LAMPIRAN

Listing Program

DefaultPanel.java

```
package lrecog.gui;
import java.awt.Container;
import java.awt.Dimension;
import java.io.PrintStream;
import java.net.URL;
import javax.swing.BoxLayout;
import javax.swing.ImageIcon;
import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.border.Border;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.SoftBevelBorder;
import lrecog.Recognition;
import lrecog.tools.ProjectEnv;

public class DefaultPanel extends JPanel
{
EmptyBorder border5 = new EmptyBorder(5, 5, 5, 5);
EmptyBorder border10 = new EmptyBorder(10, 10, 10, 10);
Border loweredBorder = new CompoundBorder(new SoftBevelBorder(1), this.border5);
public static Dimension HGAP2 = new Dimension(2, 1);
public static Dimension VGAP2 = new Dimension(1, 2);
public static Dimension HGAP5 = new Dimension(5, 1);
public static Dimension VGAP5 = new Dimension(1, 5);
public static Dimension HGAP10 = new Dimension(10, 1);
public static Dimension VGAP10 = new Dimension(1, 10);
public static Dimension HGAP15 = new Dimension(15, 1);
public static Dimension VGAP15 = new Dimension(1, 15);
public static Dimension HGAP20 = new Dimension(20, 1);
public static Dimension VGAP20 = new Dimension(1, 20);
public static Dimension HGAP25 = new Dimension(25, 1);
public static Dimension VGAP25 = new Dimension(1, 25);
public static Dimension HGAP30 = new Dimension(30, 1);
public static Dimension VGAP30 = new Dimension(1, 30);
protected Recognition lrec;
protected ProjectEnv projectEnv;
public DefaultPanel(Recognition lrec)
{
this.lrec = lrec;
this.projectEnv = lrec.getProjectEnv();
}
public JPanel createHorizontalPanel(boolean threeD)
{
JPanel p = new JPanel();
p.setLayout(new BoxLayout(p, 0));
p.setAlignmentY(0.0F);
p.setAlignmentX(0.0F);
if (threeD)
{
p.setBorder(this.loweredBorder);
}
return p;
}
public JPanel createVerticalPanel(boolean threeD)
{
JPanel p = new JPanel();
p.setLayout(new BoxLayout(p, 1));
p.setAlignmentY(0.0F);
p.setAlignmentX(0.0F);
if (threeD)
{
```

```
p.setBorder(this.loweredBorder);
}
return p;
}
public ImageIcon createImageIcon(String filename, String description)
{
URL fileURL = super.getClass().getResource("/resources/images/".concat(String.valueOf(filename)));
if (fileURL == null)
{
System.err.println(String.valueOf(new StringBuffer("Warning: ImageIcon file
[").append(filename).append("] not found.")));
return null;
}
return new ImageIcon(fileURL, description);
}
public String getString(String key)
{
return this.lrec.getString(key);
}
}
```

ErrorGraphDraw.java

```
package lrecog.gui;
import java.awt.Color;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.Image;
import javax.swing.JComponent;
import javax.swing.JPanel;

public class ErrorGraphDraw extends JPanel
{
public static final int block = 500;
private int border = 10;
private int leftBorder = 30;
private int bottomBorder = 30;
private int fontSize = 10;
private int width = 370;
private int height = 230;
private double minVal = 0.0D;
private double maxVal = 1.1D;
private double minLim = 0.0D;
private double[] error;
private int nElements;
public ErrorGraphDraw()
{
clear();
super.setPreferredSize(new Dimension(this.width + 1, this.height + 1));
super.setMinimumSize(new Dimension(this.width + 1, this.height + 1));
super.setMaximumSize(new Dimension(this.width + 1, this.height + 1));
super.setLayout(new GridLayout(1, 0));
}
public void addError(double err)
{
if (this.nElements == this.error.length)
{
double[] temp = this.error;
this.error = new double[this.error.length + 500];
System.arraycopy(temp, 0, this.error, 0, this.nElements);
```

```
}
this.error[this.nElements] = err;
if (this.maxVal < err) this.maxVal = err;
this.nElements += 1;
super.repaint();
}
public void clear()
{
this.nElements = 1;
this.error = new double[500];
this.error[0] = 0.0D;
this.minVal = 0.0D;
this.maxVal = 1.1D;
}
public void paintComponent(Graphics g)
{
super.paintComponent(g);
int stepX = (this.width - this.border - this.leftBorder) / this.nElements;
int graphH = this.height - this.border - this.bottomBorder;
int graphY = this.border;
if (stepX == 0) clear();
Image image = super.createImage(this.width, this.height);
Graphics g2 = image.getGraphics();
g2.setFont(new Font("Helvetica", 0, this.fontSize));
g2.setColor(Color.black);
g2.fillRect(0, 0, this.width, this.height);
g2.setColor(Color.white);
g2.drawLine(this.leftBorder, this.border, this.leftBorder, this.height - (this.bottomBorder * 3 / 4));
g2.drawLine(this.leftBorder * 3 / 4, this.height - this.bottomBorder, this.width - this.border, this.height -
this.bottomBorder);
g2.setColor(Color.lightGray);
double x = 0.0D;
int y;
while (x < this.maxVal)
{
y = scale(x, this.minVal, this.maxVal, 0, graphH);
g2.drawLine(this.leftBorder - 2, this.height - this.bottomBorder - y, this.width - this.border, this.height -
this.bottomBorder - y);
g2.drawString("".concat(String.valueOf((float)x)), 2, this.height - this.bottomBorder - y + this.fontSize / 2);
if (this.maxVal > 4.0D)
{
x += 2.0D;
}
if (this.maxVal > 8.0D)
{
x += 4.0D;
}
x += 0.2D;
}
if (this.minLim < this.maxVal)
{
g2.setColor(Color.red);
y = scale(this.minLim, this.minVal, this.maxVal, 0, graphH);
g2.drawLine(this.leftBorder + 1, this.height - this.bottomBorder - y, this.width - this.border, this.height -
this.bottomBorder - y);
g2.drawString("".concat(String.valueOf(this.minLim)), 2, this.height - this.bottomBorder - y + this.fontSize /
2);
}
g2.setColor(Color.yellow);
for (int i = 2; i < this.nElements; ++i)
{
int y1 = scale(this.error[(i - 1)], this.minVal, this.maxVal, 0, 0 + graphH);
int y2 = scale(this.error[i], this.minVal, this.maxVal, 0, graphH);
```

```
g2.drawLine(this.leftBorder + 1 + (i - 1) * stepX, this.height - this.bottomBorder - y1, this.leftBorder + 1 + i *
stepX, this.height - this.bottomBorder - y2);
}
g.drawImage(image, 0, 0, this);
}
public static int scale(double x, double xMin, double xMax, int toMin, int toMax)
{
return ((int)((x - xMin) / (xMax - xMin) * (toMax - toMin)) + toMin);
}
}
```

ErrorGraphPanel.java

```
package lrecog.gui;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.AbstractButton;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.text.JTextComponent;
import lrecog.Recognition;

public class ErrorGraphPanel extends DefaultPanel
{
private ErrorGraphDraw graphPanel;
private JTextField errorField;
private JTextField stepField;
private int steps = 0;
public ErrorGraphPanel(Recognition lrec)
{
super(lrec);
super.setLayout(new BorderLayout());
super.setBorder(new TitledBorder(new EmptyBorder(0, 0, 0, 0), super.getString("ERROR_TITLE_INFO"),
1, 2));
this.graphPanel = new ErrorGraphDraw();
super.add(this.graphPanel, "Center");
JButton clearButton = new JButton(super.getString("ERROR_BT_CLEAR"));
clearButton.setMargin(new Insets(0, 1, 0, 0));
clearButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
ErrorGraphPanel.this.clear();
}
});
JLabel errorLabel = new JLabel(super.getString("ERROR_LB_ERROR"));
JLabel stepLabel = new JLabel(super.getString("ERROR_LB_STEP"));
this.errorField = new JTextField(super.getString("NA"));
this.errorField.setEditable(false);
this.errorField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.stepField = new JTextField("".concat(String.valueOf(this.steps)));
```

```
this.stepField.setEditable(false);
this.stepField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
errorLabel.setLabelFor(this.errorField);
stepLabel.setLabelFor(this.stepField);
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane, 0));
buttonPane.setBorder(BorderFactory.createEmptyBorder(0, 0, 10, 4));
buttonPane.add(clearButton);
buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
buttonPane.add(errorLabel);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(this.errorField);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(stepLabel);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(this.stepField);
buttonPane.add(Box.createHorizontalGlue());
super.add(buttonPane, "South");
}
public void addError(double err)
{
this.graphPanel.addError(err);
this.errorField.setText("".concat(String.valueOf(err)));
this.stepField.setText("".concat(String.valueOf(++this.steps)));
}
public void clear()
{
this.graphPanel.clear();
this.graphPanel.repaint();
this.errorField.setText(super.getString("NA"));
this.stepField.setText("0");
this.steps = 0;
super.repaint();
}
}
```

FileOpenDialog.java

```
package lrecog.gui;
import java.awt.Component;
import java.awt.Container;
import java.awt.Dialog;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.Window;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.Hashtable;
import java.util.Set;
import java.util.TreeSet;
import java.util.Vector;
import javax.swing.AbstractButton;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JFrame;
```

```
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.ListModel;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.text.JTextComponent;

public class FileOpenDialog extends JDialog
{
private JList fileList;
private JTextField fileText;
private JComboBox c_type;
private Vector currentList;
private String selection;
private boolean state;
private Dimension screen;
private Hashtable inhalt;
public static boolean APPROVE_OPTION = true;
public static boolean CANCEL_OPTION = false;
public FileOpenDialog(Hashtable inhalt, JFrame f, boolean modal)
{
super(f, modal);
this.inhalt = inhalt;
JButton open = new JButton("Open");
open.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent ae)
{
String str = FileOpenDialog.this.fileText.getText();
if (FileOpenDialog.this.isElementOfList(str) != 0)
{
FileOpenDialog.access$2(FileOpenDialog.this, FileOpenDialog.APPROVE_OPTION);
FileOpenDialog.this.dispose();
FileOpenDialog.this.clear();
}
else
{
int len = str.length();
if (len == 0)
return;
String message = String.valueOf(new StringBuffer("""").append(str).append("""").append(" does not exist!"));
JOptionPane.showMessageDialog(null, message, "Error", 0);
FileOpenDialog.this.clear();
}
}
});
JButton cancel = new JButton("Cancel");
cancel.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent ae)
{
FileOpenDialog.access$2(FileOpenDialog.this, FileOpenDialog.CANCEL_OPTION);
FileOpenDialog.this.dispose();
FileOpenDialog.this.clear();
}
});
Set s = inhalt.keySet();
TreeSet t = new TreeSet(s);
```

```
Vector v = new Vector(t);
this.c_type = new JComboBox(v);
this.c_type.setSelectedItem(v.get(0));
this.currentList = ((Vector)inhalt.get(v.get(0)));
this.c_type.addActionListener(new ActionListener(inhalt) {
private final Hashtable val$inhalt;
public void actionPerformed(ActionEvent ae) {
String str = (String)FileOpenDialog.this.c_type.getSelectedItem();
FileOpenDialog.access$5(FileOpenDialog.this, (Vector)this.val$inhalt.get(str));
FileOpenDialog.this.fileList.setListData(FileOpenDialog.this.currentList);
}
});
this.fileList = new JList(this.currentList);
this.fileList.setSelectionMode(0);
this.fileList.addListSelectionListener(new ListSelectionListener()
{
public void valueChanged(ListSelectionEvent lse)
{
String str = (String)FileOpenDialog.this.fileList.getSelectedValue();
FileOpenDialog.this.fileText.setText(str);
}
});
this.fileList.addMouseListener(new MouseAdapter()
{
public void mouseClicked(MouseEvent e)
{
if (e.getClickCount() != 2)
return;
String str = (String)FileOpenDialog.this.fileList.getSelectedValue();
FileOpenDialog.access$8(FileOpenDialog.this, str);
FileOpenDialog.access$2(FileOpenDialog.this, FileOpenDialog.APPROVE_OPTION);
FileOpenDialog.this.dispose();
FileOpenDialog.this.clear();
}
});
JScrollPane sp = new JScrollPane(this.fileList);
Border b = BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(12, 12, 2, 12),
BorderFactory.createEtchedBorder());
sp.setBorder(b);
JLabel l_name = new JLabel("File name", 2);
JLabel l_type = new JLabel("File type", 2);
this.fileText = new JTextField();
this.fileText.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent ae)
{
String str = FileOpenDialog.this.fileText.getText();
if (FileOpenDialog.this.isElementOfList(str) != 0)
{
FileOpenDialog.access$2(FileOpenDialog.this, FileOpenDialog.APPROVE_OPTION);
FileOpenDialog.this.dispose();
FileOpenDialog.this.clear();
}
else
{
String message = String.valueOf(new StringBuffer("").append(str).append("").append(" does not exist!"));
JOptionPane.showMessageDialog(null, message, "Error", 0);
FileOpenDialog.this.clear();
}
}
});
Container c = super.getContentPane();
JPanel pan2 = new JPanel();
pan2.setLayout(new GridLayout(0, 1, 6, 12));
```

```
pan2.add(l_name);
pan2.add(l_type);
JPanel pan = new JPanel();
pan.setBorder(new EmptyBorder(6, 6, 6, 6));
pan.add(pan2, "West");
pan2 = new JPanel();
pan2.setLayout(new GridLayout(0, 1, 6, 6));
pan2.add(this.fileText);
pan2.add(this.c_type);
pan.add(pan2, "Center");
pan2 = new JPanel();
pan2.setLayout(new GridLayout(0, 1, 6, 6));
pan2.add(open);
pan2.add(cancel);
pan.add(pan2, "East");
pan2 = new JPanel();
pan2.add(pan, "West");
JPanel p = new JPanel();
p.setLayout(new BoxLayout(p, 1));
p.add(sp);
p.add(pan);
c.add(p);
super.setTitle("Open");
super.pack();
Dimension d = super.getPreferredSize();
this.screen = super.getToolkit().getScreenSize();
this.screen.height /= 2;
this.screen.height -= d.height / 2;
this.screen.width /= 2;
this.screen.width -= d.width / 2;
super.setLocation(this.screen.width, this.screen.height);
super.addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent we)
{
FileOpenDialog.access$2(FileOpenDialog.this, FileOpenDialog.CANCEL_OPTION);
FileOpenDialog.this.dispose();
FileOpenDialog.this.clear();
}
});
}
private void clear()
{
this.fileList.clearSelection();
this.fileText.setText(null);
}
private boolean isElementOfList(String str)
{
if (str != null)
{
int i = 0;
boolean found = false;
while ((!(found)) && (i < this.currentList.size()))
{
String file = (String)this.currentList.get(i);
if (file.equals(str))
{
found = true;
this.selection = str;
}
++i;
}
return found;
}
```

```
return false;
}
public boolean showDialog()
{
super.pack();
super.setLocation(this.screen.width, this.screen.height);
super.setVisible(true);
return this.state;
}
public String getSelectedItem()
{
return this.selection;
}
public String getFirstItem()
{
ListModel lm = this.fileList.getModel();
int i = this.fileList.getFirstVisibleIndex();
return ((String)lm.getElementAt(i));
}
}
```

ImagePanel.java

```
package lrecog.gui;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;
import javax.swing.JComponent;
import javax.swing.JPanel;
import lrecog.gfx.EarImage;

public class ImagePanel extends JPanel
{
private EarImage actualImage;
private int width;
private int height;
public ImagePanel(EarImage img, int maxwidth, int maxheight)
{
this.actualImage = img;
if (this.actualImage != null) { calcRatio(maxwidth, maxheight);
}
else {
this.width = maxwidth;
this.height = maxheight;
}
super.setPreferredSize(new Dimension(this.width + 2, this.height + 2));
super.setMinimumSize(new Dimension(this.width + 2, this.height + 2));
super.setMaximumSize(new Dimension(this.width + 2, this.height + 2));
}
public ImagePanel()
{
this.actualImage = null;
}
public void setImage(EarImage img, int maxwidth, int maxheight)
{
this.actualImage = img;
calcRatio(maxwidth, maxheight);
super.setPreferredSize(new Dimension(this.width + 2, this.height + 2));
super.setMinimumSize(new Dimension(this.width + 2, this.height + 2));
super.repaint();
}
public void setImage(Image img)
```

```
{
this.actualImage = new EarImage(img);
if ((this.width == 0) || (this.height == 0)) return;
super.setPreferredSize(new Dimension(this.width + 2, this.height + 2));
super.setMinimumSize(new Dimension(this.width + 2, this.height + 2));
super.repaint();
}
public void paintComponent(Graphics g)
{
super.paintComponent(g);
if (this.actualImage == null)
return;
g.drawImage(this.actualImage.getImage(), 0, 0, this.width, this.height, new Color(220, 220, 220), null);
g.setColor(new Color(0, 0, 0));
g.drawLine(0, 0, 0, this.height - 1);
g.drawLine(0, 0, this.width - 1, 0);
g.drawLine(this.width - 1, 0, this.width - 1, this.height - 1);
g.drawLine(0, this.height - 1, this.width - 1, this.height - 1);
}
private void calcRatio(int maxwidth, int maxheight)
{
EarImage img = this.actualImage;
if ((img.getWidth() == 0) || (img.getHeight() == 0)) return;
int xdiff = img.getWidth() - maxwidth;
int ydiff = img.getHeight() - maxheight;
this.width = img.getWidth();
this.height = img.getHeight();
if ((xdiff <= 0) && (ydiff <= 0))
return;
this.width = maxwidth;
this.height = (int)(1.0D / img.getWidth() * maxwidth * img.getHeight());
}
}
```

