(Iterator it = allowedSubWidgets.iterator(); it.hasNext();
)
                if(classname.equals(((Widget)it.next()).getClassname()))
                    return true;
            return false;
        }

        public boolean isInstanceOf(String classname)
        {
            if(classname.equals(getClassname()))
                return true;
            if(getParent() == null)
                return false;
            return getParent().isInstanceOf(classname);
        }

        public boolean equals(Object obj)
        {
            if(obj instanceof Widget)
                return
getClassname().equals(((Widget)obj).getClassname());
            else
                return false;
        }

        public int hashCode()
        {
            return getClassname().hashCode() + 7;
        }

        public int compareTo(Object obj)
        {
            Widget other = (Widget) obj;
            return this.getClassname().compareTo(other.getClassname());
        }

        public String toString()
        {
            return getClassname();
        }

        private void calculateAllProperties()
        {
            if(!this.allPropertiesCalculated)
            {
                TreeSet<Property> set = new TreeSet<Property>();
```

```
                Widget widget = this;
                while(widget != null)
                {
                    set.addAll(Arrays.asList(widget.properties));
                    widget = widget.getParent();
                }
                this.properties = (Property[]) set.toArray(new
Property[set.size()]);

                for(int i=0; i < this.properties.length; ++i)
                    this.propMap.put(this.properties[i].getName(),
this.properties[i]);

                this.allPropertiesCalculated = true;
            }
        }
    }

    public static class Property implements Comparable
    {
        public static final int STRING = 0;
        public static final int INTEGER = 1;
        public static final int BOOLEAN = 2;
        public static final int CHOICE = 3;
        public static final int COLOR = 4;
        public static final int ICON = 5;
        public static final int FONT = 6;
        public static final int KEYSTROKE = 7;
        public static final int METHOD = 8;
        public static final int BEAN = 9;
        public static final int PROPERTY = 10;
        public static final int COMPONENT = 11;
        private static final String[] allTypes = { "string", "integer",
            "boolean", "choice", "color", "icon", "font", "keystroke",
            "method", "bean", "property", "component" };

        private Widget widget;
        private String name;
        private int type;
        private Object defaultValue;

        Property(Widget widget, String name, String type, Object
defaultValue)
        {
            this.widget = widget;
            this.name = name;
            this.defaultValue = defaultValue;
            this.type = -1;

            for(int i = 0; i < allTypes.length; ++i)
            {
                if(allTypes[i].equals(type))
                {
                    this.type = i;
                    break;
                }
            }
```

```
            if(this.type == -1)
                throw new IllegalArgumentException("unknown property
type: " + type);
        }

        public Widget getWidget()
        {
            return widget;
        }

        public int getType()
        {
            return type;
        }

        public String getName()
        {
            return name;
        }

        public Object getDefaultValue()
        {
            return (type == CHOICE) ? getChoices()[0] : defaultValue;
        }

        public String[] getChoices()
        {
            return (type == CHOICE) ? (String[]) defaultValue : null;
        }

        public String getTypeName()
        {
            return getTypeName(getType());
        }

        public static String getTypeName(int type)
        {
            return allTypes[type];
        }

        public boolean equals(Object obj)
        {
            if(obj instanceof Property)
            {
                Property prop = (Property)obj;
                return this.getWidget().equals(prop.getWidget())
                    && this.getName().equals(prop.getName());
            }
            else
                return false;
        }

        public int hashCode()
        {
            return getWidget().hashCode() * 3 + getName().hashCode() + 7;
        }
```

```
        public int compareTo(Object obj)
        {
            Property other = (Property) obj;
            return this.getName().compareTo(other.getName());
        }

        public String toString()
        {
            return getName();
        }
    }
}
```

3.2 PACKAGE THINLETCOMMONS;

3.2.1 AntiAliasedThinlet.Java

```
package thinletcommons;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;

public class AntiAliasedThinlet extends LoggingThinlet
{

    /**
       * Número serial gerado pelo Eclipse.
       */
      private static final long serialVersionUID = 26032213611274001l8L;


      public AntiAliasedThinlet()
    {
        super();
    }


    public void paint(Graphics g)
    {
        ((Graphics2D) g).setRenderingHint(
            RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
        ((Graphics2D) g).setRenderingHint(
            RenderingHints.KEY_TEXT_ANTIALIASING,
            RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
        super.paint(g);
    }

}
```

3.2.2 ColorChoose.Java

```
package thinletcommons;
```

```
import java.awt.Color;
import java.awt.Dialog;
import java.awt.Frame;
import java.util.logging.Level;
import java.util.logging.Logger;

import thinlet.Thinlet;

public class ColorChooser
{
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    private ThinletDialog dialog;
    private Thinlet thinlet;
    private Color selectedColor;

    private Object sl_red, sl_green, sl_blue;
    private Object sb_red, sb_green, sb_blue;
    private Object tf_hue, tf_saturation, tf_brightness;
    private Object pb_hue, pb_saturation, pb_brightness;
    private Object rgb_label;


    public ColorChooser(Frame owner, String title)
    {
        this.dialog = new ThinletDialog(owner, title);
        init();
    }


    public ColorChooser(Dialog owner, String title)
    {
        this.dialog = new ThinletDialog(owner, title);
        init();
    }


    private void init()
    {
        thinlet = new AntiAliasedThinlet();

        try
        {
            Object panel =
thinlet.parse("/thinletcommons/colorchooser.xml", this);
            thinlet.add(panel);
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Error parsing colorchooser.xml", e);
        }

        dialog.setContent(thinlet);
```

```
}

public void show()
{
    if(debug()) log.fine("in show");
    dialog.pack();
    dialog.setResizable(false);
    dialog.setLocationRelativeTo(dialog.getOwner());
    dialog.setVisible(true);
}

public void setSelectedColor(Color color)
{
    int red = 0, green = 0, blue = 0;
    if(color != null)
    {
        red = color.getRed();
        green = color.getGreen();
        blue = color.getBlue();
        thinlet.setInteger(sl_red, "value", red);
        thinlet.setInteger(sl_green, "value", green);
        thinlet.setInteger(sl_blue, "value", blue);
        thinlet.setString(sb_red, "text", String.valueOf(red));
        thinlet.setString(sb_green, "text", String.valueOf(green));
        thinlet.setString(sb_blue, "text", String.valueOf(blue));
        hsbChanged();
    }
}

public Color getSelectedColor()
{
    return selectedColor;
}

public void init(Object sl_red, Object sl_green, Object sl_blue,
    Object sb_red, Object sb_green, Object sb_blue,
    Object tf_hue, Object tf_saturation, Object tf_brightness,
    Object pb_hue, Object pb_saturation, Object pb_brightness,
    Object rgb_label)
{
    if(debug()) log.fine("in init");
    this.sl_red = sl_red;
    this.sl_green = sl_green;
    this.sl_blue = sl_blue;
    this.sb_red = sb_red;
    this.sb_green = sb_green;
    this.sb_blue = sb_blue;
    this.tf_hue = tf_hue;
    this.tf_saturation = tf_saturation;
    this.tf_brightness = tf_brightness;
    this.pb_hue = pb_hue;
    this.pb_saturation = pb_saturation;
    this.pb_brightness = pb_brightness;
    this.rgb_label = rgb_label;
}

public void sliderChanged(int value, Object spinbox)
```

```
    {
        thinlet.setString(spinbox, "text", String.valueOf(value));
        hsbChanged();
    }

    public void spinboxChanged(String text, Object slider)
    {
        try
        {
            int value = Integer.parseInt(text);
            if(value >= 0 && value <= 255)
            {
                thinlet.setInteger(slider, "value", value);
                hsbChanged();
            }
        }
        catch (NumberFormatException ignore)
        {
        }
    }

