\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.document.load;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.Map;

import org.LYG.GUI.nodeEdit.LinkList;

import org.LYG.GUI.nodeEdit.LinkNode;

import org.LYG.GUI.nodeEdit.Sort;

import org.LYG.GUI.nodeView.NodeShow;

//准备把响应事件移植到这里。

import org.LYG.sets.stable.StableData;

public class LoadFile{

@SuppressWarnings({StableData.TAG\_STATIC\_ACCESS, StableData.TAG\_RESOURCE})

public static LinkNode Load(LinkNode first, NodeShow nodeView, File file, LinkList thislist) {

//get path

try {

InputStream in= new FileInputStream(file);

BufferedReader cReader= new BufferedReader(new InputStreamReader(in));

String ctempString= null;

Map<String, String> currentNodeMap= new HashMap<>();

while ((ctempString= cReader.readLine())!= null) {

if(!ctempString.contains("######################")) {

if(ctempString.contains(":")&& ctempString.split(":").length>1) {

currentNodeMap.put(ctempString.split(":")[0], ctempString.split(":")[1]);

}

}else {

LinkNode node= new LinkNode();

node.beconnect= currentNodeMap.containsKey("beconnect")

?currentNodeMap.get("beconnect").contains("false")? false: true: false;

node.dBeconnect= currentNodeMap.containsKey("dBeconnect")?

currentNodeMap.get("dBeconnect").contains("false")? false: true: false;

node.dBeconnectID= currentNodeMap.containsKey("dBeconnectID")?

Integer.parseInt(currentNodeMap.get("dBeconnectID")):0;

node.dBeconnectPrimaryKey= currentNodeMap.containsKey("dBeconnectPrimaryKey")?

currentNodeMap.get("dBeconnectPrimaryKey"):"null";

node.dBeconnectX= currentNodeMap.containsKey("dBeconnectX")?

Integer.parseInt(currentNodeMap.get("dBeconnectX")):0;

node.dBeconnectY= currentNodeMap.containsKey("dBeconnectY")?

Integer.parseInt(currentNodeMap.get("dBeconnectY")):0;

node.dBeconnetName= currentNodeMap.containsKey("dBeconnetName")?

currentNodeMap.get("dBeconnetName"):"null";

node.flash= currentNodeMap.containsKey("flash")?

Integer.parseInt(currentNodeMap.get("flash")):0;

node.ID= currentNodeMap.containsKey("NodeID")?

Integer.parseInt(currentNodeMap.get("NodeID")):0;

node.leftChoose= currentNodeMap.containsKey("leftChoose")?

currentNodeMap.get("leftChoose").contains("false")? false: true: false;

node.mBeconnect= currentNodeMap.containsKey("mBeconnect")?

currentNodeMap.get("mBeconnect").contains("false")? false: true: false;

node.mBeconnectID= currentNodeMap.containsKey("mBeconnectID")?

Integer.parseInt(currentNodeMap.get("mBeconnectID")):0;

node.mBeconnectPrimaryKey= currentNodeMap.containsKey("mBeconnectPrimaryKey")?

currentNodeMap.get("mBeconnectPrimaryKey"):"null";

node.mBeconnectX= currentNodeMap.containsKey("mBeconnectX")?

Integer.parseInt(currentNodeMap.get("mBeconnectX")):0;

node.mBeconnectY= currentNodeMap.containsKey("mBeconnectY")?

Integer.parseInt(currentNodeMap.get("mBeconnectY")):0;

node.mBeconnetName= currentNodeMap.containsKey("mBeconnetName")?currentNodeMap.get("mBeconnetName"):"null";

node.name= currentNodeMap.containsKey("NodeName")?currentNodeMap.get("NodeName"):"null";

node.rightChoose= currentNodeMap.containsKey("rightChoose")?

currentNodeMap.get("rightChoose").contains("false")? false: true: false;

node.tBeconnect= currentNodeMap.containsKey("tBeconnect")?

currentNodeMap.get("tBeconnect").contains("false")? false: true: false;

node.tBeconnectID= currentNodeMap.containsKey("tBeconnectID")?

Integer.parseInt(currentNodeMap.get("tBeconnectID")):0;

node.tBeconnectPrimaryKey= currentNodeMap.containsKey("tBeconnectPrimaryKey")?

currentNodeMap.get("tBeconnectPrimaryKey"):"null";

node.primaryKey= currentNodeMap.containsKey("primaryKey")?

currentNodeMap.get("primaryKey"):"null";

node.tBeconnectX= currentNodeMap.containsKey("tBeconnectX")?

Integer.parseInt(currentNodeMap.get("tBeconnectX")):0;

node.tBeconnectY= currentNodeMap.containsKey("tBeconnectY")?

Integer.parseInt(currentNodeMap.get("tBeconnectY")):0;

node.tBeconnetName= currentNodeMap.containsKey("tBeconnetName")?

currentNodeMap.get("tBeconnetName"):"null";

node.x= currentNodeMap.containsKey("NodeCoordinationX")?

Integer.parseInt(currentNodeMap.get("NodeCoordinationX")):0;

node.y= currentNodeMap.containsKey("NodeCoordinationY")?

Integer.parseInt(currentNodeMap.get("NodeCoordinationY")):0;

node= thislist.addNodeOnlyWithFace(node, nodeView.first);

if(null== first) {

first= node;

}else {

first.next= node;

node.pre= first;

first= first.next;

}

currentNodeMap.clear();

}

}

}catch(Exception loadE) {

loadE.printStackTrace();

}

first= new Sort().sort(first);

return first;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.document.save;

import java.io.File;

import java.io.FileWriter;

import org.LYG.GUI.nodeEdit.LinkNode;

public class SaveAndUpdateFile{

public static void update(String fileCurrentpath, LinkNode first) {

//delete file

File file= new File(fileCurrentpath);

if(file.exists()&& file.isFile()) {

file.delete();

}

//save

String fileSavepath= fileCurrentpath;

System.out.println(fileSavepath);

//create file and save

try {

FileWriter fileWriter= new FileWriter(fileSavepath);

LinkNode node= first;

while(null!= node) {

//挨个取。没难度。逐个把信息写入文件。

//节点坐标，节点名， 节点关联，

String NodeCoordinationX= ""+ node.x;

String NodeCoordinationY= ""+ node.y;

String NodeName= ""+ node.name;

String NodeID=""+ node.ID;

String flash=""+ node.flash;

String beconnect= ""+ node.beconnect;

String leftChoose= ""+ node.leftChoose;

String rightChoose= ""+ node.rightChoose;

String tBeconnect= ""+ node.tBeconnect;

String tBeconnectX= ""+ node.tBeconnectX;

String tBeconnectY= ""+ node.tBeconnectY;

String tBeconnetName= ""+ node.tBeconnetName;

String tBeconnectID= ""+ node.tBeconnectID;

String tBeconnectPrimaryKey= ""+ node.dBeconnectPrimaryKey;

String mBeconnect= ""+ node.mBeconnect;

String mBeconnectX= ""+ node.mBeconnectX;

String mBeconnectY= ""+ node.mBeconnectY;

String mBeconnetName= ""+ node.mBeconnetName;

String mBeconnectID= ""+ node.mBeconnectID;

String mBeconnectPrimaryKey= ""+ node.mBeconnectPrimaryKey;

String dBeconnect= ""+ node.dBeconnect;

String dBeconnectX= ""+ node.dBeconnectX;

String dBeconnectY= ""+ node.dBeconnectY;

String dBeconnetName= ""+ node.dBeconnetName;

String dBeconnectID= ""+ node.dBeconnectID;

String dBeconnectPrimaryKey= ""+ node.dBeconnectPrimaryKey;

String primaryKey= ""+ node.primaryKey;

String NodeConfiguration= "";

//配置

fileWriter.write("\r\n");

fileWriter.write("NodeCoordinationX:"+ NodeCoordinationX);

fileWriter.write("\r\n");

fileWriter.write("NodeName:"+ NodeName);

fileWriter.write("\r\n");

fileWriter.write("NodeCoordinationY:"+ NodeCoordinationY);

fileWriter.write("\r\n");

fileWriter.write("NodeID:"+ NodeID);

fileWriter.write("\r\n");

fileWriter.write("flash:"+ flash);

fileWriter.write("\r\n");

fileWriter.write("beconnect:"+ beconnect);

fileWriter.write("\r\n");

fileWriter.write("leftChoose:"+ leftChoose);

fileWriter.write("\r\n");

fileWriter.write("rightChoose:"+ rightChoose);

fileWriter.write("\r\n");

fileWriter.write("tBeconnect:"+ tBeconnect);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectX:"+ tBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectY:"+ tBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("tBeconnetName:"+ tBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectID:"+ tBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectPrimaryKey:"+ tBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("mBeconnect:"+ mBeconnect);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectX:"+ mBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectY:"+ mBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("mBeconnetName:"+ mBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectID:"+ mBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectPrimaryKey:"+ mBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("dBeconnect:"+ dBeconnect);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectX:"+ dBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectY:"+ dBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("dBeconnetName:"+ dBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectID:"+ dBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectPrimaryKey:"+ dBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("primaryKey:"+ primaryKey);

fileWriter.write("\r\n");

fileWriter.write("NodeConfiguration:"+ NodeConfiguration);

fileWriter.write("\r\n");

//分割

String split="##############################";

fileWriter.write("\r\n");

fileWriter.write(split);

fileWriter.flush();

if(null== node.next) {

break;

}

node= node.next;

}

fileWriter.close();

}catch(Exception saveFile) {

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.document.save;

import java.awt.FileDialog;

import java.awt.Frame;

import java.io.File;

import java.io.FileWriter;

import org.LYG.GUI.nodeEdit.LinkNode;

import comp.filenameFilter.TXTFilter;

//准备把响应事件移植到这里。

public class SaveAsANewFile{

public static void Save(LinkNode first) {

FileDialog filedialog= new FileDialog(new Frame(), "在当前文件夹下创建一个档案名", FileDialog.LOAD);

filedialog.setFilenameFilter(new TXTFilter(".etl"));

filedialog.setVisible(true);

String fileSavepath= filedialog.getDirectory()+ filedialog.getFile();

System.out.println(fileSavepath);

if(new File(fileSavepath).isFile()&& fileSavepath.contains(".etl")) {

System.out.println("文档已经存在。");

return;

}

fileSavepath= fileSavepath+ ".etl";

System.out.println(fileSavepath);

//create file and save

try {

FileWriter fileWriter= new FileWriter(fileSavepath);

LinkNode node = first;

while(node!=null) {

//挨个取。没难度。逐个把信息写入文件。

//节点坐标，节点名， 节点关联，

String NodeCoordinationX= ""+ node.x;

String NodeCoordinationY= ""+ node.y;