ImageProcPanel.java

```
package lrecog.gui;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.ArrayList;
import javax.swing.AbstractButton;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JSlider;
import javax.swing.JSplitPane;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.text.JTextComponent;
```

```
import javax.swing.tree.DefaultMutableTreeNode;
import lrecog.Recognition;
import lrecog.gfx.ImageProcessor;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;
import lrecog.tools.ImageChooser;
import lrecog.tools.ImageFilter;
import lrecog.tools.ImagePreview;
import lrecog.tools.ProjectEnv;

public class ImageProcPanel extends DefaultPanel
{
private ImageTreePanel treePanel;
private EarImage actualImage = null;
private ImagePanel orgImagePanel;
private ImagePanel procImagePanel;
private JButton tokenButton;
private JProgressBar imgControlBar;
private JTextField imgNameField;
private JTextField imgSizeField;
private JTextField imgClassField;
private JTextField imgStatusField;
private JSlider thresholdSlider;
private JSlider distanceSlider;
private JSlider minlineSlider;
private static final int ORGIMG_MAXWIDTH = 100;
private static final int ORGIMG_MAXHEIGHT = 400;
private static final int PRCIMG_MAXWIDTH = 270;
private static final int PRCIMG_MAXHEIGHT = 400;
public ImageProcPanel(Recognition lrec)
{
super(lrec);
super.setLayout(new BoxLayout(this, 0));
super.setBorder(this.border5);
JPanel leftScrollPane = createLeftPanel();
JSplitPane rightScrollPane = createRightPanel();
JSplitPane splitPane = new JSplitPane(1, leftScrollPane, rightScrollPane);
splitPane.setOneTouchExpandable(true);
splitPane.setDividerLocation(390);
Dimension minimumSize = new Dimension(100, 50);
leftScrollPane.setMinimumSize(minimumSize);
super.add(splitPane);
}
private JPanel createLeftPanel()
{
JPanel leftPane = new JPanel();
leftPane.setLayout(new BorderLayout());
leftPane.setBorder(this.loweredBorder);
this.orgImagePanel = new ImagePanel(this.actualImage, 100, 400);
leftPane.add(this.orgImagePanel, "West");
this.procImagePanel = new ImagePanel(this.actualImage, 270, 400);
leftPane.add(this.procImagePanel, "Center");
return leftPane;
}
private JSplitPane createRightPanel()
{
JPanel topScrollPane = createJTreePanel();
JPanel bottomScrollPane = createImageOpPanel();
JSplitPane splitPane = new JSplitPane(0, topScrollPane, bottomScrollPane);
splitPane.setOneTouchExpandable(true);
splitPane.setDividerLocation(200);
Dimension minimumSize = new Dimension(50, 150);
topScrollPane.setMinimumSize(minimumSize);
bottomScrollPane.setMinimumSize(minimumSize);
```

```
return splitPane;
}
private JPanel createJTreePanel()
{
JPanel rootPane = new JPanel();
rootPane.setLayout(new BorderLayout());
this.treePanel = new ImageTreePanel(this.lrec);
rootPane.add(this.treePanel, "Center");
JButton addImageButton = new JButton(super.getString("IMGPROC_BT_ADDIMG"));
addImageButton.setMargin(new Insets(0, 1, 0, 0));
addImageButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
EarImage lImage = null;
if (ImageProcPanel.this.lrec.isApplet())
{
lImage = ImageProcPanel.this.lrec.getImageChooser().open();
}
else
{
JFileChooser fc = new JFileChooser();
fc.addChoosableFileFilter(new ImageFilter());
fc.setAccessory(new ImagePreview(fc));
int returnVal = fc.showOpenDialog(ImageProcPanel.this.lrec);
if (returnVal == 0)
{
File file = fc.getSelectedFile();
lImage = new EarImage(file);
}
}
if ((lImage == null) || (lImage.getImage() == null)) {
return;
}
HumanEar lEar = (HumanEar)ImageProcPanel.this.treePanel.addChild(lImage);
if (lEar != null) lEar.addImage(lImage);
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JButton addEarButton = new JButton(super.getString("IMGPROC_BT_NEWEAR"));
addEarButton.setMargin(new Insets(0, 1, 0, 0));
addEarButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
HumanEar lEar = new HumanEar(String.valueOf(new
StringBuffer(String.valueOf(ImageProcPanel.this.getString("EAR"))).append("
").append(ImageProcPanel.this.projectEnv.numHumanEar() + 1)));
ImageProcPanel.this.treePanel.addEar(lEar);
ImageProcPanel.this.projectEnv.addHumanEar(lEar);
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JButton removeButton = new JButton(super.getString("IMGPROC_BT_DELETE"));
removeButton.setMargin(new Insets(0, 1, 0, 0));
removeButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
HumanEar lEar = ImageProcPanel.this.treePanel.removeCurrentNode();
if (lEar == null)
return;
```

```
ImageProcPanel.this.projectEnv.removeHumanEar(lEar);
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JButton renameButton = new JButton(super.getString("IMGPROC_BT_RENAME"));
renameButton.setMargin(new Insets(0, 1, 0, 0));
renameButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
ImageProcPanel.this.treePanel.renameCurrentNode();
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JPanel button1Pane = new JPanel();
button1Pane.setLayout(new BoxLayout(button1Pane, 0));
button1Pane.setBorder(BorderFactory.createEmptyBorder(0, 0, 4, 4));
button1Pane.add(Box.createHorizontalGlue());
button1Pane.add(addImageButton);
button1Pane.add(Box.createRigidArea(new Dimension(3, 0)));
button1Pane.add(addEarButton);
JPanel button2Pane = new JPanel();
button2Pane.setLayout(new BoxLayout(button2Pane, 0));
button2Pane.setBorder(BorderFactory.createEmptyBorder(0, 0, 4, 4));
button2Pane.add(Box.createHorizontalGlue());
button2Pane.add(renameButton);
button2Pane.add(Box.createRigidArea(new Dimension(3, 0)));
button2Pane.add(removeButton);
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane, 1));
buttonPane.add(button1Pane);
buttonPane.add(button2Pane);
rootPane.add(buttonPane, "South");
return rootPane;
}
private JPanel createImageOpPanel()
{
JPanel imgOpPanel = new JPanel();
imgOpPanel.setLayout(new BorderLayout());
imgOpPanel.setBorder(new TitledBorder(new EmptyBorder(0, 0, 0, 0),
super.getString("IMGPROC_TITLE_OPS"), 1, 2));
JPanel imgInfoPanel = createImageInfoPanel();
JPanel imgControlPanel = createImageControlPanel();
imgOpPanel.add(imgInfoPanel, "North");
imgOpPanel.add(imgControlPanel, "South");
return imgOpPanel;
}
private JPanel createImageInfoPanel()
{
JPanel imgInfoPanel = new JPanel();
imgInfoPanel.setLayout(new BorderLayout());
JLabel imgNameLabel = new JLabel(super.getString("IMGPROC_LB_NAME"));
JLabel imgSizeLabel = new JLabel(super.getString("IMGPROC_LB_SIZE"));
JLabel imgClassLabel = new JLabel(super.getString("IMGPROC_LB_CLASS"));
JLabel imgStatusLabel = new JLabel(super.getString("IMGPROC_LB_TOKENS"));
this.imgNameField = new JTextField(super.getString("NA"));
this.imgNameField.setEditable(false);
this.imgNameField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.imgSizeField = new JTextField(super.getString("NA"));
this.imgSizeField.setEditable(false);
this.imgSizeField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.imgClassField = new JTextField(super.getString("NA"));
```

```
this.imgClassField.setEditable(false);
this.imgClassField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.imgStatusField = new JTextField(super.getString("NONE_FOUND_YET"));
this.imgStatusField.setEditable(false);
this.imgStatusField.setForeground(Color.red);
this.imgStatusField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
imgNameLabel.setLabelFor(this.imgNameField);
imgSizeLabel.setLabelFor(this.imgSizeField);
imgClassLabel.setLabelFor(this.imgClassField);
imgStatusLabel.setLabelFor(this.imgStatusField);
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(imgNameLabel);
labelPane.add(imgSizeLabel);
labelPane.add(imgClassLabel);
labelPane.add(imgStatusLabel);
JPanel fieldPane = new JPanel();
fieldPane.setLayout(new GridLayout(0, 1));
fieldPane.add(this.imgNameField);
fieldPane.add(this.imgSizeField);
fieldPane.add(this.imgClassField);
fieldPane.add(this.imgStatusField);
imgInfoPanel.add(labelPane, "West");
imgInfoPanel.add(fieldPane, "Center");
return imgInfoPanel;
}
private JPanel createImageControlPanel()
{
JPanel imgControlPanel = new JPanel();
imgControlPanel.setLayout(new BorderLayout());
JPanel sliderPanel = createSliderPanel();
this.imgControlBar = new JProgressBar(0, 100);
this.imgControlBar.setValue(0);
this.imgControlBar.setStringPainted(true);
this.tokenButton = new JButton(super.getString("IMGPROC_BT_FINDTOKENS"));
this.tokenButton.setMargin(new Insets(0, 1, 0, 0));
this.tokenButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
if (ImageProcPanel.this.actualImage == null) return;
ImageProcPanel.this.tokenButton.setEnabled(false);
ImageProcessor imgProc = new ImageProcessor(ImageProcPanel.this.actualImage.getImage());
ImageProcPanel.this.imgControlBar.setValue(1);
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_EDGEDETECT"));
ImageProcPanel.this.lrec.refresh();
imgProc.edgeDetect(ImageProcPanel.this.thresholdSlider.getValue() * 10);
ImageProcPanel.this.procImagePanel.setImage(imgProc.getImage(ImageProcPanel.this.procImagePanel));
ImageProcPanel.this.imgControlBar.setValue(20);
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_THINNING"));
ImageProcPanel.this.lrec.refresh();
imgProc.thinning();
ImageProcPanel.this.procImagePanel.setImage(imgProc.getImage(ImageProcPanel.this.procImagePanel));
ImageProcPanel.this.imgControlBar.setValue(40);
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_LINECHECK"));
ImageProcPanel.this.lrec.refresh();
imgProc.checkLines(ImageProcPanel.this.minlineSlider.getValue() * 10);
ImageProcPanel.this.procImagePanel.setImage(imgProc.getImage(ImageProcPanel.this.procImagePanel));
ImageProcPanel.this.imgControlBar.setValue(60);
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_DISTANCE"));
ImageProcPanel.this.lrec.refresh();
imgProc.markPoints(ImageProcPanel.this.distanceSlider.getValue() * 10);
ImageProcPanel.this.procImagePanel.setImage(imgProc.getImage(ImageProcPanel.this.procImagePanel));
ImageProcPanel.this.imgControlBar.setValue(80);
```

```
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_TOKENS"));
ImageProcPanel.this.lrec.refresh();
imgProc.calcAngels();
ImageProcPanel.this.procImagePanel.setImage(imgProc.getImage(ImageProcPanel.this.procImagePanel));
ImageProcPanel.this.imgControlBar.setValue(100);
ArrayList earTokens = imgProc.getTokens();
ImageProcPanel.this.actualImage.setTokens(earTokens);
ImageProcPanel.this.updateProjectEnv();
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_FINISHED"));
ImageProcPanel.this.tokenButton.setEnabled(true);
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JButton resetButton = new JButton(super.getString("IMGPROC_BT_RESET"));
resetButton.setMargin(new Insets(0, 1, 0, 0));
resetButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
if (ImageProcPanel.this.actualImage == null) return;
ImageProcPanel.this.procImagePanel.setImage(ImageProcPanel.this.actualImage, 270, 400);
ImageProcPanel.this.actualImage.clearTokens();
ImageProcPanel.this.lrec.setStatusField(ImageProcPanel.this.getString("STATUS_IMGRESET"));
ImageProcPanel.this.imgControlBar.setValue(0);
ImageProcPanel.this.projectEnv.setModified();
ImageProcPanel.this.lrec.updateProjectEnv();
}
});
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane, 0));
buttonPane.setBorder(BorderFactory.createEmptyBorder(2, 0, 5, 10));
buttonPane.add(this.tokenButton);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(resetButton);
buttonPane.add(Box.createHorizontalGlue());
imgControlPanel.add(sliderPanel, "North");
imgControlPanel.add(this.imgControlBar, "Center");
imgControlPanel.add(buttonPane, "South");
return imgControlPanel;
}
private JPanel createSliderPanel()
{
JPanel sliderPanel = new JPanel();
sliderPanel.setLayout(new BorderLayout());
JLabel thresholdSliderLabel = new JLabel(super.getString("IMGPROC_LB_THRESHOLD"));
JLabel distanceSliderLabel = new JLabel(super.getString("IMGPROC_LB_DISTANCE"));
JLabel minlineSliderLabel = new JLabel(super.getString("IMGPROC_LB_MINLINE"));
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(thresholdSliderLabel);
labelPane.add(distanceSliderLabel);
labelPane.add(minlineSliderLabel);
this.thresholdSlider = new JSlider(0, 0, 10, 5);
this.thresholdSlider.setMajorTickSpacing(1);
this.thresholdSlider.setSnapToTicks(true);
this.distanceSlider = new JSlider(0, 0, 10, 2);
this.distanceSlider.setMajorTickSpacing(1);
this.distanceSlider.setSnapToTicks(true);
this.minlineSlider = new JSlider(0, 0, 10, 2);
this.minlineSlider.setMajorTickSpacing(1);
this.minlineSlider.setSnapToTicks(true);
JPanel sliderGroup = new JPanel();
sliderGroup.setLayout(new GridLayout(0, 1));
```

```
sliderGroup.add(this.thresholdSlider);
sliderGroup.add(this.distanceSlider);
sliderGroup.add(this.minlineSlider);
sliderGroup.setPreferredSize(new Dimension(125, 0));
sliderPanel.add(labelPane, "West");
sliderPanel.add(sliderGroup, "Center");
return sliderPanel;
}
public void setActualImage(EarImage limage)
{
this.actualImage = limage;
this.orgImagePanel.setImage(limage, 100, 400);
this.procImagePanel.setImage(limage, 270, 400);
this.imgNameField.setText(limage.toString());
this.imgSizeField.setText(String.valueOf(new
StringBuffer(String.valueOf(limage.getImage().getWidth(null))).append("x").append(limage.getImage().getH
eight(null))));
updateProjectEnv();
this.lrec.refresh();
}
public void reloadTree()
{
ArrayList lEar = this.projectEnv.getHumanEar();
this.treePanel.clear();
for (int i = 0; i < lEar.size(); ++i)
{
HumanEar lNewEar = (HumanEar)lEar.get(i);
DefaultMutableTreeNode earTree = this.treePanel.addEar(lNewEar);
for (int j = 0; j < lNewEar.numImages(); ++j)
{
EarImage lImage = lNewEar.getImage(j);
this.treePanel.addChild(earTree, lImage);
}
}
}
public void updateProjectEnv()
{
if ((this.actualImage != null) && (this.actualImage.numTokens() > 0))
{
this.imgStatusField.setForeground(Color.blue);
this.imgStatusField.setText(String.valueOf(new
StringBuffer(String.valueOf(this.actualImage.numTokens())).append("
").append(super.getString("FOUND"))));
HumanEar lEar = (HumanEar)this.treePanel.getParentOfCurrent();
if (lEar != null)
{
this.imgClassField.setText(lEar.getName());
}
else
{
this.imgClassField.setText(super.getString("NA"));
}
}
else
{
this.imgStatusField.setForeground(Color.red);
this.imgStatusField.setText(super.getString("NONE_FOUND_YET"));
this.imgClassField.setText(super.getString("NA"));
}
}
}
```