    public void ok()
    {
        int red = thinlet.getInteger(sl_red, "value");
        int green = thinlet.getInteger(sl_green, "value");
        int blue = thinlet.getInteger(sl_blue, "value");
        this.selectedColor = new Color(red, green, blue);
        dialog.setVisible(false);
    }

    public void close()
    {
        dialog.setVisible(false);
    }

    private void hsbChanged()
    {
        int red = thinlet.getInteger(sl_red, "value");
        int green = thinlet.getInteger(sl_green, "value");
        int blue = thinlet.getInteger(sl_blue, "value");

        float[] hsb = Color.RGBtoHSB(red, green, blue, null);

        thinlet.setColor(rgb_label, "background", new Color(red, green,
blue));
        thinlet.setString(tf_hue, "text", String.valueOf(hsb[0]));
        thinlet.setString(tf_saturation, "text", String.valueOf(hsb[1]));
        thinlet.setString(tf_brightness, "text", String.valueOf(hsb[2]));

        thinlet.setInteger(pb_hue, "value", (int) (100f * hsb[0]));
        thinlet.setInteger(pb_saturation, "value", (int) (100f *
hsb[1]));
        thinlet.setInteger(pb_brightness, "value", (int) (100f *
hsb[2]));
    }

}
```

3.2.3 DirChoose.Java

```
package thinletcommons;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.text.Collator;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Stack;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;
import javax.swing.filechooser.FileSystemView;
import thinlet.Thinlet;

public class DirChooser
{
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }


    private ThinletDialog dialog;
    private Thinlet thinlet;
    private File selectedDir;
    private File expandDir;
    private boolean isInitialized = false;
    private FileSystemView fsv = FileSystemView.getFileSystemView();

    public DirChooser(Frame owner, String title)
    {
        this.dialog = new ThinletDialog(owner, title);
        init();
    }

    public DirChooser(Dialog owner, String title)
    {
        this.dialog = new ThinletDialog(owner, title);
        init();
    }

    private void init()
    {
        thinlet = new AntiAliasedThinlet();

        try
        {
            Object panel =
thinlet.parse("/thinletcommons/dirchooser.xml", this);
            thinlet.add(panel);
```

```
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Error parsing dirchooser.xml", e);
        }

        dialog.setContent(thinlet);
    }

    public void show()
    {
        selectedDir = null;
        if(expandDir != null)
        {
            expandTreeTo(expandDir);
            expandDir = null;
        }
        dialog.pack();
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);
    }

    public void setSelectedDirectory(File dir)
    {
        this.expandDir = dir;
    }

    public File getSelectedDirectory()
    {
        return selectedDir;
    }

    public void init(Object tree)
    {
        if(debug()) log.fine("in init()");
        if(!isInitialized)
        {
            addFiles(fsv.getRoots(), tree);
            isInitialized = true;
        }
    }

    public void ok(Object tree)
    {
        Object node = thinlet.getSelectedItem(tree);
        if(node != null)
            selectedDir = (File) thinlet.getProperty(node, "file");
        close();
    }

    public void close()
    {
        dialog.setVisible(false);
    }

    public void nodeExpanded(Object tree, Object node)
    {
```

```
        File file = (File) thinlet.getProperty(node, "file");
        thinlet.removeAll(node);
        File[] files = fsv.getFiles(file, false);
        addFiles(files, node);
    }

    private void addFiles(File[] files, Object parent)
    {
        sort(files);

        for(int i = 0; i < files.length; ++i)
        {
            File file = files[i];
            if(fsv.isTraversable(file).booleanValue())
            {
                Object node = createNode(file);
                thinlet.setBoolean(node, "expanded", false);
                thinlet.add(parent, node);

                Object placeholder = Thinlet.create("node");
                thinlet.setBoolean(placeholder, "expanded", false);
                thinlet.setString(placeholder, "text", "#Placeholder#");
                thinlet.add(node, placeholder);
            }
        }
    }

    private Object createNode(File file)
    {
        Object node = Thinlet.create("node");
        thinlet.setString(node, "text", fsv.getSystemDisplayName(file));

        Icon icon = fsv.getSystemIcon(file);
        if(icon != null)
            thinlet.setIcon(node, "icon", convertIconToImage(thinlet,
icon));

        thinlet.putProperty(node, "file", file);
        return node;
    }

    private void sort(File[] files)
    {
        if(files.length == 0)
            return;

        File firstFile = files[0];
        if(fsv.isComputerNode(firstFile) || fsv.isDrive(firstFile)
            || fsv.isFileSystemRoot(firstFile) ||
fsv.isFloppyDrive(firstFile)
            || fsv.isRoot(firstFile))
            return;

        Arrays.sort(files, new Comparator<File>()
        {
            private  Collator collator = Collator.getInstance();
```

```
            public int compare(File obj1, File obj2)
            {
                return collator.compare(
                    fsv.getSystemDisplayName(obj1),
                    fsv.getSystemDisplayName(obj2)
                );
            }

            public boolean equals(Object other)
            {
                return false;
            }
        });
    }


    private void expandTreeTo(File file)
    {
        Stack<File> stack = new Stack<File>();
        stack.push(file);
        File root = null;
        File parent = file;

        do
        {
            parent = fsv.getParentDirectory(parent);
            if(parent != null)
            {
                stack.push(parent);
                root = parent;
            }
        } while(parent != null);

        if(root == null)
        {
            log.warning("cannot expand to directory '" + file + "'
because parent not found!");
            return;
        }

        Object tree = thinlet.find("tree");
        _expandTreeTo(tree, stack);
    }


    private void _expandTreeTo(Object node, Stack<File> stack)
    {
        File currentDir = stack.pop();
        Object[] nodes = thinlet.getItems(node);
        boolean found = false;

        for(int i = 0; i < nodes.length; ++i)
        {
            Object subnode = nodes[i];

            File file = (File) thinlet.getProperty(subnode, "file");
            if(!found && file.equals(currentDir))
```

```
            {
                thinlet.removeAll(subnode);
                File[] subfiles = fsv.getFiles(file, false);
                addFiles(subfiles, subnode);
                if(stack.empty())
                    thinlet.setBoolean(subnode, "selected", true);
                else
                {
                    thinlet.setBoolean(subnode, "expanded", true);
                    _expandTreeTo(subnode, stack);
                    found = true;
                }
            }
        }
    }


    private static Image convertIconToImage(Thinlet thinlet, Icon icon)
    {
        BufferedImage image = new BufferedImage(icon.getIconWidth(),
            icon.getIconHeight(), BufferedImage.TYPE_INT_ARGB);
        icon.paintIcon(thinlet, image.getGraphics(), 0, 0);
        return image;
    }
}
```

3.2.4 ExtensionFileFilter.Java