String NodeName= ""+ node.name;

String NodeID=""+ node.ID;

String flash=""+ node.flash;

String beconnect= ""+ node.beconnect;

String leftChoose= ""+ node.leftChoose;

String rightChoose= ""+ node.rightChoose;

String tBeconnect= ""+ node.tBeconnect;

String tBeconnectX= ""+ node.tBeconnectX;

String tBeconnectY= ""+ node.tBeconnectY;

String tBeconnetName= ""+ node.tBeconnetName;

String tBeconnectID= ""+ node.tBeconnectID;

String tBeconnectPrimaryKey= ""+ node.dBeconnectPrimaryKey;

String mBeconnect= ""+ node.mBeconnect;

String mBeconnectX= ""+ node.mBeconnectX;

String mBeconnectY= ""+ node.mBeconnectY;

String mBeconnetName= ""+ node.mBeconnetName;

String mBeconnectID= ""+ node.mBeconnectID;

String mBeconnectPrimaryKey= ""+ node.mBeconnectPrimaryKey;

String dBeconnect= ""+ node.dBeconnect;

String dBeconnectX= ""+ node.dBeconnectX;

String dBeconnectY= ""+ node.dBeconnectY;

String dBeconnetName= ""+ node.dBeconnetName;

String dBeconnectID= ""+ node.dBeconnectID;

String dBeconnectPrimaryKey= ""+ node.dBeconnectPrimaryKey;

String primaryKey= ""+ node.primaryKey;

String NodeConfiguration= "";

//配置

fileWriter.write("\r\n");

fileWriter.write("NodeCoordinationX:"+ NodeCoordinationX);

fileWriter.write("\r\n");

fileWriter.write("NodeName:"+ NodeName);

fileWriter.write("\r\n");

fileWriter.write("NodeCoordinationY:"+ NodeCoordinationY);

fileWriter.write("\r\n");

fileWriter.write("NodeID:"+ NodeID);

fileWriter.write("\r\n");

fileWriter.write("flash:"+ flash);

fileWriter.write("\r\n");

fileWriter.write("beconnect:"+ beconnect);

fileWriter.write("\r\n");

fileWriter.write("leftChoose:"+ leftChoose);

fileWriter.write("\r\n");

fileWriter.write("rightChoose:"+ rightChoose);

fileWriter.write("\r\n");

fileWriter.write("tBeconnect:"+ tBeconnect);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectX:"+ tBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectY:"+ tBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("tBeconnetName:"+ tBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectID:"+ tBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("tBeconnectPrimaryKey:"+ tBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("mBeconnect:"+ mBeconnect);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectX:"+ mBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectY:"+ mBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("mBeconnetName:"+ mBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectID:"+ mBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("mBeconnectPrimaryKey:"+ mBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("dBeconnect:"+ dBeconnect);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectX:"+ dBeconnectX);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectY:"+ dBeconnectY);

fileWriter.write("\r\n");

fileWriter.write("dBeconnetName:"+ dBeconnetName);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectID:"+ dBeconnectID);

fileWriter.write("\r\n");

fileWriter.write("dBeconnectPrimaryKey:"+ dBeconnectPrimaryKey);

fileWriter.write("\r\n");

fileWriter.write("primaryKey:"+ primaryKey);

fileWriter.write("\r\n");

fileWriter.write("NodeConfiguration:"+ NodeConfiguration);

fileWriter.write("\r\n");

//分割

String split="##############################";

fileWriter.write("\r\n");

fileWriter.write(split);

fileWriter.flush();

if(null== node.next) {

break;

}

node=node.next;

}

fileWriter.close();

}catch(Exception saveFile) {

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.extOSGI;

import org.LYG.GUI.nodeEdit.Sort;

import org.LYG.GUI.nodeEdit.LinkNode;

public class OSGI\_chansfer {

public OSGI\_chansfer(LinkNode node, LinkNode first){

first = new Sort().sort(first);

LinkNode linkNode = new LinkNode();

linkNode = first;

while(null != linkNode){

if(node.tBeconnect&&node.tBeconnectID== linkNode.ID&&node.tBeconnetName.equals(linkNode.name)

&& (node.tBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.thisRun.toptablein = linkNode.thisFace.thisView.tableout;

node.thisFace.thisRun.topgin = linkNode.thisFace.thisView.gout;

return;

}

if(node.mBeconnect&&node.mBeconnectID== linkNode.ID&&node.mBeconnetName.equals(linkNode.name)

&& (node.mBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.thisRun.midtablein = linkNode.thisFace.thisView.tableout;

node.thisFace.thisRun.midgin = linkNode.thisFace.thisView.gout;

return;

}

if(node.dBeconnect&&node.dBeconnectID== linkNode.ID&&node.dBeconnetName.equals(linkNode.name)

&& (node.dBeconnectPrimaryKey.equalsIgnoreCase(linkNode.primaryKey))){

node.thisFace.thisRun.downtablein = linkNode.thisFace.thisView.tableout;

node.thisFace.thisRun.downgin = linkNode.thisFace.thisView.gout;

return;

}

if(null != linkNode.next){

break;

}

linkNode=linkNode.next;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.extOSGI;

import java.io.IOException;

import javax.swing.JTextPane;

import org.LYG.GUI.OSGI.\*;

import org.LYG.node.ai.arffTransfer.arffTransferNodeInterface;

public class OSGI\_rigester{

JTextPane text;

Object[][] tableData\_old;

public OSGI\_rigester(Object[][] tableData\_old, JTextPane text){

this.text = text;

this.tableData\_old = tableData\_old;

}

public NodeOSGI Rigester(NodeOSGI first, LinkOSGI link) throws IOException{

//注册

ObjectInterface arffTransferNode = new arffTransferNodeInterface();

first = link.addNode(first, arffTransferNode);

return first;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.Flash;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ComponentEvent;

import java.awt.event.ComponentListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import javax.sound.sampled.UnsupportedAudioFileException;

import javax.swing.JApplet;

import javax.swing.JOptionPane;

import javax.swing.JSplitPane;

import javax.swing.JTextPane;

import javax.swing.JScrollPane;

import javax.swing.UIManager;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.FileDialog;

import java.awt.Frame;

import java.awt.MenuItem;

import java.awt.PopupMenu;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import javax.swing.event.TreeSelectionEvent;

import javax.swing.event.TreeSelectionListener;

import javax.swing.tree.DefaultMutableTreeNode;

import javax.swing.tree.TreePath;

import org.LYG.GUI.extOSGI.OSGI\_chansfer;

import org.LYG.GUI.nodeEdit.LinkList;

import org.LYG.GUI.nodeEdit.Sort;

import org.LYG.GUI.nodeEdit.LinkNode;

import org.LYG.GUI.nodeEdit.UpdateRelatedLine;

import org.LYG.GUI.nodeInfo.NodeInfo;

import org.LYG.GUI.nodeProject.NodeProject;

import org.LYG.GUI.nodeView.CacuString;

import org.LYG.GUI.nodeView.NodeShow;

import org.LYG.GUI.platForm.UnicornJSplitPane;

import org.LYG.document.load.LoadFile;

import org.LYG.document.save.SaveAndUpdateFile;

import org.LYG.document.save.SaveAsANewFile;

import org.LYG.sets.stable.StableData;

import comp.filenameFilter.TXTFilter;

public class GUIsample extends JApplet implements MouseMotionListener

, MouseListener, ItemListener, ActionListener, Runnable{

private static final long serialVersionUID = 5270675501794340912L;

public GUIsample() {

getContentPane().setBackground(new Color(255,255,255));

}

public String fileCurrentpath;

public int w, h;

public int flash= 0;

public int count= 0;

public String currentNodeName;

public int currentNodeID;

public String currentNodePrimaryKey;

public LinkList first;

// public LinkNode first;

public int currentx, currenty;

public int choose= 0;

public int oldx, oldy;

public int newx, newy;

public int isOperation= 0;

public String treeNodeName;

public NodeShow nodeView;

public NodeProject nodeProject;

public NodeInfo nodeInfo;

public UnicornJSplitPane mainSplitPane;

public UnicornJSplitPane leftSplitPane;

public UnicornJSplitPane rightSplitPane;

public UnicornJSplitPane righttopSplitPane;

public JScrollPane righttopScrollPane;

public JScrollPane rightdownScrollPane;

public JScrollPane rightrightScrollPane;

public JTextPane rightBotJTextPane;

public ThisCanvas canvas;

public PopupMenu popupMenu, nodeMenu, itemMenu, engineMenu;

public MenuItem save, saveAs, delete, load;

public MenuItem menuItem;

public MenuItem configre, run, show, dnode, dline;

public Thread thread, threadApplet;

private JTextPane text;

private Object[][] tableData\_old;

public void run() {

try {

Thread.sleep(100);

} catch (InterruptedException e1) {

e1.printStackTrace();

}

nodeProject.setBounds(0, 0, leftSplitPane.getWidth()

, leftSplitPane.getDividerLocation());

nodeProject.jPanel.newimg= nodeProject.img.getScaledInstance(nodeProject.getWidth()

, nodeProject.getHeight(), java.awt.Image.SCALE\_SMOOTH);

nodeProject.jPanel.update(getGraphics());

nodeProject.validate();

while(true){

try{

Thread.sleep(1000);

this.validate();

}catch (InterruptedException e) {}

//repaint();

}

}

public void start(){

if(thread == null){

thread = new Thread(this);

thread.start();

}

}

public void stop() {

}

public void Registrar() {

load.addActionListener(new java.awt.event.ActionListener() {

@SuppressWarnings({StableData.TAG\_RESOURCE, StableData.TAG\_STATIC\_ACCESS})

public void actionPerformed(ActionEvent e) {

try {

javax.swing.JOptionPane jOptionPane= new JOptionPane

(StableData.ATTENSION\_LOAD\_ENSURE);

int confirm= jOptionPane.showConfirmDialog

(canvas, StableData.ATTENSION\_LOAD\_ENSURE);

if(0!= confirm) {

rightBotJTextPane.setText(StableData.ATTENSION\_CANCELLED\_OPERATION);

rightBotJTextPane.validate();

return;

}

FileDialog filedialog= new FileDialog

(new Frame(), StableData.ATTENSION\_LOAD\_HISTORY

, FileDialog.LOAD);

filedialog.setFilenameFilter(new TXTFilter(StableData.FILE\_FORMAT\_ETL));

filedialog.setVisible(true);

fileCurrentpath= filedialog.getDirectory()+ filedialog.getFile();

System.out.println(fileCurrentpath);

if(null== fileCurrentpath|| fileCurrentpath.isEmpty()|| !fileCurrentpath.contains

(StableData.FILE\_FORMAT\_ETL)) {

System.out.println(StableData.ATTENSION\_RECHOICE);

return;