ImageTreePanel.java

```
package lrecog.gui;
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.Toolkit;
import javax.swing.ImageIcon;
import javax.swing.JComponent;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTree;
import javax.swing.border.EmptyBorder;
import javax.swing.border.TitledBorder;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreePath;
import lrecog.Recognition;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;

public class ImageTreePanel extends DefaultPanel
{
private DefaultMutableTreeNode rootNode;
private DefaultTreeModel treeModel;
private EarImageTree tree;
private Toolkit toolkit = Toolkit.getDefaultToolkit();
public ImageTreePanel(Recognition lrec)
{
super(lrec);
super.setBorder(new TitledBorder(new EmptyBorder(0, 0, 0, 0),
super.getString("IMGTREE_TITLE_INFO"), 1, 2));
this.rootNode = new DefaultMutableTreeNode(super.getString("IMGTREE_LB_ROOTNODE"));
this.treeModel = new DefaultTreeModel(this.rootNode);
this.treeModel.setAsksAllowsChildren(true);
this.tree = new EarImageTree(this.treeModel);
DefaultTreeCellRenderer renderer = new DefaultTreeCellRenderer();
ImageIcon earIcon = super.createImageIcon("icon_ear.gif",
super.getString("IMGTREE_EARIMG_DESC"));
if (earIcon != null) renderer.setEarIcon(earIcon);
this.tree.setCellRenderer(renderer);
this.tree.addTreeSelectionListener(new TreeSelectionListener()
{
public void valueChanged(TreeSelectionEvent e)
{
DefaultMutableTreeNode node =
(DefaultMutableTreeNode)ImageTreePanel.this.tree.getLastSelectedPathComponent();
if (node == null) return;
Object nodeInfo = node.getUserObject();
if (node.getAllowsChildren() != false)
return;
EarImage limage = (EarImage)nodeInfo;
ImageTreePanel.this.getImageProcPanel().setActualImage(limage);
}
});
JScrollPane scrollPane = new JScrollPane(this.tree);
super.setLayout(new GridLayout(1, 0));
super.add(scrollPane);
}
public ImageProcPanel getImageProcPanel()
{
return this.lrec.getImageProcPanel();
```

```
}
public void clear()
{
this.rootNode.removeAllChildren();
this.treeModel.reload();
}
public void renameCurrentNode()
{
TreePath currentSelection = this.tree.getSelectionPath();
if (currentSelection != null)
{
DefaultMutableTreeNode currentNode =
(DefaultMutableTreeNode)currentSelection.getLastPathComponent();
if ((currentNode.getAllowsChildren() == true) && (currentNode.getParent() == this.rootNode))
{
HumanEar lEar = (HumanEar)currentNode.getUserObject();
String newName = (String)JOptionPane.showInputDialog(null, super.getString("WIN_RENAME_ENTER"),
super.getString("WIN_RENAME_TITLE"), 3, null, null, lEar.getName());
if (newName != null)
{
lEar.setName(newName);
super.update(super.getGraphics());
return;
}
return;
}
}
this.toolkit.beep();
}
public HumanEar removeCurrentNode()
{
TreePath currentSelection = this.tree.getSelectionPath();
if (currentSelection != null)
{
DefaultMutableTreeNode currentNode =
(DefaultMutableTreeNode)currentSelection.getLastPathComponent();
DefaultMutableTreeNode parent = (DefaultMutableTreeNode)currentNode.getParent();
if (parent != null)
{
if (parent != this.rootNode)
{
HumanEar lEar = (HumanEar)parent.getUserObject();
EarImage lImage = (EarImage)currentNode.getUserObject();
lEar.removeImage(lImage);
this.treeModel.removeNodeFromParent(currentNode);
return null;
}
this.treeModel.removeNodeFromParent(currentNode);
return ((HumanEar)currentNode.getUserObject());
}
}
this.toolkit.beep();
return null;
}
public Object addChild(Object child)
{
DefaultMutableTreeNode parentNode = null;
TreePath parentPath = this.tree.getSelectionPath();
if (parentPath == null)
{
parentNode = this.rootNode;
}
else
{
```

```
parentNode = (DefaultMutableTreeNode)parentPath.getLastPathComponent();
if (parentNode.getAllowsChildren() == false)
{
parentNode = (DefaultMutableTreeNode)parentNode.getParent();
}
}
return addChild(parentNode, child);
}
public Object addChild(DefaultMutableTreeNode parent, Object child)
{
DefaultMutableTreeNode childNode = new DefaultMutableTreeNode(child);
childNode.setAllowsChildren(false);
if (parent == null)
{
parent = this.rootNode;
}
this.treeModel.insertNodeInto(childNode, parent, parent.getChildCount());
this.treeModel.reload();
this.tree.scrollPathToVisible(new TreePath(childNode.getPath()));
if (parent != this.rootNode) return parent.getUserObject();
return null;
}
public DefaultMutableTreeNode addEar(Object ear)
{
DefaultMutableTreeNode earNode = new DefaultMutableTreeNode(ear);
earNode.setAllowsChildren(true);
this.treeModel.insertNodeInto(earNode, this.rootNode, this.rootNode.getChildCount());
this.treeModel.reload();
this.tree.scrollPathToVisible(new TreePath(earNode.getPath()));
return earNode;
}
public Object getParentOfCurrent()
{
TreePath currentSelection = this.tree.getSelectionPath();
if (currentSelection != null)
{
DefaultMutableTreeNode currentNode =
(DefaultMutableTreeNode)currentSelection.getLastPathComponent();
if (currentNode.getParent() != currentNode.getRoot())
{
return ((DefaultMutableTreeNode)currentNode.getParent()).getUserObject();
}
}
return null;
}
}
```

EarImageTree.java

```
package lrecog.gui;
import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Component;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Insets;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.SystemColor;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.awt.dnd.Autoscroll;
```

```
import java.awt.dnd.DragGestureEvent;
import java.awt.dnd.DragGestureListener;
import java.awt.dnd.DragSource;
import java.awt.dnd.DragSourceDragEvent;
import java.awt.dnd.DragSourceDropEvent;
import java.awt.dnd.DragSourceEvent;
import java.awt.dnd.DragSourceListener;
import java.awt.dnd.DropTarget;
import java.awt.dnd.DropTargetDragEvent;
import java.awt.dnd.DropTargetDropEvent;
import java.awt.dnd.DropTargetEvent;
import java.awt.dnd.DropTargetListener;
import java.awt.geom.AffineTransform;
import java.awt.geom.Rectangle2D;
import java.awt.geom.Rectangle2D.Float;
import java.awt.geom.RectangularShape;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.PrintStream;
import java.util.Arrays;
import java.util.List;
import javax.swing.Icon;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JTree;
import javax.swing.Timer;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreeCellRenderer;
import javax.swing.tree.TreeModel;
import javax.swing.tree.TreePath;
import javax.swing.tree.TreeSelectionModel;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;

public class EarImageTree extends JTree
implements DragSourceListener, DragGestureListener, Autoscroll
{
public static final DataFlavor TREEPATH_FLAVOR = new DataFlavor("application/x-java-jvm-local-
objectref", "TreePath");
private TreePath sourcePath;
private BufferedImage imgGhost;
private Point ptOffset = new Point();
private static final int AUTOSCROLL_MARGIN = 12;
public EarImageTree(DefaultTreeModel treeModel)
{
super(treeModel);
super.putClientProperty("JTree.lineStyle", "Angled");
super.setEditable(false);
super.setShowsRootHandles(false);
super.getSelectionModel().setSelectionMode(1);
DragSource dragSource = DragSource.getDefaultDragSource();
dragSource.createDefaultDragGestureRecognizer(this, 3, this);
DropTarget dropTarget = new DropTarget(this, new OwnDropTargetListener());
dropTarget.setDefaultActions(2);
}
public void dragGestureRecognized(DragGestureEvent e)
{
Point ptDragOrigin = e.getDragOrigin();
TreePath path = super.getPathForLocation(ptDragOrigin.x, ptDragOrigin.y);
if (path == null) return;
if (isRootPath(path)) return;
Rectangle raPath = super.getPathBounds(path);
this.ptOffset.setLocation(ptDragOrigin.x - raPath.x, ptDragOrigin.y - raPath.y);
```

```
JLabel lbl = (JLabel)super.getCellRenderer().getTreeCellRendererComponent(this,
path.getLastPathComponent(), false, super.isExpanded(path),
super.getModel().isEar(path.getLastPathComponent()), 0, false);
lbl.setSize((int)raPath.getWidth(), (int)raPath.getHeight());
this.imgGhost = new BufferedImage((int)raPath.getWidth(), (int)raPath.getHeight(), 3);
Graphics2D g2 = this.imgGhost.createGraphics();
g2.setComposite(AlphaComposite.getInstance(2, 0.5F));
lbl.paint(g2);
Icon icon = lbl.getIcon();
int nStartOfText = (icon == null) ? 0 : icon.getIconWidth() + lbl.getIconTextGap();
g2.setComposite(AlphaComposite.getInstance(4, 0.5F));
g2.setPaint(new GradientPaint(nStartOfText, 0.0F, SystemColor.controlShadow, super.getWidth(), 0.0F, new
Color(255, 255, 255, 0)));
g2.fillRect(nStartOfText, 0, super.getWidth(), this.imgGhost.getHeight());
g2.dispose();
super.setSelectionPath(path);
Transferable transferable = new TransferableTreePath(path);
this.sourcePath = path;
e.startDrag(null, this.imgGhost, new Point(5, 5), transferable, this);
}
public void dragEnter(DragSourceDragEvent e)
{
}
public void dragOver(DragSourceDragEvent e)
{
}
public void dragExit(DragSourceEvent e)
{
}
public void dropActionChanged(DragSourceDragEvent e)
{
}
public void dragDropEnd(DragSourceDropEvent e) {
if (!(e.getDropSuccess()))
return;
int nAction = e.getDropAction();
if (nAction != 2)
return;
this.sourcePath = null;
}
public void autoscroll(Point pt)
{
int nRow = super.getRowForLocation(pt.x, pt.y);
if (nRow < 0) {
return;
}
Rectangle raOuter = super.getBounds();
nRow = (nRow < super.getRowCount() - 1) ? nRow + 1 : (pt.y + raOuter.y <= 12) ? nRow - 1 : (nRow <= 0)
? 0 : nRow;
super.scrollRowToVisible(nRow);
}
public Insets getAutoscrollInsets()
{
Rectangle raOuter = super.getBounds();
Rectangle raInner = super.getParent().getBounds();
return new Insets(raInner.y - raOuter.y + 12, raInner.x - raOuter.x + 12, raOuter.height - raInner.height -
raInner.y + raOuter.y + 12, raOuter.width - raInner.width - raInner.x + raOuter.x + 12);
}
private TreePath getChildPath(TreePath pathParent, int nChildIndex)
{
TreeModel model = super.getModel();
return pathParent.pathByAddingChild(model.getChild(pathParent.getLastPathComponent(), nChildIndex));
}
private boolean isRootPath(TreePath path)
```

```
{
return ((super.isRootVisible()) && (super.getRowForPath(path) == 0));
}
class OwnDropTargetListener
implements DropTargetListener
{
private TreePath _pathLast = null;
private Rectangle2D _raCueLine = new Rectangle2D.Float();
private Rectangle2D _raGhost = new Rectangle2D.Float();
private Color _colorCueLine;
private Point _ptLast = new Point();
private Timer _timerHover;
private int _nLeftRight = 0;
private int _nShift = 0;
public OwnDropTargetListener()
{
this._colorCueLine = new Color(SystemColor.controlShadow.getRed(),
SystemColor.controlShadow.getGreen(), SystemColor.controlShadow.getBlue(), 64);
this._timerHover = new Timer(1000, new EarImageTree.1((OwnDropTargetListener)this));
this._timerHover.setRepeats(false);
}
public void dragEnter(DropTargetDragEvent e)
{
if (!(isDragAcceptable(e)))
e.rejectDrag();
else
e.acceptDrag(e.getDropAction());
}
public void dragExit(DropTargetEvent e)
{
if (DragSource.isDragImageSupported())
return;
EarImageTree.this.repaint(this._raGhost.getBounds());
}
public void dragOver(DropTargetDragEvent e)
{
Point pt = e.getLocation();
if (pt.equals(this._ptLast)) {
return;
}
int nDeltaLeftRight = pt.x - this._ptLast.x;
if (((this._nLeftRight > 0) && (nDeltaLeftRight < 0)) || ((this._nLeftRight < 0) && (nDeltaLeftRight > 0)))
this._nLeftRight = 0;
this._nLeftRight += nDeltaLeftRight;
this._ptLast = pt;
Graphics2D g2 = (Graphics2D)EarImageTree.this.getGraphics();
if (!(DragSource.isDragImageSupported()))
{
EarImageTree.this.paintImmediately(this._raGhost.getBounds());
this._raGhost.setRect(pt.x - EarImageTree.this.ptOffset.x, pt.y - EarImageTree.this.ptOffset.y,
EarImageTree.this.imgGhost.getWidth(), EarImageTree.this.imgGhost.getHeight());
g2.drawImage(EarImageTree.this.imgGhost, AffineTransform.getTranslateInstance(this._raGhost.getX(),
this._raGhost.getY()), null);
}
else {
EarImageTree.this.paintImmediately(this._raCueLine.getBounds());
}
TreePath path = EarImageTree.this.getClosestPathForLocation(pt.x, pt.y);
if (path != this._pathLast)
{
this._nLeftRight = 0;
this._pathLast = path;
this._timerHover.restart();
}
```

```
Rectangle raPath = EarImageTree.this.getPathBounds(path);
this._raCueLine.setRect(0.0D, raPath.y + (int)raPath.getHeight(), EarImageTree.this.getWidth(), 2.0D);
g2.setColor(this._colorCueLine);
g2.fill(this._raCueLine);
if (this._nLeftRight > 20)
{
this._nShift = 1;
}
else if (this._nLeftRight < -20)
{
this._nShift = -1;
}
else {
this._nShift = 0;
}
this._raGhost = this._raGhost.createUnion(this._raCueLine);
}
public void dropActionChanged(DropTargetDragEvent e)
{
if (!(isDragAcceptable(e)))
e.rejectDrag();
else
e.acceptDrag(e.getDropAction());
}
public void drop(DropTargetDropEvent e)
{
this._timerHover.stop();
if (!(isDropAcceptable(e)))
{
e.rejectDrop();
return;
}
e.acceptDrop(e.getDropAction());
Transferable transferable = e.getTransferable();
DataFlavor[] flavors = transferable.getTransferDataFlavors();
for (int i = 0; i < flavors.length; ++i)
{
DataFlavor flavor = flavors[i];
if (!(flavor.isMimeTypeEqual("application/x-java-jvm-local-objectref")))
continue;
try
{
Point pt = e.getLocation();
TreePath pathTarget = EarImageTree.this.getClosestPathForLocation(pt.x, pt.y);
TreePath pathSource = (TreePath)transferable.getTransferData(flavor);
DefaultTreeModel model = (DefaultTreeModel)EarImageTree.this.getModel();
TreePath pathNewChild = null;
DefaultMutableTreeNode sourceNode = (DefaultMutableTreeNode)pathSource.getLastPathComponent();
DefaultMutableTreeNode destNode = (DefaultMutableTreeNode)pathTarget.getLastPathComponent();
if (destNode.getAllowsChildren() == false)
{
destNode = (DefaultMutableTreeNode)destNode.getParent();
}
if ((sourceNode.getAllowsChildren() == true) && (destNode != sourceNode.getRoot()))
{
destNode = (DefaultMutableTreeNode)destNode.getRoot();
}
if ((destNode.getAllowsChildren() == true) && (sourceNode.getAllowsChildren() == false) && (destNode
!= destNode.getRoot()))
{
EarImage srcImage = (EarImage)sourceNode.getUserObject();
HumanEar destEar = (HumanEar)destNode.getUserObject();
if (sourceNode.getParent() != sourceNode.getRoot())
{
```

```
HumanEar srcEar = (HumanEar)((DefaultMutableTreeNode)sourceNode.getParent()).getUserObject();
srcEar.removeImage(srcImage);
}
destEar.addImage(srcImage);
model.removeNodeFromParent(sourceNode);
model.insertNodeInto(sourceNode, destNode, destNode.getChildCount());
model.reload();
EarImageTree.this.setSelectionPath(new TreePath(sourceNode.getPath()));
}
}
catch (UnsupportedFlavorException ufe)
{
System.out.println(ufe);
e.dropComplete(false);
return;
}
catch (IOException ioe)
{
System.out.println(ioe);
e.dropComplete(false);
return;
}
}
e.dropComplete(true);
}
public boolean isDragAcceptable(DropTargetDragEvent e)
{
if ((e.getDropAction() & 0x2) == 0) {
return false;
}
return (e.isDataFlavorSupported(EarImageTree.TREEPATH_FLAVOR));
}
public boolean isDropAcceptable(DropTargetDropEvent e)
{
if ((e.getDropAction() & 0x2) == 0) {
return false;
}
return (e.isDataFlavorSupported(EarImageTree.TREEPATH_FLAVOR));
}
}
class TransferableTreePath
implements Transferable
{
private TreePath _path;
private DataFlavor[] _flavors = { EarImageTree.TREEPATH_FLAVOR };
public TransferableTreePath(TreePath path)
{
this._path = path;
}
public DataFlavor[] getTransferDataFlavors()
{
return this._flavors;
}
public boolean isDataFlavorSupported(DataFlavor flavor)
{
return Arrays.asList(this._flavors).contains(flavor);
}
public synchronized Object getTransferData(DataFlavor flavor) throws UnsupportedFlavorException
{
if (flavor.isMimeTypeEqual(EarImageTree.TREEPATH_FLAVOR.getMimeType())) {
return this._path;
}
throw new UnsupportedFlavorException(flavor);
}
```

```
}
}
```

