/\*////////////////////////////////////////////////////////////////////////>>Main

./src/drivers/DatabaseMain.java

//////////////////////////////////////////////////////////////////////////////\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*A NOTE ON GENERAL COMMENTING\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* class name/ path is notated by that thing at the top of this class, if the

\* class is a main, ">>Main" will be in the top right corner (like above)

\*

\* comments depend on how log each line is

\* each line in this program is attempted to stay on one line for each page

\* for readability, pleasing graders, etc...

\* Block comments are in the Netbeans/javadoc style, and are generally put

\* before methods

\* lines that are too long in block comments are tabbed in (see ↑), (if the line

\* is independent. if the block comment is a paragraph, the comment is one

\* big block.)

\*

\* line comments are different from what you'll normally expect (maybe), but

\* it's a system I like. Instead of explaining everything, i'll give some

\* examples (i'm sure you'll understand)

\*

\* line of code 1;//comment 1

\*

\* really really really really really really really long line of code 2;

\* //comment 2

\*

\* really really really really really really really really really long line of

\* code 3;

\* //comment 3

\*

\* line of code 4;

\* //really really really really really really really really long comment 4

\*

\* line of code 5;

\* //really really really really really really really really really long

\* //comment 5

\*

\* really really really really really really really really really long line of

\* code 6;

\* //really really really really really really really really really long

\* //comment 5

\*

\* each tab is (and probably all is) four spaces wide

\*

\* a note on V 3.0 commenting: not all of the speling is checked/corected :)

\*

\* another note on comments:

\* @version Is the version number of the specific class.

\* Version number goes up if the program is modified

\* @author Is the author of the code (me) notated as "Kieda"

\* @since Is the the last day the class was edited (including comments!)

\* NOTE: good luck to anyone who is reading this source code ^\_^

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TOTAL LINES IN THIS PROJECT: 3555 and counting\*\*\*\*\*\*\*\*\*\*\*\*

\*/

package drivers;//package, etc...

// <editor-fold defaultstate="collapsed" desc="Imports">

import gui.mainframe.ChineseFrame;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.EventQueue;

import universals.LogManager;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import mechanics.database.DatabaseCreator;

import mechanics.database.DatabaseLoader;

import universals.FontFinder;

import universals.TimeManager;// </editor-fold>

/\*\*

\* Main class for the Chinese Character Finder Project

\* This is the starting positon to create a database

\* @version 1.9

\* @author Kieda

\* @since 2-26-2011

\*/

// <editor-fold defaultstate="collapsed" desc="Main">

public class DatabaseMain {

public static ChineseFrame f;//calls a new ChineseFrame. This is the main

//GUI that runs this program

/\*\*

\* main: the start of the program

\* initializes the ChineseFrame to the Database Frame

\* @param args arguements for the command line

\*/

public static void main(String args[]) {//start of main

LogManager.logThing(TimeManager.getCurrentTimeAndDate()+

"--\tPROGRAM STARTING UP");

//First logs that the program has started in the logmanager

/\*Adjusts the Default look&feel to be native to the OS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

//Sets the look & feel to be the OS default

} catch (UnsupportedLookAndFeelException e) {

LogManager.logError(e,"System LookAndFeel unsupported in java", 33

, "Main", true);

//logs and prints if the OS look and feel is unsupported

}

//prints error and logs it if there is a UnsupportedLookAndFeelException

catch (ClassNotFoundException e) {

LogManager.logError(e,"Cannot Find System's LookAndFeel", 39, "Main"

, true);

//logs and prints error if it Cannot Find the system's

//LookAndFeel

} catch (InstantiationException e) {

LogManager.logError(e,"Cannot System's Instantiate LookAndFeel", 44

, "Main");

//logs error if it cannot instantiate the system's

//LookAndFeel

} catch (IllegalAccessException e) {

LogManager.logError(e,"Cannot Access System's LookAndFeel", 49

, "Main");

//logs error if it cannot access the system's LookAndFeel

//(Probably due to restrictions, etc)

}

DatabaseCreator.init();

//initializes the DatabaseCreator so it can be added to the frame

//initialized outside of frame to prevent errors

DatabaseLoader.load();

//loads all of the Characters into the Java Virtual Machine

//increases speed

FontFinder.findFonts();

//finds and records all of the fonts that can properly

//display Chinese Characters

/\*Creates a new DatabaseCreator\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EventQueue.invokeLater(new Runnable() {

//starts a new frame in a Runnable

public void run() {

f = new ChineseFrame();

//initializes ChineseFrame

f.initFrame(ChineseFrameComponent.DATABASE);

//initialized outside of frame to prevent errors, Creates a

//database ChineseFrame

f.setVisible(true);

//makes the frame visible

}

});

}

}// </editor-fold>

/\*////////////////////////////////////////////////////////////////////////>>Main

./src/drivers/FinderMain.java

//////////////////////////////////////////////////////////////////////////////\*/

/\*

\* Confused about the commenting techniques? Go to the DatabaseMain in package

\* drivers for more clarification.

\*/

package drivers;

// <editor-fold defaultstate="collapsed" desc="Imports">

import gui.mainframe.ChineseFrame;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import universals.LogManager;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import mechanics.database.DatabaseCreator;

import mechanics.database.DatabaseLoader;

import universals.FontFinder;// </editor-fold>

import universals.TimeManager;

/\*\*

\* Main class for the Chinese IB Project

\* This is where everything begins...

\* @version 1.9

\* @author Kieda

\* @since 2-26-2011

\*/

// <editor-fold defaultstate="collapsed" desc="Main">

public class FinderMain {

public static ChineseFrame f;//calls a new ChineseFrame. This is the main

//GUI that runs this program

/\*\*

\* main: the start of the program

\* initializes the ChineseFrame to the Finder Frame

\* @param args arguements for the command line

\*/

public static void main(String args[]) {//start of main

LogManager.logThing(TimeManager.getCurrentTimeAndDate()+

"--\tPROGRAM STARTING UP");

//First logs that the program has started in the logmanager

/\*Adjusts the Default look&feel to be native to the OS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

//Sets the look & feel to be the OS default

} catch (UnsupportedLookAndFeelException e) {

LogManager.logError(e,"System LookAndFeel unsupported in java", 23

, "Main", true);

//logs and prints if the OS look and feel is unsupported

}

//prints error and logs it if there is a UnsupportedLookAndFeelException

catch (ClassNotFoundException e) {

LogManager.logError(e,"Cannot Find System's LookAndFeel", 26, "Main"

, true);

//logs and prints error if it Cannot Find the system's

//LookAndFeel

} catch (InstantiationException e) {

LogManager.logError(e,"Cannot System's Instantiate LookAndFeel", 28

, "Main");

//logs error if it cannot instantiate the system's

//LookAndFeel

} catch (IllegalAccessException e) {

LogManager.logError(e,"Cannot Access System's LookAndFeel", 30

, "Main");

//logs error if it cannot access the system's LookAndFeel

//(Probably due to restrictions, etc)

}

DatabaseCreator.init();

//initializes the DatabaseCreator so it can be added to the frame

//initialized outside of frame to prevent errors

DatabaseLoader.load();

//loads all of the Characters into the Java Virtual Machine

//increases speed

FontFinder.findFonts();

//finds and records all of the fonts that can properly

//display Chinese Characters

/\*Creates a new ChineseFrame\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

java.awt.EventQueue.invokeLater(new Runnable() {

//starts a new frame in a Runnable

public void run() {

f = new ChineseFrame();

//initializes ChineseFrame

f.initFrame(ChineseFrameComponent.FINDER);

//initialized outside of frame to prevent errors, Creates a

//finder ChineseFrame

f.setVisible(true);

//makes the frame visible

}

});

}

}// </editor-fold>

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/ChineseFrame.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe;

// <editor-fold defaultstate="collapsed" desc="imports">

import gui.mainframe.componentcreator.ChineseFrameComponents;

import gui.mainframe.componentcreator.drawpanel.InternalDrawFrame;

import gui.mainframe.componentcreator.drawpanel.InternalDrawFramePanel;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.WindowEvent;

import java.awt.event.WindowListener;

import java.io.IOException;

import java.util.ArrayList;

import javax.swing.GroupLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.LayoutStyle;

import universals.LogManager;

import universals.UniversalDataStorage;// </editor-fold>

/\*\*

\* ChineseFrame.java

\* The core GUI for the CS IB project

\* Created on Dec 27, 2010, 9:08:04 PM

\* @version BETA 3.0

\* @author Kieda

\* @since 2-25-2011

\*/

public class ChineseFrame extends JFrame implements KeyListener, WindowListener{

// <editor-fold defaultstate="collapsed" desc="declarations">

private static boolean dead = false;

//called to kill the program, i'll use this eventually

public static ChineseFrameComponents components;

//the components to be added to the frame

private static JPanel manualInputPanel;

//the manulInputPanel (panel on the left)

private static JPanel drawPanel;

//the panel which is drawn on (panel on the right)

private static JPanel bottomPanel;

//the panel that is either output or the database creation panel

//(panel on the bottom)

public boolean frametype;

//the kind of frametype used (either database or recognition)

// </editor-fold>

/\*\*

\* Constructor for the finder frame, the gui component responsibe for

\* interfacing between the user, the text writing algorithms, and the

\* dictionary

\*/

public ChineseFrame() {

super("CHINESE CHARACTER RECOGNITION BETA 3.0");

//sets the frame's title

UniversalDataStorage.intersections = new ArrayList<Integer>(1);

//initializes intersections to prevent errors

}

/\*\*

\* initializes the frame and its components

\* @param frameType the kind of frame to be used

\* (finder or database creation)

\*/

@SuppressWarnings("unchecked")

public void initFrame(boolean frameType){

frametype = frameType;// sets the frameType

addWindowListener(this);

//sets the what should happen when the frame is closed

components = new ChineseFrameComponents();

//creates the components for the ChineseFrame

components.initStuff();

//initializes the variables of the ChineseFrameComponents

components.initFinderFrameComponents();

//initializes the components in ChineseFrameComponents for use

//in the ChineseFrame

manualInputPanel = components.getManualInputPanel();

//initializes a local copy of the ManualInputPanel

drawPanel = components.getDrawPanel();

//initializes a local copy of the DrawPanel

if(frametype)//decides what to do based on the frametype given (in

//this situation, output)

bottomPanel = components.getOutputPanel();

//initializes a local copy of the BottomPanel in the outputPanel

//form

else//decides what to do based on the frametype given(in

//this situation, database)

bottomPanel = components.getDatabaseCreationPanel();

//initializes a local copy of the BottomPanel in the database

//creator form

setJMenuBar(components.getChineseMenuBar());

//sets the JMenuBar to the ChineseMenuBar

GroupLayout layout = new GroupLayout(getContentPane());

//creates a new layout based on this frame

getContentPane().setLayout(layout);

//sets his frame's layout to the GroupLayout

// <editor-fold defaultstate="collapsed" desc="template code from the

Netbeans IDE">

layout.setHorizontalGroup(//starts the horizontal grouping

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

//adds a parralel group (basically the entire container)

.addGroup(layout.createSequentialGroup()

//creates a squential group in the parallel for the drawpanel

//and the manualInputPanel (top left and top right)

.addComponent(manualInputPanel, 250, 250, 250)

//creates a maualInputPanel with the horizontal default

//size of 250

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a gap between the manualInputPanel and the drawPanel

.addComponent(drawPanel, 300, 300, Short.MAX\_VALUE)

//creates a maualInputPanel with the horizontal default

//size of 325

)//end of sequential group

.addComponent(bottomPanel, GroupLayout.Alignment.TRAILING

, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE

, Short.MAX\_VALUE)