}

File file= new File(fileCurrentpath);

if(!file.isFile()) {

System.out.println(StableData.ATTENSION\_RECHOICE);

return;

}

LinkNode needDeleteNode= first.first;

while(needDeleteNode!= null) {

first.first= first.deletNode(first.first, needDeleteNode.name,

needDeleteNode.ID

, needDeleteNode.primaryKey);

if(null== needDeleteNode.next) {

break;

}

needDeleteNode= needDeleteNode.next;

}

canvas.repaint();

first.first= LoadFile.Load(first.first, nodeView, file, first);

}catch(Exception loadE) {

loadE.printStackTrace();

}

canvas.repaint();

righttopScrollPane.validate();

}

});

save.addActionListener(new java.awt.event.ActionListener() {

@SuppressWarnings({StableData.TAG\_UNUSED, StableData.TAG\_STATIC\_ACCESS})

public void actionPerformed(ActionEvent e) {

if(null== fileCurrentpath) {

System.out.println(StableData.ATTENSION\_UNCURRENT\_CHOICE);

return;

}

javax.swing.JOptionPane jOptionPane= new JOptionPane(StableData.ATTENSION\_UPDATE\_ENSURE

+ fileCurrentpath + StableData.MARK\_QUESTION);

int confirm= jOptionPane.showConfirmDialog(canvas, StableData.ATTENSION\_UPDATE\_ENSURE

+ fileCurrentpath + StableData.MARK\_QUESTION);

if(0!= confirm) {

rightBotJTextPane.setText(StableData.ATTENSION\_CANCELLED\_OPERATION);

rightBotJTextPane.validate();

return;

}

SaveAndUpdateFile.update(fileCurrentpath, first.first);

}

});

saveAs.addActionListener(new java.awt.event.ActionListener() {

@SuppressWarnings(StableData.TAG\_UNUSED)

public void actionPerformed(ActionEvent e) {

SaveAsANewFile.Save(first.first);

}

});

//delete

delete.addActionListener(new java.awt.event.ActionListener() {

@SuppressWarnings(StableData.TAG\_STATIC\_ACCESS)

public void actionPerformed(ActionEvent e) {

try {

javax.swing.JOptionPane jOptionPane

= new JOptionPane(StableData.ATTENSION\_CANCEL\_ENSURE);

int confirm= jOptionPane.showConfirmDialog(canvas,

StableData.ATTENSION\_CANCEL\_ENSURE);

if(0!= confirm) {

rightBotJTextPane.setText(StableData.ATTENSION\_CANCELLED\_OPERATION);

rightBotJTextPane.validate();

return;

}

LinkNode node= first.first;

while(node!= null) {

first.first= first.deletNode(first.first, node.name, node.ID,

node.primaryKey);

if(null== node.next) {

break;

}

node= node.next;

}

node= node.next;

canvas.repaint();

}catch(Exception E) {

canvas.repaint();

}

rightBotJTextPane.setText(StableData.ATTENSION\_DELETE);

rightBotJTextPane.validate();

}

});

leftSplitPane.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

public void propertyChange(java.beans.PropertyChangeEvent evt) {

if (evt.getPropertyName().equals(JSplitPane.DIVIDER\_LOCATION\_PROPERTY)) {

//action code

nodeProject.setBounds(0, 0,leftSplitPane.getWidth(),leftSplitPane

.getDividerLocation());

nodeProject.jPanel.newimg = nodeProject.img.getScaledInstance

(nodeProject.getWidth(), nodeProject.getHeight()

,java.awt.Image.SCALE\_SMOOTH );

nodeProject.jPanel.repaint();

nodeProject.validate();

}

}

});

mainSplitPane.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

public void propertyChange(java.beans.PropertyChangeEvent evt) {

if (evt.getPropertyName().equals(JSplitPane.DIVIDER\_LOCATION\_PROPERTY)) {

//action code

nodeProject.setBounds(0, 0,mainSplitPane.getDividerLocation()

, leftSplitPane.getDividerLocation());

nodeProject.jPanel.newimg= nodeProject.img.getScaledInstance(nodeProject.getWidth()

, nodeProject.getHeight(),java.awt.Image.SCALE\_SMOOTH );

nodeProject.jPanel.repaint();

nodeProject.validate();

}

}

});

righttopScrollPane.addComponentListener(new ComponentListener(){

public void componentHidden(ComponentEvent arg0) {}

public void componentMoved(ComponentEvent arg0) {}

public void componentResized(ComponentEvent arg0) {

righttopScrollPane.validate();

}

public void componentShown(ComponentEvent arg0) {}

});

getContentPane().addComponentListener(new ComponentListener(){

public void componentHidden(ComponentEvent arg0) {}

public void componentMoved(ComponentEvent arg0) {}

public void componentResized(ComponentEvent arg0) {

w=getContentPane().getWidth();

h=getContentPane().getHeight();

mainSplitPane.setBounds(10, 50, w-20, h-80);

mainSplitPane.setDividerLocation(0.11);

leftSplitPane.setDividerLocation(0.25);

rightSplitPane.setDividerLocation(0.85);

righttopSplitPane.setDividerLocation(0.9);

nodeProject.setBounds(0, 0,mainSplitPane.getDividerLocation()

, leftSplitPane.getDividerLocation());

nodeProject.jPanel.newimg = nodeProject.img.getScaledInstance

(nodeProject.getWidth(), nodeProject.getHeight()

, java.awt.Image.SCALE\_SMOOTH );

nodeProject.jPanel.repaint();

nodeProject.validate();

mainSplitPane.validate();

System.out.println(w + "<>" + h);

}

public void componentShown(ComponentEvent arg0) {

}

});

addMouseListener(this);

addMouseMotionListener(this);

nodeProject.addMouseListener(this);

nodeView.addMouseListener(this);

nodeView.tree.addMouseListener(this);

nodeView.tree.addTreeSelectionListener(new TreeSelectionListener() {

public void valueChanged(TreeSelectionEvent evt) {

DefaultMutableTreeNode note= (DefaultMutableTreeNode) nodeView.tree.getLastSelectedPathComponent();

String tr = null;

if(note!= null){

tr= new CacuString().cauString(note.toString());

}

if(tr!=null){

treeNodeName= new String(tr);

rightBotJTextPane.setText("节点名："+ treeNodeName);

rightBotJTextPane.validate();

}

}

});

menuItem.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

if(treeNodeName!=null){

try {

first.first= first.addNode(first.first, treeNodeName, 100, 50, nodeView.first);

righttopScrollPane.validate();

} catch (CloneNotSupportedException e1) {

rightBotJTextPane.setText(StableData.NODE\_ADD\_ERROR);

rightBotJTextPane.validate();

} catch (InstantiationException e1) {

rightBotJTextPane.setText(StableData.NODE\_ADD\_ERROR);

rightBotJTextPane.validate();

} catch (IllegalAccessException e1) {

rightBotJTextPane.setText(StableData.NODE\_ADD\_ERROR);

rightBotJTextPane.validate();

} catch (IOException e1) {

rightBotJTextPane.setText(StableData.NODE\_ADD\_ERROR);

rightBotJTextPane.validate();

}

rightBotJTextPane.setText("节点名："+ "treeNodeName");

rightBotJTextPane.validate();

}

}

});

configre.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

LinkNode node= new LinkNode();

first.first= new Sort().sort(first.first);

node= first.first;

while(node!= null){

if(node.name.equals(canvas.currentNodeName)&&node.ID== canvas.currentNodeID

&& node.primaryKey.equals(canvas.currentNodePrimaryKey)){

try {

node.thisFace.config(rightBotJTextPane);

node.thisFace.thisPanel.setLocation(node.x, node.y);

node.thisFace.thisPanel.setSize(300, 300);

node.thisFace.thisPanel.setResizable(true);

node.thisFace.thisPanel.jsp.setBounds(0, 0, node.thisFace.thisPanel.getWidth()-10

, node.thisFace.thisPanel.getHeight()-45);

node.thisFace.thisPanel.jp.setPreferredSize(new Dimension(800,600));

node.thisFace.thisPanel.setBackground(Color.BLUE);

node.thisFace.thisPanel.setVisible(true);

node.thisFace.thisPanel.validate();

new OSGI\_chansfer(node, first.first);

} catch (IOException e1){

rightBotJTextPane.setText(StableData.NODE\_UPDATE\_ERROR);

rightBotJTextPane.validate();

}

}

node= node.next;

}

rightBotJTextPane.setText(StableData.NODE\_UPDATE\_SUCCESS);

rightBotJTextPane.validate();

}

});

run.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

LinkNode node= new LinkNode();

first.first= new Sort().sort(first.first);

node= first.first;

while(node!= null){

if(node.name.equals(canvas.currentNodeName)&&node.ID == canvas.currentNodeID

&& node.primaryKey.equals(canvas.currentNodePrimaryKey)){

try {

node.thisFace.execute(rightBotJTextPane);

} catch (FileNotFoundException e1) {

rightBotJTextPane.setText(StableData.NODE\_EXEC\_ERROR);

rightBotJTextPane.validate();

} catch (IOException e1) {

rightBotJTextPane.setText(StableData.NODE\_EXEC\_ERROR);

rightBotJTextPane.validate();

} catch (UnsupportedAudioFileException e2) {

rightBotJTextPane.setText(StableData.NODE\_EXEC\_ERROR);

rightBotJTextPane.validate();

} catch (InterruptedException e3) {

rightBotJTextPane.setText(StableData.NODE\_EXEC\_ERROR);

rightBotJTextPane.validate();

}

}

node= node.next;

}

rightBotJTextPane.setText(StableData.NODE\_EXEC\_SUCCESS);

rightBotJTextPane.validate();

}

});

show.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

LinkNode node= new LinkNode();

first.first= new Sort().sort(first.first);

node= first.first;

while(node!= null){

if(node.name.equals(canvas.currentNodeName)&&node.ID==canvas.currentNodeID

&& node.primaryKey.equals(canvas.currentNodePrimaryKey)){

if(!node.thisFace.showed){

try {

node.thisFace.view(rightBotJTextPane);

node.thisFace.thisView.setLocation(node.x, node.y);

node.thisFace.thisView.setSize(500, 500);

node.thisFace.thisView.setResizable(true);

node.thisFace.thisView.jsp.setBounds(0, 0, node.thisFace.thisPanel.getWidth()-10

, node.thisFace.thisPanel.getHeight()-45);

node.thisFace.thisView.jp.setPreferredSize(new Dimension(800,600));

node.thisFace.thisView.setVisible(true);

node.thisFace.thisView.validate();

} catch (Exception e1) {

//e1.printStackTrace();

rightBotJTextPane.setText(StableData.NODE\_INSPECT\_ERROR);

rightBotJTextPane.validate();

}

}else{

node.thisFace.thisView.setVisible(true);