NetworkPanel.java

```
package lrecog.gui;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;
import javax.swing.AbstractButton;
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JSplitPane;
import javax.swing.JTextField;
import javax.swing.border.TitledBorder;
import javax.swing.text.JTextComponent;
import lrecog.Recognition;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;
import lrecog.nnetwork.BackProp;
import lrecog.tools.ProjectEnv;

public class NetworkPanel extends DefaultPanel
{
private ErrorGraphPanel errorPanel;
private JProgressBar netProgressBar;
private JTextField numEarImagesField;
private JTextField numHumanEarField;
private JTextField maxTokenField;
private JTextField statusField;
private JButton trainButton;
private JTextField inputNeuronField;
private JTextField hiddenNeuronField;
private JTextField outputNeuronField;
private JTextField learnRateField;
private JTextField momentumField;
private JTextField stepsField;
public NetworkPanel(Recognition lrec)
{
super(lrec);
super.setLayout(new BoxLayout(this, 0));
super.setBorder(this.border5);
JPanel leftPane = createLeftPanel();
JPanel rightPane = createRightPanel();
JSplitPane splitPane = new JSplitPane(1, leftPane, rightPane);
splitPane.setOneTouchExpandable(true);
splitPane.setDividerLocation(390);
Dimension minimumSize = new Dimension(100, 50);
leftPane.setMinimumSize(minimumSize);
super.add(splitPane);
```

```
}
private JPanel createLeftPanel()
{
JPanel leftPane = new JPanel();
leftPane.setLayout(new BorderLayout());
leftPane.setBorder(this.loweredBorder);
this.errorPanel = new ErrorGraphPanel(this.lrec);
leftPane.add(this.errorPanel, "North");
return leftPane;
}
private JPanel createRightPanel()
{
JPanel rightPane = new JPanel();
rightPane.setLayout(new BorderLayout());
rightPane.add(createNetworkInfoPanel(), "North");
rightPane.add(createNetworkOpPanel(), "South");
return rightPane;
}
private JPanel createNetworkInfoPanel()
{
JPanel netInfoPanel = new JPanel();
netInfoPanel.setLayout(new BorderLayout());
netInfoPanel.setBorder(new TitledBorder(super.getString("NETWORK_TITLE_INFO")));
JLabel numEarImagesLabel = new JLabel(super.getString("NETWORK_LB_EARIMAGES"));
JLabel numHumanEarLabel = new JLabel(super.getString("NETWORK_LB_HUMANEAR"));
JLabel maxTokenLabel = new JLabel(super.getString("NETWORK_LB_MAXTOKEN"));
JLabel statusLabel = new JLabel(super.getString("NETWORK_LB_STATUS"));
this.numEarImagesField = new JTextField(super.getString("NA"));
this.numEarImagesField.setEditable(false);
this.numEarImagesField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.numHumanEarField = new JTextField(super.getString("NA"));
this.numHumanEarField.setEditable(false);
this.numHumanEarField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.maxTokenField = new JTextField(super.getString("NA"));
this.maxTokenField.setEditable(false);
this.maxTokenField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.statusField = new JTextField(super.getString("NOT_TRAINED_YET"));
this.statusField.setForeground(Color.red);
this.statusField.setEditable(false);
this.statusField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
numEarImagesLabel.setLabelFor(this.numEarImagesField);
numHumanEarLabel.setLabelFor(this.numHumanEarField);
maxTokenLabel.setLabelFor(this.maxTokenField);
statusLabel.setLabelFor(this.statusField);
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(numEarImagesLabel);
labelPane.add(numHumanEarLabel);
labelPane.add(maxTokenLabel);
labelPane.add(statusLabel);
JPanel fieldPane = new JPanel();
fieldPane.setLayout(new GridLayout(0, 1));
fieldPane.add(this.numEarImagesField);
fieldPane.add(this.numHumanEarField);
fieldPane.add(this.maxTokenField);
fieldPane.add(this.statusField);
netInfoPanel.add(labelPane, "West");
netInfoPanel.add(fieldPane, "Center");
return netInfoPanel;
}
private JPanel createNetworkOpPanel()
{
JPanel netOpPanel = new JPanel();
netOpPanel.setLayout(new BorderLayout());
```

```
netOpPanel.setBorder(new TitledBorder(super.getString("NETWORK_TITLE_OPS")));
JLabel inputNeuronLabel = new JLabel(super.getString("NETWORK_LB_INPUT"));
JLabel hiddenNeuronLabel = new JLabel(super.getString("NETWORK_LB_HIDDEN"));
JLabel outputNeuronLabel = new JLabel(super.getString("NETWORK_LB_OUTPUT"));
JLabel learnRateLabel = new JLabel(super.getString("NETWORK_LB_LEARNRATE"));
JLabel momentumLabel = new JLabel(super.getString("NETWORK_LB_MOMENTUM"));
JLabel stepsLabel = new JLabel(super.getString("NETWORK_LB_STEPS"));
this.inputNeuronField = new JTextField("10");
this.hiddenNeuronField = new JTextField("5");
this.outputNeuronField = new JTextField("2");
this.learnRateField = new JTextField("0.3");
this.momentumField = new JTextField("1.0");
this.stepsField = new JTextField("100");
inputNeuronLabel.setLabelFor(this.inputNeuronField);
hiddenNeuronLabel.setLabelFor(this.hiddenNeuronField);
outputNeuronLabel.setLabelFor(this.outputNeuronField);
learnRateLabel.setLabelFor(this.learnRateField);
momentumLabel.setLabelFor(this.momentumField);
stepsLabel.setLabelFor(this.stepsField);
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(inputNeuronLabel);
labelPane.add(hiddenNeuronLabel);
labelPane.add(outputNeuronLabel);
labelPane.add(learnRateLabel);
labelPane.add(momentumLabel);
labelPane.add(stepsLabel);
JPanel fieldPane = new JPanel();
fieldPane.setLayout(new GridLayout(0, 1));
fieldPane.add(this.inputNeuronField);
fieldPane.add(this.hiddenNeuronField);
fieldPane.add(this.outputNeuronField);
fieldPane.add(this.learnRateField);
fieldPane.add(this.momentumField);
fieldPane.add(this.stepsField);
JPanel contrPane = createNetworkControlPanel();
netOpPanel.add(labelPane, "West");
netOpPanel.add(fieldPane, "Center");
netOpPanel.add(contrPane, "South");
return netOpPanel;
}
private JPanel createNetworkControlPanel()
{
JPanel netContrPanel = new JPanel();
netContrPanel.setLayout(new BorderLayout());
this.netProgressBar = new JProgressBar(0, 50);
this.netProgressBar.setValue(0);
this.netProgressBar.setStringPainted(true);
this.trainButton = new JButton(super.getString("NETWORK_BT_TRAIN"));
this.trainButton.setEnabled(false);
this.trainButton.setMargin(new Insets(0, 1, 0, 0));
this.trainButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
NetworkPanel.this.trainNetwork();
NetworkPanel.this.projectEnv.setModified();
NetworkPanel.this.lrec.updateProjectEnv();
}
});
JButton defaultButton = new JButton(super.getString("NETWORK_BT_DEFAULT"));
defaultButton.setMargin(new Insets(0, 1, 0, 0));
defaultButton.addActionListener(new ActionListener()
{
```

```
public void actionPerformed(ActionEvent e)
{
int maxToken = NetworkPanel.this.projectEnv.getMaxToken() * 2;
NetworkPanel.this.inputNeuronField.setText("".concat(String.valueOf(maxToken)));
NetworkPanel.this.hiddenNeuronField.setText("20");
NetworkPanel.this.outputNeuronField.setText("".concat(String.valueOf(NetworkPanel.this.projectEnv.numH
umanEar())));
NetworkPanel.this.learnRateField.setText("0.3");
NetworkPanel.this.momentumField.setText("1.0");
NetworkPanel.this.stepsField.setText("500");
NetworkPanel.this.netProgressBar.setValue(0);
NetworkPanel.this.projectEnv.setNetwork(null);
NetworkPanel.this.projectEnv.setModified();
NetworkPanel.this.lrec.updateProjectEnv();
}
});
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane, 0));
buttonPane.setBorder(BorderFactory.createEmptyBorder(2, 0, 5, 10));
buttonPane.add(this.trainButton);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(defaultButton);
buttonPane.add(Box.createHorizontalGlue());
netContrPanel.add(this.netProgressBar, "Center");
netContrPanel.add(buttonPane, "South");
return netContrPanel;
}
public void trainNetwork()
{
int inputs = Integer.parseInt(this.inputNeuronField.getText());
int hiddens = Integer.parseInt(this.hiddenNeuronField.getText());
int outputs = Integer.parseInt(this.outputNeuronField.getText());
double learnRate = Double.parseDouble(this.learnRateField.getText());
double momentum = Double.parseDouble(this.momentumField.getText());
int steps = Integer.parseInt(this.stepsField.getText());
this.netProgressBar.setMaximum(steps);
BackProp nNetwork = new BackProp(inputs, hiddens, outputs);
this.projectEnv.setNetwork(nNetwork);
nNetwork.setAlpha(learnRate);
nNetwork.setMomentum(momentum);
ArrayList HumanEar = this.projectEnv.getHumanEar();
Random rand = new Random();
for (int i = 0; i < HumanEar.size(); ++i)
{
HumanEar lEar = (HumanEar)HumanEar.get(i);
double[] outputVals = new double[outputs];
for (j = 0; j < outputs; ++j)
{
if (j <= i)
{
outputVals[j] = 1.0D;
}
else outputVals[j] = 0.0D;
}
lEar.setID(outputVals);
}
this.errorPanel.clear();
ArrayList imageList = new ArrayList();
for (int j = 0; j < HumanEar.size(); ++j)
{
HumanEar lEar = (HumanEar)HumanEar.get(j);
for (int k = 0; k < lEar.numImages(); ++k)
{
EarImage lImage = lEar.getImage(k);
```

```
lImage.setEar(lEar);
imageList.add(lImage);
}
}
for (int i = 0; i < steps; ++i)
{
double sumError = 0.0D;
Collections.shuffle(imageList);
for (int j = 0; j < imageList.size(); ++j)
{
EarImage lImage = (EarImage)imageList.get(j);
sumError += nNetwork.learnVector(lImage.getTokens(inputs), lImage.getEar().getID());
}
this.errorPanel.addError(sumError);
this.netProgressBar.setValue(i + 1);
this.lrec.refresh();
}
}
public void updateProjectEnv()
{
this.numHumanEarField.setText("".concat(String.valueOf(this.projectEnv.numHumanEar())));
this.numEarImagesField.setText("".concat(String.valueOf(this.projectEnv.numEarImages())));
this.maxTokenField.setText("".concat(String.valueOf(this.projectEnv.getMaxToken())));
BackProp nNetwork = this.projectEnv.getNetwork();
if (nNetwork != null)
{
this.inputNeuronField.setText("".concat(String.valueOf(nNetwork.numInput())));
this.hiddenNeuronField.setText("".concat(String.valueOf(nNetwork.numHidden())));
this.outputNeuronField.setText("".concat(String.valueOf(nNetwork.numOutput())));
this.learnRateField.setText("".concat(String.valueOf(nNetwork.getAlpha())));
this.momentumField.setText("".concat(String.valueOf(nNetwork.getMomentum())));
}
if (this.projectEnv.getMaxToken() > 0)
{
this.trainButton.setEnabled(true);
}
else this.trainButton.setEnabled(false);
if (this.projectEnv.getNetwork() != null)
{
this.statusField.setForeground(Color.blue);
this.statusField.setText(super.getString("TRAINED"));
}
else
{
this.statusField.setForeground(Color.red);
this.statusField.setText(super.getString("NOT_TRAINED_YET"));
}
}
}
```

RecognitionPanel.java

```
package lrecog.gui;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.ArrayList;
import javax.swing.AbstractButton;
```

```
import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JSlider;
import javax.swing.JSplitPane;
import javax.swing.JTextField;
import javax.swing.border.TitledBorder;
import javax.swing.text.JTextComponent;
import lrecog.Recognition;
import lrecog.gfx.ImageProcessor;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;
import lrecog.nnetwork.BackProp;
import lrecog.tools.ImageChooser;
import lrecog.tools.ImageFilter;
import lrecog.tools.ImagePreview;
import lrecog.tools.ProjectEnv;

public class RecognitionPanel extends DefaultPanel
{
private EarImage actualImage = null;
private ImagePanel recogImagePanel;
private RecogTablePanel recogTablePanel;
private JButton recogButton;
private JButton resetButton;
private JTextField imageNameField;
private JTextField imageTokenField;
private JTextField statusField;
private JProgressBar recogProgressBar;
private JSlider thresholdSlider;
private JSlider distanceSlider;
private JSlider minlineSlider;
private static final int IMG_MAXWIDTH = 370;
private static final int IMG_MAXHEIGHT = 400;
public RecognitionPanel(Recognition lrec)
{
super(lrec);
super.setLayout(new BoxLayout(this, 0));
super.setBorder(this.border5);
JPanel leftPane = createLeftPanel();
JPanel rightPane = createRightPanel();
JSplitPane splitPane = new JSplitPane(1, leftPane, rightPane);
splitPane.setOneTouchExpandable(true);
splitPane.setDividerLocation(390);
Dimension minimumSize = new Dimension(100, 50);
leftPane.setMinimumSize(minimumSize);
super.add(splitPane);
}
private JPanel createLeftPanel()
{
JPanel leftPane = new JPanel();
leftPane.setLayout(new BorderLayout());
leftPane.setBorder(this.loweredBorder);
this.recogImagePanel = new ImagePanel(this.actualImage, 370, 400);
leftPane.add(this.recogImagePanel, "West");
return leftPane;
}
private JPanel createRightPanel()
```

```
{
JPanel rightPane = new JPanel();
rightPane.setLayout(new BorderLayout());
rightPane.add(createRecogInfoPanel(), "North");
rightPane.add(createRecogResultPanel(), "Center");
rightPane.add(createRecogControlPanel(), "South");
return rightPane;
}
private JPanel createRecogInfoPanel()
{
JPanel recogInfoPanel = new JPanel();
recogInfoPanel.setLayout(new BorderLayout());
recogInfoPanel.setBorder(new
TitledBorder(super.getString("RECOG_TITLE_INFO")));
JLabel imageNameLabel = new JLabel(super.getString("RECOG_LB_IMAGE"));
JLabel imageTokenLabel = new JLabel(super.getString("RECOG_LB_TOKEN"));
JLabel statusLabel = new JLabel(super.getString("RECOG_LB_STATUS"));
this.imageNameField = new JTextField(super.getString("NA"));
this.imageNameField.setEditable(false);
this.imageNameField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.imageTokenField = new JTextField(super.getString("NA"));
this.imageTokenField.setEditable(false);
this.imageTokenField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
this.statusField = new JTextField(super.getString("NOT_RECOGNIZED_YET"));
this.statusField.setForeground(Color.red);
this.statusField.setEditable(false);
this.statusField.setBorder(BorderFactory.createEmptyBorder(1, 1, 1, 1));
imageNameLabel.setLabelFor(this.imageNameField);
imageTokenLabel.setLabelFor(this.imageTokenField);
statusLabel.setLabelFor(this.statusField);
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(imageNameLabel);
labelPane.add(imageTokenLabel);
labelPane.add(statusLabel);
JPanel fieldPane = new JPanel();
fieldPane.setLayout(new GridLayout(0, 1));
fieldPane.add(this.imageNameField);
fieldPane.add(this.imageTokenField);
fieldPane.add(this.statusField);
recogInfoPanel.add(labelPane, "West");
recogInfoPanel.add(fieldPane, "Center");
return recogInfoPanel;
}
private JPanel createRecogResultPanel()
{
JPanel recogResultPanel = new JPanel();
recogResultPanel.setLayout(new BorderLayout());
recogResultPanel.setBorder(new TitledBorder(super.getString("RECOG_TITLE_RESULT")));
this.recogTablePanel = new RecogTablePanel(this.lrec);
recogResultPanel.add(this.recogTablePanel, "Center");
return recogResultPanel;
}
private JPanel createRecogControlPanel()
{
JPanel recogControlPanel = new JPanel();
recogControlPanel.setLayout(new BorderLayout());
recogControlPanel.setBorder(new TitledBorder(super.getString("RECOG_TITLE_CONTROL")));
JPanel sliderPanel = createSliderPanel();
this.recogProgressBar = new JProgressBar(0, 100);
this.recogProgressBar.setValue(0);
this.recogProgressBar.setStringPainted(true);
JButton loadButton = new JButton(super.getString("RECOG_BT_LOADIMAGE"));
loadButton.setMargin(new Insets(0, 1, 0, 0));
```

```
loadButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
EarImage lImage = null;
if (RecognitionPanel.this.lrec.isApplet())
{
lImage = RecognitionPanel.this.lrec.getImageChooser().open();
}
else
{
JFileChooser fc = new JFileChooser();
fc.addChoosableFileFilter(new ImageFilter());
fc.setAccessory(new ImagePreview(fc));
int returnVal = fc.showOpenDialog(RecognitionPanel.this.lrec);
if (returnVal == 0)
{
File file = fc.getSelectedFile();
lImage = new EarImage(file);
}
}
if ((lImage == null) || (lImage.getImage() == null)) {
return;
}
RecognitionPanel.access$0(RecognitionPanel.this, lImage);
RecognitionPanel.this.recogImagePanel.setImage(lImage, 370, 400);
RecognitionPanel.this.imageNameField.setText(lImage.toString());
RecognitionPanel.this.imageTokenField.setText("0");
RecognitionPanel.this.statusField.setForeground(Color.red);
RecognitionPanel.this.statusField.setText(RecognitionPanel.this.getString("NOT_RECOGNIZED_YET"));
RecognitionPanel.this.updateProjectEnv();
RecognitionPanel.this.lrec.refresh();
}
});
this.recogButton = new JButton(super.getString("RECOG_BT_RECOGNIZE"));
if (!(this.lrec.isApplet())) this.recogButton.setEnabled(false);
this.recogButton.setMargin(new Insets(0, 1, 0, 0));
this.recogButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
RecognitionPanel.this.recogTablePanel.clear();
ImageProcessor imgProc = new ImageProcessor(RecognitionPanel.this.actualImage.getImage());
RecognitionPanel.this.recogProgressBar.setValue(1);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_EDGEDETECT"));
RecognitionPanel.this.lrec.refresh();
imgProc.edgeDetect(RecognitionPanel.this.thresholdSlider.getValue() * 10);
RecognitionPanel.this.recogImagePanel.setImage(imgProc.getImage(RecognitionPanel.this.recogImagePanel
));
RecognitionPanel.this.recogProgressBar.setValue(16);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_THINNING"));
RecognitionPanel.this.lrec.refresh();
imgProc.thinning();
RecognitionPanel.this.recogImagePanel.setImage(imgProc.getImage(RecognitionPanel.this.recogImagePanel
));
RecognitionPanel.this.recogProgressBar.setValue(32);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_LINECHECK"));
RecognitionPanel.this.lrec.refresh();
imgProc.checkLines(RecognitionPanel.this.minlineSlider.getValue() * 10);
RecognitionPanel.this.recogImagePanel.setImage(imgProc.getImage(RecognitionPanel.this.recogImagePanel
));
RecognitionPanel.this.recogProgressBar.setValue(48);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_DISTANCE"));
RecognitionPanel.this.lrec.refresh();
```

```
imgProc.markPoints(RecognitionPanel.this.distanceSlider.getValue() * 10);
RecognitionPanel.this.recogImagePanel.setImage(imgProc.getImage(RecognitionPanel.this.recogImagePanel
));
RecognitionPanel.this.recogProgressBar.setValue(64);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_TOKENS"));
RecognitionPanel.this.lrec.refresh();
imgProc.calcAngels();
RecognitionPanel.this.recogImagePanel.setImage(imgProc.getImage(RecognitionPanel.this.recogImagePanel
));
RecognitionPanel.this.recogProgressBar.setValue(96);
ArrayList EarTokens = imgProc.getTokens();
RecognitionPanel.this.actualImage.setTokens(EarTokens);
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_RECOGIMAGE"));
BackProp nNetwork = RecognitionPanel.this.projectEnv.getNetwork();
double[] resultID =
nNetwork.propagate(RecognitionPanel.this.actualImage.getTokens(nNetwork.numInput()));
ArrayList HumanEar = RecognitionPanel.this.projectEnv.getHumanEar();
double error = 0.0D;
for (int i = 0; i < HumanEar.size(); error = 0.0D)
{
HumanEar lEar = (HumanEar)HumanEar.get(i);
double[] ID = lEar.getID();
for (int j = 0; j < nNetwork.numOutput(); ++j)
{
if (ID[j] >= resultID[j])
{
error += ID[j] - resultID[j];
}
else error += resultID[j] - ID[j];
}
RecognitionPanel.this.recogTablePanel.addResult(lEar, 100.0D - (100 / HumanEar.size() * error));
++i;
}
RecognitionPanel.this.recogProgressBar.setValue(100);
RecognitionPanel.this.imageTokenField.setText("".concat(String.valueOf(RecognitionPanel.this.actualImage.
numTokens())));
RecognitionPanel.this.statusField.setForeground(Color.blue);
RecognitionPanel.this.statusField.setText(RecognitionPanel.this.getString("RECOGNIZED"));
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_FINISHED"));
}
});
this.resetButton = new JButton("Reset");
if (!(this.lrec.isApplet())) this.resetButton.setEnabled(false);
this.resetButton.setMargin(new Insets(0, 1, 0, 0));
this.resetButton.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
RecognitionPanel.this.recogImagePanel.setImage(RecognitionPanel.this.actualImage, 370, 400);
RecognitionPanel.this.actualImage.clearTokens();
RecognitionPanel.this.lrec.setStatusField(RecognitionPanel.this.getString("STATUS_IMGRESET"));
RecognitionPanel.this.imageTokenField.setText("0");
RecognitionPanel.this.statusField.setForeground(Color.red);
RecognitionPanel.this.statusField.setText(RecognitionPanel.this.getString("NOT_RECOGNIZED_YET"));
RecognitionPanel.this.recogProgressBar.setValue(0);
RecognitionPanel.this.recogTablePanel.clear();
}
});
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane, 0));
buttonPane.setBorder(BorderFactory.createEmptyBorder(2, 0, 5, 10));
buttonPane.add(loadButton);
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(this.recogButton);
```

```
buttonPane.add(Box.createRigidArea(new Dimension(3, 0)));
buttonPane.add(this.resetButton);
buttonPane.add(Box.createHorizontalGlue());
recogControlPanel.add(sliderPanel, "North");
recogControlPanel.add(this.recogProgressBar, "Center");
recogControlPanel.add(buttonPane, "South");
return recogControlPanel;
}
private JPanel createSliderPanel()
{
JPanel sliderPanel = new JPanel();
sliderPanel.setLayout(new BorderLayout());
JLabel thresholdSliderLabel = new JLabel(super.getString("RECOG_LB_THRESHOLD"));
JLabel distanceSliderLabel = new JLabel(super.getString("RECOG_LB_DISTANCE"));
JLabel minlineSliderLabel = new JLabel(super.getString("RECOG_LB_MINLINE"));
JPanel labelPane = new JPanel();
labelPane.setLayout(new GridLayout(0, 1));
labelPane.add(thresholdSliderLabel);
labelPane.add(distanceSliderLabel);
labelPane.add(minlineSliderLabel);
this.thresholdSlider = new JSlider(0, 0, 10, 5);
this.thresholdSlider.setMajorTickSpacing(1);
this.thresholdSlider.setSnapToTicks(true);
this.distanceSlider = new JSlider(0, 0, 10, 2);
this.distanceSlider.setMajorTickSpacing(1);
this.distanceSlider.setSnapToTicks(true);
this.minlineSlider = new JSlider(0, 0, 10, 2);
this.minlineSlider.setMajorTickSpacing(1);
this.minlineSlider.setSnapToTicks(true);
JPanel sliderGroup = new JPanel();
sliderGroup.setLayout(new GridLayout(0, 1));
sliderGroup.add(this.thresholdSlider);
sliderGroup.add(this.distanceSlider);
sliderGroup.add(this.minlineSlider);
sliderGroup.setPreferredSize(new Dimension(125, 0));
sliderPanel.add(labelPane, "West");
sliderPanel.add(sliderGroup, "Center");
return sliderPanel;
}
public void updateProjectEnv()
{
if ((this.projectEnv.getNetwork() != null) && (this.actualImage != null))
{
this.recogButton.setEnabled(true);
this.resetButton.setEnabled(true);
}
else
{
this.recogButton.setEnabled(false);
this.resetButton.setEnabled(false);
}
}
}
```