//adds the bottomPanel beneath

//the manualInputPanel/drawPanel

);//end of horizontal group

layout.setVerticalGroup(//starts the vertical grouping

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

//adds a parralel group (basically the entire container)

.addGroup(layout.createSequentialGroup()

//creates a squential group for the drawpanel/manualInputPanel

//then finally the bottomPanel

.addGroup(layout.createParallelGroup(GroupLayout.Alignment

.TRAILING, false)

//creates a parallel group for the drawpanel and

//manualInputPanel

.addComponent(drawPanel, GroupLayout.Alignment.LEADING, 300

, 325, Short.MAX\_VALUE)

//adds the drawPanel

.addComponent(manualInputPanel, GroupLayout.Alignment

.LEADING, GroupLayout.DEFAULT\_SIZE

, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE))

//adds the outputPanel

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a a gap between the manualInputPanel/drawPanel and

//the bottomPanel

.addComponent(bottomPanel, 150, 210

, Short.MAX\_VALUE)

//adds the bottomPanel

)

);//end of vertical group

// </editor-fold>

pack();//packs all of the components in

}

/\*\*

\* kills the program

\*/

public static void kill(){

dead = true;//stops the running loops

InternalDrawFrame.kill(); //stops the drawing threads

System.exit(0);//exits the sytem

}

// <editor-fold defaultstate="collapsed" desc="key events">

public void windowOpened(WindowEvent e) {}

public void windowClosing(WindowEvent e) {

try {

LogManager.closeAllWriters();

} catch (IOException ex) {}

kill();

}

public void windowClosed(WindowEvent e) {}

public void windowIconified(WindowEvent e) {}

public void windowDeiconified(WindowEvent e) {}

public void windowActivated(WindowEvent e) {}

public void windowDeactivated(WindowEvent e) {}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/ChineseFrameActionHandler.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe;

// <editor-fold defaultstate="collapsed" desc="imports">

import gui.mainframe.componentcreator.drawpanel.InternalDrawFramePanel;

import gui.mainframe.componentcreator.options.FloatingPartsCounterPanel;

import gui.mainframe.componentcreator.options.IntersectionStringInputPanel;

import gui.mainframe.componentcreator.options.IntersectionsCrudeAmoutPanel;

import gui.mainframe.componentcreator.options.StrokeComplexityPanel;

import gui.mainframe.componentcreator.options.StrokeCrudeCountPanel;

import gui.mainframe.componentcreator.databasecreator.DatabaseCreatorPanel;

import universals.LogManager;

import java.awt.event.ActionEvent;

import java.awt.event.MouseEvent;

import java.beans.PropertyChangeEvent;

import java.util.ArrayList;

import javax.swing.event.CaretEvent;

import javax.swing.event.ChangeEvent;

import mechanics.ChineseCharacter;

import mechanics.database.DatabaseCreator;

import mechanics.database.SearchEngine;

import universals.UniversalDataStorage;// </editor-fold>

/\*\*

\* This is another "static method class" where all of the methods are static.

\* This class has a variety of events used by the ChineseFrame

\* @version 1.5

\* @author Kieda

\* @since 2-25-2011

\*/

public class ChineseFrameActionHandler extends ChineseFrame{

/\*\*

\* the event for the exit button; exits the program

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void exitMenuButtonActionPerformed(ActionEvent evt) {

exit();//passes to exit(), quits program

}

/\*\*

\* exits the program

\*/

public static void exit(){

ChineseFrame.kill();//passes to kill(), which exits the program

}

/\*\*

\* the event for the strokes counter; updates the numberOfStrokes

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void numberOfStrokesCounterStateChanged(ChangeEvent evt) {

UniversalDataStorage.numberOfStrokes = StrokeCrudeCountPanel

.getNumberOfStrokes();

//sets the numberOfStrokes

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

/\*\*

\* the event for the floating parts counter; updates the

\* numberOfFloatingParts

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void numberOfFloatingPartsCounterPropertyChange(

PropertyChangeEvent evt) {

UniversalDataStorage.numberOfFloatingParts = FloatingPartsCounterPanel

.getNumberOfFloatingParts();

//sets the numberOfFloatingParts

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

/\*\*

\* the event for the intersections total counter; updates the

\* totalNumberOfIntersections

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void totalNumberOfIntersectionsCounterPropertyChange(

PropertyChangeEvent evt) {

UniversalDataStorage.totalNumberOfIntersections =

IntersectionsCrudeAmoutPanel.getTotalNumberOfIntersections();

//sets the totalNumberOfIntersections

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

/\*\*

\* the event for mouse pressed in the internalDrawFrame;

\* updates the x and y position of the mouse

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void characterDrawInternalFrameMousePressed(MouseEvent evt) {

UniversalDataStorage.getXMouseDrawing = evt.getX();

//sets the getXMouseDrawing

UniversalDataStorage.getYMouseDrawing = evt.getY();

//sets the getYMouseDrawing

}

/\*\*

\* the event for the straight strokes counter;

\* updates the numberOfStraightStrokes

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void numberOfStraightStrokesCounterPropertyChange(

PropertyChangeEvent evt) {

UniversalDataStorage.numberOfStraightStrokes = StrokeComplexityPanel

.getNumberOfStraightStrokes();

//sets the numberOfStraightStrokes

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

/\*\*

\* the event for the complex strokes counter;

\* updates the numberOfComplexStrokes

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void numberOfComplexStrokesCounterPropertyChange(

PropertyChangeEvent evt) {

UniversalDataStorage.numberOfComplexStrokes = StrokeComplexityPanel

.getNumberOfComplexStrokes();

//sets the numberOfComplexStrokes

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

/\*\*

\* the event for the intersections JTextField; updates the intersections

\* @param evt the event that is passed on by new CaretListener(){...}

\*/

public static void intersectionsStringInputCaretUpdate(CaretEvent evt) {

if(!IntersectionStringInputPanel.getIntersectionsString().isEmpty()

&&IntersectionStringInputPanel.getIntersectionsString()

.contains(",")){//ensures the string input is valid

UniversalDataStorage.intersections = new ArrayList<Integer>();

//clears the universal intersections

String[] intersectionsInput = IntersectionStringInputPanel

.getIntersectionsString().split(",");

//splits up the string inputted to an array

for(String intersection: intersectionsInput)

//goes through all values inputted

try {

UniversalDataStorage.intersections.add(Integer.parseInt(

intersection));

//adds the integer value of the array string

//inputted to the intersections

} catch (Exception e) {LogManager.logError(e,"CANNOT PARSE " +

"STRING IN INPUT TEXTBOX", 63

, "FinderFrameActionHandler");

//logs error if someone puts in an invalid string

}

SearchEngine.searchForChineseCharacter();

//searches if the character exists

}

}

/\*\*

\* for the addtodatabase button; adds a string of line to the database,

\* preferabley an actual chinese character (otherwise errors!)

\* @param evt the event that is passed on by new ActionListener(){...}

\* @param characterToWrite the character to add to the database

\*/

public static void addToDatabaseActionEvent(ActionEvent evt

, String characterToWrite){

DatabaseCreatorPanel.setCharacterToWrite(DatabaseCreator.next());

//goes to the next character to write into the database

InternalDrawFramePanel.clearPanel();

//clears the InternalDrawFramePanel

UniversalDataStorage.character = characterToWrite.split(":")[0];

//gets the String chinese character

UniversalDataStorage.pinyin = characterToWrite.split(":")[1];

//gets the String of pinyin

DatabaseCreator.add(new ChineseCharacter(UniversalDataStorage.character

, ""//meaning- not filled out

, UniversalDataStorage.pinyin

, UniversalDataStorage.numberOfFloatingParts

, UniversalDataStorage.intersections

, UniversalDataStorage.numberOfStraightStrokes

, UniversalDataStorage.numberOfComplexStrokes));

//adds the new chinese character to the database

System.out.println(DatabaseCreatorPanel.getCharacterToWrite());

//prints out the next character to write

}

/\*\*

\* the event for the clear button; clears the InternalDrawFramePanel

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void clearActionPerformed(ActionEvent evt) {

InternalDrawFramePanel.clearPanel();

//clears the InternalDrawFramePanel

}

/\*\*

\* the event for the next button; goes to the next character for the

\* database CharacterFrame

\* @param evt the event that is passed on by new ActionListener(){...}

\*/

public static void nextButtonActionPerformed(ActionEvent evt) {

DatabaseCreatorPanel.setCharacterToWrite(DatabaseCreator.next());

//sets the next character to write

InternalDrawFramePanel.clearPanel();

//clears the InternalDrawFramePanel

System.out.println(DatabaseCreatorPanel.getCharacterToWrite());

//prints out the next character to write

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/ChineseFrameComponent.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator;

import java.awt.Component;//import

/\*\*

\* This is an interface that manages all of the components in the ChineseFrame.

\* All components that are added to the chinese frame implement this class.

\* This class also holds a few key final values for the GUI components

\* @author Kieda

\* @since 3-13-2011

\*/

public interface ChineseFrameComponent{//start of interface

/\*\*

\* The method getComponent() allows the easy addition of new GUI components

\* call getComponent() in the GUI part needed, and the GUI component will be

\* returned

\* @return the component that the class will return, usually will be

\* return this;

\*/

public Component getComponent();

public static final int deltaMouseMax = 1;

//the minimum amount that the mouse will have to move to add on another

//point in the current ChineseStroke. This allows safety in the fact

//that points won't be coninuallt added (slowing down the computer)

//This value used to be alot higher, but after deciding to 'can' the

//idea of having two individual lines intersect more than once,

//I have found that this method works alot better, and is alot more

//smooth with more points

public static final boolean DATABASE = false;

//final value to call on to initialize a database creator. This value is

//called by the main classes to decide whether or not to create a

//database or to find chinese characters. This value, DATABASE (which is

// false) is called on in ChineseFrame's initFrame(boolean frameType)

//(in package gui.mainframe) to decide the frame type

public static final boolean FINDER = true;

//final value to call on to initialize a character finder. This value is

//called by the main classes to decide whether or not to create a

//database or to find chinese characters. This value, FINDER (which is

// true) is called on in ChineseFrame's initFrame(boolean frameType)

//(in package gui.mainframe) to decide the frame type

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/ChineseFrameComponents.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator;

// <editor-fold defaultstate="collapsed" desc="imports">

import gui.mainframe.ChineseFrame;

import gui.mainframe.componentcreator.bottompanel.CharacterFieldPanel;

import gui.mainframe.componentcreator.bottompanel.PossibleChineseCharactersPanel;

import gui.mainframe.componentcreator.drawpanel.InternalDrawFrame;

import gui.mainframe.componentcreator.menubar.MenuBar;

import gui.mainframe.componentcreator.options.FloatingPartsCounterPanel;

import gui.mainframe.componentcreator.options.IntersectionStringInputPanel;

import gui.mainframe.componentcreator.options.IntersectionsCrudeAmoutPanel;

import gui.mainframe.componentcreator.options.StrokeComplexityPanel;

import gui.mainframe.componentcreator.options.StrokeCrudeCountPanel;

import gui.mainframe.componentcreator.databasecreator.DatabaseCreatorPanel;

import java.awt.Font;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JInternalFrame;

import javax.swing.JLabel;

import javax.swing.JMenuBar;

import javax.swing.JPanel;

import javax.swing.LayoutStyle;

import javax.swing.border.BevelBorder;

import mechanics.database.DatabaseCreator;// </editor-fold>

/\*\*

\* This class stores all of the components used by the ChineseFrame gui

\* This class also builds the the main components for use in the ChineseFrame

\* (the manualInputPanel, the drawPanel, and the bottomPanel)

\* @author Kieda

\* @since 3-5-2011

\*/

public class ChineseFrameComponents extends ChineseFrame{

//extends the main ChineseFrame

// <editor-fold defaultstate="collapsed" desc="component declaration">

/\* components are sorted first by JPanel they are in, and second by

\* their location

\*

\* components in a main JPanel are sorted by

\* JPanel

\* JLabel for the JPanel

\* Other components (sorted by thier location vertically)

\* please add components in this manner

\*/

private static MenuBar menuBar;

//menubar for providing basic program funtions

// <editor-fold defaultstate="collapsed" desc="top left panel">

private static JPanel manualInputPanel = new JPanel();

//manual input panel. allows a person to see if there is a

//mistake in his/her writing, and allows them to adjust

private static JLabel optionsLabel = new JLabel();

//the label for the manualInputPanel

private static FloatingPartsCounterPanel numberOfFloatingPartsPanel;

//the number of floating parts

private static IntersectionsCrudeAmoutPanel

totalNumberOfIntersectionsPanel;

//the total number of intersections, will probably replace

private static StrokeCrudeCountPanel numberOfStrokesPanel;

//the total number of intersections, will probably replace

private static StrokeComplexityPanel strokeComplexityPanel;

//the number of complex/simple strokes. updates automatically

private static IntersectionStringInputPanel

intersectionsStringInputPanel;

//the input for the intersections on each stroke.