}

}

node=node.next;

}

rightBotJTextPane.setText(StableData.NODE\_INDICATE\_SUCCESS);

rightBotJTextPane.validate();

}

});

dnode.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

LinkNode node=new LinkNode();

first.first=new Sort().sort(first.first);

node=first.first;

while(node!=null){

if(node.name.equals(canvas.currentNodeName)&&node.ID== canvas.currentNodeID

&& node.primaryKey.equalsIgnoreCase(canvas.currentNodePrimaryKey) ){

first.first= first.deletNode(first.first, node.name, node.ID,

node.primaryKey);

new UpdateRelatedLine(first.first, canvas.currentNodeName,

canvas.currentNodeID

, canvas.currentNodePrimaryKey);

}

node= node.next;

}

canvas.repaint();

}

});

dline.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(ActionEvent e) {

LinkNode node=new LinkNode();

first.first=new Sort().sort(first.first);

node=first.first;

while(node!=null){

if(node.beconnect&&node.name.equals(canvas.currentNodeName)&& node.ID==canvas.currentNodeID

&& node.primaryKey.equals(canvas.currentNodePrimaryKey)){

node.beconnect=false;

node.tBeconnect=false;

node.mBeconnect=false;

node.dBeconnect=false;

}

node= node.next;

}

canvas.repaint();

}

});

}

public void init(){

try {

CreatMap();

} catch (IOException e) {

e.printStackTrace();

}

Registrar();

this.resize(w,h);

}

public void init(Object[][] tableData\_old,JTextPane text){

try {

this.text= text;

this.tableData\_old= tableData\_old;

CreatMap();

} catch (IOException e) {

e.printStackTrace();

}

Registrar();

this.resize(w,h);

}

private void CreatMap() throws IOException {

w= 1446- 130;

h= 820- 110;

getContentPane().setLayout(null);

UIManager.put("SplitPaneUI", "org.LYG.GUI.platForm.UnicornSplitPaneUI");

UIManager.put("ScrollBarUI", "org.LYG.GUI.platForm.UnicornScrollBarUI");

UIManager.put("TreeUI", "org.LYG.GUI.platForm.UnicornTreeUI");

currentNodeName= new String("");

first= new LinkList();

nodeInfo= new NodeInfo();

nodeView= new NodeShow(this.tableData\_old, this.text);

nodeView.tree.setBackground(Color.white);

nodeView.setBounds(10, 168, 137, 222);

nodeProject= new NodeProject();

nodeProject.setBounds(10, 38, 137, 124);

mainSplitPane = new UnicornJSplitPane();

mainSplitPane.setAutoscrolls(true);

//mainSplitPane.setEnabled(false);//

mainSplitPane.setBounds(10, 50, w-20, h-80);

mainSplitPane.setVisible(true);

getContentPane().add(mainSplitPane);

leftSplitPane= new UnicornJSplitPane();

leftSplitPane.setOrientation(JSplitPane.VERTICAL\_SPLIT);

mainSplitPane.setLeftComponent(leftSplitPane);

leftSplitPane.setLeftComponent(nodeProject);

leftSplitPane.setRightComponent(nodeView);

rightSplitPane= new UnicornJSplitPane();

rightSplitPane.setOrientation(JSplitPane.VERTICAL\_SPLIT);

mainSplitPane.setRightComponent(rightSplitPane);

righttopSplitPane= new UnicornJSplitPane();

rightSplitPane.setLeftComponent(righttopSplitPane);

rightBotJTextPane= new JTextPane();

rightBotJTextPane.setText("你好，亲~");

nodeMenu= new PopupMenu();

canvas= new ThisCanvas(threadApplet, first, nodeView, nodeMenu, rightBotJTextPane);

canvas.setPreferredSize(new Dimension(1500,1000));

canvas.setEnabled(true);

righttopScrollPane= new JScrollPane();

righttopScrollPane.setViewportView(canvas);

righttopSplitPane.setLeftComponent(righttopScrollPane);

rightrightScrollPane= new JScrollPane();

righttopSplitPane.setRightComponent(nodeInfo);

rightdownScrollPane= new JScrollPane(rightBotJTextPane);

rightSplitPane.setRightComponent(rightdownScrollPane);

popupMenu= new PopupMenu();

menuItem= new MenuItem();

menuItem.setLabel("add");

popupMenu.add(menuItem);

configre= new MenuItem();

configre.setLabel("配置");

run= new MenuItem();

run.setLabel("运行");

show= new MenuItem();

show.setLabel("显示");

dnode= new MenuItem();

dnode.setLabel("删除该节");

dline= new MenuItem();

dline.setLabel("删除链接");

nodeMenu.add(configre);

nodeMenu.add(run);

nodeMenu.add(show);

nodeMenu.add(dnode);

nodeMenu.add(dline);

getContentPane().add(popupMenu);

getContentPane().add(nodeMenu);

engineMenu= new PopupMenu();

load= new MenuItem();

load.setLabel(StableData.CONFIG\_LOAD);

save= new MenuItem();

save.setLabel(StableData.CONFIG\_UPDATE);

saveAs= new MenuItem();

saveAs.setLabel(StableData.CONFIG\_SAVE);

delete= new MenuItem();

delete.setLabel(StableData.CONFIG\_DELETE);

engineMenu.add(load);

engineMenu.add(save);

engineMenu.add(saveAs);

engineMenu.add(delete);

getContentPane().add(engineMenu);

getContentPane().setVisible(true);

}

public void actionPerformed(ActionEvent arg0) {}

public void itemStateChanged(ItemEvent arg0) {}

public void mouseClicked(MouseEvent arg0) {}

public void mouseEntered(MouseEvent arg0) {}

public void mouseExited(MouseEvent arg0) {}

public void mousePressed(MouseEvent arg0) {}

public void mouseReleased(MouseEvent arg0) {

TreePath path = nodeView.tree.getPathForLocation(arg0.getX(), arg0.getY());

if (path != null){

nodeView.tree.setSelectionPath(path);

if (arg0.getButton() == 3){

popupMenu.show(nodeView.tree, arg0.getX(), arg0.getY());

}else {

engineMenu.show(canvas, 0, 0);

}

}else {

engineMenu.show(canvas, 0, 0);

}

}

public void mouseDragged(MouseEvent arg0) {}

public void mouseMoved(MouseEvent arg0) {}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.Flash;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.MenuItem;

import java.awt.PopupMenu;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextPane;

import org.LYG.GUI.nodeEdit.CheckRange;

import org.LYG.GUI.nodeEdit.ChooseCheck;

import org.LYG.GUI.nodeEdit.DrawArrow;

import org.LYG.GUI.nodeEdit.DrawFlashSide;

import org.LYG.GUI.nodeEdit.DynamicLineUpdater;

import org.LYG.GUI.nodeEdit.LinkList;

import org.LYG.GUI.nodeEdit.LinkNode;

import org.LYG.GUI.nodeEdit.Sort;

import org.LYG.GUI.nodeInfo.NodeInfo;

import org.LYG.GUI.nodeProject.NodeProject;

import org.LYG.GUI.nodeView.NodeShow;

import org.LYG.GUI.platForm.UnicornJSplitPane;

import org.LYG.sets.stable.StableData;

public class ThisCanvas extends JPanel implements MouseMotionListener

, MouseListener, ItemListener, ActionListener, Runnable{

private static final long serialVersionUID = 1L;

public Thread threadApplet;

public String fileCurrentpath;

public int w, h;

public int flash= 0;

public int count= 0;

public String currentNodeName;

public int currentNodeID;

public String currentNodePrimaryKey;

public LinkList first;

public int currentx, currenty;

public int choose= 0;

public int oldx, oldy;

public int newx, newy;

public int isOperation= 0;

public String treeNodeName;

public NodeShow nodeView;

public NodeProject nodeProject;

public NodeInfo nodeInfo;

public UnicornJSplitPane mainSplitPane;

public UnicornJSplitPane leftSplitPane;

public UnicornJSplitPane rightSplitPane;

public UnicornJSplitPane righttopSplitPane;

public JScrollPane righttopScrollPane;

public JScrollPane rightdownScrollPane;

public JScrollPane rightrightScrollPane;

public JTextPane rightBotJTextPane;

public PopupMenu popupMenu, nodeMenu, itemMenu, engineMenu;

public MenuItem save, saveAs, delete, load;

public MenuItem menuItem;

public MenuItem configre, run, show, dnode, dline;

public ThisCanvas(Thread threadApplet, LinkList first, NodeShow nodeView

, PopupMenu nodeMenu, JTextPane rightBotJTextPane){

this.setLayout(null);

this.addMouseListener(this);

this.addMouseMotionListener(this);

this.start();

this.setOpaque(false);

this.threadApplet= threadApplet;

this.first= first;

this.nodeView= nodeView;

this.nodeMenu= nodeMenu;

this.rightBotJTextPane= rightBotJTextPane;

}

@SuppressWarnings(StableData.TAG\_DEPRECATION)

public void run() {

while(true){

try{

Thread.sleep(1000);

this.updateUI();

}catch (InterruptedException e) {

threadApplet.destroy();

e.printStackTrace();

}

}

}

public void start(){

if(null== threadApplet){

threadApplet = new Thread(this);

threadApplet.start();

}

}

@SuppressWarnings("deprecation")

public void stop() {

threadApplet.destroy();

}

public void actionPerformed(ActionEvent arg0) {}

public void itemStateChanged(ItemEvent arg0) {}

public void mouseClicked(MouseEvent arg0) {}

public void mouseEntered(MouseEvent arg0) {}

public void mouseExited(MouseEvent arg0) {}

public void mousePressed(MouseEvent arg0) {

isOperation = 1;

oldx = arg0.getX();

oldy = arg0.getY();

currentx = arg0.getX();

currenty = arg0.getY();

LinkNode node= new ChooseCheck().chooseCheckNode(first.first, arg0);

currentNodeName = node.name;

currentNodeID = node.ID;

currentNodePrimaryKey = node.primaryKey;

rightBotJTextPane.setText("坐标位："+arg0.getX()+"|"+arg0.getY());

rightBotJTextPane.validate();

}

public void mouseReleased(MouseEvent arg0){

isOperation = 0;

currentx = arg0.getX();

currenty = arg0.getY();

LinkNode node = first.first;

while(node != null){

if(node.rightChoose && !node.leftChoose){

if(oldx == arg0.getX()&&oldy == arg0.getY()){

nodeMenu.show(this, arg0.getX(), arg0.getY());

}

else{

new CheckRange(first.first, node,arg0);

}

}

node.setchoose(false);

node.rightChoose = false;

node = node.next;

}

}

public void mouseDragged(MouseEvent e) {

isOperation=1;

try {

Thread.sleep(100);

} catch (InterruptedException e1) {

e1.printStackTrace();