```
package thinletcommons;

import java.io.File;

public class ExtensionFileFilter implements FileFilter
{
    private String extension;
    private String description;

    public ExtensionFileFilter(String extension, String description)
    {
        this.extension = "." + extension;
        this.description = description;
    }

    public boolean accept(File file)
    {
        return file.getName().endsWith(extension);
    }

    public String getDescription()
    {
        return description;
    }

}
```

3.2.5 FileChooser.Java

```
package thinletcommons;

import java.awt.Dialog;
import java.awt.Frame;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.text.Collator;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.Icon;
import javax.swing.filechooser.FileSystemView;
import thinlet.Thinlet;

public class FileChooser extends AntiAliasedThinlet
{
    private static final Logger log = Logger.getLogger("thinletcommons");
      private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    public static final int MODE_OPEN = 0;
    public static final int MODE_SAVE = 1;

    private static final FileSystemView fsv =
FileSystemView.getFileSystemView();
    private static final DecimalFormat decimalFormat = new
DecimalFormat("#,##0.0");
    private static final DecimalFormat integerFormat = new
DecimalFormat("###,###,###,###,###,###,###,###,###,###");
    private static final DateFormat dateFormat =
DateFormat.getInstance();

    private ThinletDialog dialog;
    private Thinlet thinlet;
    private File selectedFile;
    private File showFile;
    private File currentDir;
    private int mode;
    private boolean showHiddenFiles = false;
    private int sortColumn = 0;
    private boolean sortAscending = true;
    private FileFilter[] filters;
    private FileFilter selectedFilter;

    private int indexSelectedFilter;
```

```
      private Object cbPath, bUp, bShowHidden, tbFilelist, tFilename,
bOk, cbFilter;

    public FileChooser(Frame owner, String title, int mode)
    {
        this.dialog = new ThinletDialog(owner, title);
        this.mode = mode;
        init();
    }

    public FileChooser(Dialog owner, String title, int mode)
    {
        this.dialog = new ThinletDialog(owner, title);
        this.mode = mode;
        init();
    }

   private void init()
    {
        thinlet = new AntiAliasedThinlet();
        try
        {  Object panel = null;
            panel = thinlet.parse("/thinletcommons/filechooser.xml",
this);
            thinlet.add(panel);
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Error parsing filechooser.xml", e);
        }

        tbFilelist = thinlet.find("tbFilelist");
        cbPath = thinlet.find("cbPath");
        bUp = thinlet.find("bUp");
        bShowHidden = thinlet.find("bShowHidden");
        bOk = thinlet.find("bOk");
        tFilename = thinlet.find("tFilename");
        cbFilter = thinlet.find("cbFilter");
        dialog.setContent(thinlet);
    }

    public void show()
    {
        selectedFile = null;
        setupFileFilters();
        if(showFile == null || !showFile.exists())
            showFile = fsv.getHomeDirectory();
        addFiles(showFile);
        showFile = null;
        thinlet.setString(thinlet.find("bOk"), "text", mode == MODE_OPEN
? "Open" : "Save");
        thinlet.setBoolean(thinlet.find("bShowHidden"), "selected",
showHiddenFiles);
        thinlet.requestFocus(thinlet.find("tbFilelist"));
        dialog.pack();
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);
```

```
    }

    public void setSelectedFile(File file)
    {
        this.showFile = file;
        if (file != null)
            currentDir = file.getParentFile();
    }

    public File getSelectedFile()
    {
        return selectedFile;
    }

    public void setShowHiddenFiles(boolean showHiddenFiles)
    {
        this.showHiddenFiles = showHiddenFiles;
    }

    public boolean getShowHiddenFiles()
    {
        return showHiddenFiles;
    }

    public void setFileFilters(FileFilter[] filters)
    {
        this.filters = filters;
    }

    public FileFilter getSelectedFileFilter()
    {
        return selectedFilter;
    }

    public int getIndexOfSelectedFileFilter() {
      return indexSelectedFilter;
    }

    public void ok()
    {
        String filename = thinlet.getString(tFilename, "text");
        if(filename == null || filename.trim().length() == 0)
            new MessageDialog(dialog, "Nada selecionado", "Por favor
selecione um arquivo.").show();
        else
        {
            File file = new File(currentDir, filename);
            if(fsv.isTraversable(file).booleanValue())
                addFiles(file);
            else
            {
                selectedFile = file;
                close();
            }
        }

    }
```

```
    public void close()
    {
        dialog.setVisible(false);

    }

    public void goUp()
    {
        if(currentDir != null)
        {
            File parent = fsv.getParentDirectory(currentDir);
            if(parent != null)
                addFiles(parent);
        }
    }

    public void goHome()
    {
        File home = new File(System.getProperty("user.home"));
        addFiles(home);
    }

    public void toggleShowHiddenFiles(boolean showHidden)
    {
        this.showHiddenFiles = showHidden;
        addFiles(currentDir);
    }

    public void tableRowDoubleClicked()
    {   File file = getSelectedTableRowFile();
        if(file != null && fsv.isTraversable(file).booleanValue())
            addFiles(file);
        else
        {
            selectedFile = file;
            close();
        }
    }

    public void tableRowSelected()
    {
        thinlet.setString(tFilename, "text",
getSelectedTableRowFilename());
    }

    public void tableHeaderChanged(Object colName, Object colType, Object
colSize, Object colMod)
    {
        final Object[] column = { colName, colType, colSize, colMod };
        for(int i = 0; i < column.length; ++i)
        {
            String sort = thinlet.getChoice(column[i], "sort");
            if(!sort.equals("none"))
            {
                this.sortColumn = i;
                this.sortAscending = sort.equals("ascent");
```

```
                break;
            }
        }
        addFiles(currentDir);
    }

    public void cbPathChanged()
    {
        Object item = thinlet.getSelectedItem(cbPath);
        if(item != null)
        {
            File dir = (File) thinlet.getProperty(item, "file");
            addFiles(dir);
        }
    }

    public void cbFilterChanged(int index)
    {
        indexSelectedFilter = index;
        if(filters != null)
        {
            selectedFilter = filters[index];
            if (currentDir != null) {
                  addFiles(currentDir);
            }
            else {
                  addFiles(fsv.getHomeDirectory());
            }
       }
    }

    public String getFilterChanged(){
      String filterChanged = this.getString(cbFilter,"text");
      return  filterChanged;
    }

    public String getFileName(){
      String fileName = this.getString(tFilename,"text");
      return fileName;
    }

    public void addFiles(File file)
    {
        currentDir = file;
        if(fsv.isTraversable(file).booleanValue())
            thinlet.setString(tFilename, "text", "");
        else
        {
            thinlet.setString(tFilename, "text",
fsv.getSystemDisplayName(file));
            currentDir = fsv.getParentDirectory(file);
        }

        File[] files = fsv.getFiles(currentDir, false);
        sort(files, sortColumn, sortAscending);
        thinlet.removeAll(tbFilelist);
        for(int i = 0; i < files.length; ++i) {
```

```
            if(acceptFile(files[i]))
                thinlet.add(tbFilelist, createTableRow(files[i]));
        }
        thinlet.removeAll(cbPath);
        File dir = currentDir;
        while(dir != null)
        {
            Object item = Thinlet.create("choice");
            thinlet.setString(item, "text",
fsv.getSystemDisplayName(dir));
            thinlet.putProperty(item, "file", dir);
            setFileIcon(item, dir);
            thinlet.add(cbPath, item, 0);
            dir = fsv.getParentDirectory(dir);
        }
        thinlet.setInteger(cbPath, "selected", thinlet.getCount(cbPath) -
1);
        thinlet.setBoolean(bUp, "enabled",
fsv.getParentDirectory(currentDir) != null);

    }

    private boolean acceptFile(File file)
    {
        if(!showHiddenFiles && fsv.isHiddenFile(file)) return false;
        if(fsv.isTraversable(file).booleanValue()) return true;
        if(selectedFilter == null) return true;
        return selectedFilter.accept(file);
    }

    private void setupFileFilters()
    {
        thinlet.removeAll(cbFilter);
        if(filters == null)
        {
            addFileFilter("All files (*.*)");
            selectedFilter = null;
        }
        else
        {
            for(int i = 0; i < filters.length; ++i)
                addFileFilter(filters[i].getDescription());
            if (indexSelectedFilter > 0) {
                  selectedFilter = filters[indexSelectedFilter];
            }
            else {
                  selectedFilter = filters[0];
            }
        }
        if (indexSelectedFilter > 0) {
            thinlet.setInteger(cbFilter, "selected",
indexSelectedFilter);
        }
        else {
            thinlet.setInteger(cbFilter, "selected", 0);
        }
```