//Does so by analyzing an inputted string

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="top right panel">

private static JPanel drawPanel = new JPanel();

//the panel that contains the InternalDrawFrame.

//will later be updated for more funtionality/user experience.

private static JLabel drawPanelLabel = new JLabel();

//the label for the drawPanel

private static Runnable drawInternalFrame;

//the internalDrawFrame: what you actually draw on.

//is a Runnable to be multithreaded to allow drawing

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="bottom panel">

//OPTION 1: Database Training

private static JPanel databaseCreationPanel;

//the panel responsible for creating the database. Already

//contains all of its componants

//OPTION 2: Character Finding

private static JPanel outputPanel = new JPanel();

//the panel which displays the possible chinese characters and

//the character text field. Will update later to add quickly

//accesible punctuation to improve user experience with a pen.

private static JLabel outputLabel = new JLabel();

//the label for the outputPanel

private static PossibleChineseCharactersPanel

possibleChineseCharactersPanel;

//a list of the possible chinese characters

private static CharacterFieldPanel characterFieldPanel;

//the character field that is wrote to when writing

//chinese characters

// </editor-fold>

// </editor-fold>

/\*\*

\* creates all of the main panels

\* call this method first to initialize all of the components at once, then

\* retrive the components by the get methods

\*/

public void initFinderFrameComponents() {

initOutputPanel();

//creates the outputPanel; which is used for finding characters

//and putting characters together in asentences

initDrawPanel();

//creates a drawPanel; which is used for drawing characters

initManualInputPanel();

//creates a manualInputPanel; which is used to manually input

//values for characters

initMenuBar();

//creates a menubar; which is useful in the UI

}

/\*\*

\* initializes all of the individual components

\* call this method first before ANYTHING ELSE to avoid errors in frame

\* initialization

\*/

public void initStuff(){

menuBar = new MenuBar();

//initializes the menuBar

// <editor-fold defaultstate="collapsed" desc="initialization of the

components in the options panel">

numberOfFloatingPartsPanel = new FloatingPartsCounterPanel();

totalNumberOfIntersectionsPanel = new IntersectionsCrudeAmoutPanel();

numberOfStrokesPanel = new StrokeCrudeCountPanel();

strokeComplexityPanel = new StrokeComplexityPanel();

intersectionsStringInputPanel = new IntersectionStringInputPanel();

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="initialization of the

components in the output panel">

possibleChineseCharactersPanel = new PossibleChineseCharactersPanel();

characterFieldPanel = new CharacterFieldPanel();

// </editor-fold>

databaseCreationPanel =new DatabaseCreatorPanel(DatabaseCreator.next());

//initializes the databaseCreationPanel, sets the first character

//to display as the first character from the DatabaseCreatorPanel

drawInternalFrame = new InternalDrawFrame();

//creates the Runnable drawInternalFrame as an InternalDrawFrame

Thread drawInternalFrameThread = new Thread(drawInternalFrame);

//creates a thread from the drawInternalFrame Runnable

drawInternalFrameThread.start();

//starts the drawInternalFrameThread, which allows the

//InternalDrawFrame to draw constantly

}

// <editor-fold defaultstate="collapsed" desc="component initialization">

/\*\*

\* Initializes the menubar used in the ChineseFrame

\* call this method first, then add it to the ChineseFrame

\* call getMenuBar() to retreive this component

\*/

public void initMenuBar() {/\*Nothing goes here, initialization currently

happens by creating a new MenuBar. Keep this for external initialization

that may need to occur\*/

}

/\*\*

\* creates an output panel

\* call this method first, then add it to the ChineseFrame

\* call getOutputPanel() to retreive this component

\*/

public void initOutputPanel() {

outputPanel.setBorder(BorderFactory

.createBevelBorder(BevelBorder.RAISED));

//puts a raised bevel border around the output panel

outputLabel.setFont(new Font("Tahoma", 0, 14));

//sets the label to go on the outputPanel to a larger size

outputLabel.setText("Output");

//sets the label's text

GroupLayout outputPanelLayout = new GroupLayout(outputPanel);

//creates a GroupLayout for the outputPanel

outputPanel.setLayout(outputPanelLayout);

//sets the outputPanel's layout to the GroupPanel

outputPanelLayout.setHorizontalGroup(//starts the horizontal grouping

outputPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)

//creates a parallel group to contain everything in the

//outputPanel

.addGroup(

outputPanelLayout.createSequentialGroup()

//creates a sequentialGroup to place the

//possibleChineseCharactersPanel and characterFieldPanel

.addContainerGap()//adds a gap on the left-most part

.addComponent(possibleChineseCharactersPanel, 100, 300, 300)

//adds the possibleChineseCharactersPanel after the gap

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a gap between the possibleChineseCharactersPanel and

//the characterFieldPanel

.addComponent(characterFieldPanel, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

//adds a resizable characterFieldPanel next

.addContainerGap()

//finally adds a container gap in the horizontal direction

)

.addGroup(

outputPanelLayout.createSequentialGroup()

//creates a sequentialGroup to place the outputLabel on

//the outputPanel

.addContainerGap()

//adds a containerGap (for label alignment)

.addComponent(outputLabel, GroupLayout.DEFAULT\_SIZE, 102

, Short.MAX\_VALUE)

//adds the outputLabel

)

);

outputPanelLayout.setVerticalGroup(//starts the vertical grouping

outputPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)

//creates a parallelGroup for containing the components

.addGroup(

outputPanelLayout.createSequentialGroup()

//creates a sequentialGroup for containing the

//outputLabel, characterFieldPanel, and

//possibleChineseCharactersPanel

.addComponent(outputLabel)

//adds the outputLabel to the top of the outputPanel

.addGroup(

outputPanelLayout.createParallelGroup(GroupLayout

.Alignment.TRAILING, true)

//adds a resizable parallel group that contains

//the possibleChineseCharactersPanel and

//characterFieldPanel

.addGroup(

outputPanelLayout.createSequentialGroup()

//adds a group to contain the characterFieldPanel

.addGap(6, 6, 6)

//adds a gap above the characterFieldPanel

.addComponent(characterFieldPanel, 50, 200

, Short.MAX\_VALUE)

//adds the characterFieldPanel

)

.addGroup(

outputPanelLayout.createSequentialGroup()

//adds a group to contain the

//possibleChineseCharactersPanel

.addPreferredGap(LayoutStyle.ComponentPlacement

.RELATED)

//adds a gap above the

//possibleChineseCharactersPanel

.addComponent(possibleChineseCharactersPanel, 50

, 200, Short.MAX\_VALUE)

//adds the possibleChineseCharactersPanel

)

)

.addContainerGap()//adds a containerGap on the bottom

)

);

}

/\*\*

\* creates a drawPanel

\* call this method first, then add it to the ChineseFrame

\* call getDrawPanel() to retreive this component

\*/

public void initDrawPanel() {

drawPanel.setBorder(BorderFactory.createBevelBorder(BevelBorder

.LOWERED));//makes a lowered bevel border around the draw panel

drawPanelLabel.setFont(new Font("Tahoma", 0, 14));

//sets the label to a larger font

drawPanelLabel.setText("Draw Panel");

//draws the label on the drawPanel

GroupLayout drawPanelLayout = new GroupLayout(drawPanel);

//creates a layout for the drawPanel

drawPanel.setLayout(drawPanelLayout);

//sets the layout for the drawpanel

drawPanelLayout.setHorizontalGroup(//starts the horizontal grouping

drawPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)

//adds a parallel group (basically the entire drawPanel)

.addGroup(drawPanelLayout.createSequentialGroup()

//group contains the drawPanel

.addGap(65, 65, 65)

//adds a gap before adding the drawPanel

.addComponent((JInternalFrame) drawInternalFrame, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout

.PREFERRED\_SIZE)

//adds the drawFrame

.addContainerGap(118, Short.MAX\_VALUE)

//adds the closing containerGap

)

.addGroup(drawPanelLayout.createSequentialGroup()

//group contains the drawPanelLabel, added to the drawPanel

.addContainerGap()

//adds a containerGap (for alignment and consistancy)

.addComponent(drawPanelLabel, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

//adds the drawPanelLabel

)

);

drawPanelLayout.setVerticalGroup(//starts the vertical grouping

drawPanelLayout.createParallelGroup(GroupLayout.Alignment.LEADING)

//adds a parallel group for basically the entire drawPanel

.addGroup(drawPanelLayout.createSequentialGroup()

//creates a sequential group for the drawPanelLabel and

//drawInternalFrame

.addComponent(drawPanelLabel)

//adds the drawPanelLabel on top

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a gap between the label and the drawInternalFrame

.addComponent((JInternalFrame) drawInternalFrame, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout

.PREFERRED\_SIZE)

//adds the drawInternalFrame to the drawPanel

.addContainerGap(23, Short.MAX\_VALUE)

//finally adds a resizable containerGap

)

);

}

/\*\*

\* creates a manualInputPanel

\* call this method first, then add it to the ChineseFrame

\* call getManualInputPanel() to retreive this component

\*/

public void initManualInputPanel() {

manualInputPanel.setBorder(BorderFactory.createBevelBorder(BevelBorder

.LOWERED));

//sets the border of the manualInputPanel to a lowered

//BevelBorder

optionsLabel.setFont(new Font("Tahoma", 0, 14));

//sets the optionsLabel's font to something larger

optionsLabel.setText("Options");

//sets the optionsLabel's text

GroupLayout manualInputPanelLayout = new GroupLayout(manualInputPanel);

//creates a new GroupLayout for the manualInputPanel

manualInputPanel.setLayout(manualInputPanelLayout);

//sets the manualInputLabel's layout

manualInputPanelLayout.setHorizontalGroup(

//start of the horizontal grouping

manualInputPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

//parallel group to allow many objects to be added that are

//veritcally aligned

.addGroup(manualInputPanelLayout.createSequentialGroup()

.addContainerGap()

//creates a sequential group for addition of components

.addGroup(

manualInputPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

//creates a parallel group to contain everything

//after the containerGap()

.addGroup(manualInputPanelLayout.createSequentialGroup()

//adds a group to contain the optionsLabel

.addComponent(optionsLabel, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

//adds a resizable optionsLabel

)

.addGroup(

manualInputPanelLayout.createSequentialGroup()

//creates a sequentialGroup to contain the other

//components

.addGroup(manualInputPanelLayout.createParallelGroup(

GroupLayout.Alignment.TRAILING, true)