}

currentx= e.getX();

currenty= e.getY();

first.first= new Sort().sort(first.first);

LinkNode node= first.first;

Graphics g= getGraphics();

Graphics2D g2= (Graphics2D)g;

g2.setColor(Color.black);

while(null!= node){

if(node.leftChoose&& !node.rightChoose){

node.setxy(e.getX(), e.getY());

new DynamicLineUpdater().exec(first.first, node);

}

if(!node.leftChoose&&node.rightChoose){

new DrawArrow(g2,oldx, oldy, e.getX(), e.getY());

}

node= node.next;

this.update(g);

g.dispose();

}

}

public void mouseMoved(MouseEvent arg0) {

}

public void paint(Graphics g){

nodeView.validate();

Graphics2D g2= (Graphics2D)g;

g2.clearRect(0, 0, this.getWidth(), this.getHeight());

first.first = new Sort().sort(first.first);

LinkNode node= first.first;

while(node!= null){

if(node.x< 0){

node.x= 10;

}

if(node.x> (this.getWidth()-100)){

node.x= this.getWidth()-100;

}

if(node.y < 0){

node.y = 10;

}

if(node.y > (this.getHeight()-100)){

node.y = this.getHeight()-100;

}

g.drawImage(node.thisFace.thisImage, node.x+19, node.y+12, this);

if(node.flash > 100){

node.flash = 0;

}

if(0 == isOperation) {

new DrawFlashSide(g2, node.x, node.y, node.flash++ % 3);

}else {

new DrawFlashSide(g2, node.x, node.y, node.flash);

}

g2.setColor(Color.black);

g.drawString(node.name + "->" + node.ID,node.x - 5, node.y-20);

g2.setColor(new Color(25, 25, 112));

if(node.beconnect){

if(node.tBeconnect){

new DrawArrow(g2, node.tBeconnectX+62, node.tBeconnectY+28, node.x+14, node.y-6);

if(!node.leftChoose&&node.rightChoose){

g2.setColor(Color.black);

new DrawArrow(g2, oldx, oldy, currentx, currenty);

g2.setColor(new Color(25,25,112));

}

}

if(node.mBeconnect){

new DrawArrow(g2, node.mBeconnectX+62, node.mBeconnectY+28, node.x-4, node.y+25);

if(!node.leftChoose&& node.rightChoose){

g2.setColor(Color.black);

new DrawArrow(g2, oldx, oldy, currentx, currenty);

g2.setColor(new Color(25, 25, 112));

}

}

if(node.dBeconnect){

new DrawArrow(g2, node.dBeconnectX+ 62, node.dBeconnectY+ 28, node.x+ 6, node.y+ 55);

if(!node.leftChoose&& node.rightChoose){

g2.setColor(Color.black);

new DrawArrow(g2, oldx, oldy, currentx, currenty);

g2.setColor(new Color(25, 25, 112));

}

}

}else if(!node.leftChoose&& node.rightChoose){

g2.setColor(Color.black);

new DrawArrow(g2, oldx, oldy, currentx, currenty);

g2.setColor(new Color(25, 25, 112));

}

node = node.next;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeEdit;

**import** java.awt.event.MouseEvent;

**public** **class** CheckRange{

**public** CheckRange(LinkNode first,LinkNode node, MouseEvent arg0) {

LinkNode linkNode=first;

**int** x,y;

x= arg0.getX();

y= arg0.getY();

**while**(**null** != linkNode){

**if**((x>linkNode.x-20)&& (x<linkNode.x+100)&& (y>linkNode.y-100)&& (y<linkNode.y+16)

&& (!node.primaryKey.equalsIgnoreCase(linkNode.primaryKey))){

linkNode.beconnect= **true**;

linkNode.tBeconnect= **true**;

linkNode.tBeconnectX= node.x;

linkNode.tBeconnectY= node.y;

linkNode.tBeconnectID= node.ID;

linkNode.tBeconnectPrimaryKey= node.primaryKey;

linkNode.tBeconnetName= **new** String(node.name);

**return**;

}

**if**((x>linkNode.x-20)&& (x<linkNode.x+50)&& (y>linkNode.y+16)&& (y<linkNode.y+32)

&& (!node.primaryKey.equalsIgnoreCase(linkNode.primaryKey))){

linkNode.beconnect= **true**;

linkNode.mBeconnect= **true**;

linkNode.mBeconnectX= node.x;

linkNode.mBeconnectY= node.y;

linkNode.mBeconnectID= node.ID;

linkNode.mBeconnectPrimaryKey= node.primaryKey;

linkNode.mBeconnetName= **new** String(node.name);

**return**;

}

**if**((x>linkNode.x-20)&& (x<linkNode.x+50)&& (y>linkNode.y+32)&& (y<linkNode.y+100)

&& (!node.primaryKey.equalsIgnoreCase(linkNode.primaryKey))){

linkNode.beconnect= **true**;

linkNode.dBeconnect= **true**;

linkNode.dBeconnectX= node.x;

linkNode.dBeconnectY= node.y;

linkNode.dBeconnectID= node.ID;

linkNode.dBeconnectPrimaryKey= node.primaryKey;

linkNode.dBeconnetName= **new** String(node.name);

**return**;

}

linkNode= linkNode.next;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeEdit;

**import** java.awt.\*;

**public** **class** DrawArrow{

**public** DrawArrow(Graphics2D g2, **int** x, **int** y, **int** connectX, **int** connectY) {

x+= 10;

connectX-= 10;

g2.setStroke(**new** BasicStroke(2, BasicStroke.***CAP\_SQUARE***, BasicStroke.***JOIN\_ROUND***));

drawCurve(g2, x, y, connectX, connectY, 6);

DrawSinLine.*drawHead*(connectX-8, connectY-3, g2);

}

**private** **void** drawCurve(Graphics2D g2, **int** x, **int** y, **int** connectX, **int** connectY, **double** scale) {

**double** distanceX = Math.*abs*(x - connectX);

**double** distanceY = Math.*abs*(y - connectY);

**double** signOfPointX = (x - connectX < 0)? 1: -1;

**double** signOfPointY = (y - connectY < 0)? 1: -1;

**double** averageOfDistanceY = (distanceX == 0)?0: distanceY/distanceX;

**double** signOfPointYWithaverageOfDistanceY= averageOfDistanceY\*signOfPointY;

**double** oldRegisterY=0;

**boolean** firstTime= **true**;

**if**(signOfPointX == 1) {

**for**(**int** c = 0, i = x; i < connectX - 16; c+= 8, i+= 8) {

**double** registerY = y + signOfPointYWithaverageOfDistanceY \* c + scale

\* Math.*sin*(averageOfDistanceY \* c / 6);

g2.drawLine(i, **true**== firstTime? (**int**)registerY: (**int**)oldRegisterY

, i+8, (**int**)registerY);

oldRegisterY= registerY;

firstTime= **false**;

}

}

**if**(signOfPointX == -1) {

**for**(**int** c = 0, i = x; i > connectX + 2; c+= 8, i-= 8) {

**double** registerY = y + signOfPointYWithaverageOfDistanceY \* c + scale

\* Math.*sin*(averageOfDistanceY \* c / 6 );

g2.drawLine(i, **true**== firstTime? (**int**)registerY: (**int**)oldRegisterY

, i-8, (**int**)registerY);

oldRegisterY=registerY;

firstTime= **false**;

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeEdit;

import java.awt.Color;

import java.awt.Graphics2D;

public class DrawFlashSide{

public DrawFlashSide(Graphics2D g2, int x, int y, int flash) {

if(0 >= flash){

g2.setColor(Color.blue);

DrawSinLine.drawCosLine(x, y , g2);

g2.setColor(Color.pink);

DrawSinLine.drawSinLine(x, y , g2);

}

if(1 == flash){

g2.setColor(Color.ORANGE);

DrawSinLine.drawCosLine(x, y , g2);

g2.setColor(Color.blue);

DrawSinLine.drawSinLine(x, y , g2);

}

if(2 <= flash){

g2.setColor(Color.ORANGE);

DrawSinLine.drawCosLine(x, y , g2);

g2.setColor(Color.RED);

DrawSinLine.drawSinLine(x, y , g2);

}

drawConnect(g2, x, y);

}

//for cell postfix

private void drawConnect(Graphics2D g2, int x, int y) {

g2.drawOval(x + 10, y - 8, 4, 4);

g2.drawOval(x - 8, y + 22, 4, 4);

g2.drawOval(x + 2, y + 52, 4, 4);

g2.drawOval(x + 62, y + 26, 4, 4);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeEdit;

import java.awt.Graphics2D;

import org.LYG.GUI.theme.neroCell.DrawArrowHead;

import org.LYG.GUI.theme.neroCell.DrawNeroCellMask31;

import org.LYG.GUI.theme.neroCell.DrawNeroCellMask32;

public class DrawSinLine{

public static void drawCosLine(int x0, int y0, Graphics2D g2) {

for(int y = 0; y < DrawNeroCellMask31.neroShape.length; y++) {

for(int x = 0; x < DrawNeroCellMask31.neroShape[0].length; x++) {

if(1 == DrawNeroCellMask31.neroShape[y][x]) {

g2.drawLine(x + x0, y + y0, x + x0, y + y0);

}

}

}

}

public static void drawSinLine(int x0, int y0, Graphics2D g2) {

for(int y = 0; y < DrawNeroCellMask32.neroShape.length; y++) {

for(int x = 0; x < DrawNeroCellMask32.neroShape[0].length; x++) {

if(1 == DrawNeroCellMask32.neroShape[y][x]) {

g2.drawLine(x + x0, y + y0, x + x0, y + y0);

}

}

}

}

public static void drawHead(int x0, int y0, Graphics2D g2) {

for(int y = 0; y < DrawArrowHead.neroShape.length; y++) {

for(int x = 0; x < DrawArrowHead.neroShape[0].length; x++) {

if(1 == DrawArrowHead.neroShape[y][x]) {

g2.drawLine(x + x0, y + y0, x + x0, y + y0);

}

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeEdit;

**public** **class** DynamicLineUpdater{

**public** **void** exec(LinkNode first,LinkNode node){

LinkNode linkNode= first;

**while**(**null** != linkNode) {

**if**(linkNode.primaryKey.equalsIgnoreCase(node.tBeconnectPrimaryKey)){

node.tBeconnectX= linkNode.x;

node.tBeconnectY= linkNode.y;

}

**if**(linkNode.tBeconnectPrimaryKey.equalsIgnoreCase(node.primaryKey)){

linkNode.tBeconnectX= node.x;

linkNode.tBeconnectY= node.y;

}

**if**(linkNode.primaryKey.equalsIgnoreCase(node.mBeconnectPrimaryKey)){

node.mBeconnectX= linkNode.x;

node.mBeconnectY= linkNode.y;