```
    }

    private void addFileFilter(String description)
    {
        Object choice = Thinlet.create("choice");
        thinlet.setString(choice, "text", description);
        thinlet.add(cbFilter, choice);
    }

    private Object createTableRow(File file)
    {
        Object row = Thinlet.create("row");
        Object cellName = Thinlet.create("cell");
        Object cellType = Thinlet.create("cell");
        Object cellSize = Thinlet.create("cell");
        Object cellMod = Thinlet.create("cell");
        thinlet.add(row, cellName);
        thinlet.add(row, cellType);
        thinlet.add(row, cellSize);
        thinlet.add(row, cellMod);
        thinlet.setString(cellName, "text",
fsv.getSystemDisplayName(file));
        thinlet.setString(cellType, "text", formatTypeString(file));
        thinlet.setString(cellMod, "text",
formatDateString(file.lastModified()));
        thinlet.setString(cellSize, "text",
fsv.isTraversable(file).booleanValue() ? "" :
formatByteString(file.length()));
        thinlet.setChoice(cellSize, "alignment", "right");
        setFileIcon(cellName, file);
        thinlet.putProperty(row, "file", file);
        return row;
    }

    private static void sort(File[] files, int sortColumn, boolean
sortAscending)
    {
        if(files.length == 0)
            return;

        File firstFile = files[0];
        if(fsv.isComputerNode(firstFile) || fsv.isDrive(firstFile)
            || fsv.isFileSystemRoot(firstFile) ||
fsv.isFloppyDrive(firstFile)
            || fsv.isRoot(firstFile))
            return;

        Arrays.sort(files, new FileComparator(sortColumn,
sortAscending));
    }

    private File getSelectedTableRowFile()
    {
        Object row = thinlet.getSelectedItem(tbFilelist);
        if(row != null)
            return (File) thinlet.getProperty(row, "file");
        else
```

```
            return null;
    }

    private String getSelectedTableRowFilename()
    {
        File file = getSelectedTableRowFile();
        if(file != null)
            return fsv.getSystemDisplayName(file);
        else
            return "";
    }

    private void setFileIcon(Object component, File file)
    {
        Icon icon = fsv.getSystemIcon(file);
        if(icon != null)
            thinlet.setIcon(component, "icon",
convertIconToImage(thinlet, icon));
    }

    private static String formatTypeString(File file)
    {
        String desc = fsv.getSystemTypeDescription(file);
        if(desc == null || desc.length() == 0)
        {
            if(fsv.isComputerNode(file))
                desc = "Computer";
            else if(fsv.isDrive(file))
                desc = "Disk";
            else if(fsv.isFloppyDrive(file))
                desc = "Floppy";
            else if(fsv.isTraversable(file).booleanValue())
                desc = "Directory";
            else
                desc = "Unknown";
        }
        return desc;
    }

    private static String formatByteString(long size)
    {
        double d = size;
        int magnitude = 0;
        for(; d >= 1024.0; ++magnitude)
            d = d / 1024.0;
        switch(magnitude)
        {
            case 1: return decimalFormat.format(d) + " KB";
            case 2: return decimalFormat.format(d) + " MB";
            case 3: return decimalFormat.format(d) + " GB";
            case 4: return decimalFormat.format(d) + " TB";
            default: return integerFormat.format(size);
        }
    }


    private static String formatDateString(long date)
```

```
    {
        return dateFormat.format(new Date(date));
    }


    private static Image convertIconToImage(Thinlet thinlet, Icon icon)
    {
        BufferedImage image = new BufferedImage(icon.getIconWidth(),
            icon.getIconHeight(), BufferedImage.TYPE_INT_ARGB);
        icon.paintIcon(thinlet, image.getGraphics(), 0, 0);
        return image;
    }

    private static class FileComparator implements Comparator<File>
    {
        private Collator collator = Collator.getInstance();
        private int sortColumn;
        private boolean sortAscending;

        public FileComparator(int sortColumn, boolean sortAscending)
        {
            this.sortColumn = sortColumn;
            this.sortAscending = sortAscending;
        }

        public int compare(File obj1, File obj2)
        {
            File f1 = (File) obj1;
            File f2 = (File) obj2;
            boolean f1Traversable = fsv.isTraversable(f1).booleanValue();
            boolean f2Traversable = fsv.isTraversable(f2).booleanValue();

            if(f1Traversable && !f2Traversable)
                return sortAscending ? Integer.MIN_VALUE :
Integer.MAX_VALUE;
            else if(!f1Traversable && f2Traversable)
                return sortAscending ? Integer.MAX_VALUE :
Integer.MIN_VALUE;
            else
            {
                int retval = 0;
                switch(sortColumn)
                {
                    case 0:                          retval =
collator.compare(
                            fsv.getSystemDisplayName(f1),
                            fsv.getSystemDisplayName(f2));
                        break;
                    case 1:                          retval =
collator.compare(
                            fsv.getSystemTypeDescription(f1),
                            fsv.getSystemTypeDescription(f2));
                        break;
                    case 2:                          retval = f1.length() >
f2.length() ? 1
                            : f1.length() < f2.length() ? -1 : 0;
                        break;
```

```
                    case 3:                         retval =
f1.lastModified() > f2.lastModified() ? 1
                            : f1.lastModified() < f2.lastModified() ? -1
: 0;
                        break;
                    default:
                        throw new IllegalArgumentException("unexpected
column: " + sortColumn);
                }

                if(retval == 0 && sortColumn != 0)
                    retval = collator.compare(
                        fsv.getSystemDisplayName(f1),
                        fsv.getSystemDisplayName(f2));

                return sortAscending ? retval : (-1 * retval);
            }
        }

        public boolean equals(Object other)
        {
            return false;
        }
    }

    public Dialog getDialog() {
      return dialog;
    }

    public void setSelectedFiler(int index) {
      thinlet.setInteger(cbFilter, "selected", index);
      cbFilterChanged(index);
    }



    public static void main(String[] args)
    {
        FileChooser chooser = new FileChooser(new Frame(), "Choose
File...", FileChooser.MODE_OPEN);
        if(args.length > 0)
            chooser.setSelectedFile(new File(args[0]));
        chooser.setFileFilters(new FileFilter[]
        {
            new ExtensionFileFilter("xml", "XML files (*.xml)"),
            new ExtensionFileFilter("java", "Java source code (*.java)"),
            new FileFilter()
            {
                public boolean accept(File file) { return true; }
                public String getDescription() { return "All files
(*.*)"; }

            }
        });
        chooser.show();
```

```
        System.out.println("File selected: " +
chooser.getSelectedFile());
        System.exit(0);
    }

}
```

3.2.6 FileFilter.Java

```
package thinletcommons;

import java.io.File;

public interface FileFilter
{
    public boolean accept(File file);

    public String getDescription();
}
```

3.2.7 FontChooser.Java

```
package thinletcommons;

import java.awt.Dialog;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Frame;
import java.awt.GraphicsEnvironment;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
import thinlet.Thinlet;

public class FontChooser
{
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }


    private ThinletDialog dialog;
    private Thinlet thinlet;
    private Font selectedFont;


    public FontChooser(Frame owner, String title)
    {
        this.dialog = new ThinletDialog(owner, title);
        init();
    }

    public FontChooser(Dialog owner, String title)
    {
```

```
        this.dialog = new ThinletDialog(owner, title);
        init();
    }

    private void init()
    {
        thinlet = new AntiAliasedThinlet();

        try
        {
            Object panel =
thinlet.parse("/thinletcommons/fontchooser.xml", this);
            thinlet.add(panel);
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Error parsing fontchooser.xml", e);
        }

        dialog.setContent(thinlet);
    }

    public void show()
    {
        dialog.setSize(new Dimension(320, 360));
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);
    }

    public void setSelectedFont(Font font)
    {
        String family = null;
        int size = 12;
        boolean isBold = false;
        boolean isItalic = false;

        if(font != null)
        {
            family = font.getFamily();
            size = font.getSize();
            isBold = font.isBold();
            isItalic = font.isItalic();
            thinlet.setFont(thinlet.find("preview"), "font", font);
        }

        thinlet.setString(thinlet.find("fontsize"), "text",
String.valueOf(size));
        thinlet.setBoolean(thinlet.find("cb_bold"), "selected", isBold);
        thinlet.setBoolean(thinlet.find("cb_italic"), "selected",
isItalic);