RecogTablePanel.java

```
package lrecog.gui;
import java.awt.Color;
import java.awt.Container;
import java.awt.GridLayout;
import java.util.ArrayList;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
```

```
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import lrecog.Recognition;
import lrecog.gfx.HumanEar;

public class RecogTablePanel extends DefaultPanel
{
private RecognitionPanel recogPanel;
private JTable table;
private RecogTableModel recogTableModel;
public RecogTablePanel(Recognition lrec)
{
super(lrec);
this.recogPanel = lrec.getRecogPanel();
super.setLayout(new GridLayout(1, 0));
DefaultTableCellRenderer colorRenderer = new DefaultTableCellRenderer()
{
public void setValue(Object value)
{
if (value instanceof RecogTablePanel.RecogResult)
{
RecogTablePanel.RecogResult rResult = (RecogTablePanel.RecogResult)value;
if (rResult.percentage >= 90.0D) super.setForeground(Color.red);
else if (rResult.percentage >= 80.0D) super.setForeground(Color.blue);
else if (rResult.percentage > 50.0D) super.setForeground(Color.green);
else super.setForeground(Color.black);
String tmpString = new String("".concat(String.valueOf(rResult.percentage)));
super.setText(String.valueOf(tmpString.substring(0, tmpString.indexOf(".") + 4)).concat(" %"));
}
else if (value instanceof HumanEar)
{
HumanEar lEar = (HumanEar)value;
super.setForeground(Color.black);
super.setText(lEar.getName());
}
else
{
super.setValue(value);
}
}
};
this.recogTableModel = new RecogTableModel();
this.table = new JTable(this.recogTableModel);
this.table.setRowSelectionAllowed(false);
this.table.setColumnSelectionAllowed(false);
this.table.setShowGrid(false);
Class cClass = this.table.getColumnClass(0);
this.table.setDefaultRenderer(cClass, colorRenderer);
JScrollPane scrollPane = new JScrollPane(this.table);
super.add(scrollPane);
}
public void addResult(HumanEar lEar, double perc)
{
this.recogTableModel.add(new RecogResult(lEar, perc));
}
public void clear()
{
this.recogTableModel.clear();
}
class RecogResult
{
protected HumanEar lEar;
protected double percentage;
public RecogResult(HumanEar lEar, double percent)
```

```
{
this.lEar = lEar;
this.percentage = percent;
}
}
class RecogTableModel extends AbstractTableModel
{
private final int COLUMS = 2;
private ArrayList resultList = new ArrayList();
public String getColumnName(int col)
{
if (col == 0) return RecogTablePanel.this.getString("RECOGTABLE_EAR");
if (col == 1) return "%";
return "ERROR";
}
public void add(RecogTablePanel.RecogResult result)
{
this.resultList.add(result);
super.fireTableStructureChanged();
}
public void clear()
{
this.resultList.clear();
super.fireTableStructureChanged();
}
public int getColumnCount()
{
return 2;
}
public int getRowCount()
{
return this.resultList.size();
}
public boolean isCellEditable(int row, int col)
{
return false;
}
public Object getValueAt(int row, int col)
{
RecogTablePanel.RecogResult rResult = (RecogTablePanel.RecogResult)this.resultList.get(row);
if (rResult != null)
{
if (col == 0) return rResult.lEar;
if (col == 1) return rResult;
}
return "ERROR";
}
}
}
```