//creates a parallelGroup to add the

//input components

.addComponent(intersectionsStringInputPanel,

GroupLayout.Alignment.LEADING, 100, 100,

Short.MAX\_VALUE)

//adds the intersectionsStringInputPanel

//at the top

.addComponent(numberOfFloatingPartsPanel,

GroupLayout.Alignment.LEADING, 100, 100,

Short.MAX\_VALUE)

//adds the numberOfFloatingPartsPanel after

//the intersectionsStringInputPanel

.addComponent(totalNumberOfIntersectionsPanel,

GroupLayout.Alignment.LEADING, 100, 100,

Short.MAX\_VALUE)

//adds the totalNumberOfIntersectionsPanel

//after the numberOfFloatingPartsPanel

.addComponent(strokeComplexityPanel, GroupLayout

.Alignment.LEADING, 100, 100, Short.MAX\_VALUE)

//adds the strokeComplexityPanel after

//the totalNumberOfIntersectionsPanel

.addComponent(numberOfStrokesPanel, GroupLayout

.Alignment.LEADING, 100, 100, Short.MAX\_VALUE)

//adds the numberOfStrokesPanel last

)

.addContainerGap()//the final gap

)

)

)

);

manualInputPanelLayout.setVerticalGroup(

//start of the vertical grouping

manualInputPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

.addGroup(manualInputPanelLayout.createSequentialGroup()

//creates a group to contain the components

.addComponent(optionsLabel).addPreferredGap(LayoutStyle

.ComponentPlacement.RELATED)

//adds the optionsLabel at the top

.addComponent(numberOfStrokesPanel, GroupLayout.PREFERRED\_SIZE,

GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

//adds the numberOfStrokesPanel below the optionsLabel

.addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)

//adds a gap between the numberOfStrokesPanel and the

//numberOfFloatingPartsPanel

.addComponent(numberOfFloatingPartsPanel, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

//adds the numberOfFloatingPartsPanel below the

//numberOfStrokesPanel

.addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)

//adds a gap between the numberOfFloatingPartsPanel and the

//totalNumberOfIntersectionsPanel

.addComponent(totalNumberOfIntersectionsPanel, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout

.PREFERRED\_SIZE)

//adds the totalNumberOfIntersectionsPanel below the

//numberOfFloatingPartsPanel

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a gap between the totalNumberOfIntersectionsPanel and

//the strokeComplexityPanel

.addComponent(strokeComplexityPanel, GroupLayout.PREFERRED\_SIZE,

GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

//adds the strokeComplexityPanel below the

//totalNumberOfIntersectionsPanel

.addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)

//adds a gap between the strokeComplexityPanel and the

//intersectionsStringInputPanel

.addComponent(intersectionsStringInputPanel, GroupLayout

.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

//adds the intersectionsStringInputPanel at the bottom

//of the panel

.addContainerGap()

//adds the end container gap

)

);

}// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="get components">

/\*\*

\* returns the menuBar component

\* menubar is the menubar (uppermost component) of the ChineseFrame

\* @return MenuBar component

\*/

public JMenuBar getChineseMenuBar() {

return menuBar;//returns the menuBar

}

/\*\*

\* databaseCreationPanel is the bottom panel in the ChineseFrame, in

\* Database mode

\* @return the DatabaseCreatorPanel component

\*/

public DatabaseCreatorPanel getDatabaseCreationPanel() {

return (DatabaseCreatorPanel) databaseCreationPanel;

//returns databaseCreationPanel (from type JPanel to

//DatabaseCreatorPanel)

}

/\*\*

\* outputPanel is the bottom panel in the ChineseFrame, in Finder mode

\* @return the outputPanel component

\*/

public JPanel getOutputPanel() {

return outputPanel;//returns the outputPanel

}

/\*\*

\* drawPanel is the right-most panel in the ChineseFrame

\* @return the drawPanel component

\*/

public JPanel getDrawPanel() {

return drawPanel;//returns the drawPanel

}

/\*\*

\* manualInputPanel is the left-mostcomponent in the ChineseFrame

\* @return the manualInputPanel component

\*/

public JPanel getManualInputPanel() {

return manualInputPanel;//returns the manualInputPanel

}

// </editor-fold>

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/componentcreator/bottompanel/CharacterFieldPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.bottompanel;

// <editor-fold defaultstate="collapsed" desc="imports are important">

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Component;

import java.awt.Font;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.LayoutStyle;

import universals.FontFinder;

// </editor-fold>

/\*\*

\* This class is a JPanel of the type ChineseFrameComponent.

\* This panel is used on the right-most side in the outputPanel on the

\* ChineseFrame.

\* The use of this panel is to be an output for the found character, so

\* paragraphs could be made continuoulsy.

\* @author Kieda

\* @since 3-13-2011

\*/

public class CharacterFieldPanel extends JPanel

implements ChineseFrameComponent{

private static JScrollPane characterScrollField = new JScrollPane();

//the scrollPane that contains the textArea, allowing the paragraphs

//made to be scrollable

private static JLabel characterFieldLabel = new JLabel();

//the label for the characterField

private static JTextArea characterTextField = new JTextArea();

//the editable textArea that contains the chinese characters

/\*\*

\* The constructor for the CharacterFieldPanel. This contructor builds and

\* initializes the CharacterFieldPanel

\*/

public CharacterFieldPanel() {

setBorder(BorderFactory.createEtchedBorder());

//sets the CharacterFieldPanel's border to an etchedBorder

characterFieldLabel.setText("Character Field");

//sets what the text for the CharacterFieldPanel's label

characterTextField.setColumns(19);

//sets the number of columns in the characterTextField to 19

characterTextField.setRows(4);

//sets the number of rows in the characterTextField to 4

characterTextField.setFont(new Font(FontFinder.chinesefonts.get(0)

,Font.PLAIN,20));

//sets the font in the characterTextField to a font compatible

//with Chinese character, and makes the font larger (for

//readability and user friendly-ness)

characterScrollField.setViewportView(characterTextField);

//sets the characterScrollField to display the characterTextField

GroupLayout characterFieldPanelLayout = new GroupLayout(this);

//creates a groupLayout for the CharacterFieldPanel

setLayout(characterFieldPanelLayout);

//sets CharacterFieldPanel's layout to the above grouplayout

characterFieldPanelLayout.setHorizontalGroup(

//start of horizontal group layout

characterFieldPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

//creates a parallel group to contain the sequentialGroup

.addGroup(characterFieldPanelLayout.createSequentialGroup()

//adds a sequential group to contain the characterFieldLabel

//and characterScrollField

.addGroup(characterFieldPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING)

//creates a parallel group to have the

//characterFieldLabel and characterScrollField placed

//vertically

.addGroup(characterFieldPanelLayout.createSequentialGroup()

//add a sequential group to contain the

//characterFieldLabel

.addContainerGap()

//gap for alignment

.addComponent(characterFieldLabel)

//adding the characterFieldLabel to the right of the

//gap

)

.addGroup(characterFieldPanelLayout.createSequentialGroup()

//add a sequential group to contain the

//characterScrollField

.addContainerGap()

//gap for alignment

.addComponent(characterScrollField, 50, 357, Short

.MAX\_VALUE)

//adds a resizable characterScrollField

)

)

.addContainerGap()

//finally a containerGap

)

);

characterFieldPanelLayout.setVerticalGroup(

//start of vertical group layout

characterFieldPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

//creates a parallel group to contain the sequential

//addition of the vertical components

.addGroup(characterFieldPanelLayout.createSequentialGroup()

//creates a sequential group to contain the components

.addContainerGap()

//adds a gap

.addComponent(characterFieldLabel)

//adds the characterFieldLabel below the gap

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

//adds a gap between the characterFieldLabel and the

//characterScrollField

.addComponent(characterScrollField, 50, 96, Short.MAX\_VALUE)

//adds the characterScrollField at the bottom

.addContainerGap()

//creates the final gap

)

);

}

/\*\*

\* This method adds a string to the characterTextField. This should be used

\* to add found characters to the characterTextField.

\* @param s

\*/

public static void addToList(String s){

characterTextField.append(s);

}

/\*\*

\* the required method for classes implementing ChineseFrameComponent,

\* returns the instance of this class

\* @return the CharacterFieldPanel

\*/

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/bottompanel/PossibleChineseCharacters.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.bottompanel;//package

import gui.mainframe.componentcreator.ChineseFrameComponent;

import gui.mainframe.componentcreator.drawpanel.InternalDrawFramePanel;

import java.awt.Color;

import java.awt.Component;

import java.awt.Font;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.util.ArrayList;

import javax.swing.AbstractListModel;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.LayoutStyle;

import mechanics.ChineseCharacter;

import universals.FontFinder;

/\*\*

\* This is another component implementing ChineseFrameComponent

\* @author Kieda

\* @since 3-13-2011

\*/

public class PossibleChineseCharactersPanel extends JPanel implements

ChineseFrameComponent, MouseListener{

private static JLabel possibleChineseCharactersLabel = new JLabel();

private static JList listOfPossibleChineseCharacters = new JList();

private static JScrollPane JSP;

static ArrayList<String> strings = new ArrayList<String>();

public PossibleChineseCharactersPanel() {

listOfPossibleChineseCharacters.setFont(new Font(FontFinder.chinesefonts

.get(0),Font.PLAIN,14));

setBorder(BorderFactory.createEtchedBorder());

possibleChineseCharactersLabel.setText("Possible Chinese Characters");

listOfPossibleChineseCharacters.setBorder(BorderFactory.createLineBorder

(new Color(0, 0, 0)));

listOfPossibleChineseCharacters.addMouseListener(this);

listOfPossibleChineseCharacters.setModel(new AbstractListModel() {

String[] strings = {""};

public int getSize() { return strings.length; }

public Object getElementAt(int i) { return strings[i]; }

});

JSP = new JScrollPane(listOfPossibleChineseCharacters);

GroupLayout possibleChineseCharactersPanelLayout

= new GroupLayout(this);

setLayout(possibleChineseCharactersPanelLayout);

possibleChineseCharactersPanelLayout.setHorizontalGroup(

possibleChineseCharactersPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

.addGroup(possibleChineseCharactersPanelLayout

.createSequentialGroup()

.addContainerGap()

.addGroup(possibleChineseCharactersPanelLayout

.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(possibleChineseCharactersPanelLayout

.createSequentialGroup()

.addComponent(JSP, GroupLayout.DEFAULT\_SIZE, 315

, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(possibleChineseCharactersPanelLayout

.createSequentialGroup()

.addComponent(possibleChineseCharactersLabel)

.addGap(89, 89, 89))))

);

possibleChineseCharactersPanelLayout.setVerticalGroup(

possibleChineseCharactersPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

.addGroup(possibleChineseCharactersPanelLayout

.createSequentialGroup()

.addContainerGap()

.addComponent(possibleChineseCharactersLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(JSP, GroupLayout.DEFAULT\_SIZE, 96

, Short.MAX\_VALUE)

.addContainerGap())

);

}

public static void addToList(ChineseCharacter c){

strings.add(c.toString());

listOfPossibleChineseCharacters.setModel(new AbstractListModel() {

public int getSize() { return strings.size(); }

public Object getElementAt(int i) { return strings.get(i); }

});

}

public static void clearList(){

strings = new ArrayList<String>();

listOfPossibleChineseCharacters.setModel(new AbstractListModel() {

public int getSize() { return strings.size(); }

public Object getElementAt(int i) { return strings.get(i); }

});

}

public Component getComponent() {

return this;

}

public void mouseClicked(MouseEvent e) {

}

public void mousePressed(MouseEvent e) {

String s = (((String) listOfPossibleChineseCharacters.getSelectedValue()