        Object fontlist = thinlet.find("fontlist");
        Object[] items = thinlet.getItems(fontlist);
        for(int i = 0; i < items.length; ++i)
        {
            String text = thinlet.getString(items[i], "text");
```

```
            thinlet.setBoolean(items[i], "selected",
text.equals(family));
        }

        thinlet.setBoolean(thinlet.find("b_ok"), "enabled", font !=
null);
    }

    public Font getSelectedFont()
    {
        return selectedFont;
    }

    public void init(Object fontlist)
    {
        if(debug()) log.fine("in init()");
        String[] names =
GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyN
ames();
        Arrays.sort(names);
        for(int i = 0; i < names.length; ++i)
        {
            Object item = Thinlet.create("item");
            thinlet.add(fontlist, item);
            thinlet.setString(item, "text", names[i]);
        }
    }

    public void ok(Object fontlist, Object fontsize, Object cbBold,
Object cbItalic)
    {
        this.selectedFont = getFont(fontlist, fontsize, cbBold,
cbItalic);
        if(debug()) log.fine("selected: " + this.selectedFont);
        dialog.setVisible(false);
    }

    public void close()
    {
        dialog.setVisible(false);
    }

    public void fontChanged(Object fontlist, Object fontsize, Object
cbBold, Object cbItalic, Object preview, Object bOk)
    {
        Font font = getFont(fontlist, fontsize, cbBold, cbItalic);
        if(font == null)
            thinlet.setBoolean(bOk, "enabled", false);
        else
        {
            thinlet.setBoolean(bOk, "enabled", true);
            thinlet.setFont(preview, "font", font);
        }
    }

    private Font getFont(Object fontlist, Object fontsize, Object cbBold,
Object cbItalic)
```

```
    {
        Object item = thinlet.getSelectedItem(fontlist);
        if(item == null)
            return null;

        String family = thinlet.getString(item, "text");
        int size = 12;
        String fontsizeText = thinlet.getString(fontsize, "text");
        if(fontsizeText.length() > 0)
            size = Integer.parseInt(fontsizeText);
        boolean isBold = thinlet.getBoolean(cbBold, "selected");
        boolean isItalic = thinlet.getBoolean(cbItalic, "selected");

        Font font = new Font(family,
            isBold ? (isItalic ? Font.BOLD|Font.ITALIC : Font.BOLD)
                : (isItalic ? Font.ITALIC : Font.PLAIN), size);

        return font;
    }

}
```

3.2.8 LoggingThinlet.Java

```
package thinletcommons;

import java.util.logging.Level;
import java.util.logging.Logger;
import thinlet.Thinlet;

public class LoggingThinlet extends Thinlet
{
    private static final long serialVersionUID = -4488739744278519797L;
    private static final Logger log = Logger.getLogger("thinletcommons");
      private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    public LoggingThinlet()
    {
        super();
    }

    protected void handleException(Throwable throwable)
    {
        log.log(Level.SEVERE, "Exception in business logic", throwable);
    }
}
```

3.2.9 MessageDialog.Java

```
package thinletcommons;

import java.awt.Dialog;
import java.awt.Dimension;
```

```
import java.awt.EventQueue;
import java.awt.Frame;
import java.awt.Image;
import java.util.logging.Level;
import java.util.logging.Logger;
import thinlet.Thinlet;
import utils.AWTUtils;

public class MessageDialog
{
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    public static final int MODE_OK = 0;
    public static final int MODE_OK_CANCEL = 1;
    public static final int MODE_YES_NO = 2;
    public static final int MODE_YES_NO_CANCEL = 3;
    public static final int ACTION_OK = 0;
    public static final int ACTION_CANCEL = 1;
    public static final int ACTION_YES = 2;
    public static final int ACTION_NO = 3;
    private Thinlet thinlet;
    private ThinletDialog dialog;
    private String msg;
    private Image icon;
    private int mode;
    private int returnValue;

    public MessageDialog(Dialog owner, String title, String msg)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, null, MODE_OK);
    }

    public MessageDialog(Dialog owner, String title, Image icon, String
msg)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, icon, MODE_OK);
    }

    public MessageDialog(Frame owner, String title, String msg)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, null, MODE_OK);
    }

    public MessageDialog(Frame owner, String title, Image icon, String
msg)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, icon, MODE_OK);
    }

    public MessageDialog(Dialog owner, String title, String msg, int
mode)
```

```
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, null, mode);
    }

    public MessageDialog(Dialog owner, String title, Image icon, String
msg, int mode)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, icon, mode);
    }


    public MessageDialog(Frame owner, String title, String msg, int mode)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, null, mode);
    }

    public MessageDialog(Frame owner, String title, Image icon, String
msg, int mode)
    {
        this.dialog = new ThinletDialog(owner, title);
        init(msg, icon, mode);
    }

    private void init(String msg, Image icon, int mode)
    {
        this.thinlet = new AntiAliasedThinlet();
        this.msg = msg;
        this.mode = mode;
        this.icon = icon;

        try
        {
            Object panel =
thinlet.parse("/thinletcommons/messagedialog.xml", this);
            thinlet.add(panel);
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Error parsing messagedialog.xml", e);
        }

        this.dialog.setContent(thinlet);
    }

    public int show()
    {
        try
        {
            Dimension dim = AWTUtils.getBounds(msg, dialog);
            if(debug()) log.fine("msg width=" + dim.getWidth() + "
height=" + dim.getHeight());
            Object message = thinlet.find("message");
            thinlet.setInteger(message, "width", (int)dim.getWidth() +
30);
```

```
            thinlet.setInteger(message, "height", (int)dim.getHeight() +
30);
        }
        catch(Exception e)
        {
            log.log(Level.SEVERE, "Exception getting bounds", e);
        }
        returnValue = ACTION_CANCEL;
        dialog.pack();
        dialog.setLocationRelativeTo(dialog.getOwner());
        dialog.setVisible(true);

        return returnValue;
    }

    public void init(Object iconLabel, Object textarea, Object button1,
Object button2, Object button3)
    {
        thinlet.setString(textarea, "text", this.msg);

        if(this.icon != null)
            thinlet.setIcon(iconLabel, "icon", this.icon);

        final Object defaultButton;

        switch(this.mode)
        {
            case MODE_OK:
                hideButton(button1);
                hideButton(button2);
                showButton(button3, "cancel", "Ok", 0, "ok.gif",
ACTION_OK);
                defaultButton = button3;
                break;
            case MODE_OK_CANCEL:
                hideButton(button1);
                showButton(button2, "default", "Ok", 0, "ok.gif",
ACTION_OK);
                showButton(button3, "cancel", "Cancel", 0, "cancel.gif",
ACTION_CANCEL);
                defaultButton = button3;
                break;
            case MODE_YES_NO:
                hideButton(button1);
                showButton(button2, "normal", "Yes", 0, "ok.gif",
ACTION_YES);
                showButton(button3, "normal", "No", 0, "cancel.gif",
ACTION_NO);
                defaultButton = button3;
                break;
            case MODE_YES_NO_CANCEL:
                showButton(button1, "normal", "Yes", 0, "ok.gif",
ACTION_YES);
                showButton(button2, "normal", "No", 0, "cancel.gif",
ACTION_NO);
                showButton(button3, "cancel", "Cancel", 0, null,
ACTION_CANCEL);
```