ImageProcessor.java

```
package lrecog.gfx;
import java.awt.Component;
import java.awt.Image;
import java.awt.MediaTracker;
import java.awt.Panel;
import java.awt.image.MemoryImageSource;
import java.awt.image.PixelGrabber;
import java.io.PrintStream;
import java.util.ArrayList;
public class ImageProcessor extends Panel
{
private int[] Pixels;
```

```
private ArrayList alltokens = null;
private int width;
private int height;
public static final int PIXEL_MASK = 255;
private static final int COLOR_BACKGROUND = -1;
private static final int COLOR_FOREGROUND = -16777216;
private static final int COLOR_SKELETON = -16776961;
private static final int COLOR_REMOVABLE = -16711681;
private static final int COLOR_GOODLINE = -16711936;
private static final int COLOR_BADLINE = -65536;
private static final int COLOR_DONE = -512;
private static final int COLOR_POINT_MARK = -65281;
private static final int COLOR_POINT_DONE = -256;
private static int LINE_MIN = 20;
private static int POINT_DIFF = 20;
private static long points_pos;
private static int root_x;
private static int root_y;

public ImageProcessor(int _w, int _h)
{
this.width = _w;
this.height = _h;
this.Pixels = new int[this.width * this.height];
}
public ImageProcessor(Image myImage)
{
this.width = myImage.getWidth(null);
this.height = myImage.getHeight(null);
this.Pixels = new int[this.width * this.height];
PixelGrabber myPixelGrabber = new PixelGrabber(myImage, 0, 0, this.width, this.height, this.Pixels, 0,
this.width);
try
{
myPixelGrabber.grabPixels();
}
catch (Exception e)
{
System.out.println(e);
}
}
public Image getImage(Component theComponent)
{
MediaTracker myMediaTracker = new MediaTracker(theComponent);
Image myImage = super.createImage(new MemoryImageSource(this.width, this.height, this.Pixels, 0,
this.width));
myMediaTracker.addImage(myImage, 0);
try
{
myMediaTracker.waitForAll();
}
catch (Exception e)
{
System.out.println(e);
}
return myImage;
}
public void Clear()
{
for (int i = 0; i < this.height * this.width; ++i)
{
this.Pixels[i] = -1;
}
}
```

```
public int getWidth()
{
return this.width;
}
public int getHeight()
{
return this.height;
}
public void setPixel(int x, int y, int color)
{
if ((x < 0) || (y < 0) || (x >= this.width) || (y >= this.height))
{
return;
}
this.Pixels[(y * this.width + x)] = color;
}
public void setGrayPixel(int x, int y, int gray)
{
if ((x < 0) || (y < 0) || (x >= this.width) || (y >= this.height))
{
return;
}
if (gray < 0) gray = 0;
if (gray > 255) gray = 255;
this.Pixels[(y * this.width + x)] = (0xFF000000 | gray | gray << 8 | gray << 16);
}
public int getPixel(int x, int y)
{
if (x < 0) x = 0;
if (y < 0) y = 0;
if (x >= this.width) x = this.width - 1;
if (y >= this.height) y = this.height - 1;
return this.Pixels[(y * this.width + x)];
}
public int getGrayPixel(int x, int y)
{
if (x < 0) x = 0;
if (y < 0) y = 0;
if (x >= this.width) x = this.width - 1;
if (y >= this.height) y = this.height - 1;
return ((this.Pixels[(y * this.width + x)] & 0xFF) / 3 + (this.Pixels[(y * this.width + x)] >> 8 & 0xFF) / 3 +
(this.Pixels[(y * this.width + x)] >> 16 & 0xFF) / 3);
}
public void drawCircle(int x, int y, int r, int color)
{
for (int i = 0; i < 360; i += 5)
{
setPixel((int)Math.round(x + Math.sin(i) * r), (int)Math.round(y + Math.cos(i) * r), color);
}
}
public void drawSquare(int x, int y, int r, int color)
{
for (int i = x - (r / 2); i <= x + r / 2; ++i)
{
setPixel(i, y - (r / 2), color);
setPixel(i, y + r / 2, color);
}
for (i = y - (r / 2); i <= y + r / 2; ++i)
{
setPixel(x - (r / 2), i, color);
setPixel(x + r / 2, i, color);
}
}
void drawLine(int x0, int y0, int x1, int y1, int color)
```

```
{
int ex = x1 - x0;
int ey = y1 - y0;
int dx;
if (ex > 0)
{
dx = 1;
}
else if (ex < 0)
{
dx = -1;
ex = -ex;
} else {
dx = 0;
}
int dy;
if (ey > 0)
{
dy = 1;
}
else if (ey < 0)
{
dy = -1;
ey = -ey;
} else {
dy = 0;
}
int x = x0; int y = y0;
if (ex > ey)
{
height = 2 * ey - ex;
while (true) { if (x == x1)
return;
if (height >= 0)
{
height -= 2 * ex;
y += dy;
}
height += 2 * ey;
x += dx;
setPixel(x, y, color);
}
}
int height = 2 * ex - ey;
while (y != y1)
{
if (height >= 0)
{
height -= 2 * ey;
x += dx;
}
height += 2 * ex;
y += dy;
setPixel(x, y, color);
}
}
public void edgeDetect(int threshold)
{
int max = 0; int min = 0;
ImageProcessor Source = new ImageProcessor(this.width, this.height);
Source.Clear();
int y;
for (int x = this.width - 2; x > 0; --x)
{
```

```
for (y = this.height - 2; y > 0; --y)
{
int dx = getGrayPixel(x - 1, y + 1) + getGrayPixel(x, y + 1) + getGrayPixel(x + 1, y + 1) - (getGrayPixel(x -
1, y - 1) + getGrayPixel(x, y - 1) + getGrayPixel(x + 1, y - 1));
int dy = getGrayPixel(x + 1, y - 1) + getGrayPixel(x + 1, y) + getGrayPixel(x + 1, y + 1) - (getGrayPixel(x -
1, y - 1) + getGrayPixel(x - 1, y) + getGrayPixel(x - 1, y + 1));
int z = (int)(Math.sqrt(dx * dx + dy * dy) / 3);
max = (z > max) ? z : max;
min = (z < min) ? z : min;
Source.setPixel(x, y, z);
}
}
float a = 255.0F / (max - min);
float b = a * min;
for (x = this.width - 1; x >= 0; --x)
{
for (y = this.height - 1; y >= 0; --y)
{
setPixel(x, y, (a * Source.getPixel(x, y) + b > threshold) ? -16777216 : -1);
}
}
}
public void thinning()
{
boolean remain = true;
if (!(remain))
return;
remain = false;
for (int j = 0; ; j += 2) { if (j <= 6);
int y;
for (int x = 0; x < this.width; ++x)
{
for (y = 0; y < this.height; ++y)
{
if ((getPixel(x, y) != -16777216) || (this.Pixels[neighbour(x, y, j)] != -1))
continue;
if (matchPatterns(x, y))
{
setPixel(x, y, -16776961);
}
else
{
setPixel(x, y, -16711681);
remain = true;
}
}
}
for (x = 0; x < this.width; ++x)
{
for (y = 0; y < this.height; ++y)
{
if (getPixel(x, y) != -16711681)
continue;
setPixel(x, y, -1);
}
}
}
}
private int neighbour(int x, int y, int j)
{
switch (j)
{
case 0:
++x; break;
```

```
case 1:
++x; --y; break;
case 2:
--y; break;
case 3:
--x; --y; break;
case 4:
--x; break;
case 5:
--x; ++y; break;
case 6:
++y; break;
case 7:
++x; ++y;
}
if (x >= this.width - 1) x = this.width - 1;
if (x < 0) x = 0;
if (y >= this.height - 1) y = this.height - 1;
if (y < 0) y = 0;
return (y * this.width + x);
}
private boolean matchPatterns(int x, int y)
{
if (x >= this.width - 1) x = this.width - 1;
if (x < 0) x = 0;
if (y >= this.height - 1) y = this.height - 1;
if (y < 0) y = 0;
if ((this.Pixels[neighbour(x, y, 0)] == -1) && (this.Pixels[neighbour(x, y, 4)] == -1) &&
(((this.Pixels[neighbour(x, y, 1)] != -1) || (this.Pixels[neighbour(x, y, 2)] != -1) || (this.Pixels[neighbour(x, y,
3)] != -1))) && (((this.Pixels[neighbour(x, y, 5)] != -1) || (this.Pixels[neighbour(x, y, 6)] != -1) ||
(this.Pixels[neighbour(x, y, 7)] != -1))))
return true;
}
if ((this.Pixels[neighbour(x, y, 2)] == -1) && (this.Pixels[neighbour(x, y, 6)] == -1) &&
(((this.Pixels[neighbour(x, y, 7)] != -1) || (this.Pixels[neighbour(x, y, 0)] != -1) || (this.Pixels[neighbour(x, y,
1)] != -1))) && (((this.Pixels[neighbour(x, y, 3)] != -1) || (this.Pixels[neighbour(x, y, 4)] != -1) ||
(this.Pixels[neighbour(x, y, 5)] != -1))))
{
return true;
}
if ((this.Pixels[neighbour(x, y, 7)] == -16776961) && (this.Pixels[neighbour(x, y, 0)] == -1) &&
(this.Pixels[neighbour(x, y, 6)] == -1) && (((this.Pixels[neighbour(x, y, 1)] != -1) || (this.Pixels[neighbour(x,
y, 2)] != -1) || (this.Pixels[neighbour(x, y, 3)] != -1) || (this.Pixels[neighbour(x, y, 4)] != -1) ||
(this.Pixels[neighbour(x, y, 5)] != -1))))
{
return true;
}
if ((this.Pixels[neighbour(x, y, 5)] == -16776961) && (this.Pixels[neighbour(x, y, 4)] == -1) &&
(this.Pixels[neighbour(x, y, 6)] == -1) && (((this.Pixels[neighbour(x, y, 7)] != -1) || (this.Pixels[neighbour(x,
y, 0)] != -1) || (this.Pixels[neighbour(x, y, 1)] != -1) || (this.Pixels[neighbour(x, y, 2)] != -1) ||
(this.Pixels[neighbour(x, y, 3)] != -1))))
{
return true;
}
if ((this.Pixels[neighbour(x, y, 3)] == -16776961) && (this.Pixels[neighbour(x, y, 2)] == -1) &&
(this.Pixels[neighbour(x, y, 4)] == -1) && (((this.Pixels[neighbour(x, y, 5)] != -1) || (this.Pixels[neighbour(x,
y, 6)] != -1) || (this.Pixels[neighbour(x, y, 7)] != -1) || (this.Pixels[neighbour(x, y, 0)] != -1) ||
(this.Pixels[neighbour(x, y, 1)] != -1))))
{
return true;
}
return ((this.Pixels[neighbour(x, y, 1)] == -16776961) && (this.Pixels[neighbour(x, y, 0)] == -1) &&
(this.Pixels[neighbour(x, y, 2)] == -1) && (((this.Pixels[neighbour(x, y, 3)] != -1) || (this.Pixels[neighbour(x,
```

```
y, 4)] != -1) || (this.Pixels[neighbour(x, y, 5)] != -1) || (this.Pixels[neighbour(x, y, 6)] != -1) ||
(this.Pixels[neighbour(x, y, 7)] != -1))));
}
public void checkLines(int minline)
{
int length = 0;
LINE_MIN = minline;
for (int y = 0; y < this.height; ++y)
{
for (int x = 0; x < this.width; ++x)
{
if (getPixel(x, y) != -16776961)
continue;
length = checkLineLength(x, y, 1);
if (length > LINE_MIN)
{
paintLines(x, y, -16711936);
}
else
{
paintLines(x, y, -65536);
}
length = 0;
}
}
}
private int checkLineLength(int x, int y, int length)
{
setPixel(x, y, -512);
try
{
if (this.Pixels[neighbour(x, y, 0)] == -16776961)
length = checkLineLength(x + 1, y, length + 1);
if (this.Pixels[neighbour(x, y, 1)] == -16776961)
length = checkLineLength(x + 1, y - 1, length + 1);
if (this.Pixels[neighbour(x, y, 2)] == -16776961)
length = checkLineLength(x, y - 1, length + 1);
if (this.Pixels[neighbour(x, y, 3)] == -16776961)
length = checkLineLength(x - 1, y - 1, length + 1);
if (this.Pixels[neighbour(x, y, 4)] == -16776961)
length = checkLineLength(x - 1, y, length + 1);
if (this.Pixels[neighbour(x, y, 5)] == -16776961)
length = checkLineLength(x - 1, y + 1, length + 1);
if (this.Pixels[neighbour(x, y, 6)] == -16776961)
length = checkLineLength(x, y + 1, length + 1);
if (this.Pixels[neighbour(x, y, 7)] == -16776961)
length = checkLineLength(x + 1, y + 1, length + 1);
}
catch (Exception e)
{
System.out.println("Maybe Stackoverflow!!\n".concat(String.valueOf(e)));
}
return length;
}
private void paintLines(int x, int y, int color)
{
setPixel(x, y, color);
try
{
if (this.Pixels[neighbour(x, y, 0)] == -512) paintLines(x + 1, y, color);
if (this.Pixels[neighbour(x, y, 1)] == -512) paintLines(x + 1, y - 1, color);
if (this.Pixels[neighbour(x, y, 2)] == -512) paintLines(x, y - 1, color);
if (this.Pixels[neighbour(x, y, 3)] == -512) paintLines(x - 1, y - 1, color);
if (this.Pixels[neighbour(x, y, 4)] == -512) paintLines(x - 1, y, color);
```

```
if (this.Pixels[neighbour(x, y, 5)] == -512) paintLines(x - 1, y + 1, color);
if (this.Pixels[neighbour(x, y, 6)] == -512) paintLines(x, y + 1, color);
if (this.Pixels[neighbour(x, y, 7)] != -512) return; paintLines(x + 1, y + 1, color);
}
catch (Exception e)
{
System.out.println("Maybe Stackoverflow\n".concat(String.valueOf(e)));
}
}
public void markPoints(int distance)
{
int length = 0;
POINT_DIFF = distance;
points_pos = POINT_DIFF - 1;
for (int y = 0; y < this.height; ++y)
{
for (int x = 0; x < this.width; ++x)
{
if (getPixel(x, y) != -16711936)
continue;
paintPoints(x, y);
}
}
}
private boolean paintPoints(int x, int y)
{
boolean result = false;
points_pos += 1L;
if (points_pos == POINT_DIFF)
{
setPixel(x, y, -65281);
points_pos = 0L;
} else {
setPixel(x, y, -256);
}
try
{
if (this.Pixels[neighbour(x, y, 0)] == -16711936) result = paintPoints(x + 1, y);
if (this.Pixels[neighbour(x, y, 1)] == -16711936) result = paintPoints(x + 1, y - 1);
if (this.Pixels[neighbour(x, y, 2)] == -16711936) result = paintPoints(x, y - 1);
if (this.Pixels[neighbour(x, y, 3)] == -16711936) result = paintPoints(x - 1, y - 1);
if (this.Pixels[neighbour(x, y, 4)] == -16711936) result = paintPoints(x - 1, y);
if (this.Pixels[neighbour(x, y, 5)] == -16711936) result = paintPoints(x - 1, y + 1);
if (this.Pixels[neighbour(x, y, 6)] == -16711936) result = paintPoints(x, y + 1);
if (this.Pixels[neighbour(x, y, 7)] == -16711936) result = paintPoints(x + 1, y + 1);
}
catch (Exception e)
{
System.out.println("Maybe Stackoverflow\n".concat(String.valueOf(e)));
}
if (result == false) setPixel(x, y, -65281);
return true;
}
public void calcAngels()
{
int length = 0;
root_x = ImageProcessor.root_y = -1;
this.alltokens = new ArrayList();
for (int y = 0; y < this.height; ++y)
{
for (int x = 0; x < this.width; ++x)
{
if (getPixel(x, y) != -65281)
continue;
```

```
searchNeighbour(x, y, true);
}
}
for (int i = 0; i < this.alltokens.size(); ++i)
{
EarToken actToken = (EarToken)this.alltokens.get(i);
drawSquare(actToken.getX1(), actToken.getY1(), 5, -65536);
}
}
private void searchNeighbour(int x, int y, boolean isRoot)
{
if (getPixel(x, y) == -65281)
{
if (isRoot == true)
{
root_x = x;
root_y = y;
}
else if ((isRoot == false) && (root_x != -1) && (root_y != -1))
{
if (root_x + root_y - (x + y) == 0) return;
drawLine(root_x, root_y, x, y, -16776961);
EarToken ltoken = new EarToken(root_x, root_y, x, y);
this.alltokens.add(ltoken);
setPixel(root_x, root_y, -65281);
setPixel(x, y, -65281);
return;
}
}
else setPixel(x, y, -16711936);
try
{
if ((this.Pixels[neighbour(x, y, 0)] == -256) || (this.Pixels[neighbour(x, y, 0)] == -65281))
searchNeighbour(x + 1, y, false);
if ((this.Pixels[neighbour(x, y, 1)] == -256) || (this.Pixels[neighbour(x, y, 1)] == -65281))
searchNeighbour(x + 1, y - 1, false);
if ((this.Pixels[neighbour(x, y, 2)] == -256) || (this.Pixels[neighbour(x, y, 2)] == -65281))
searchNeighbour(x, y - 1, false);
if ((this.Pixels[neighbour(x, y, 3)] == -256) || (this.Pixels[neighbour(x, y, 3)] == -65281))
searchNeighbour(x - 1, y - 1, false);
if ((this.Pixels[neighbour(x, y, 4)] == -256) || (this.Pixels[neighbour(x, y, 4)] == -65281))
searchNeighbour(x - 1, y, false);
if ((this.Pixels[neighbour(x, y, 5)] == -256) || (this.Pixels[neighbour(x, y, 5)] == -65281))
searchNeighbour(x - 1, y + 1, false);
if ((this.Pixels[neighbour(x, y, 6)] == -256) || (this.Pixels[neighbour(x, y, 6)] == -65281))
searchNeighbour(x, y + 1, false);
if ((this.Pixels[neighbour(x, y, 7)] == -256) || (this.Pixels[neighbour(x, y, 7)] == -65281))
searchNeighbour(x + 1, y + 1, false);
}
catch (Exception e)
{
System.out.println("Maybe Stackoverflow\n".concat(String.valueOf(e)));
}
if (isRoot != true) return; setPixel(x, y, -16776961);
}
public ArrayList getTokens()
{
return this.alltokens;
}
}
```

EarImage.java

```
package lrecog.gfx;
import java.awt.Image;
import java.io.File;
import java.net.URL;
import java.util.ArrayList;
import javax.swing.ImageIcon;

public class EarImage
{
private Image image;
private ArrayList tokens;
private File filename;
private HumanEar ear;
public EarImage(File fileopen)
{
this.image = null;
this.tokens = null;
this.filename = null;
this.ear = null;
if (fileopen.isFile())
{
this.filename = fileopen;
this.image = new ImageIcon(fileopen.getPath()).getImage();
}
this.tokens = new ArrayList();
}
public EarImage(URL urlopen)
{
this.image = null;
this.tokens = null;
this.filename = null;
this.ear = null;
this.filename = new File(urlopen.toString());
this.image = new ImageIcon(urlopen).getImage();
this.tokens = new ArrayList();
}
public EarImage(Image img)
{
this.image = null;
this.tokens = null;
this.filename = null;
this.ear = null;
this.image = img;
this.tokens = new ArrayList();
}
public EarImage(Image img, String filename)
{
this(img);
this.filename = new File(filename);
}
public void setImage(Image img)
{
this.image = img;
}
public void setEar(HumanEar ear)
{
this.ear = ear;
}
public HumanEar getEar()
{
return this.ear;
}
```

```
public Image getImage()
{
return this.image;
}
public int getWidth()
{
return this.image.getWidth(null);
}
public int getHeight()
{
return this.image.getHeight(null);
}
public void addToken(EarToken token)
{
this.tokens.add(token);
}
public void setTokens(ArrayList tokens)
{
this.tokens = tokens;
}
public double[] getTokens(int inputs)
{
double[] tokenVector = new double[inputs];
for (int i = 0; i < inputs; ++i)
{
if (i < this.tokens.size())
{
EarToken token = (EarToken)this.tokens.get(i);
if (i % 2 == 0) tokenVector[i] = token.getCOS();
else tokenVector[i] = token.getSIN();
}
else {
tokenVector[i] = -1.0D;
}
}
return tokenVector;
}
public double[] getTokens()
{
return getTokens(this.tokens.size() * 2);
}
public EarToken getToken(int idx)
{
return ((EarToken)this.tokens.get(idx));
}
public int numTokens()
{
return this.tokens.size();
}
public void clearTokens()
{
this.tokens.clear();
}
public File getFileName()
{
return this.filename;
}
public String toString()
{
return this.filename.getName();
}
}
```

HumanEar.java

```
package lrecog.gfx;
import java.util.AbstractCollection;
import java.util.ArrayList;
public class HumanEar
{
private String earName;
private ArrayList images = null;
private double[] ID;
public HumanEar(String name)
{
this.earName = name;
this.images = new ArrayList();
this.ID = new double[0];
}
public void setName(String name)
{
this.earName = name;
}
public String getName()
{
return this.earName;
}
public void addImage(EarImage limage)
{
this.images.add(limage);
}
public void removeImage(EarImage limage)
{
this.images.remove(limage);
}
public EarImage getImage(int idx)
{
return ((EarImage)this.images.get(idx));
}
public int numImages()
{
return this.images.size();
}
public void setID(double[] id)
{
this.ID = id;
}
public double[] getID()
{
return this.ID;
}
public String toString()
{
return String.valueOf(new StringBuffer(String.valueOf(this.earName)).append("
(").append(this.images.size()).append(")"));
}
}
```

EarToken.java

```
package lrecog.gfx;
public class EarToken
{
private int x1;
private int y1;
private int x2;
private int y2;
```

```
private double cos;
private double sin;
public EarToken(int x1, int y1, int x2, int y2)
{
this.x1 = x1;
this.y1 = y1;
this.x2 = x2;
this.y2 = y2;
calcCosinus();
calcSinus();
}
private void calcCosinus()
{
int ax = this.x2 - this.x1;
int ay = this.y2 - this.y1;
double hyp = Math.sqrt(ax * ax + ay * ay);
if (hyp == 0.0D) this.cos = 0.0D;
else this.cos = (ay / hyp);
}
private void calcSinus()
{
int ax = this.x2 - this.x1;
int ay = this.y2 - this.y1;
double hyp = Math.sqrt(ax * ax + ay * ay);
if (hyp == 0.0D) hyp = Math.abs(ax);
this.sin = (ax / hyp);
}
public int getX1()
{
return this.x1;
}
public int getY1()
{
return this.y1;
}
public int getX2()
{
return this.x2;
}
public int getY2()
{
return this.y2;
}
public double getCOS()
{
return this.cos;
}
public double getSIN()
{
return this.sin;
}
}
```