).split(":")[0]);

if(s.length() >1)

s = s.charAt(1)+"";

CharacterFieldPanel.addToList(s);

InternalDrawFramePanel.clearPanel();

}

public void mouseReleased(MouseEvent e) {

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/databasecreator/CharacterPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.databasecreator;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.image.BufferedImage;

import javax.swing.JPanel;

import javax.swing.border.LineBorder;

/\*\*

\*

\* @author Kieda

\*/

public class CharacterPanel extends JPanel{

public CharacterPanel(String s) {

asd = s;

initComponents();

}

private void initComponents() {

setBorder(new LineBorder(Color.BLACK));

setSize(100,100);

repaint();

}

public String asd;

@Override

public void paint(Graphics g) {

super.paint(g);

Graphics2D g2d = (Graphics2D)g;

BufferedImage bi = (BufferedImage) createImage(20, 20);

Graphics2D big = bi.createGraphics();

big.drawString(asd.split(":")[0]+"", 1, 11);

g2d.scale(4.6, 4.6);//transformation

g2d.drawImage(bi, 1, 1, this);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/databasecreator/DatabaseCreatorPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.databasecreator;

import com.sun.java.swing.plaf.motif.MotifBorders.BevelBorder;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.GroupLayout;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.LayoutStyle;

import universals.FontFinder;

/\*\*

\* @version 1.6

\* @author Kieda

\*/

public class DatabaseCreatorPanel extends JPanel

implements ChineseFrameComponent{

private static String characterToWrite;

private JButton addToDatabaseButton;

private JButton clear;

private JButton next;

private JLabel characterToWriteLiteral;

private JLabel characterInJPanel;

static CharacterPanel characterPanel;

static JLabel jl;

public static String getCharacterToWrite() {

return characterToWrite;

}

public static void setCharacterToWrite(String characterToWrite) {

DatabaseCreatorPanel.characterToWrite = characterToWrite;

if(characterToWrite.split(":")[0].length()>1)

jl.setText(characterToWrite.split(":")[0].charAt(1)+"");

else

jl.setText(characterToWrite.split(":")[0].charAt(0)+"");

}

public DatabaseCreatorPanel(String characterToWrite) {

DatabaseCreatorPanel.characterToWrite = characterToWrite;

initComponents();

}

@SuppressWarnings("unchecked")

private void initComponents() {

jl = new JLabel();

if(characterToWrite.split(":")[0].length()>1)

jl.setText(characterToWrite.split(":")[0].charAt(1)+"");

else

jl.setText(characterToWrite.split(":")[0].charAt(0)+"");

jl.setFont(new Font(FontFinder.chinesefonts.get(0),Font.PLAIN,70));

setBorder(new BevelBorder(true, Color.DARK\_GRAY, Color.LIGHT\_GRAY));

addToDatabaseButton = new JButton();

clear = new JButton();

next = new JButton();

characterToWriteLiteral = new JLabel();

characterInJPanel = new JLabel();

addToDatabaseButton.setText("Add to Database");

clear.setText("Clear");

next.setText("Next");

addToDatabaseButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ChineseFrameActionHandler

.addToDatabaseActionEvent(e,characterToWrite);

}

});

clear.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ChineseFrameActionHandler.clearActionPerformed(e);

}

});

next.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ChineseFrameActionHandler.nextButtonActionPerformed(e);

}

});

characterToWriteLiteral.setFont(new java.awt.Font("Tahoma", 0, 15));

characterToWriteLiteral.setText("Character To Write");

characterInJPanel.setText(characterToWrite);

characterPanel = new CharacterPanel(characterToWrite);

System.out.println(characterToWrite);

GroupLayout layout = new GroupLayout(this);

this.setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment

.LEADING)

.addComponent(characterToWriteLiteral)

.addComponent(jl, GroupLayout.PREFERRED\_SIZE, GroupLayout

.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

.addGap(180)

.addComponent(addToDatabaseButton,50,125,200)

.addGap(10)

.addComponent(clear,50,125,200)

.addContainerGap(10,Short.MAX\_VALUE)

.addComponent(next,50,125,200))

.addGap(30)

);

layout.setVerticalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(characterToWriteLiteral)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jl, GroupLayout.PREFERRED\_SIZE, GroupLayout

.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(169, Short.MAX\_VALUE))

.addGap(30)

.addComponent(addToDatabaseButton,100,100,100)

.addComponent(clear,100,100,100)

.addGap(30)

.addComponent(next,100,100,100)

);

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/drawpanel/InternalDrawFrame.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.drawpanel;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import gui.mainframe.componentcreator.menubar.MenuBar;

import universals.LogManager;

import java.awt.BorderLayout;

import java.awt.Component;

import java.awt.Cursor;

import java.awt.Graphics2D;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

import java.util.Arrays;

import javax.swing.JInternalFrame;

import mechanics.database.DatabaseDriver;

import mechanics.database.SearchEngine;

import mechanics.recognition.managers.FloatingPartManager;

import mechanics.recognition.managers.IntersectionManager;

import mechanics.recognition.managers.StrokeComplexityManager;

import mechanics.recognition.managers.StrokeNumberManager;

import universals.ChineseStroke;

import universals.CreatedCharacter;

import universals.UniversalDataStorage;

/\*\*

\* @version 2.3

\* @author Kieda

\*/

public class InternalDrawFrame extends JInternalFrame

implements Runnable, MouseListener, ChineseFrameComponent{

private int positionX;

private int positionY;

private int frameWidth;

private int xMouseDrawing;

private int yMouseDrawing;

private boolean firstTime = true;

private static boolean dead = false;

IntersectionManager im = new IntersectionManager();

StrokeComplexityManager cm = new StrokeComplexityManager();

FloatingPartManager fpm = new FloatingPartManager();

StrokeComplexityManager scm = new StrokeComplexityManager();

StrokeNumberManager snm = new StrokeNumberManager();

private InternalDrawFramePanel cp;

BufferedImage bi;

Graphics2D big;

public InternalDrawFrame() {

init();

}

public InternalDrawFrame(int x, int y) {

init(x,y);

}

public InternalDrawFrame(int x, int y, int width) {

init(x,y,width);

}

private void init(){

init(0,0);

}

private void init(int x, int y){

init(x,y,300);

}

private void init(int x, int y, int width){

setCursor(new Cursor(Cursor.CROSSHAIR\_CURSOR));

UniversalDataStorage.points = new ChineseStroke();

positionX = x;

positionY = y;

frameWidth = width;

setLocation(positionX, positionY);

setBounds(positionX, positionY, frameWidth, frameWidth);

addMouseListener(this);

setDefaultCloseOperation(JInternalFrame.DISPOSE\_ON\_CLOSE);

setVisible(true);

setClosable(true);

setMaximumSize(new java.awt.Dimension(width, width));

setMinimumSize(new java.awt.Dimension(width, width));

setPreferredSize(new java.awt.Dimension(width, width));

cp = new InternalDrawFramePanel(getBounds());

getContentPane().add(cp, BorderLayout.CENTER);

pack();

}

public static void setUniversalPoints(CreatedCharacter points) {

UniversalDataStorage.p = points;

}

public static void setPoints(ChineseStroke points) {

UniversalDataStorage.points = points;

}

public static ArrayList<ArrayList<Integer[]>> getUniversalPoints() {

return UniversalDataStorage.p;

}

public static ChineseStroke getPoints() {

return UniversalDataStorage.points;

}

public int getXMouseDrawing() {

return xMouseDrawing;

}

public int getYMouseDrawing() {

return yMouseDrawing;

}

/\*ITERATION\*/

public void run() {

fast:while(true){

pointcheck();

UniversalDataStorage.getXMouseDrawing = getXMouseDrawing();

UniversalDataStorage.getYMouseDrawing = getYMouseDrawing();

repaint();

cp.repaint();

try{

Thread.sleep(15);

}

catch(InterruptedException ex){LogManager.logError(ex,

"Thread interrupted", 115, "InternalDrawFrame");}

/\*USE OF FLAGS\*/

if(dead){

break fast;

}

}

this.dispose();

}

boolean create = false;

public void pointcheck(){

if(create){

/\*ADVANCED SELECTION\*/

if(firstTime){

firstTime = false;

xMouseDrawing = getMousePosition().x;

yMouseDrawing = getMousePosition().y;

UniversalDataStorage.points.add(new Integer[]{xMouseDrawing-8

,yMouseDrawing-27});

}

else{

try {

if(deltaMouseMax<=Math.abs(xMouseDrawing-

getMousePosition().x)||deltaMouseMax<=

Math.abs(yMouseDrawing-getMousePosition().y)){

xMouseDrawing = getMousePosition().x;

yMouseDrawing = getMousePosition().y;

UniversalDataStorage.points.add(new Integer[]{xMouseDrawing

-8,yMouseDrawing-27});

//corrects mouse offset

}

} catch (Exception e) {LogManager.logError(e

,"mouse goes off drawpanel", 144, "InternalDrawFrame");}

}

}

}

public void mousePressed(MouseEvent e) {

DatabaseDriver.halt();

create = true;

pointcheck();

}

public void mouseClicked(MouseEvent e) {}

public void mouseReleased(MouseEvent e) {

if(MenuBar.getPenOrEraser())

UniversalDataStorage.p.add(UniversalDataStorage.points);

else

UniversalDataStorage.p.erase(UniversalDataStorage.points);

UniversalDataStorage.numberOfStrokes = snm.processData(

UniversalDataStorage.p);

UniversalDataStorage.intersections = im.processData(

UniversalDataStorage.p);

UniversalDataStorage.numberOfComplexStrokes = cm.processData(

UniversalDataStorage.p)[1];

UniversalDataStorage.numberOfStraightStrokes = cm.processData(

UniversalDataStorage.p)[0];

UniversalDataStorage.numberOfFloatingParts = fpm.processData(

UniversalDataStorage.p);

System.out.printf("Intersections %s; Complex %d; Simple %d; Floating" +

" %d; Stroke Count %d\n"

, Arrays.toString(UniversalDataStorage.intersections.toArray())

, UniversalDataStorage.numberOfComplexStrokes

, UniversalDataStorage.numberOfStraightStrokes

, UniversalDataStorage.numberOfFloatingParts

, UniversalDataStorage.numberOfStrokes

);

SearchEngine.searchForChineseCharacter();

create = false;

UniversalDataStorage.points = new ChineseStroke();

UniversalDataStorage.updateOptions();

}

public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}

public Component getComponent() {

return(this);

}

public static void kill(){

dead = true;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/drawpanel/InternalDrawFramePanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.drawpanel;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Rectangle;

import java.awt.image.BufferedImage;

import javax.swing.JPanel;

import universals.ChineseStroke;

import universals.UniversalDataStorage;

/\*\*

\*

\* @author Kieda

\*/

public class InternalDrawFramePanel extends JPanel{

public InternalDrawFramePanel(Rectangle bounds) {

setBounds(bounds);

}

private BufferedImage bi;

private Graphics2D big;

static boolean firstTime = true;

@Override

public void paintComponent(Graphics g){

ChineseStroke points = InternalDrawFrame.getPoints();

Graphics2D g2= (Graphics2D) g;

if(firstTime){

Dimension dim = getSize();

bi = (BufferedImage) createImage(dim.width, dim.height);

big = (Graphics2D) bi.createGraphics();

big.clearRect(0, 0, bi.getWidth(), bi.getHeight());

big.setStroke(new BasicStroke(3, BasicStroke.CAP\_SQUARE, BasicStroke

.JOIN\_ROUND, 1.0f, new float[] {10,10,10,10,10}, 0));

big.setPaint(Color.LIGHT\_GRAY);

big.drawLine(0, dim.height/2, dim.width, dim.height/2);

big.drawLine(dim.width/2, 0, dim.width/2, dim.height);

big.setStroke(new BasicStroke(1));

big.setPaint(Color.BLACK);

firstTime = false;