```
                defaultButton = button3;
                break;
            default:
                throw new IllegalArgumentException("illegal message
dialog mode: " + this.mode);
        }

        EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
                thinlet.requestFocus(defaultButton);
            }
        });
    }

    public void buttonPressed(Object button)
    {
        returnValue =
Integer.parseInt((String)thinlet.getProperty(button, "action"));
        dialog.setVisible(false);
    }

    private void hideButton(Object button)
    {
        thinlet.setBoolean(button, "visible", false);
        thinlet.setChoice(button, "type", "normal");
    }

    private void showButton(Object button, String type, String text, int
mnemonic, String icon, int action)
    {
        thinlet.setChoice(button, "type", type);
        thinlet.setString(button, "text", text);
        thinlet.setInteger(button, "mnemonic", mnemonic);
        if(icon != null)
            thinlet.setIcon(button, "icon", AWTUtils.getIcon(dialog,
"/thinletcommons/icons/" + icon));
        thinlet.putProperty(button, "action", Integer.toString(action));
    }

}
```

3.2.10 ThinletDialog.Java

```
package thinletcommons;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dialog;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import thinlet.Thinlet;

public class ThinletDialog extends Dialog
{
    private static final long serialVersionUID = -4935536880140349 61L;
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    private transient Thinlet content;
    private transient Image doublebuffer;


    public ThinletDialog(Frame owner)
    {
        super(owner, true);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Frame owner, boolean modal)
    {
        super(owner, modal);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Frame owner, String title)
    {
        super(owner, title, true);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Frame owner, String title, boolean modal)
    {
        super(owner, title, modal);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Dialog owner)
    {
        super(owner);
        setModal(false);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Dialog owner, boolean modal)
    {
        super(owner);
        setModal(modal);
        addWindowListener(new WindowHandler());
    }

    public ThinletDialog(Dialog owner, String title)
    {
        super(owner, title, true);
        addWindowListener(new WindowHandler());
```

```
}

public ThinletDialog(Dialog owner, String title, boolean modal)
{
    super(owner, title, modal);
    addWindowListener(new WindowHandler());
}

public void setContent(Thinlet content)
{
    this.content = content;
    removeAll();
    setLayout(new BorderLayout());
    _addImpl(content, BorderLayout.CENTER);
    pack();
}

public Thinlet getContent()
{
    return content;
}

public void update(Graphics g)
{
    paint(g);
}

public void paint(Graphics g)
{
    if(doublebuffer == null)
    {
        Dimension d = getSize();
        doublebuffer = createImage(d.width, d.height);
    }
    Graphics dg = doublebuffer.getGraphics();
    dg.setClip(g.getClipBounds());
    super.paint(dg);
    dg.dispose();
    g.drawImage(doublebuffer, 0, 0, this);
}

public void doLayout()
{
    if(doublebuffer != null)
    {
        doublebuffer.flush();
        doublebuffer = null;
    }
    super.doLayout();
}

protected void addImpl(Component comp, Object constraints, int index)
{
    setContent((Thinlet)comp);
}

private void _addImpl(Component comp, String constraints)
```

```
    {
        super.addImpl(comp, constraints, -1);
    }

    private class WindowHandler extends WindowAdapter
    {
        public void windowClosing(WindowEvent e)
        {
            setVisible(content != null && !content.destroy());
            dispose();
        }
    }
}
```

3.2.11 ThinletTester.Java

```
package thinletcommons;

import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.InputStream;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import thinlet.FrameLauncher;
import thinlet.Thinlet;

public class ThinletTester
{
    private static String file = null;
    private static String thinletClassname = "thinlet.Thinlet";
    private static String handlerClassname = null;

    public static void main(String[] args) throws Throwable
    {
        getArgs(args);
        run();
    }

    private static void getArgs(String[] args)
    {
        if(args.length < 1 || args.length > 5)
            usage();

        for(int i = 0; i < args.length; ++i)
        {
            if(args[i].equals("--help"))
                usage();
            else if(args[i].equals("--thinlet"))
            {
                if(i == args.length - 1)
                    usage();
                else
                    thinletClassname = args[++i];
            }
            else if(args[i].equals("--handler"))
            {
```

```
                if(i == args.length - 1)
                    usage();
                else
                    handlerClassname = args[++i];
            }
            else if(file == null)
                file = args[i];
            else
                usage();
        }

        if(file == null)
            usage();
    }

    private static void usage()
    {
        System.out.println("ThinletTester, version 1.0");
        System.out.println();
        System.out.println("Usage: java [-cp <classpath>]
thinletcommons.ThinletTester [--thinlet <classname>]");
        System.out.println("         [--handler <classname>]
thinletfile.xml");
        System.out.println();
        System.out.println("  --thinlet  Thinlet classname, e.g.
thinlet.Thinlet");
        System.out.println("  --handler  Handler classname, e.g.
foo.bar.MyHandler");
        System.out.println();
        System.out.println("  The classpath must include both the Thinlet
class and the handler class (if");
        System.out.println("  given on the commandline) as well as
thinlet.jar.");
        System.out.println("  The specified Thinlet class and Handler
class must have a default");
        System.out.println("  constructor.");
        System.out.println();
        System.exit(1);
    }

    private static void run() throws Throwable
    {
        Class thinletClass = Class.forName(thinletClassname);
        Object thinlet = thinletClass.newInstance();

        if(!Thinlet.class.isInstance(thinlet))
        {
            System.err.println("error: " + thinletClassname + " is not a
Thinlet class.");
            System.exit(2);
        }

        System.out.println("using thinlet class: " +
thinletClass.getName());

        Object handler = thinlet;
        if(handlerClassname != null)
```

```
        {
            Class handlerClass = Class.forName(handlerClassname);
            handler = handlerClass.newInstance();
            System.out.println("using handler class: " +
handlerClass.getName());
        }

        InputStream is = getInputStream(file);
        System.out.println("parsing file " + file + " ...");

        Method parseMethod = thinletClass.getMethod("parse", new Class[]
{ InputStream.class, Object.class });
        Object result;
        try
        {
            result = parseMethod.invoke(thinlet, new Object[] { is,
handler });
        }
        catch(InvocationTargetException e)
        {
            throw e.getTargetException();
        }

        Method addMethod = thinletClass.getMethod("add", new Class[] {
Object.class });
        try
        {
            addMethod.invoke(thinlet, new Object[] { result });
        }
        catch(InvocationTargetException e)
        {
            throw e.getTargetException();
        }

        System.out.println("showing frame...");

        FrameLauncher frame = new FrameLauncher("ThinletTester", null,
(Thinlet)thinlet, 400, 400, false);
        frame.pack();
        frame.setVisible(true);