BackProp.java

```
package lrecog.nnetwork;
import java.util.Random;
public class BackProp
{
private double[] inputA;
private double[] hiddenA;
private double[] hiddenN;
private double[] hiddenD;
private double[][] hiddenW;
```

```
private double[] outputA;
private double[] outputN;
private double[] outputD;
private double[] oldD;
private double[][] outputW;
private double[] biasH;
private double[] biasO;
private int numInput;
private int numHidden;
private int numOutput;
private int epoch;
private double momentum;
private double alpha;
private double absError;
private Random rand;

public BackProp(int input, int hidden, int output)
{
this.absError = 0.0D;
this.numInput = input;
this.numHidden = hidden;
this.numOutput = output;
this.inputA = new double[this.numInput];
this.hiddenW = new double[this.numHidden][this.numInput];
this.hiddenA = new double[this.numHidden];
this.hiddenN = new double[this.numHidden];
this.hiddenD = new double[this.numHidden];
this.biasH = new double[this.numHidden];
this.outputW = new double[this.numOutput][this.numHidden];
this.outputA = new double[this.numOutput];
this.outputN = new double[this.numOutput];
this.outputD = new double[this.numOutput];
this.oldD = new double[this.numOutput];
this.biasO = new double[this.numOutput];
this.alpha = 0.3D;
this.momentum = 1.0D;
this.rand = new Random();
init();
}
public BackProp(int input, int hidden, int output, double alpha, double mom)
{
this(input, hidden, output);
this.alpha = alpha;
this.momentum = mom;
}
private void init()
{
this.epoch = 0;
for (int i = 0; i < this.numInput; ++i)
{
this.inputA[i] = frandom(-1.0D, 1.0D);
}
int m;
for (i = 0; i < this.numHidden; ++i)
{
this.hiddenA[i] = frandom(-1.0D, 1.0D);
this.biasH[i] = frandom(-1.0D, 1.0D);
for (m = 0; m < this.numInput; ++m)
{
this.hiddenW[i][m] = frandom(-1.0D, 1.0D);
}
}
for (i = 0; i < this.numOutput; ++i)
{
```

```
this.biasO[i] = frandom(-1.0D, 1.0D);
for (m = 0; m < this.numHidden; ++m)
{
this.outputW[i][m] = frandom(-1.0D, 1.0D);
}
}
}
private double sigmoid(double x)
{
return (1.0D / (1.0D + Math.exp(-x)));
}
private double sigmoidDeriv(double x)
{
return (sigmoid(x) * (1 - sigmoid(x)));
}
private void feedForward()
{
double sum2 = 0.0D;
int j;
for (int i = 0; i < this.numHidden; ++i)
{
sum2 = this.biasH[i];
for (j = 0; j < this.numInput; ++j)
{
sum2 += this.hiddenW[i][j] * this.inputA[j];
}
this.hiddenN[i] = sum2;
this.hiddenA[i] = sigmoid(sum2);
}
for (i = 0; i < this.numOutput; ++i)
{
sum2 = this.biasO[i];
for (j = 0; j < this.numHidden; ++j)
{
sum2 += this.outputW[i][j] * this.hiddenA[j];
}
this.outputN[i] = sum2;
}
}
private void computeDelta(int m)
{
this.outputD[m] = ((this.outputA[m] - sigmoid(this.outputN[m])) * sigmoidDeriv(this.outputN[m]));
for (int i = 0; i < this.numHidden; ++i)
{
this.outputW[m][i] += this.outputD[m] * this.hiddenA[i] * this.alpha;
}
for (i = 0; i < this.numOutput; ++i)
{
this.biasO[i] += this.outputD[m] * this.alpha;
}
}
private void updateWeights()
{
for (int j = 0; j < this.numHidden; ++j)
{
double sum2 = 0.0D;
for (int i = 0; i < this.numOutput; ++i)
{
sum2 += this.outputD[i] * this.outputW[i][j];
}
sum2 *= sigmoidDeriv(this.hiddenN[j]);
this.biasH[j] += sum2 * this.alpha;
for (i = 0; i < this.numInput; ++i)
{
```

```
this.hiddenW[j][i] += this.alpha * sum2 * this.inputA[i];
}
}
}
private double frandom(double min, double max)
{
return (this.rand.nextDouble() * (max - min) + min);
}
public double[] propagate(double[] vector)
{
for (int i = 0; i < this.numInput; ++i)
{
this.inputA[i] = vector[i];
}
double sum2;
int j;
for (i = 0; i < this.numHidden; ++i)
{
sum2 = this.biasH[i];
for (j = 0; j < this.numInput; ++j)
{
sum2 += this.hiddenW[i][j] * this.inputA[j];
this.hiddenA[i] = sigmoid(sum2);
}
}
for (i = 0; i < this.numOutput; ++i)
{
sum2 = this.biasO[i];
for (j = 0; j < this.numHidden; ++j)
{
sum2 += this.outputW[i][j] * this.hiddenA[j];
}
this.outputN[i] = sum2;
this.outputA[i] = sigmoid(sum2);
}
return this.outputA;
}
public double learnVector(double[] in, double[] out)
{
for (int i = 0; i < this.numInput; ++i)
{
this.inputA[i] = in[i];
}
for (i = 0; i < this.numOutput; ++i)
{
this.outputA[i] = out[i];
}
feedForward();
this.absError = 0.0D;
for (int j = 0; j < this.numOutput; ++j)
{
computeDelta(j);
this.absError += Math.pow(this.outputA[j] - sigmoid(this.outputN[j]), 2.0D);
}
updateWeights();
this.alpha *= this.momentum;
return this.absError;
}
public int numInput()
{
return this.numInput;
}
public int numHidden()
{
```

```
return this.numHidden;
}
public int numOutput()
{
return this.numOutput;
}
public double[] getBiasH()
{
return this.biasH;
}
public double[] getBiasO()
{
return this.biasO;
}
public double[][] getHiddenW()
{
return this.hiddenW;
}
public double[][] getOutputW()
{
return this.outputW;
}
public double getAlpha()
{
return this.alpha;
}
public double getMomentum()
{
return this.momentum;
}
public int getEpoch()
{
return this.epoch;
}
public void setAlpha(double a)
{
this.alpha = a;
}
public void setMomentum(double mom)
{
this.momentum = mom;
}
public void setHiddenW(double weight, int hidden, int input)
{
this.hiddenW[hidden][input] = weight;
}
public void setBiasH(double bias, int hidden)
{
this.biasH[hidden] = bias;
}
public void setOutputW(double weight, int output, int hidden)
{
this.outputW[output][hidden] = weight;
}
public void setBiasO(double bias, int output)
{
this.biasO[output] = bias;
}
}
```

ImageChooser.java

```
package lrecog.tools;
import java.awt.Image;
```

```
import java.io.PrintStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;
import java.util.Vector;
import javax.swing.ImageIcon;
import lrecog.Recognition;
import lrecog.gfx.EarImage;
import lrecog.gui.FileOpenDialog;

public class ImageChooser
{
private Recognition lrec;
private ProjectEnv projectEnv;
private Vector imageVector;
private Hashtable imageTable;
private Image img;
private FileOpenDialog fileOpenDialog;
private String first;
public ImageChooser(Recognition lrec)
{
this.lrec = lrec;
this.projectEnv = lrec.getProjectEnv();
Vector images = this.projectEnv.getImageVector();
String imageDir = this.projectEnv.getImageDir();
URL codebase = this.projectEnv.getCodeBase();
this.imageVector = new Vector();
this.imageTable = new Hashtable();
SortedSet filetypes = new TreeSet();
for (int i = 0; i < images.size(); ++i)
{
String name = (String)images.get(i);
String path = String.valueOf(imageDir).concat(String.valueOf(name));
try
{
Image img = new ImageIcon(new URL(codebase, path)).getImage();
if (img.getWidth(null) != -1)
{
this.imageTable.put(name, img);
int index = name.lastIndexOf(".");
String type = name.substring(index);
filetypes.add(type); break label247:
}
label247: System.out.println(String.valueOf(new StringBuffer("Image ").append(name).append(" couldn't be
loaded...")));
}
catch (MalformedURLException m)
{
System.out.println(String.valueOf(new StringBuffer("URL for the image ").append(name).append(" badly
specified")));
}
}
Vector[] files = new Vector[filetypes.size()];
Hashtable content = new Hashtable();
Set s = this.imageTable.keySet();
TreeSet set = new TreeSet(s);
int i = 0;
for (Iterator j = filetypes.iterator(); j.hasNext(); ++i)
{
files[i] = new Vector();
```

```
String a = (String)j.next();
for (Iterator z = set.iterator(); z.hasNext(); )
{
String name = (String)z.next();
int index = name.lastIndexOf(".");
String type = name.substring(index);
if (!(a.equalsIgnoreCase(type))) continue; files[i].add(name);
}
if (a.equalsIgnoreCase(".gif")) content.put("GIF Image Files (*.gif)", files[i]);
else if (a.equalsIgnoreCase(".jpg")) content.put("JPEG Image Files (*.jpg)", files[i]);
else content.put(a, files[i]);
}
if (content.size() <= 0)
return;
this.fileOpenDialog = new FileOpenDialog(content, lrec.getFrame(), true);
this.first = this.fileOpenDialog.getFirstItem();
this.img = ((Image)this.imageTable.get(this.first));
}
public EarImage open()
{
if (this.fileOpenDialog == null) return null;
boolean b = this.fileOpenDialog.showDialog();
if (b == FileOpenDialog.APPROVE_OPTION)
{
String s = this.fileOpenDialog.getSelectedItem();
this.img = ((Image)this.imageTable.get(s));
return new EarImage(this.img, s);
}
return null;
}
public EarImage getEarImage(String imageName)
{
this.img = ((Image)this.imageTable.get(imageName));
if (this.img != null) return new EarImage(this.img, imageName);
return null;
}
public String getFirstImageName()
{
return this.first;
}
}
```

ImageFilter.java

```
package lrecog.tools;
import java.io.File;
import javax.swing.filechooser.FileFilter;

public class ImageFilter extends FileFilter
{
public boolean accept(File f)
{
if (f.isDirectory())
{
return true;
}
String extension = Utils.getExtension(f);
if (extension != null)
{
return ((extension.equals("tiff")) || (extension.equals("tif")) || (extension.equals("gif")) ||
(extension.equals("jpeg")) || (extension.equals("jpg")));
}
return false;
}
```

```
public String getDescription()
{
return "Images (*.jpg/*.gif/*.tiff)";
}
}
```

ImagePreview.java

```
package lrecog.tools;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.File;
import javax.swing.ImageIcon;
import javax.swing.JComponent;
import javax.swing.JFileChooser;

public class ImagePreview extends JComponent
implements PropertyChangeListener
{
ImageIcon thumbnail = null;
File file = null;
public ImagePreview(JFileChooser fc)
{
super.setPreferredSize(new Dimension(100, 50));
fc.addPropertyChangeListener(this);
}
public void loadImage()
{
if (this.file == null)
{
return;
}
ImageIcon tmpIcon = new ImageIcon(this.file.getPath());
if (tmpIcon.getIconWidth() > 90)
{
this.thumbnail = new ImageIcon(tmpIcon.getImage().getScaledInstance(90, -1, 1));
}
else
{
this.thumbnail = tmpIcon;
}
}
public void propertyChange(PropertyChangeEvent e)
{
String prop = e.getPropertyName();
if (!(prop.equals("SelectedFileChangedProperty")))
return;
this.file = ((File)e.getNewValue());
if (!(super.isShowing()))
return;
loadImage();
super.repaint();
}
public void paintComponent(Graphics g)
{
if (this.thumbnail == null)
{
loadImage();
}
if (this.thumbnail == null)
```

```
return;
int x = super.getWidth() / 2 - (this.thumbnail.getIconWidth() / 2);
int y = super.getHeight() / 2 - (this.thumbnail.getIconHeight() / 2);
if (y < 0)
{
y = 0;
}
if (x < 5)
{
x = 5;
}
this.thumbnail.paintIcon(this, g, x, y);
}
}
```

ProjectEnv.java

```
package lrecog.tools;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintStream;
import java.net.URL;
import java.util.AbstractCollection;
import java.util.ArrayList;
import java.util.Vector;
import lrecog.Recognition;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;
import lrecog.gfx.EarToken;
import lrecog.nnetwork.BackProp;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.AttributesImpl;
import org.xml.sax.helpers.XMLReaderFactory;

public class ProjectEnv
{
private boolean modified = false;
private File configfile = null;
private Recognition lrec = null;
private URL codeBase = null;
private Vector imageVector = null;
private String imageDir = null;
private ArrayList HumanEar;
private BackProp nNetwork = null;
private int numInputNeurons;
private int numHiddenNeurons;
private int numOutputNeurons;
public ProjectEnv(Recognition lrec)
{
this.lrec = lrec;
this.HumanEar = new ArrayList();
}
public void addHumanEar(HumanEar lEar)
{
this.HumanEar.add(lEar);
}
public void removeHumanEar(HumanEar lEar)
```

```
{
this.HumanEar.remove(lEar);
}
public ArrayList getHumanEar()
{
return this.HumanEar;
}
public Recognition getRecognition()
{
return this.lrec;
}
public int numHumanEar()
{
return this.HumanEar.size();
}
public int numEarImages()
{
int images = 0;
for (int i = 0; i < this.HumanEar.size(); ++i)
{
HumanEar lEar = (HumanEar)this.HumanEar.get(i);
images += lEar.numImages();
}
return images;
}
public void setNetwork(BackProp nNetwork)
{
this.nNetwork = nNetwork;
}
public void setImageVector(Vector images)
{
this.imageVector = images;
}
public void setImageDir(String imageDir)
{
this.imageDir = imageDir;
}
public void setModified()
{
this.modified = true;
}
public void setCodeBase(URL cbase)
{
this.codeBase = cbase;
}
public URL getCodeBase()
{
return this.codeBase;
}
public boolean wasModified()
{
return this.modified;
}
public BackProp getNetwork()
{
return this.nNetwork;
}
public File getFileName()
{
return this.configfile;
}
public Vector getImageVector()
{
return this.imageVector;
```

```
}
public String getImageDir()
{
return this.imageDir;
}
public boolean Open(File fileopen)
{
XMLReader parser = null;
XMLCfgReader cfgReader = new XMLCfgReader(this);
try
{
parser = XMLReaderFactory.createXMLReader();
}
catch (Exception e)
{
try
{
parser = XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser");
}
catch (Exception ee)
{
System.err.println("Couldn't locate a SAX parser:");
System.err.println(ee);
return 0;
}
}
parser.setContentHandler(cfgReader);
try
{
InputStream fIn = new FileInputStream(fileopen);
parser.parse(new InputSource(fIn));
}
catch (SAXParseException e)
{
System.out.println(String.valueOf(fileopen).concat(" is not well formed."));
System.out.println(String.valueOf(new StringBuffer(String.valueOf(e.getMessage())).append(" at line
").append(e.getLineNumber()).append(", column ").append(e.getColumnNumber())));
}
catch (SAXException e)
{
System.out.println(e.getMessage());
}
catch (IOException e)
{
System.out.println("Could not report on ".concat(String.valueOf(fileopen)));
System.out.println("IOException: ".concat(String.valueOf(e)));
}
this.HumanEar = cfgReader.getHumanEar();
this.configfile = fileopen;
this.modified = false;
return true;
}
public boolean Open(URL configURL)
{
XMLReader parser = null;
XMLCfgReader cfgReader = new XMLCfgReader(this);
try
{
parser = XMLReaderFactory.createXMLReader();
}
catch (Exception e)
{
 try
{
```

```
parser = XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser");
}
catch (Exception ee)
{
System.err.println("Couldn't locate a SAX parser:");
System.err.println(ee);
/* 262 */         return 0;
}
}
parser.setContentHandler(cfgReader);
try
{
InputStream fIn = configURL.openStream();
parser.parse(new InputSource(fIn));
}
catch (SAXParseException e)
{
System.out.println(String.valueOf(configURL).concat(" is not well formed."));
System.out.println(String.valueOf(new StringBuffer(String.valueOf(e.getMessage())).append(" at line
").append(e.getLineNumber()).append(", column ").append(e.getColumnNumber())));
}
catch (SAXException e)
{
System.out.println(e.getMessage());
}
catch (IOException e)
{
System.out.println("Could not report on ".concat(String.valueOf(configURL)));
System.out.println("IOException: ".concat(String.valueOf(e)));
}
this.HumanEar = cfgReader.getHumanEar();
this.configfile = null;
this.modified = false;
return true;
}
public boolean Save(File filesave)
{
XMLCfgWriter writer = new XMLCfgWriter();
AttributesImpl attr = new AttributesImpl();
try
{
FileOutputStream fOut = new FileOutputStream(filesave);
writer.setOutput(fOut);
writer.startDocument();
for (int i = 0; i < this.HumanEar.size(); ++i)
{
HumanEar lEar = (HumanEar)this.HumanEar.get(i);
attr.clear();
attr.addAttribute(null, null, "name", null, lEar.getName());
attr.addAttribute(null, null, "IDlen", null, "".concat(String.valueOf(lEar.getID().length)));
writer.startElement("HumanEar", attr);
if (lEar.getID().length > 0);
double[] ID = lEar.getID();
for (int i2 = 0; i2 < ID.length; ++i2)
{
attr.clear();
attr.addAttribute(null, null, "value", null, "".concat(String.valueOf(ID[i2])));
writer.startElement("ID", attr);
writer.endElement("ID");
}
for (int j = 0; j < lEar.numImages(); ++j)
{
EarImage limage = lEar.getImage(j);
attr.clear();
```

```
attr.addAttribute(null, null, "file", null, limage.getFileName().getPath());
writer.startElement("EarImage", attr);
for (int k = 0; k < limage.numTokens(); ++k)
{
EarToken ltoken = limage.getToken(k);
attr.clear();
attr.addAttribute(null, null, "x1", null, "".concat(String.valueOf(ltoken.getX1())));
attr.addAttribute(null, null, "y1", null, "".concat(String.valueOf(ltoken.getY1())));
attr.addAttribute(null, null, "x2", null, "".concat(String.valueOf(ltoken.getX2())));
attr.addAttribute(null, null, "y2", null, "".concat(String.valueOf(ltoken.getY2())));
writer.startElement("EarToken", attr);
writer.endElement("EarToken");
}
writer.endElement("EarImage");
}
writer.endElement("HumanEar");
}
if (this.nNetwork != null)
{
attr.clear();
attr.addAttribute(null, null, "input", null, "".concat(String.valueOf(this.nNetwork.numInput())));
attr.addAttribute(null, null, "hidden", null, "".concat(String.valueOf(this.nNetwork.numHidden())));
attr.addAttribute(null, null, "output", null, "".concat(String.valueOf(this.nNetwork.numOutput())));
attr.addAttribute(null, null, "alpha", null, "".concat(String.valueOf(this.nNetwork.getAlpha())));
attr.addAttribute(null, null, "momentum", null, "".concat(String.valueOf(this.nNetwork.getMomentum())));
writer.startElement("backProp", attr);
double[] biasH = this.nNetwork.getBiasH();
double[][] hiddenW = this.nNetwork.getHiddenW();
for (int i = 0; i < this.nNetwork.numHidden(); ++i)
{
attr.clear();
writer.startElement("hidden", attr);
for (int j = 0; j < this.nNetwork.numInput(); ++j)
{
attr.clear();
attr.addAttribute(null, null, "H", null, "".concat(String.valueOf(hiddenW[i][j])));
writer.startElement("hiddenW", attr);
writer.endElement("hiddenW");
}
attr.clear();
attr.addAttribute(null, null, "H", null, "".concat(String.valueOf(biasH[i])));
writer.startElement("biasH", attr);
writer.endElement("biasH");
writer.endElement("hidden");
}
double[] biasO = this.nNetwork.getBiasO();
double[][] outputW = this.nNetwork.getOutputW();
for (int i = 0; i < this.nNetwork.numOutput(); ++i)
{
attr.clear();
writer.startElement("output", attr);
for (int j = 0; j < this.nNetwork.numHidden(); ++j)
{
attr.clear();
attr.addAttribute(null, null, "O", null, "".concat(String.valueOf(outputW[i][j])));
writer.startElement("outputW", attr);
writer.endElement("outputW");
}
attr.clear();
attr.addAttribute(null, null, "O", null, "".concat(String.valueOf(biasO[i])));
writer.startElement("biasO", attr);
writer.endElement("biasO");
writer.endElement("output");
}
```

```
writer.endElement("backProp");
}
writer.endDocument();
}
catch (FileNotFoundException e)
{
System.err.println("Error: File could not be open.");
}
this.configfile = filesave;
this.modified = false;
return true;
}
public int getMaxToken()
{
int maxToken = 0;
for (int i = 0; i < this.HumanEar.size(); ++i)
{
HumanEar lEar = (HumanEar)this.HumanEar.get(i);
for (int j = 0; j < lEar.numImages(); ++j)
{
EarImage lImage = lEar.getImage(j);
if (lImage.numTokens() <= maxToken)
continue;
maxToken = lImage.numTokens();
}
}
return maxToken;
}
}
```