}

if(points.size()>1){

for(int i = 1; i<points.size();i++){

big.setPaint(Color.BLACK);

big.drawLine(points.get(i-1)[0], points.get(i-1)[1]

, points.get(i)[0] , points.get(i)[1]);

big.setPaint(Color.GRAY);

big.drawOval(points.get(i-1)[0]-1, points.get(i-1)[1]-1, 2, 2);

}

}

g2.drawImage(bi, 0, 0, this);

}

public static void clearPanel(){

firstTime = true;

UniversalDataStorage.p.clear();

UniversalDataStorage.points.clear();

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./gui/mainframe/componentcreator/menubar/MenuBar.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.menubar;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Component;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JRadioButtonMenuItem;

/\*\*

\*

\* @author Kieda

\*/

public class MenuBar extends JMenuBar implements ChineseFrameComponent {

private static JMenu edit = new JMenu();

private static JMenuItem exitMenuButton = new JMenuItem();

private static JMenuItem clear = new JMenuItem();

private static JMenu file = new JMenu();

private static JMenu chineseCharacterFinder = new JMenu();

private static JRadioButtonMenuItem eraser = new JRadioButtonMenuItem();

private static JRadioButtonMenuItem pen = new JRadioButtonMenuItem();

private static JRadioButtonMenuItem addToDatabase = new JRadioButtonMenuItem();

private static JMenuItem newCharacterMenuButton = new JMenuItem();

public MenuBar() {

file.setText("File");

exitMenuButton.setText("Exit");

clear.setText("Clear");

exitMenuButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent evt) {

MenuBarActionHandler.exitActionPerformed(evt);

}

});

clear.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

MenuBarActionHandler.clearActionPerformed(e);

}

});

file.add(exitMenuButton);

newCharacterMenuButton.setText("New Character");

//file.add(newCharacterMenuButton);

add(file);

edit.setText("Edit");

edit.add(clear);

eraser.setSelected(false);

eraser.setText("Eraser");

eraser.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

MenuBarActionHandler.eraserStateChanged(e);

}

});

//edit.add(eraser);

pen.setSelected(true);

pen.setText("Pen");

pen.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

MenuBarActionHandler.penStateChanged(e);

}

});

//edit.add(pen);

add(edit);

chineseCharacterFinder.setText("Chinese Character Finder");

add(chineseCharacterFinder);

}

@Override

public Component getComponent() {

return this;

}

public static void penOrEraser(boolean b){

eraser.setSelected(b);

pen.setSelected(!b);

}

public static boolean getPenOrEraser(){

return pen.isSelected();

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./gui/mainframe/componentcreator/menubar/MenuBarActionHandler.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.menubar;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.drawpanel.InternalDrawFramePanel;

import java.awt.event.ActionEvent;

/\*\*

\*

\* @author Kieda

\*/

public class MenuBarActionHandler extends MenuBar{

public static boolean penOrEraser = true;

public static void exitActionPerformed(ActionEvent evt) {

ChineseFrameActionHandler.exitMenuButtonActionPerformed(evt);

}

public static void clearActionPerformed(ActionEvent evt) {

InternalDrawFramePanel.clearPanel();

}

public static void penStateChanged(ActionEvent evt) {

MenuBar.penOrEraser(penOrEraser);

}

public static void eraserStateChanged(ActionEvent evt) {

penOrEraser = false;

MenuBar.penOrEraser(!penOrEraser);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/options/FloatingPartsCounterPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.options;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSpinner;

import javax.swing.LayoutStyle;

/\*\*

\*

\* @author Kieda

\*/

public class FloatingPartsCounterPanel extends JPanel

implements ChineseFrameComponent{

private static JSpinner numberOfFloatingPartsCounter = new JSpinner();

private static JLabel numberOfFloatingPartsLabel = new JLabel();

public FloatingPartsCounterPanel() {

setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

numberOfFloatingPartsLabel.setText("Number of Floating Parts");

numberOfFloatingPartsCounter.addPropertyChangeListener(

new PropertyChangeListener() {

public void propertyChange(PropertyChangeEvent evt) {

ChineseFrameActionHandler

.numberOfFloatingPartsCounterPropertyChange(evt);

}

});

GroupLayout numberOfFloatingPartsPanelLayout = new GroupLayout(this);

setLayout(numberOfFloatingPartsPanelLayout);

numberOfFloatingPartsPanelLayout.setHorizontalGroup(

numberOfFloatingPartsPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

.addGroup(GroupLayout.Alignment.TRAILING,

numberOfFloatingPartsPanelLayout.createSequentialGroup()

.addContainerGap()

.addComponent(numberOfFloatingPartsCounter, GroupLayout

.PREFERRED\_SIZE, 49, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED

, 37, Short.MAX\_VALUE)

.addComponent(numberOfFloatingPartsLabel)

.addContainerGap())

);

numberOfFloatingPartsPanelLayout.setVerticalGroup(

numberOfFloatingPartsPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

.addGroup(numberOfFloatingPartsPanelLayout.createSequentialGroup()

.addGap(11, 11, 11)

.addGroup(numberOfFloatingPartsPanelLayout.createParallelGroup(

GroupLayout.Alignment.BASELINE)

.addComponent(numberOfFloatingPartsLabel)

.addComponent(numberOfFloatingPartsCounter,

GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE

, GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

}

public static int getNumberOfFloatingParts() {

return Integer.parseInt(numberOfFloatingPartsCounter.getValue()+"");

}

public static void setNumberOfFloatingParts(int i) {

numberOfFloatingPartsCounter.setValue(i);

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/options/IntersectionStringInputPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.options;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.LayoutStyle;

import javax.swing.event.CaretEvent;

import javax.swing.event.CaretListener;

/\*\*

\*

\* @author Kieda

\*/

public class IntersectionStringInputPanel extends JPanel

implements ChineseFrameComponent{

private static JTextField intersectionsStringInput = new JTextField();

private static JLabel intersectionsStringInputLabel = new JLabel();

public IntersectionStringInputPanel() {

setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

intersectionsStringInput.addCaretListener(new CaretListener() {

public void caretUpdate(CaretEvent evt) {

ChineseFrameActionHandler

.intersectionsStringInputCaretUpdate(evt);

}

});

intersectionsStringInputLabel

.setText("String Input (separate by commas)");

GroupLayout intersectionsStringInputPanelLayout = new GroupLayout(this);

setLayout(intersectionsStringInputPanelLayout);

intersectionsStringInputPanelLayout.setHorizontalGroup(

intersectionsStringInputPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING)

.addGroup(intersectionsStringInputPanelLayout.createSequentialGroup

().addContainerGap()

.addGroup(intersectionsStringInputPanelLayout

.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(intersectionsStringInputPanelLayout

.createSequentialGroup()

.addComponent(intersectionsStringInput, GroupLayout

.DEFAULT\_SIZE, 205, Short.MAX\_VALUE)

.addContainerGap())

.addGroup(GroupLayout.Alignment.TRAILING,

intersectionsStringInputPanelLayout

.createSequentialGroup()

.addComponent(intersectionsStringInputLabel)

.addGap(25, 25, 25))))

);

intersectionsStringInputPanelLayout.setVerticalGroup(

intersectionsStringInputPanelLayout.createParallelGroup(GroupLayout

.Alignment.LEADING)

.addGroup(GroupLayout.Alignment.TRAILING,

intersectionsStringInputPanelLayout.createSequentialGroup()

.addContainerGap(13, Short.MAX\_VALUE)

.addComponent(intersectionsStringInputLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(intersectionsStringInput, GroupLayout

.PREFERRED\_SIZE, 20, GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

}

public static String getIntersectionsString() {

return intersectionsStringInput.getText();

}

public static void setIntersectionsString(String s) {

intersectionsStringInput.setText(s);

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/options/IntersectionsCrudeAmountPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.options;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSpinner;

import javax.swing.LayoutStyle;

/\*\*

\*

\* @author Kieda

\*/

public class IntersectionsCrudeAmoutPanel extends JPanel

implements ChineseFrameComponent{

private static JSpinner totalNumberOfIntersectionsCounter = new JSpinner();

private static JLabel totalNumberOfIntersectionsLabel = new JLabel();

public IntersectionsCrudeAmoutPanel() {

setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

totalNumberOfIntersectionsLabel.setText("Total Number of Intersections");

totalNumberOfIntersectionsCounter.addPropertyChangeListener(

new PropertyChangeListener() {

public void propertyChange(PropertyChangeEvent evt) {

ChineseFrameActionHandler

.totalNumberOfIntersectionsCounterPropertyChange(evt);

}

});

GroupLayout totalNumberOfIntersectionsPanelLayout=new GroupLayout(this);

setLayout(totalNumberOfIntersectionsPanelLayout);

totalNumberOfIntersectionsPanelLayout.setHorizontalGroup(

totalNumberOfIntersectionsPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING)

.addGroup(GroupLayout.Alignment.TRAILING,

totalNumberOfIntersectionsPanelLayout

.createSequentialGroup()

.addContainerGap()

.addComponent(totalNumberOfIntersectionsCounter,

GroupLayout.PREFERRED\_SIZE, 54, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 8

, Short.MAX\_VALUE)

.addComponent(totalNumberOfIntersectionsLabel)

.addContainerGap())

);

totalNumberOfIntersectionsPanelLayout.setVerticalGroup(

totalNumberOfIntersectionsPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING)

.addGroup(totalNumberOfIntersectionsPanelLayout

.createSequentialGroup()

.addGap(11, 11, 11)

.addGroup(totalNumberOfIntersectionsPanelLayout

.createParallelGroup(GroupLayout.Alignment.BASELINE)

.addComponent(totalNumberOfIntersectionsLabel)

.addComponent(totalNumberOfIntersectionsCounter, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout

.PREFERRED\_SIZE))

.addContainerGap())

);

}

public static int getTotalNumberOfIntersections() {

return Integer

.parseInt(totalNumberOfIntersectionsCounter.getValue()+"");

}

public static void setTotalNumberOfIntersections(int i) {

totalNumberOfIntersectionsCounter.setValue(i);

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/options/StrokeComplexityPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.options;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import java.beans.PropertyChangeEvent;

import java.beans.PropertyChangeListener;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSpinner;

import javax.swing.LayoutStyle;

/\*\*

\*

\* @author Kieda

\*/

public class StrokeComplexityPanel extends JPanel

implements ChineseFrameComponent{

private static JSpinner numberOfStraightStrokesCounter = new JSpinner();

private static JLabel numberOfStraightStrokesLabel = new JLabel();

private static JLabel numberOfComplexStrokesLabel= new JLabel();

private static JSpinner numberOfComplexStrokesCounter = new JSpinner();

public StrokeComplexityPanel() {

setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

numberOfStraightStrokesLabel.setText("Number of Straight Strokes");

numberOfStraightStrokesCounter.addPropertyChangeListener(

new PropertyChangeListener() {

public void propertyChange(PropertyChangeEvent evt) {

ChineseFrameActionHandler

.numberOfStraightStrokesCounterPropertyChange(evt);

}

});

numberOfComplexStrokesLabel.setText("Number of Complex Strokes");

numberOfComplexStrokesCounter.addPropertyChangeListener(

new PropertyChangeListener() {

public void propertyChange(PropertyChangeEvent evt) {

ChineseFrameActionHandler

.numberOfComplexStrokesCounterPropertyChange(evt);

}

});

GroupLayout strokeComplexityPanelLayout = new GroupLayout(this);

setLayout(strokeComplexityPanelLayout);

strokeComplexityPanelLayout.setHorizontalGroup(

strokeComplexityPanelLayout.createParallelGroup(GroupLayout.