        System.out.println("done.");
    }


    private static InputStream getInputStream(String filename) throws
Exception
    {
        return new BufferedInputStream(new FileInputStream(filename));
    }
}
```

3.2.12 ColorChoose.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!-- jEdit settings:
:tabSize=4:indentSize=4:noTabs=false:folding=explicit:collapseFolds=1: --
>

<panel columns="3" top="10" left="10" right="10" bottom="10" gap="4"
init="init(sl_red, sl_green, sl_blue, sb_red, sb_green, sb_blue, tf_hue,
tf_saturation, tf_brightness, pb_hue, pb_saturation, pb_brightness,
rgb_label)">
      <label text="Red, green, and blue values" colspan="3" />

      <label text="Red:" alignment="right" for="sb_red" mnemonic="0" />
      <slider name="sl_red" minimum="0" maximum="255" valign="center"
action="sliderChanged(this.value, sb_red)" />
      <spinbox name="sb_red" minimum="0" maximum="255" text="0"
columns="3" action="spinboxChanged(this.text, sl_red)" />

      <label text="Green:" alignment="right" for="sb_green" mnemonic="0"
/>
      <slider name="sl_green" minimum="0" maximum="255" valign="center"
action="sliderChanged(this.value, sb_green)" />
      <spinbox name="sb_green" minimum="0" maximum="255" text="0"
columns="3" action="spinboxChanged(this.text, sl_green)" />

      <label text="Blue:" alignment="right" for="sb_blue" mnemonic="0" />
      <slider name="sl_blue" minimum="0" maximum="255" valign="center"
action="sliderChanged(this.value, sb_blue)" />
      <spinbox name="sb_blue" minimum="0" maximum="255" text="0"
columns="3" action="spinboxChanged(this.text, sl_blue)" />

      <separator colspan="3" />

      <label text="Hue, saturation, and brightness values" colspan="3" />

      <label text="Hue:" alignment="right" />
      <textfield name="tf_hue" text="0.0" editable="false" />
      <progressbar name="pb_hue" valign="center" />

      <label text="Saturation:" alignment="right" />
      <textfield name="tf_saturation" text="0.0" editable="false" />
      <progressbar name="pb_saturation" valign="center" />

      <label text="Brightness:" alignment="right" />
      <textfield name="tf_brightness" text="0.0" editable="false" />
      <progressbar name="pb_brightness" valign="center" />

      <separator colspan="3"/>

      <label colspan="3" name="rgb_label" valign="fill" weighty="1"
background="#000000" text="ThinG!" font="40 bold italic"
alignment="center"/>

    <panel colspan="3" gap="10" halign="right" top="12" weightx="1">
        <button action="ok()" icon="/thinletcommons/icons/ok.gif"
mnemonic="0" name="b_ok" text="Ok" type="default"/>
        <button action="close()" icon="/thinletcommons/icons/cancel.gif"
mnemonic="0" name="b_cancel" text="Cancel" type="cancel"/>
    </panel>
```

```
</panel>
```

3.2.13 DirChooser.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- jEdit settings:
:tabSize=4:indentSize=4:noTabs=false:folding=explicit:collapseFolds=1: --
>

<panel
      columns="1"
      gap="6"
      top="6"
      left="6"
      right="6"
      bottom="6"
      width="300"
      height="400"
      init="init(tree)"
>
      <tree
            name="tree"
            angle="true"
            line="false"
            weightx="1"
            weighty="1"
            perform="ok(this)"
            expand="nodeExpanded(this, item)"
      />
      <panel gap="6" top="12" weightx="1">
            <panel weightx="1"/>
            <button name="ok" text="Ok"
icon="/thinletcommons/icons/ok.gif" mnemonic="0" type="default"
action="ok(tree)"/>
            <button name="cancel" text="Cancela"
icon="/thinletcommons/icons/cancel.gif" mnemonic="0" type="cancel"
action="close()"/>
      </panel>
</panel>
```

3.2.14 FileChooser.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- jEdit settings:
:tabSize=4:indentSize=4:noTabs=true:folding=explicit:collapseFolds=1: -->

<panel bottom="12" columns="1" gap="12" left="12" right="12" top="12"
weightx="1" weighty="1">
    <panel gap="6" weightx="1">
        <label for="cbPath" mnemonic="0" text="Visualizando:"/>
        <combobox action="cbPathChanged()" editable="false" name="cbPath"
weightx="1"/>
```

```
        <button action="goUp()" icon="/thinletcommons/icons/up.gif"
name="bUp" tooltip="Ir ao diretório superior."/>
        <button action="goHome()" icon="/thinletcommons/icons/home.gif"
name="bHome" tooltip="Ir para pasta pessoal."/>
        <togglebutton action="toggleShowHiddenFiles(this.selected)"
icon="/thinletcommons/icons/hidden_files.gif" name="bShowHidden"
tooltip="Se pressionado mostra os arquivos ocultos."/>
    </panel>

    <table action="tableRowSelected()" height="300" line="false"
name="tbFilelist" perform="tableRowDoubleClicked()" weightx="1"
weighty="1" width="420">
        <header action="tableHeaderChanged(colName, colType, colSize,
colMod)">
            <column name="colName" sort="ascent" text="Nome"
width="170"/>
            <column name="colType" text="Tipo" width="140"/>
            <column name="colSize" text="Tamanho" width="80"/>
            <column name="colMod" text="Última Modificação" width="120"/>
        </header>
    </table>

    <panel columns="4" gap="6">
        <label for="tFilename" mnemonic="0" text="Nome do Arquivo:"/>
        <textfield name="tFilename" perform="ok()" weightx="1"/>

        <!-- some free space between filename field and ok/cancel buttons
-->
        <panel left="12" rowspan="2"/>

        <button action="ok()" icon="/thinletcommons/icons/ok.gif"
mnemonic="0" name="bOk" text="Ok" type="default"/>
        <label for="cbFilter" halign="right" mnemonic="1"
text="Filtro:"/>
        <combobox action="cbFilterChanged(this.selected)"
editable="false" name="cbFilter" weightx="1"/>
        <button action="close()" icon="/thinletcommons/icons/cancel.gif"
mnemonic="0" name="bCancel" text="Cancela" type="cancel"/>
    </panel>
</panel>
```

3.2.15 FontChooser.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- jEdit settings:
:tabSize=4:indentSize=4:noTabs=true:folding=explicit:collapseFolds=1: -->

<panel bottom="12" columns="3" gap="10" left="12" right="12" top="12"
weightx="1" weighty="1" init="init(fontlist)">
    <label colspan="3" for="fontlist" text="Font:" mnemonic="0"
valign="bottom"/>
    <list action="fontChanged(this, fontsize, cb_bold, cb_italic,
preview, b_ok)" line="false" name="fontlist" rowspan="3" weightx="4"
weighty="4"/>
    <label for="fontsize" text="Size:" mnemonic="0" />
```

```
    <spinbox action="fontChanged(fontlist, this, cb_bold, cb_italic,
preview, b_ok)" maximum="200" minimum="0" name="fontsize"
perform="fontChanged(fontlist, this, cb_bold, cb_italic, preview, b_ok)"
text="12" weightx="0"/>
    <panel/>
    <checkbox action="fontChanged(fontlist, fontsize, this, cb_italic,
preview, b_ok)" mnemonic="0" name="cb_bold" text="Bold"/>
    <panel/>
    <checkbox action="fontChanged(fontlist, fontsize, cb_bold, this,
preview, b_ok)" mnemonic="0" name="cb_italic" text="Italic"
valign="top"/>
    <panel border="true" bottom="5" colspan="3" left="3" right="3"
text="Preview" top="5" weightx="1" weighty="1">
        <textarea border="false" editable="false" end="35" height="30"
name="preview" start="35" text="The quick brown fox jumps over the lazy
dog." weightx="1" weighty="1" width="100" wrap="true"/>
    </panel>
    <panel colspan="3" gap="10" halign="right" weightx="1">
        <button action="ok(fontlist, fontsize, cb_bold, cb_italic)"
icon="/thinletcommons/icons/ok.gif" enabled="false" mnemonic="0"
name="b_ok" text="Ok" type="default"/>
        <button action="close()" icon="/thinletcommons/icons/cancel.gif"
mnemonic="0" name="b_cancel" text="Cancel" type="cancel"/>
    </panel>
</panel>
```

3.2.16 MessageDialog.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- jEdit settings:
:tabSize=4:indentSize=4:noTabs=false:folding=explicit:collapseFolds=1: --
>

<panel
      columns="2"
      gap="12" top="12" left="12" right="12" bottom="12"
      init="init(iconLabel, message, button1, button2, button3)"
>
      <label
            name="iconLabel"
            icon="/thinletcommons/icons/inform.gif"
            valign="top"
            rowspan="2"
      />
      <textarea
            name="message"
            editable="false"
            border="false"
            wrap="true"
            weightx="1"
            weighty="1"
      />
      <panel gap="12" halign="right">
            <button name="button1" text="..."
action="buttonPressed(this)"/>
```

```
            <button name="button2" text="..."
action="buttonPressed(this)"/>
            <button name="button3" text="..."
action="buttonPressed(this)"/>
      </panel>
</panel>
```