}

**if**(linkNode.mBeconnectPrimaryKey.equalsIgnoreCase(node.primaryKey)){

linkNode.mBeconnectX= node.x;

linkNode.mBeconnectY= node.y;

}

**if**(linkNode.primaryKey.equalsIgnoreCase(node.dBeconnectPrimaryKey)){

node.dBeconnectX= linkNode.x;

node.dBeconnectY= linkNode.y;

}

**if**(linkNode.dBeconnectPrimaryKey.equalsIgnoreCase(node.primaryKey)){

linkNode.dBeconnectX= node.x;

linkNode.dBeconnectY= node.y;

}

linkNode= linkNode.next;

}

linkNode = **null**;

}

**public** DynamicLineUpdater() {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeEdit;

import java.io.IOException;

import org.LYG.GUI.OSGI.\*;

public class LinkList{

int index= 0;

String key;

public LinkNode first;

public int sum\_of\_nude= 0;

public LinkList(){}

public boolean search(LinkNode linkNode, String key){

if(null== linkNode){

return false;

}

if(linkNode.name.equals(key)){

return true;

}

while(null != linkNode.next){

linkNode=linkNode.next;

if(linkNode.name.equals(key)){

while(null != linkNode.pre){

linkNode=linkNode.pre;

}

return true;

}

}

return false;

}

public LinkNode addNodeOnlyWithFace(LinkNode linkNode, NodeOSGI nOSGI)

throws CloneNotSupportedException, InstantiationException

, IllegalAccessException, IOException {

NodeOSGI currentOSGI= nOSGI;

while(null!= currentOSGI && null!= currentOSGI.pre){

currentOSGI= currentOSGI.pre;

}

if(null!= linkNode){

while(null!= currentOSGI){

if(currentOSGI.thisName.equals(linkNode.name)){

linkNode.thisFace= currentOSGI.currentFace.luoyaoguang();

sum\_of\_nude++;

index++;

return linkNode;

}

currentOSGI= currentOSGI.next;

}

}

index++;

sum\_of\_nude++;

return linkNode;

}

public LinkNode addNode(LinkNode linkNode, String treeNodeName,int x,int y,NodeOSGI nOSGI )

throws CloneNotSupportedException, InstantiationException, IllegalAccessException

, IOException {

NodeOSGI currentOSGI= nOSGI;

while(null!= currentOSGI && null!= currentOSGI.pre){

currentOSGI= currentOSGI.pre;

}

if(null== linkNode){

while(null!= currentOSGI){

if(currentOSGI.thisName.equals(treeNodeName)){

linkNode= new LinkNode();

linkNode.addName(treeNodeName,x,y,++index);

linkNode.thisFace= currentOSGI.currentFace.luoyaoguang();

linkNode.next= null;

linkNode.pre= null;

sum\_of\_nude++;

return linkNode;

}

currentOSGI=currentOSGI.next;

}

}

while(null!= linkNode.next){

linkNode = linkNode.next;

}

while(null!= currentOSGI){

if(currentOSGI.thisName.equals(treeNodeName)){

//linkNode=new linkNode();

LinkNode node = new LinkNode();

node.addName(treeNodeName, x, y, ++index);

node.thisFace= currentOSGI.currentFace.luoyaoguang();

node.pre= linkNode;

linkNode.next= node;

sum\_of\_nude ++;

return linkNode;

}

currentOSGI = currentOSGI.next;

}

while(null != linkNode.pre){

linkNode = linkNode.pre;

}

sum\_of\_nude++;

return linkNode;

}

public LinkNode deletNode(LinkNode linkNode, String name, int ID, String primaryKey){

if(null!= linkNode){

if(linkNode.name.equals(name)&& linkNode.ID== ID

&& linkNode.primaryKey.equalsIgnoreCase(primaryKey)){

if(null!= linkNode.next){

linkNode= linkNode.next;

linkNode.pre= null;

return linkNode;

}

if(null== linkNode.next){

linkNode= null;

return linkNode;

}

}

while(null!= linkNode.next){

linkNode = linkNode.next;

if(linkNode.name.equals(name)&& linkNode.ID== ID

&& linkNode.primaryKey.equalsIgnoreCase(primaryKey)){

if(null!= linkNode.next){

@SuppressWarnings("unused")

LinkNode node= linkNode;

linkNode= linkNode.next;

linkNode.pre= linkNode.pre.pre;

linkNode.pre.next= linkNode;

node= null;

linkNode= new Sort().sort(linkNode);

return linkNode;

}

if(null== linkNode.next){

linkNode= linkNode.pre;

linkNode.next= null;

linkNode= new Sort().sort(linkNode);

return linkNode;

}

}

}

}

return linkNode;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeEdit;

**import** org.LYG.GUI.OSGI.ObjectInterface;

**public** **class** LinkNode **extends** Thread{

**public** String primaryKey= "";

**public** **int** flash= 0;

**public** Boolean beconnect;

**public** Boolean leftChoose;

**public** Boolean rightChoose;

**public** Boolean tBeconnect;

**public** **int** tBeconnectX;

**public** **int** tBeconnectY;

**public** String tBeconnetName;

**public** String tBeconnectPrimaryKey= "";

**public** **int** tBeconnectID;

**public** Boolean mBeconnect;

**public** **int** mBeconnectX;

**public** **int** mBeconnectY;

**public** String mBeconnetName;

**public** String mBeconnectPrimaryKey= "";

**public** **int** mBeconnectID;

**public** Boolean dBeconnect;

**public** **int** dBeconnectX;

**public** **int** dBeconnectY;

**public** String dBeconnetName;

**public** String dBeconnectPrimaryKey= "";

**public** **int** dBeconnectID;

**public** String name;

**public** LinkNode pre;

**public** LinkNode next;

**public** **int** ID;

**public** **int** x, y;

**public** ObjectInterface thisFace;

**public** LinkNode(){}

**public** **void** addName(String thisName, **int** x1,**int** y1,**int** id1){

beconnect= **false**;

rightChoose= **false**;

leftChoose= **false**;

tBeconnect= **false**;

mBeconnect= **false**;

dBeconnect= **false**;

x= x1;

y= y1;

name= **new** String(thisName);

ID= id1;

tBeconnectPrimaryKey= "";

mBeconnectPrimaryKey= "";

dBeconnectPrimaryKey= "";

tBeconnectID= 0;

mBeconnectID= 0;

dBeconnectID= 0;

primaryKey=""+ Math.*random*();

}

**public** **void** setxy(**int** x1,**int** y1){

x= x1;

y= y1;

}

**public** **void** setchoose(Boolean choose){

leftChoose= choose;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeEdit;

**public** **class** UpdateRelatedLine{

**public** UpdateRelatedLine(LinkNode first, String currentNodeName

, **int** currentNodeID, String currentNodePrimaryKey){

first = **new** Sort().sort(first);

**while**(**null**!= first) {

**if**(first.tBeconnect&& first.tBeconnetName.equals(currentNodeName)

&& first.tBeconnectID==currentNodeID

&& first.tBeconnectPrimaryKey.equalsIgnoreCase(currentNodePrimaryKey)){

first.tBeconnect= **false**;

}

**if**(first.mBeconnect&& first.mBeconnetName.equals(currentNodeName)

&& first.mBeconnectID==currentNodeID

&& first.mBeconnectPrimaryKey.equalsIgnoreCase(currentNodePrimaryKey)){

first.mBeconnect= **false**;

}

**if**(first.dBeconnect&& first.dBeconnetName.equals(currentNodeName)

&& first.dBeconnectID==currentNodeID

&& first.dBeconnectPrimaryKey.equalsIgnoreCase(currentNodePrimaryKey)){

first.dBeconnect= **false**;

}

**if**(**null**== first.next) {

**break**;

}

first= first.next;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeInfo;

import javax.swing.ImageIcon;

import javax.swing.JScrollPane;

import java.awt.\*;

import javax.swing.\*;

@SuppressWarnings({"unchecked","rawtypes"})

public class NodeInfo extends JScrollPane {

private static final long serialVersionUID= 866589699634559456L;

String[] countryStrings= {"china", "ca", "denmark", "fr", "genmany"

, "india", "norway", "uk", "us"};

private ImageIcon[] images= {

new ImageIcon(this.getClass().getResource("china.gif")),

new ImageIcon(this.getClass().getResource("us.gif")),

new ImageIcon(this.getClass().getResource("denmark.gif")),

new ImageIcon(this.getClass().getResource("fr.gif")),

new ImageIcon(this.getClass().getResource("germany.gif")),

new ImageIcon(this.getClass().getResource("india.gif")),

new ImageIcon(this.getClass().getResource("norway.gif")),

new ImageIcon(this.getClass().getResource("uk.gif")),

new ImageIcon(this.getClass().getResource("ca.gif")) };

public NodeInfo() {

Integer[] intArray= new Integer[countryStrings.length];

JComboBox countryList= new JComboBox(intArray);

countryList.setMaximumRowCount(5);

for (int i= 0; i< countryStrings.length; i++) {

intArray[i]= new Integer(i);

if (images[i]!= null) {

images[i].setImage(images[i].getImage().getScaledInstance

(50, 50, Image.SCALE\_DEFAULT));

images[i].setDescription(countryStrings[i]);

}

}

countryList.removeAllItems();

for(int i=0; i<images.length; i++) {

countryList.addItem(images[i]);

}

this.setViewportView(countryList);

this.validate();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeProject;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import javax.swing.ImageIcon;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

public class NodeProject extends JScrollPane {

private static final long serialVersionUID = 866589699634559456L;

private ImageIcon images;

public Image newimg;

public MyPanel jPanel;

public Image img;

public NodeProject() {

images= new ImageIcon(this.getClass().getResource("LUO.jpg"));

img= images.getImage();

jPanel= new MyPanel();

jPanel.repaint();

this.setViewportView(jPanel);

}

public class MyPanel extends JPanel {

public Image newimg;

private static final long serialVersionUID = 1L;

public MyPanel(){

setLayout(null);

}

public void paint(Graphics g) {

((Graphics2D) g).drawImage(newimg, 0, 0, this);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.nodeView;

**public** **class** CacuString {

**public** String cauString(String tr){

String currentstr = **new** String("");

**if**(tr.equals("Node")){**return** **null**;}

**char**[] a = **new** **char**[tr.length()];

**for**(**int** i = 0;i < tr.length(); i++) {

a[i] = tr.charAt(i);

}

**for**(**int** i = 0;i < tr.length(); i++){

**if**(a[i] == 't' && a[i + 1] == 'e' && a[i + 2] == 'x' && a[i + 3] == 't'){

**for**(**int** j = i + 5; a[j] != ','; j++){

currentstr = currentstr + a[j];