Alignment.LEADING)

.addGroup(strokeComplexityPanelLayout.createSequentialGroup()

.addContainerGap()

.addGroup(strokeComplexityPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING, false)

.addComponent(numberOfComplexStrokesCounter

, GroupLayout.Alignment.TRAILING)

.addComponent(numberOfStraightStrokesCounter, GroupLayout.

Alignment.TRAILING, GroupLayout.DEFAULT\_SIZE, 51

, Short.MAX\_VALUE))

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 21

, Short.MAX\_VALUE)

.addGroup(strokeComplexityPanelLayout.createParallelGroup(

GroupLayout.Alignment.LEADING)

.addComponent(numberOfStraightStrokesLabel, GroupLayout

.Alignment.TRAILING)

.addComponent(numberOfComplexStrokesLabel, GroupLayout

.Alignment.TRAILING))

.addContainerGap())

);

strokeComplexityPanelLayout.setVerticalGroup(

strokeComplexityPanelLayout.createParallelGroup(GroupLayout.

Alignment.LEADING)

.addGroup(strokeComplexityPanelLayout.createSequentialGroup()

.addGap(11, 11, 11)

.addGroup(strokeComplexityPanelLayout.createParallelGroup(

GroupLayout.Alignment.BASELINE)

.addComponent(numberOfStraightStrokesLabel)

.addComponent(numberOfStraightStrokesCounter, GroupLayout.

PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.

PREFERRED\_SIZE))

.addGap(11, 11, 11)

.addGroup(strokeComplexityPanelLayout.createParallelGroup(

GroupLayout.Alignment.BASELINE)

.addComponent(numberOfComplexStrokesLabel)

.addComponent(numberOfComplexStrokesCounter, GroupLayout.

PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.

PREFERRED\_SIZE))

.addContainerGap())

);

}

public static int getNumberOfStraightStrokes() {

return Integer.parseInt(numberOfStraightStrokesCounter.getValue()+"");

}

public static int getNumberOfComplexStrokes() {

return Integer.parseInt(numberOfComplexStrokesCounter.getValue()+"");

}

public static void setNumberOfStraightStrokes(int i) {

numberOfStraightStrokesCounter.setValue(i);

}

public static void setNumberOfComplexStrokes(int i) {

numberOfComplexStrokesCounter.setValue(i);

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/gui/mainframe/componentcreator/options/StrokeCrudeCountPanel.java

//////////////////////////////////////////////////////////////////////////////\*/

package gui.mainframe.componentcreator.options;

import gui.mainframe.ChineseFrameActionHandler;

import gui.mainframe.componentcreator.ChineseFrameComponent;

import java.awt.Color;

import java.awt.Component;

import javax.swing.BorderFactory;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSpinner;

import javax.swing.LayoutStyle;

import javax.swing.event.ChangeEvent;

import javax.swing.event.ChangeListener;

/\*\*

\*

\* @author Kieda

\*/

public class StrokeCrudeCountPanel extends JPanel implements

ChineseFrameComponent{

private static JSpinner numberOfStrokesCounter = new JSpinner();

private static JLabel numberOfStrokesLabel = new JLabel();

public StrokeCrudeCountPanel() {

setBorder(BorderFactory.createLineBorder(new Color(0, 0, 0)));

numberOfStrokesLabel.setText("Number of Strokes");

numberOfStrokesCounter.addChangeListener(new ChangeListener() {

public void stateChanged(ChangeEvent evt) {

ChineseFrameActionHandler

.numberOfStrokesCounterStateChanged(evt);

}

});

GroupLayout numberOfStrokesPanelLayout = new GroupLayout(this);

setLayout(numberOfStrokesPanelLayout);

numberOfStrokesPanelLayout.setHorizontalGroup(

numberOfStrokesPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

.addGroup(GroupLayout.Alignment.TRAILING, numberOfStrokesPanelLayout

.createSequentialGroup()

.addContainerGap()

.addComponent(numberOfStrokesCounter, GroupLayout.PREFERRED\_SIZE

, 53, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 63

, Short.MAX\_VALUE)

.addComponent(numberOfStrokesLabel)

.addContainerGap())

);

numberOfStrokesPanelLayout.setVerticalGroup(

numberOfStrokesPanelLayout.createParallelGroup(GroupLayout.Alignment

.LEADING)

.addGroup(numberOfStrokesPanelLayout.createSequentialGroup()

.addGap(11, 11, 11)

.addGroup(numberOfStrokesPanelLayout.createParallelGroup(

GroupLayout.Alignment.BASELINE)

.addComponent(numberOfStrokesCounter, GroupLayout

.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE

, GroupLayout.PREFERRED\_SIZE)

.addComponent(numberOfStrokesLabel))

.addContainerGap())

);

}

public static int getNumberOfStrokes() {

return Integer.parseInt(numberOfStrokesCounter.getValue()+"");

}

public Component getComponent() {

return this;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/ChineseCharacter.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics;

import java.util.ArrayList;

import java.util.Arrays;

/\*\*

\*

\* @author Kieda

\*/

public class ChineseCharacter {

private String pinyin;

private String character;

private String meaning;

private int floatingParts;

private ArrayList<Integer> intersections;

private int straightStrokes;

private int complexStrokes;

public ChineseCharacter(String pinyin, String character, String meaning

, int floatingParts, ArrayList<Integer> intersections

, int straightStrokes, int complexStrokes) {

this.character = character;

this.pinyin = pinyin;

this.meaning = meaning;

this.floatingParts = floatingParts;

this.intersections = intersections;

this.straightStrokes = straightStrokes;

this.complexStrokes = complexStrokes;

}

public String getCharacter() {

return character;

}

public int getComplexStrokes() {

return complexStrokes;

}

public int getFloatingParts() {

return floatingParts;

}

public ArrayList<Integer> getIntersections() {

return intersections;

}

public String getMeaning() {

return meaning;

}

public String getPinyin() {

return pinyin;

}

public int getStraightStrokes() {

return straightStrokes;

}

public void setCharacter(String character) {

this.character = character;

}

public void setComplexStrokes(int complexStrokes) {

this.complexStrokes = complexStrokes;

}

public void setFloatingParts(int floatingParts) {

this.floatingParts = floatingParts;

}

public void setIntersections(ArrayList<Integer> intersections) {

this.intersections = intersections;

}

public void setMeaning(String meaning) {

this.meaning = meaning;

}

public void setPinyin(String pinyin) {

this.pinyin = pinyin;

}

public void setStraightStrokes(int straightStrokes) {

this.straightStrokes = straightStrokes;

}

@Override

public String toString() {

if(this.equals(null))

return "nuthing found";

else{

String intersectionsString = "";

for(Integer i:intersections){

intersectionsString+=(i+",");

}

return String.format("%s:%s:%s#%d:%s:%d:%d"

,pinyin, character,meaning,floatingParts,intersectionsString

,straightStrokes,complexStrokes);

}

}

public static ChineseCharacter toChineseCharacter(String currentLine){

String currentLineData = currentLine.split("#")[1];

String[] infrormation = currentLineData.split(":");

String intersectionString = infrormation[1];

String[] intersectionsString = intersectionString.split(",");

Integer[] intersections = new Integer[intersectionsString.length];

for (int counter = 0; counter < intersectionsString.length; counter++) {

//System.out.println("A");

intersections[counter] = Integer.parseInt(

intersectionsString[counter]);

}

int floatingParts = Integer.parseInt(currentLineData.split(":")[0]);

int straightStrokes = Integer.parseInt(currentLineData.split(":")[2]);

int complexStrokes = Integer.parseInt(currentLineData.split(":")[3]);

String pinyin = currentLine.split("#")[0].split(":")[0];

String character = currentLine.split("#")[0].split(":")[1];

String meaning = currentLine.split("#")[0].split(":")[2];

Arrays.sort(intersections);

return new ChineseCharacter(pinyin, character, meaning, floatingParts,

new ArrayList<Integer>(Arrays.asList(intersections)),

straightStrokes, complexStrokes);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/database/DatabaseCreator.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.database;

import java.io.File;

import java.io.IOException;

import java.util.Scanner;

import universals.LogManager;

import mechanics.ChineseCharacter;

/\*\*

\* @version 1.0

\* @author Kieda

\*/

public class DatabaseCreator {

private static File loadfile = new File("new.txt");

private static Scanner in;

public static String next(){

if(!done())

return in.nextLine();

else

return "";

}

public static boolean done(){

return !in.hasNext();

}

public static void init(){

try {

in = new Scanner(loadfile);

} catch (IOException e) {}

}

public static void add(ChineseCharacter c){

LogManager.logThing(c.toString(), LogManager.DATABASE);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/database/DatabaseDriver.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.database;

import gui.mainframe.componentcreator.bottompanel.PossibleChineseCharactersPanel;

import java.awt.EventQueue;

import java.io.BufferedWriter;

import universals.LogManager;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

import mechanics.ChineseCharacter;

import universals.TimeManager;

import universals.UniversalDataStorage;

/\*\*

\* @version 1.9

\* @author Kieda

\*/

public class DatabaseDriver {

static Scanner in;

public static ChineseCharacter searchCharacter(Object[] intersections

,int floatingParts,int straightStrokes,int complexStrokes){

Integer[] intIntersections = new Integer[intersections.length];

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

intIntersections[i] = Integer.parseInt(intersections[i]+"");

}

return searchCharacter(intIntersections, floatingParts, straightStrokes

, complexStrokes);

}

public static void addToDatabase(ChineseCharacter c){

File f = new File("NEWEST\_VERSION.data");

try {

ArrayList<String> originalLog = new ArrayList<String>();

Scanner in = new Scanner(f);

while (in.hasNext()) {

originalLog.add(in.nextLine());

//adds the file to be written to an ArrayList<String>

}

FileWriter bestBuddy = new FileWriter(f);

BufferedWriter bff = new BufferedWriter(bestBuddy);

for (String line : originalLog) {

bff.write(line);//rewriting the log

bff.newLine();

}

bff.write(c.toString());

bff.close();

} catch (IOException ex) {LogManager.logError(ex, "something wrong " +

"with making dictionary", 46, "DatabaseDriver", true);}

}

static ChineseCharacter chinesecharacter = null;

static Integer[] intersections;

static int floatingParts;

static int straightStrokes;

static int complexStrokes;

private static boolean stop = false;

public static ChineseCharacter searchCharacter(Integer[] intersection

,int floatingPart, int straightStroke, int complexStroke){

intersections = intersection;

floatingParts = floatingPart;

straightStrokes = straightStroke;

complexStrokes = complexStroke;

EventQueue.invokeLater(new Runnable() {

public void run() {

PossibleChineseCharactersPanel.clearList();

Arrays.sort(intersections);

no:for(ChineseCharacter c: UniversalDataStorage.database){

if (c.getFloatingParts() == floatingParts

&& Arrays.toString(c.getIntersections().toArray())

.equals(Arrays.toString(intersections))

&& (c.getComplexStrokes() + c.getStraightStrokes())

== (complexStrokes + straightStrokes)

//IF NEEDED

// &&straightStrokeSearch==straightStrokes

// &&complexStrokeSearch==complexStrokes

) {

if(stop){

stop = false;

break no;

}

String pinyin = c.getPinyin();

String character = c.getCharacter();

String meaning = c.getMeaning();

chinesecharacter = new ChineseCharacter(pinyin

, character, meaning, floatingParts, new

ArrayList<Integer>(Arrays.asList(intersections))

, straightStrokes, complexStrokes);