3.3 PACKAGE UTILS;

3.3.1 AWTUtils.Java

```
package utils;

import java.awt.AWTKeyStroke;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.font.FontRenderContext;
import java.awt.font.TextLayout;
import java.awt.geom.Rectangle2D;
import java.lang.reflect.Field;
import java.lang.reflect.Modifier;
import java.net.URL;
import java.util.HashMap;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;

public class AWTUtils
{
    private static final Logger log = Logger.getLogger("thinletcommons");
    private static final boolean debug() { return
log.isLoggable(Level.FINE); }

    private static HashMap<Integer,String> keyDescriptions = new
HashMap<Integer,String>();


    static
    {
        Field[] fields = KeyEvent.class.getDeclaredFields();
        for(int i = 0; i < fields.length; ++i)
        {
            Field field = fields[i];
            int modifiers = field.getModifiers();
            if(((modifiers & (Modifier.STATIC | Modifier.PUBLIC)) != 0)
                && field.getName().startsWith("VK_"))
            {
                try
```

```
                {
                    int keyCode = field.getInt(null);
                    String keyDescription = field.getName().substring(3);
                    keyDescriptions.put(new Integer(keyCode),
keyDescription);
                }
                catch(IllegalAccessException e)
                {
                    log.log(Level.SEVERE, "can't get value of field " +
field, e);
                }
            }
        }
    }

    public static String getColorString(Color c)
    {
        int red = c.getRed();
        int green = c.getGreen();
        int blue = c.getBlue();
        StringBuffer s = new StringBuffer("#");
        if(red < 10) s.append("0");
        s.append(Integer.toHexString(red));
        if(green < 10) s.append("0");
        s.append(Integer.toHexString(green));
        if(blue < 10) s.append("0");
        s.append(Integer.toHexString(blue));
        return s.toString();
    }

    public static String getFontString(Font font)
    {
        StringBuffer s = new StringBuffer();
        s.append(font.getFamily());
        s.append(" ");
        s.append(font.getSize());
        if(font.isItalic())
            s.append(" italic");
        if(font.isBold())
            s.append(" bold");
        return s.toString();
    }

    public static String getFontString(Font font, Font defaultFont)
    {
        StringBuffer s = new StringBuffer();
        if(!font.getFamily().equals(defaultFont.getFamily()))
            s.append(font.getFamily());
        if(font.getSize() != defaultFont.getSize())
        {
            if(s.length() > 0) s.append(' ');
            s.append(font.getSize());
        }
        if(font.isItalic() != defaultFont.isItalic())
        {
            if(s.length() > 0) s.append(' ');
            s.append("italic");
```

```
        }
        if(font.isBold() != defaultFont.isBold())
        {
            if(s.length() > 0) s.append(' ');
            s.append("bold");
        }
        return s.toString();
    }

    public static Frame getFrame(Component comp)
    {
        while(comp != null && !(Frame.class.isInstance(comp)))
            comp = comp.getParent();
        return (Frame)comp;
    }

    public static Dimension getBounds(String text, Component component)
    {
        Graphics2D graphics = (Graphics2D)component.getGraphics();
        StringTokenizer st = new StringTokenizer(text, "\n", true);
        Dimension dim = new Dimension(0, 0);
        while(st.hasMoreTokens())
        {
            String token = st.nextToken();
            if(token.equals("\n"))
                token = "W";
            TextLayout textLayout = new TextLayout(token,
component.getFont(),
                (graphics != null) ? graphics.getFontRenderContext()
                    : new FontRenderContext(null, true, false));
            Rectangle2D rect = textLayout.getBounds();
            dim.height += (int)rect.getHeight();
            dim.width = Math.max(dim.width, (int)rect.getWidth());
        }
        return dim;
    }

    public static Image getIcon(Component component, String path)
    {
        URL url = component.getClass().getResource(path);
        return getIcon(component, url);
    }

    public static Image getIcon(Component component, URL url)
    {
        if(debug()) log.fine("loading icon url=" + url + "...");
        Image icon = component.getToolkit().getImage(url);
        MediaTracker mediatracker = new MediaTracker(component);
        mediatracker.addImage(icon, 1);
        try
        {
            mediatracker.waitForID(1);
        }
        catch(InterruptedException e)
        {
            log.warning("loading of icon " + url + " has been
interrupted!");
```

```
        }

        return icon;
    }

    public static String getAWTKeyStrokeDescription(AWTKeyStroke k)
    {
        StringBuffer buf = new StringBuffer();

        int mod = k.getModifiers();
        if((mod & InputEvent.ALT_DOWN_MASK) != 0 || (mod &
InputEvent.ALT_MASK) != 0)
            buf.append("alt ");
        if((mod & InputEvent.ALT_GRAPH_DOWN_MASK) != 0 || (mod &
InputEvent.ALT_GRAPH_MASK) != 0)
            buf.append("altGraph ");
        if((mod & InputEvent.META_DOWN_MASK) != 0 || (mod &
InputEvent.META_MASK) != 0)
            buf.append("meta ");
        if((mod & InputEvent.CTRL_DOWN_MASK) != 0 || (mod &
InputEvent.CTRL_MASK) != 0)
            buf.append("ctrl ");
        if((mod & InputEvent.SHIFT_DOWN_MASK) != 0 || (mod &
InputEvent.SHIFT_MASK) != 0)
            buf.append("shift ");

        buf.append(getKeyText(k.getKeyCode()));

        return buf.toString();
    }

    public static String getKeyText(int keyCode)
    {
        String desc = (String) keyDescriptions.get(new Integer(keyCode));
        if(desc == null)
        {
            log.warning("KeyEvent field for keyCode " + keyCode + " not
found!"
                + " Returning default description.");
            return KeyEvent.getKeyText(keyCode);
        }
        else
            return desc;
    }
}
```

3.3.2 ReaderInputStream.Java

```
package utils;

import java.io.IOException;
import java.io.InputStream;
import java.io.Reader;

public class ReaderInputStream extends InputStream
{
```

```
    protected Reader reader;

    public ReaderInputStream(Reader reader)
    {
        this.reader = reader;
    }

    public int available() throws IOException
    {
        return reader.ready() ? 1 : 0;
    }

    public void close() throws IOException
    {
        reader.close();
    }

    public void mark(int readAheadLimit)
    {
        try { reader.mark(readAheadLimit); } catch(IOException ignore) {}
    }

    public boolean markSupported()
    {
        return reader.markSupported();
    }

    public int read() throws IOException
    {
        return reader.read();
    }

    public void reset() throws IOException
    {
        reader.reset();
    }

    public long skip(long n) throws IOException
    {
        return reader.skip(n);
    }

}
```