}

**return** currentstr;

}

}

**return** currentstr;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.nodeView;

import java.awt.Font;

import javax.swing.ImageIcon;

import javax.swing.JLabel;

import javax.swing.JScrollPane;

import javax.swing.JTextPane;

import javax.swing.JTree;

import java.awt.event.\*;

import java.io.IOException;

import javax.swing.tree.\*;

import org.LYG.GUI.OSGI.\*;

import org.LYG.GUI.extOSGI.\*;

import org.LYG.GUI.platForm.UnicornTreeCellRenderer;

public class NodeShow extends JScrollPane implements MouseListener, ItemListener, ActionListener {

private static final long serialVersionUID = 1L;

public JTree tree;

public NodeOSGI first;

public LinkOSGI link;

DefaultTreeModel treeModel;

DefaultMutableTreeNode root;

ImageIcon test;

public String labelname;

JTextPane text;

Object[][] tableData\_old;

public NodeShow(Object[][] tableData\_old, JTextPane text) throws IOException{

this.text= text;

this.tableData\_old= tableData\_old;

link= new LinkOSGI();

first= new OSGI\_rigester(this.tableData\_old, this.text).Rigester(first, link);

DefaultMutableTreeNode root = new DefaultMutableTreeNode("Node");

treeModel= new DefaultTreeModel(root);

tree= new JTree(treeModel);

tree.setExpandsSelectedPaths(true);

tree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE\_TREE\_SELECTION);

tree.putClientProperty("JTree.lineStyle" , "Horizontal");

tree.setEditable(false);

UnicornTreeCellRenderer myCellRenderer = new UnicornTreeCellRenderer();

myCellRenderer.setFont(new Font("Serif", Font.ITALIC, 12));

tree.setCellRenderer(myCellRenderer);

DefaultMutableTreeNode BI = new DefaultMutableTreeNode("BI");

DefaultMutableTreeNode SOUND = new DefaultMutableTreeNode("SOUND");

DefaultMutableTreeNode IMAGE = new DefaultMutableTreeNode("IMAGE");

DefaultMutableTreeNode MOVIE = new DefaultMutableTreeNode("MOVIE");

root.add(BI);

root.add(SOUND);

root.add(IMAGE);

root.add(MOVIE);

if(first!=null){

if(first.currentFace.position == null){

JLabel label;

label = new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node = new DefaultMutableTreeNode(label);

root.add(node);

}

else if(first.currentFace.position.equals("BI")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

BI.add(node);

}

else if(first.currentFace.position.equals("SOUND")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

SOUND.add(node);

}

else if(first.currentFace.position.equals("IMAGE")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

IMAGE.add(node);

}

else if(first.currentFace.position.equals("MOVIE")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

MOVIE.add(node);

}

else{

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

root.add(node);

}

while(first.next!=null) {

first=first.next;

if(first.currentFace.position==null){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

root.add(node);

}

else if(first.currentFace.position.equals("BI")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

BI.add(node);

}

else if(first.currentFace.position.equals("SOUND")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

SOUND.add(node);

}

else if(first.currentFace.position.equals("MOVIE")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

MOVIE.add(node);

}

else if(first.currentFace.position.equals("IMAGE")){

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

IMAGE.add(node);

}

else{

JLabel label;

label=new JLabel();

label.setIcon(first.thisIcon);

label.setText(first.thisName);

DefaultMutableTreeNode node=new DefaultMutableTreeNode(label);

root.add(node);

}

}

}

this.setViewportView(tree);

//add(tree);

}

public void actionPerformed(ActionEvent e) {

}

public void itemStateChanged(ItemEvent arg0) {

}

public void mouseClicked(MouseEvent arg0) {

}

public void mouseEntered(MouseEvent arg0) {

}

public void mouseExited(MouseEvent arg0) {

}

public void mousePressed(MouseEvent arg0) {

}

public void mouseReleased(MouseEvent arg0) {

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.OSGI;

**public** **class** LinkOSGI{

**public** NodeOSGI addNode(NodeOSGI currentNode, ObjectInterface currentFace) {

**if**(**null**== currentNode){

currentNode= **new** NodeOSGI();

currentNode.addName(currentFace);

currentNode.next= **null**;

currentNode.pre= **null**;

**return** currentNode;

}

**while**(currentNode.next!= **null**){

currentNode= currentNode.next;

}

NodeOSGI node= **new** NodeOSGI();

node.addName(currentFace);

node.pre= currentNode;

currentNode.next= node;

**while**(currentNode.pre!= **null**){

currentNode= currentNode.pre;

}

**return** currentNode;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.OSGI;

**import** javax.swing.ImageIcon;

**public** **class** NodeOSGI{

**public** NodeOSGI next;

**public** NodeOSGI pre;

**public** ObjectInterface currentFace;

**public** ImageIcon thisIcon;

**public** String thisName;

@Override

**public** Object clone() {

NodeOSGI obj = **null**;

**try**{

obj = (NodeOSGI)**super**.clone();

}**catch**(CloneNotSupportedException e) {

e.printStackTrace();

}

**return** obj;

}

**public** NodeOSGI(){

next=**null**;

pre=**null**;

currentFace=**null**;

thisIcon=**null**;

thisName=**null**;

}

**public** **void** addName(ObjectInterface thisface){

next=**null**;

pre=**null**;

currentFace=thisface;

thisIcon=currentFace.thisIcon;

thisName=**new** String(currentFace.thisName);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.OSGI;

import java.awt.Image;

import java.io.FileNotFoundException;

import java.io.IOException;

import javax.sound.sampled.UnsupportedAudioFileException;

import javax.swing.ImageIcon;

import javax.swing.JTextPane;

public class ObjectInterface implements Cloneable{

public ImageIcon thisIcon;

public Image thisImage;

public String thisName;

public String position;

public ObjectPanel thisPanel;

public ObjectRun thisRun;

public ObjectView thisView;

public ObjectInterface thisInterface;

public boolean showed = false;

public ObjectInterface luoyaoguang() throws CloneNotSupportedException

, IOException {

return thisInterface;

}

public ObjectInterface(){

thisIcon = null;

thisImage = null;

thisName = null;

thisPanel = new ObjectPanel();

thisRun = new ObjectRun();

thisView = new ObjectView();

}

public void config(JTextPane rightBotJTextPane) throws IOException{

}

public void execute(JTextPane rightBotJTextPane) throws FileNotFoundException

, IOException, UnsupportedAudioFileException, InterruptedException{

}

public void view(JTextPane rightBotJTextPane) throws Exception{

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.OSGI;

import java.awt.Panel;

import java.awt.ScrollPane;

import javax.swing.JFrame;

public class ObjectPanel extends JFrame implements Cloneable{

private static final long serialVersionUID = 1L;

public boolean close = false;

public ObjectPanel addr;

public ScrollPane jsp;

public String textPane;

public Panel jp;

public int h;

public int w;

protected ObjectPanel(){

}

public void config() {

}

public ObjectPanel luoyaoguang() {

return addr;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.OSGI;

import java.util.Map;

import javax.sound.sampled.AudioInputStream;

import javax.swing.JPanel;

import javax.swing.JTable;

public class ObjectRun extends JPanel implements Cloneable{

private static final long serialVersionUID = 1L;

public ObjectRun addr;

public JTable toptablein;

public Map<String, Integer> topMapIn;

public int[][] topgin;

public String topsin;

public AudioInputStream topaisin;

// public LYGFileIO toplygin;

public JTable midtablein;

public int[][] midgin;

public AudioInputStream midaisin;

// public LYGFileIO midlygin;

public JTable downtablein;

public int[][] downgin;

public AudioInputStream downaisin;

// public LYGFileIO downlygin;

public ObjectRun(){

}

@Override

public ObjectRun clone() {

return addr;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.GUI.OSGI;

import java.awt.Panel;

import java.awt.ScrollPane;

import java.awt.image.BufferedImage;

import java.util.Map;

import javax.sound.sampled.AudioInputStream;

import javax.swing.JFrame;

import javax.swing.JTable;

public class ObjectView extends JFrame implements Cloneable{

private static final long serialVersionUID = 1L;

public ObjectView addr;

public ScrollPane jsp;

public Panel jp;

public int h;

public int w;

public boolean close=false;

public JTable tableout;

public Map<String, Integer> topMapOut;

public int[][] gout;

public AudioInputStream aisout;

public AudioInputStream aiscurout;

// public LYGFileIO lygout;

public BufferedImage imageout;

public ObjectView() {

}

public void view() throws Exception{

}

public ObjectView clone() {

return addr;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.theme.neroCell;

**public** **class** DrawArrowHead{

**public** **static** **final** **int**[][] ***neroShape*** = {

{0,0,1,0,0,0,0,0,0,0},

{0,0,1,1,1,1,0,0,0,0},

{0,0,1,1,1,1,1,1,1,0},

{1,1,1,1,1,1,1,1,1,1},

{0,0,1,1,1,1,1,1,1,0},

{0,0,1,1,1,1,0,0,0,0},

{0,0,1,0,0,0,0,0,0,0},

};

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.theme.neroCell;

**public** **class** DrawNeroCellMask31{

**public** **static** **final** **int**[][] ***neroShape*** = {

{0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0},

{1,1,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0},

{1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

};

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.theme.neroCell;

**public** **class** DrawNeroCellMask32{

**public** **static** **final** **int**[][] ***neroShape*** = {

{0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{1,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0},

{1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0}, {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0}, {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1}, {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

};

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.sets.stable;