Utils.java

```
package lrecog.tools;
import java.io.File;

public class Utils
{
public static final String jpeg = "jpeg";
public static final String jpg = "jpg";
public static final String gif = "gif";
public static final String tiff = "tiff";
public static final String tif = "tif";
public static final String xml = "xml";
public static String getExtension(File f)
{
String ext = null;
String s = f.getName();
int i = s.lastIndexOf(46);
if ((i > 0) && (i < s.length() - 1))
{
ext = s.substring(i + 1).toLowerCase();
}
return ext;
}
}
```

XMLCfgReader.java

```
package lrecog.tools;
import java.io.File;
import java.util.ArrayList;
import lrecog.Recognition;
import lrecog.gfx.EarImage;
import lrecog.gfx.HumanEar;
```

```
import lrecog.gfx.EarToken;
import lrecog.nnetwork.BackProp;
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;

public class XMLCfgReader
implements ContentHandler
{
private ProjectEnv projectEnv;
private boolean correctVersion = false;
private ArrayList humanEar;
private HumanEar actEar;
private double[] actEarID;
private double actID;
private int numID = -1;
private EarImage actImage;
private EarToken actToken;
private BackProp actNetwork;
private int numEarID = 0;
private int numHidden = -1;
private int numHiddenW = -1;
private int numOutput = -1;
private int numOutputW = -1;
private double actHiddenW;
private double actBiasH;
private double actOutputW;
private double actBiasO;
public XMLCfgReader(ProjectEnv prjEnv)
{
this.projectEnv = prjEnv;
this.humanEar = new ArrayList();
}
public void setDocumentLocator(Locator locator)
{
}
public void startDocument()
throws SAXException
{
}
public void endDocument()
throws SAXException
{
}
public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
throws SAXException
{
if (qName.compareToIgnoreCase("lrecog") == 0)
{
if (atts.getValue("version").compareToIgnoreCase("1.0") != 0)
return;
this.correctVersion = true;
}
else {
if (!(this.correctVersion))
return;
if (qName.compareToIgnoreCase("humanEar") == 0)
{
this.actEar = new HumanEar(atts.getValue("name"));
if (atts.getValue("IDlen") != null)
{
this.actEarID = new double[Integer.parseInt(atts.getValue("IDlen"))];
}
```

```
this.numID = -1;
}
else if (qName.compareToIgnoreCase("ID") == 0)
{
this.numID += 1;
this.actID = Double.parseDouble(atts.getValue("value"));
}
else if (qName.compareToIgnoreCase("EarImage") == 0)
{
if (this.projectEnv.getCodeBase() == null)
{
this.actImage = new EarImage(new File(atts.getValue("file")));
}
else
{
this.actImage = this.projectEnv.getRecognition().getImageChooser().getEarImage(atts.getValue("file"));
}
}
else if (qName.compareToIgnoreCase("earToken") == 0)
{
int x1 = Integer.parseInt(atts.getValue("x1"));
int y1 = Integer.parseInt(atts.getValue("y1"));
int x2 = Integer.parseInt(atts.getValue("x2"));
int y2 = Integer.parseInt(atts.getValue("y2"));
this.actToken = new EarToken(x1, y1, x2, y2);
}
else if (qName.compareToIgnoreCase("backProp") == 0)
{
int input = Integer.parseInt(atts.getValue("input"));
int hidden = Integer.parseInt(atts.getValue("hidden"));
int output = Integer.parseInt(atts.getValue("output"));
double alpha = Double.parseDouble(atts.getValue("alpha"));
double momentum = Double.parseDouble(atts.getValue("momentum"));
this.actNetwork = new BackProp(input, hidden, output, alpha, momentum);
this.numHidden = (this.numHiddenW = -1);
this.numOutput = (this.numOutputW = -1);
}
else if (qName.compareToIgnoreCase("hidden") == 0)
{
this.numHidden += 1;
this.numHiddenW = -1;
}
else if (qName.compareToIgnoreCase("hiddenW") == 0)
{
this.numHiddenW += 1;
this.actHiddenW = Double.parseDouble(atts.getValue("H"));
}
else if (qName.compareToIgnoreCase("biasH") == 0)
{
this.actBiasH = Double.parseDouble(atts.getValue("H"));
}
else if (qName.compareToIgnoreCase("output") == 0)
{
this.numOutput += 1;
this.numOutputW = -1;
}
else if (qName.compareToIgnoreCase("outputW") == 0)
{
this.numOutputW += 1;
this.actOutputW = Double.parseDouble(atts.getValue("O"));
} else {
if (qName.compareToIgnoreCase("biasO") != 0)
return;
this.actBiasO = Double.parseDouble(atts.getValue("O"));
```

```
}
}
}
public void endElement(String namespaceURI, String localName, String qName)
throws SAXException
{
if (!(this.correctVersion))
return;
if ((qName.compareToIgnoreCase("humanEar") == 0) && (this.actEar != null))
{
this.actEar.setID(this.actEarID);
this.humanEar.add(this.actEar);
this.actEar = null;
}
else if (qName.compareToIgnoreCase("ID") == 0)
{
this.actEarID[this.numID] = this.actID;
}
else if ((qName.compareToIgnoreCase("EarImage") == 0) && (this.actImage != null) && (this.actEar !=
null))
{
this.actEar.addImage(this.actImage);
this.actImage = null;
}
else if ((qName.compareToIgnoreCase("earToken") == 0) && (this.actToken != null) && (this.actImage !=
null))
{
this.actImage.addToken(this.actToken);
this.actToken = null;
}
else if ((qName.compareToIgnoreCase("backProp") == 0) && (this.actNetwork != null))
{
this.projectEnv.setNetwork(this.actNetwork);
this.actNetwork = null;
this.numHidden = (this.numHiddenW = -1);
this.numOutput = (this.numOutputW = -1);
}
else if ((qName.compareToIgnoreCase("hiddenW") == 0) && (this.actNetwork != null))
{
this.actNetwork.setHiddenW(this.actHiddenW, this.numHidden, this.numHiddenW);
}
else if ((qName.compareToIgnoreCase("biasH") == 0) && (this.actNetwork != null))
{
this.actNetwork.setBiasH(this.actBiasH, this.numHidden);
}
else if ((qName.compareToIgnoreCase("outputW") == 0) && (this.actNetwork != null))
{
this.actNetwork.setOutputW(this.actOutputW, this.numOutput, this.numOutputW);
} else {
if ((qName.compareToIgnoreCase("biasO") != 0) || (this.actNetwork == null))
return;
this.actNetwork.setBiasO(this.actBiasO, this.numOutput);
}
}
public void characters(char[] text, int start, int length)
throws SAXException
{
}
public void ignorableWhitespace(char[] text, int start, int length)
throws SAXException
{
}
public void processingInstruction(String target, String data)
throws SAXException
```

```
{
}
public void startPrefixMapping(String prefix, String uri)
throws SAXException
{
}
public void endPrefixMapping(String prefix)
throws SAXException
{
}
public void skippedEntity(String name)
throws SAXException
{
}
public ArrayList getHumanEar()
{
return this.humanEar;
}
}
```

XMLCfgWriter.java

```
package lrecog.tools;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import org.xml.sax.Attributes;

public class XMLCfgWriter
{
protected PrintWriter fOut;
protected boolean fCanonical;
protected int fElementDepth;
public void setOutput(OutputStream stream)
{
try
{
Writer writer = new OutputStreamWriter(stream, "UTF-8");
this.fOut = new PrintWriter(writer);
}
catch (UnsupportedEncodingException e)
{
System.err.println(e);
}
}
public void startDocument()
{
this.fElementDepth = 0;
this.fOut.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
this.fOut.println("<lrecog version=\"1.0\">");
this.fOut.flush();
}
public void endDocument()
{
if (this.fElementDepth != 0) System.err.println("Error: ElementDepth > 0 on end of document!");
this.fOut.println("</lrecog>");
this.fOut.flush();
}
public void startElement(String element, Attributes attrs)
{
this.fElementDepth += 1;
```

```
this.fOut.print('<');
this.fOut.print(element);
if (attrs != null)
{
int len = attrs.getLength();
for (int i = 0; i < len; ++i)
{
this.fOut.print(' ');
this.fOut.print(attrs.getQName(i));
this.fOut.print("=\"");
normalizeAndPrint(attrs.getValue(i));
this.fOut.print("");
}
}
this.fOut.print('>');
this.fOut.flush();
}
public void endElement(String element)
{
this.fElementDepth -= 1;
this.fOut.print("</");
this.fOut.print(element);
this.fOut.print('>');
this.fOut.flush();
}
public void characters(String s)
{
normalizeAndPrint(s);
this.fOut.flush();
}
protected void normalizeAndPrint(String s)
{
int len = (s != null) ? s.length() : 0;
for (int i = 0; i < len; ++i)
{
char c = s.charAt(i);
normalizeAndPrint(c);
}
}
protected void normalizeAndPrint(char c)
{
switch (c)
{
case '<':
this.fOut.print("&lt;");
break;
case '>':
this.fOut.print("&gt;");
break;
case '&':
this.fOut.print("&amp;");
break;
case '"':
this.fOut.print("&quot;");
break;
case '\n':
case '\r':
this.fOut.print("&#");
this.fOut.print(Integer.toString(c));
this.fOut.print(';');
break;
default:
this.fOut.print(c);
}
```

```
}
}
```

XMLFilter.java

```
package lrecog.tools;
import java.io.File;
import javax.swing.filechooser.FileFilter;

public class XMLFilter extends FileFilter
{
public boolean accept(File f)
{
if (f.isDirectory())
{
return true;
}
String extension = Utils.getExtension(f);
if (extension != null)
{
return (extension.equals("xml"));
}
return false;
}
public String getDescription()
{
return "XML Files (*.xml)";
}
}
```

ContrastTheme.java

```
package lrecog.themes;
import javax.swing.UIDefaults;
import javax.swing.border.Border;
import javax.swing.border.CompoundBorder;
import javax.swing.border.LineBorder;
import javax.swing.plaf.BorderUIResource;
import javax.swing.plaf.ColorUIResource;
import javax.swing.plaf.basic.BasicBorders.MarginBorder;
import javax.swing.plaf.metal.DefaultMetalTheme;
import javax.swing.plaf.metal.MetalTheme;

public class ContrastTheme extends DefaultMetalTheme
{
private final ColorUIResource primary1 = new ColorUIResource(0, 0, 0);
private final ColorUIResource primary2 = new ColorUIResource(204, 204, 204);
private final ColorUIResource primary3 = new ColorUIResource(255, 255, 255);
private final ColorUIResource primaryHighlight = new ColorUIResource(102, 102, 102);

private final ColorUIResource secondary2 = new ColorUIResource(204, 204, 204);
private final ColorUIResource secondary3 = new ColorUIResource(255, 255, 255);
private final ColorUIResource controlHighlight = new ColorUIResource(102, 102, 102);
public String getName()
{
return "Contrast";
}

protected ColorUIResource getPrimary1()
{
return this.primary1; }
protected ColorUIResource getPrimary2() { return this.primary2; }
protected ColorUIResource getPrimary3() { return this.primary3; }
public ColorUIResource getPrimaryControlHighlight() { return this.primaryHighlight; }
```

```
protected ColorUIResource getSecondary2() { return this.secondary2; }
protected ColorUIResource getSecondary3() { return this.secondary3; }
public ColorUIResource getControlHighlight() { return super.getSecondary3(); }
public ColorUIResource getFocusColor() { return super.getBlack(); }
public ColorUIResource getTextHighlightColor() { return super.getBlack(); }
public ColorUIResource getHighlightedTextColor() { return super.getWhite(); }
public ColorUIResource getMenuSelectedBackground() { return super.getBlack(); }
public ColorUIResource getMenuSelectedForeground() { return super.getWhite(); }
public ColorUIResource getAcceleratorForeground() { return super.getBlack(); }
public ColorUIResource getAcceleratorSelectedForeground() { return super.getWhite();
}

public void addCustomEntriesToTable(UIDefaults table)
{
Border blackLineBorder = new BorderUIResource(new LineBorder(super.getBlack()));
Border whiteLineBorder = new BorderUIResource(new LineBorder(super.getWhite()));

Object textBorder = new BorderUIResource(new CompoundBorder(blackLineBorder, new
BasicBorders.MarginBorder()));

table.put("ToolTip.border", blackLineBorder);
table.put("TitledBorder.border", blackLineBorder);
table.put("Table.focusCellHighlightBorder", whiteLineBorder);
table.put("Table.focusCellForeground", super.getWhite());
table.put("TextField.border", textBorder);
table.put("PasswordField.border", textBorder);
table.put("TextArea.border", textBorder);
table.put("TextPane.font", textBorder);
}
}
```

RubyTheme.java

```
package lrecog.themes;
import javax.swing.plaf.ColorUIResource;
import javax.swing.plaf.metal.DefaultMetalTheme;

public class RubyTheme extends DefaultMetalTheme
{
private final ColorUIResource primary1 = new ColorUIResource(80, 10, 22);
private final ColorUIResource primary2 = new ColorUIResource(193, 10, 44);
private final ColorUIResource primary3 = new ColorUIResource(244, 10, 66);

public String getName()
{
return "Ruby";
}

protected ColorUIResource getPrimary1()
{
return this.primary1; }
protected ColorUIResource getPrimary2() { return this.primary2; }
protected ColorUIResource getPrimary3() { return this.primary3;
}
}
```

AquaTheme.java

```
package lrecog.themes;
import javax.swing.plaf.ColorUIResource;
import javax.swing.plaf.metal.DefaultMetalTheme;

public class AquaTheme extends DefaultMetalTheme
```

```
{
private final ColorUIResource primary1 = new ColorUIResource(102, 153, 153);
private final ColorUIResource primary2 = new ColorUIResource(128, 192, 192);
private final ColorUIResource primary3 = new ColorUIResource(159, 235, 235);

public String getName()
{
return "Aqua";
}

protected ColorUIResource getPrimary1()
{
return this.primary1; }
protected ColorUIResource getPrimary2() { return this.primary2; }
protected ColorUIResource getPrimary3() { return this.primary3;
}
}
```

CharcoalTheme.java

```
package lrecog.themes;
import javax.swing.plaf.ColorUIResource;
import javax.swing.plaf.metal.DefaultMetalTheme;

public class CharcoalTheme extends DefaultMetalTheme
{
 private final ColorUIResource primary1 = new ColorUIResource(66, 33, 66);
private final ColorUIResource primary2 = new ColorUIResource(90, 86, 99);
private final ColorUIResource primary3 = new ColorUIResource(99, 99, 99);

private final ColorUIResource secondary1 = new ColorUIResource(0, 0, 0);
private final ColorUIResource secondary2 = new ColorUIResource(51, 51, 51);
private final ColorUIResource secondary3 = new ColorUIResource(102, 102, 102);

private final ColorUIResource black = new ColorUIResource(222, 222, 222);
private final ColorUIResource white = new ColorUIResource(0, 0, 0);

public String getName()
{
 return "Charcoal";
}

protected ColorUIResource getPrimary1()
{
return this.primary1; }
protected ColorUIResource getPrimary2() { return this.primary2; }
protected ColorUIResource getPrimary3() { return this.primary3; }

protected ColorUIResource getSecondary1() { return this.secondary1; }
protected ColorUIResource getSecondary2() { return this.secondary2; }
protected ColorUIResource getSecondary3() { return this.secondary3; }
protected ColorUIResource getBlack() { return this.black; }
protected ColorUIResource getWhite() { return this.white;
}
}
```

EmeraldTheme.java

```
package lrecog.themes;
import javax.swing.plaf.ColorUIResource;
import javax.swing.plaf.metal.DefaultMetalTheme;

public class EmeraldTheme extends DefaultMetalTheme
{
```

```
private final ColorUIResource primary1 = new ColorUIResource(51, 142, 71);
private final ColorUIResource primary2 = new ColorUIResource(102, 193, 122);
private final ColorUIResource primary3 = new ColorUIResource(153, 244, 173);

public String getName()
{
return "Emerald";
}

protected ColorUIResource getPrimary1()
{
return this.primary1; }
protected ColorUIResource getPrimary2() { return this.primary2; }
protected ColorUIResource getPrimary3() { return this.primary3;
}
}
```