**public** **interface** StableData {

**public** **static** **final** String ***ATTENSION\_UNCURRENT\_CHOICE***= "当前没有选中文档。";

**public** **static** **final** String ***ATTENSION\_UPDATE\_ENSURE***= "确认更新在该文档:";

**public** **static** **final** String ***ATTENSION\_CANCELLED\_OPERATION***= "亲，您刚取消了当前操作~";

**public** **static** **final** String ***ATTENSION\_RECHOICE***= "不是.etl格式文档，请重新选择。";

**public** **static** **final** String ***ATTENSION\_CANCEL\_ENSURE***= "再次确认要删除吗？是否已经保存？";

**public** **static** **final** String ***ATTENSION\_DELETE***= "亲，当前ETL流删除的干干净净~";

**public** **static** **final** String ***ATTENSION\_LOAD\_ENSURE***= "再次确认要导入吗？当前已经保存？";

**public** **static** **final** String ***ATTENSION\_LOAD\_HISTORY***= "选择历史档案";

**public** **static** **final** String ***FILE\_FORMAT\_ETL***= ".etl";

**public** **static** **final** String ***NODE\_ADD\_ERROR***= "节点添加失败~请重试。";

**public** **static** **final** String ***NODE\_UPDATE\_ERROR***= "节点配置失败~请重试。";

**public** **static** **final** String ***NODE\_UPDATE\_SUCCESS***= "配置成功~";

**public** **static** **final** String ***NODE\_EXEC\_ERROR***= "节点运行失败~请重试。";

**public** **static** **final** String ***NODE\_INSPECT\_ERROR***= "节点查看失败，请重试~";

**public** **static** **final** String ***NODE\_INDICATE\_SUCCESS***= "显示成功~";

**public** **static** **final** String ***NODE\_EXEC\_SUCCESS***= "运行成功~";

**public** **static** **final** String ***TAG\_DEPRECATION***= "deprecation";

**public** **static** **final** String ***TAG\_STATIC\_ACCESS***= "static-access";

**public** **static** **final** String ***TAG\_UNUSED***= "unused";

**public** **static** **final** String ***TAG\_UNCHECKED***= "unchecked";

**public** **static** **final** String ***TAG\_RAW\_TYPES***= "rawtypes";

**public** **static** **final** String ***TAG\_SERIAL***= "serial";

**public** **static** **final** String ***TAG\_RESOURCE***= "resource";

**public** **static** **final** String ***CONFIG\_LOAD***= "载入已有ETL";

**public** **static** **final** String ***CONFIG\_UPDATE***= "保存并更新当前ETL";

**public** **static** **final** String ***CONFIG\_SAVE***= "创建一个新的文档并保存";

**public** **static** **final** String ***CONFIG\_DELETE***= "删除当前ETL";

**public** **static** **final** String ***DOC\_CREATE***= "在当前文件夹下创建一个档案名";

**public** **static** **final** String ***DOC\_EXIST***= "文档已经存在。";

**public** **static** **final** String ***MARK\_QUESTION***= "？"; //......

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.platForm;

**import** javax.swing.JSplitPane;

@SuppressWarnings("serial")

**public** **class** UnicornJSplitPane **extends** JSplitPane {

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.platForm;

**import** javax.swing.plaf.basic.BasicScrollBarUI ;

**public** **final** **class** UnicornScrollBarUI **extends** BasicScrollBarUI {

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.platForm;

**import** javax.swing.plaf.basic.BasicSplitPaneUI;

**public** **class** UnicornSplitPaneUI **extends** BasicSplitPaneUI {

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.platForm;

**import** javax.swing.tree.\*;

@SuppressWarnings("serial")

**public** **class** UnicornTreeCellRenderer **extends** DefaultTreeCellRenderer {

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.platForm;

**import** javax.swing.plaf.basic.\*;

**public** **class** UnicornTreeUI **extends** BasicTreeUI {

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.document.delete;

import javax.swing.JTextPane;

import org.LYG.GUI.Flash.ThisCanvas;

import org.LYG.GUI.nodeEdit.LinkList;

import org.LYG.GUI.nodeEdit.LinkNode;

import org.LYG.sets.stable.StableData;

public class DeleteFile{

@SuppressWarnings(StableData.TAG\_STATIC\_ACCESS)

public void delete(JTextPane rightBotJTextPane, LinkNode first, LinkList thislist, ThisCanvas canvas) {}

}

节点 部分例子代码：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.node.ai.arffTransfer;

import java.awt.\*;

import java.io.FileNotFoundException;

import java.io.IOException;

import javax.swing.\*;

import org.LYG.GUI.OSGI.\*;

public class arffTransferNodeInterface extends ObjectInterface{

public arffTransferNodeInterface() throws IOException{

thisIcon=new ImageIcon(this.getClass().getResource("1.jpg"));

thisName=new String("arffTransfer");

position=new String("BI");

Image img = ((ImageIcon) thisIcon).getImage();

Image newimg = img.getScaledInstance(30,30,java.awt.Image.SCALE\_SMOOTH );

thisImage=img.getScaledInstance(30,30,java.awt.Image.SCALE\_SMOOTH );

thisIcon = new ImageIcon(newimg);

}

public void config(JTextPane jTextPane) throws IOException{

thisView=new arffTransferNodeView();

thisRun=new arffTransferNodeRun();

thisPanel=new arffTransferNodePanel((arffTransferNodeRun) thisRun);

thisPanel.config();

showed=false;

}

public void execute(JTextPane jTextPane) throws FileNotFoundException, IOException{

((arffTransferNodeRun) thisRun).run((arffTransferNodeView) thisView);

}

public void view(JTextPane jTextPane) throws Exception{

thisView.view();

showed=true;

}

public ObjectInterface luoyaoguang() throws CloneNotSupportedException, IOException {

thisInterface = new arffTransferNodeInterface();

return thisInterface;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.node.ai.arffTransfer;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import org.LYG.GUI.OSGI.\*;

import java.awt.FileDialog;

import java.awt.Frame;

import java.awt.Panel;

import java.awt.ScrollPane;

import java.awt.Color;

public class arffTransferNodePanel extends ObjectPanel{

private static final long serialVersionUID = 1L;

private FileDialog filedialog;

public arffTransferNodePanel(final arffTransferNodeRun thisRun){

setLayout(null);

jsp = new ScrollPane();

add(jsp);

jp=new Panel();

jp.setLayout(null);

jp.setBackground(Color.white);

JButton button = new JButton("Finish");

button.setBounds(0, 0, 100, 30);

button.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

System.out.println(e.getSource());

close=true;

thisRun.value=1;

}

});

jp.add(button);

JButton readfile= new JButton("Write File");

readfile.setBounds(0, 35, 100, 65);

readfile.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent arg0) {

filedialog=new FileDialog(new Frame(),"filechoose",FileDialog.LOAD);

filedialog.setVisible(true);

thisRun.filepath=filedialog.getDirectory()+filedialog.getFile();

System.out.println(thisRun.filepath);

}

});

jp.add(readfile);

jsp.add(jp);

close=false;

}

public void config(){

System.out.println("configued");

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.node.ai.arffTransfer;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import org.LYG.GUI.OSGI.\*;

public class arffTransferNodeRun extends ObjectRun{

private static final long serialVersionUID = 1L;

public int value = 0;

public String filepath;

public arffTransferNodeRun( ) throws IOException{

}

public void run(final arffTransferNodeView thisView) throws IOException{

System.out.println("runed" + value);

System.out.println(toptablein.getModel().getValueAt(0, 0));

System.out.println("runed" + value);

File file= new File(filepath);

file.createNewFile();

BufferedWriter wr = new BufferedWriter

(new OutputStreamWriter(new FileOutputStream(file),"GBK"));

arffLink link= new arffLink();

arffNode node= new arffNode();

wr.write("@relation "+"'ARFF'"+"\n");

for(int i= 0;i< toptablein.getModel().getColumnCount(); i++){

if(toptablein.getModel().getColumnName(i).contains("String")){

wr.write("@attribute " + "'" + toptablein.getModel().getColumnName(i) + i + "'" + " {");

for(int j=0; j<toptablein.getModel().getRowCount(); j++){

Object obj = toptablein.getModel().getValueAt(j, i);

if(obj != null) {

if(!link.search(node, obj.toString())){

link.addNode(node, obj.toString());

wr.write("'"+obj.toString()+"'");

wr.write(",");

}

}

}

wr.write("}\n");

}

if(toptablein.getModel().getColumnName(i).contains("Number")){

wr.write("@attribute "+"'"+toptablein.getModel().getColumnName(i)+i+"'"+" real");

wr.write("\n");

}

if(toptablein.getModel().getColumnName(i).contains("Date")){

wr.write("@attribute "+"'"+toptablein.getModel().getColumnName(i)+i+"'"+" string");

wr.write("\n");

}

}

wr.write("@data\n");

for(int i=0; i<toptablein.getModel().getRowCount(); i++){

for(int j=0; j<toptablein.getModel().getColumnCount();j++){

if(toptablein.getModel().getColumnName(j).contains("String")

||toptablein.getModel().getColumnName(j).contains("Date")){

Object obj = toptablein.getModel().getValueAt(i, j);

if(obj != null) {

wr.write("'"+obj.toString()+"'");

wr.write(",");

}

}else{

Object obj = toptablein.getModel().getValueAt(i, j);

if(obj != null) {

wr.write(obj.toString());

wr.write(",");

}

}

}

wr.write("\n");

}

System.out.println("===完成省份：");

System.out.println("全部完成。。。。。。。。");

wr.flush();

wr.close();

thisView.tableout=toptablein;

//thisView.out=new JTable(content,spec);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package org.LYG.node.ai.arffTransfer;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Panel;

import java.awt.ScrollPane;

import javax.swing.JButton;

import javax.swing.JScrollPane;

import org.LYG.GUI.OSGI.\*;

public class arffTransferNodeView extends ObjectView{

private static final long serialVersionUID = 1L;

public JButton button;

public arffTransferNodeView(){

}

public void view(){

jsp = new ScrollPane();

jp=new Panel();

jp.setBackground(Color.yellow);

JScrollPane j=new JScrollPane();

tableout.setBackground(new Color(240, 128, 128));

tableout.setPreferredSize(new Dimension(200,200));

tableout.setVisible(true);

j.setViewportView(tableout);

jp.add(j);

jsp.add(jp);

add(jsp);

close=false;

}

@Override

public ObjectView clone() {

addr = (ObjectView)super.clone();

return addr;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.node.ai.arffTransfer;

**public** **class** arffNode{

**public** String thisName;

**public** arffNode next;

**public** arffNode pre;

**public** arffNode(){

next=**null**;

pre=**null**;

thisName=**null**;

}

**public** **void** addName(String name){

next=**null**;

pre=**null**;

thisName=name;

thisName=**new** String(name);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.node.ai.arffTransfer;

**public** **class** arffLink{

**public** **boolean** search(arffNode first2, String key){

**while**(first2 != **null** && first2.pre != **null**){

first2 = first2.pre;

}

**if**(first2 == **null** || first2.thisName == **null**){

**return** **false**;

}

**if**(first2.thisName.equals(key)){

**return** **true**;

}

**while**(first2.next != **null**){

first2 = first2.next;

**if**(first2.thisName.equals(key)){

**while**(first2.pre != **null**){

first2 = first2.pre;

}

**return** **true**;

}

}

**return** **false**;

}

**public** arffNode addNode(arffNode currentnode, String name) {

**if**(currentnode == **null**){

currentnode = **new** arffNode();

currentnode.addName(name);

currentnode.next = **null**;

currentnode.pre = **null**;

**return** currentnode;

}

**while**(currentnode.next != **null**){

currentnode = currentnode.next;

}

arffNode node = **new** arffNode();

node.addName(name);

node.pre = currentnode;

currentnode.next = node;

**while**(currentnode.pre != **null**){

currentnode = currentnode.pre;

}

**return** currentnode;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** org.LYG.GUI.theme.neroCell;

**public** **class** DrawNeroCellMask33{

**public** **static** **final** **int**[][] ***neroShape*** = { {0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1},

{1,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0},

{1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0},

{1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0},

{1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0},

{1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0},

{0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

};

}