System.out.println(chinesecharacter);

PossibleChineseCharactersPanel.addToList(

chinesecharacter);

LogManager.logThing("Character found:" +chinesecharacter

+ " " + TimeManager.getCurrentTimeAndDate());

}

}

}});

return chinesecharacter;

}

public static void halt(){

stop = true;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/database/DatabaseLoader.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.database;

import java.awt.Font;

import java.io.File;

import java.util.Scanner;

import javax.swing.JFrame;

import javax.swing.JLabel;

import mechanics.ChineseCharacter;

import universals.LogManager;

import universals.UniversalDataStorage;

/\*\*

\*

\* @author Kieda

\*/

public class DatabaseLoader {

static Scanner in;

static int databaseCapacity;

public static JFrame f;

public static JLabel j1;

public static void load(){

showMessage();

File f = new File("characters.DAT");

try {

in = new Scanner(f.getAbsoluteFile(), "UTF8");

} catch (Exception bbb) {

in = new Scanner("");

LogManager.logError(bbb, "chinese data file not found", 30

, "DatabaseDriver");

}

databaseCapacity = 0;

while (in.hasNext()) {

in.nextLine();

databaseCapacity++;

}

ChineseCharacter[] data = new ChineseCharacter[databaseCapacity];

in.close();

try {

in = new Scanner(f.getAbsoluteFile(), "UTF8");

} catch (Exception bbb) {

in = new Scanner("");

LogManager.logError(bbb, "chinese data file not found", 30

, "DatabaseDriver");

}

int i = 0;

while (in.hasNext()) {

data[i] = ChineseCharacter.toChineseCharacter(in.nextLine());

i++;

}

UniversalDataStorage.database = data;

try {

Thread.sleep(500);

} catch (InterruptedException ex) {}

}

private static void showMessage(){

f = new JFrame("LOADING...");

f.setResizable(false);

j1 = new JLabel(" Loading, Please Wait");

f.setBounds(200, 200, 400, 200);

j1.setFont(new Font("Arial", Font.BOLD, 30));

j1.setLocation(90, 200);

f.add(j1);

f.setDefaultCloseOperation(JFrame.DO\_NOTHING\_ON\_CLOSE);

f.setVisible(true);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/database/SearchEngine.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.database;

import universals.UniversalDataStorage;

/\*\*

\* @version .3 crappy version :P

\* @author Kieda

\*/

public class SearchEngine {

/\*\*

\* Searches for a character match in the database

\*/

public static void searchForChineseCharacter(){

if(UniversalDataStorage.intersections.size()>0)

DatabaseDriver.searchCharacter(UniversalDataStorage.intersections

.toArray(), UniversalDataStorage.numberOfFloatingParts

, UniversalDataStorage.numberOfStraightStrokes

, UniversalDataStorage.numberOfComplexStrokes);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/recognition/ChineseManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.recognition;

import universals.CreatedCharacter;

/\*\*

\*

\* @author Kieda

\*/

public interface ChineseManager<T> {

public T returnValue();

public T processData(CreatedCharacter p);

public void updateManager();

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/recognition/managers/FloatingPartManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.recognition.managers;

import java.awt.geom.Line2D;

import java.util.ArrayList;

import mechanics.recognition.ChineseManager;

import universals.UniversalDataStorage;

import universals.CreatedCharacter;

/\*\*

\* @version 1.2

\* @author Kieda

\*/

public class FloatingPartManager implements ChineseManager{

@Override

public Integer returnValue() {

return UniversalDataStorage.numberOfFloatingParts;

}

public int nextToDo(ArrayList<Integer> i, int length){

for(int p = 0; p<length;p++){

if(!i.contains(p))

return p;

}

return -1;

}

public Integer processData(CreatedCharacter p) {

ArrayList<Integer> i = new ArrayList<Integer>();

ArrayList<Integer> zaq = new ArrayList<Integer>();

int intersection = 0;

/\*NESTED LOOPS, ADVANCED SELECTION\*/

if(p.size()>0)

heart:for(int n = 0;true;){

if(!i.contains(n)){

i.add(n);

for(int s = 0; s<(p.get(n).size()-1);s++){

Line2D l1 = new Line2D.Double(p.get(n).get(s)[0]

, p.get(n).get(s)[1], p.get(n).get(s+1)[0]

, p.get(n).get(s+1)[1]);

for(int z = 0; z<p.size();z++){

if(z!=n)

for(int q = 0; q<(p.get(z).size()-1);q++){

Line2D l2 = new Line2D.Double(p.get(z).get(q)

[0], p.get(z).get(q)[1], p.get(z)

.get(q+1)[0] , p.get(z).get(q+1)[1]);

if(l1.intersectsLine(l2)){

if(!i.contains(z)){

zaq.add(z);

}

}

}

}

}

}

if(zaq.size()>0)

n = zaq.remove(zaq.size()-1);

else{

n = nextToDo(i, p.size());

intersection++;

}

if(n==-1)

/\*USE OF FLAGS\*/

break heart;

}

return intersection;

}

public void updateManager() {

UniversalDataStorage.numberOfFloatingParts = (

processData(UniversalDataStorage.p));

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/recognition/managers/IntersectionManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.recognition.managers;

import java.awt.geom.Line2D;

import java.util.ArrayList;

import java.util.Arrays;

import mechanics.recognition.ChineseManager;

import universals.UniversalDataStorage;

import universals.CreatedCharacter;

/\*\*

\* V2.0 Intersection Manager

\* @author Kieda

\*/

public class IntersectionManager implements ChineseManager{

public ArrayList<Integer> transferArray(int[] s){

Integer[] ai = new Integer[s.length];

for( int d = 0; d<s.length;d++){

ai[d] = s[d];

}

return new ArrayList<Integer>(Arrays.asList(ai));

}

@Override

public ArrayList<Integer> processData(CreatedCharacter p) {

ArrayList<Integer> i = new ArrayList<Integer>(p.size());

i.ensureCapacity(p.size());

boolean b = true;

/\*NESTED LOOPS\*/

for(int n = 0; n<p.size(); n++){

int intersection = 0;

ArrayList<Integer> nnn = new ArrayList<Integer>();

for(int s = 0; s<(p.get(n).size()-1);s++){

Line2D l1 = new Line2D.Double(p.get(n).get(s)[0], p.get(n).get(s)

[1], p.get(n).get(s+1)[0] , p.get(n).get(s+1)[1]);

for(int z = 0; z<p.size();z++){

if(z!=n&&!nnn.contains(z))

stop:for(int q = 0; q<(p.get(z).size()-1);q++){

Line2D l2 = new Line2D.Double(p.get(z).get(q)[0], p

.get(z).get(q)[1], p.get(z).get(q+1)[0]

, p.get(z).get(q+1)[1]);

if(l1.intersectsLine(l2)){

intersection++;

nnn.add(z);

break stop;

}

}

}

}

i.add( intersection);

}

return i;

}

@Override

public ArrayList<Integer> returnValue() {

return UniversalDataStorage.intersections;

}

public void updateManager() {

UniversalDataStorage.intersections = processData(

UniversalDataStorage.p);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/recognition/managers/StrokeComplexityManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.recognition.managers;

import java.util.ArrayList;

import mechanics.recognition.ChineseManager;

import universals.ChineseLine;

import universals.ChineseStroke;

import universals.UniversalDataStorage;

import universals.CreatedCharacter;

/\*\*

\*

\* @author Kieda

\*/

public class StrokeComplexityManager implements ChineseManager{

@Override

public Integer returnValue() {

return UniversalDataStorage.numberOfComplexStrokes;

}

public void updateManager() {

UniversalDataStorage.numberOfComplexStrokes = processData(

UniversalDataStorage.p)[1];

UniversalDataStorage.numberOfStraightStrokes = processData(

UniversalDataStorage.p)[0];

}

public Integer[] processData(CreatedCharacter p) {

Integer complex = 0;

UniversalDataStorage.p = p;

for(ArrayList<Integer[]> s: p){

if(complexStroke((ChineseStroke) s)){

complex++;

}

}

Integer simple = p.size()-complex;

UniversalDataStorage.numberOfComplexStrokes = complex;

UniversalDataStorage.numberOfStraightStrokes = simple;

return new Integer[]{simple,complex};

}

public boolean complexStroke(ChineseStroke p){

boolean b = false;

if(p.size()>=2){

for(int i = 0; i<(p.size()-2);i++){

ChineseLine cl1 = new ChineseLine(p.get(i)[0], p.get(i+1)[0]

, p.get(i+1)[1], p.get(i)[1]);

ChineseLine cl2 = new ChineseLine(p.get(i+1)[0], p.get(i+2)[0]

, p.get(i+2)[1], p.get(i+1)[1]);

if(Math.abs(cl1.totaldegrees-cl2.totaldegrees)>70

&&(cl1.totaldegrees>80||cl2.totaldegrees>80)){

b = true;

}

}

}

return b;

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/mechanics/recognition/managers/StrokeNumberManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package mechanics.recognition.managers;

import mechanics.recognition.ChineseManager;

import universals.UniversalDataStorage;

import universals.CreatedCharacter;

/\*\*

\*

\* @author Kieda

\*/

public class StrokeNumberManager implements ChineseManager<Integer>{

public Integer returnValue() {

return UniversalDataStorage.numberOfStrokes;

}

public Integer processData(CreatedCharacter p) {

return p.size();

}

public void updateManager() {

UniversalDataStorage.numberOfStrokes = processData(UniversalDataStorage

.p);

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/ChineseLine.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

/\*\*

\* This class is used for analyzing specific lines on a ChineseStroke

\* a line is a two-point protion of a stroke. The use of slopes found

\* @author Kieda

\* @version 1.2

\* @since 3-13-2011

\*/

public class ChineseLine {

private int x1;//the first x coordinate in the line

private int x2;//the second x coordinate in the line

private int y1;//the first y coordinate in the line

private int y2;//the second y coordinate in the line

public double degrees;

//the degrees the line is pointe from its nearest quandrant

public double totaldegrees;

//the degrees the line is pointed from the first quadrant

public int quadrant;

//the quandrant the line goes in

public double magnitude;

//the magnitude of length of a line

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//these quadrant values are used when //

//defining a line. Other classes can use //

//these values to deterine where a line is //

//pointing or to define a custom line //

//quadrants are as shown in the ascii image //

//below. //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

// |Y //

// | //

// QUADRANT II | QUADRANT I //

// | //

// | //

// ---------------+--------------- //

// | X //

// QUADRANT III | QUADRANT IV //

// | //

// | //

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

public static final int FIRST\_QUADRANT = 0;//value for the first quadrant

public static final int SECOND\_QUADRANT = 1;//value for the second quadrant

public static final int THIRD\_QUADRANT = 2;//value for the third quadrant

public static final int FOURTH\_QUADRANT = 3;//value for the fourth quadrant

/\*\*

\* this is the constructor for the ChineseLine

\* this constuctor takes the position points and translates the line into a

\* magnitude and direction

\* @param x1 the first x coordinate in the line

\* @param x2 the second x coordinate in the line

\* @param y1 the first y coordinate in the line

\* @param y2 the second y coordinate in the line

\*/

public ChineseLine(int x1, int x2, int y1, int y2) {

this.x1 = x1;//sets the value of the first x coordinate in the line

this.x2 = x2;//sets the value of the second x coordinate in the line

this.y1 = y1;//sets the value of the first y coordinate in the line

this.y2 = y2;//sets the value of the second y coordinate in the line

if((x2-x1) == 0)//if the line is vertical

degrees = 90;//the line is going up or down 90 degrees

else//otherwise

degrees = Math.toDegrees(Math.abs(Math.atan((y2-y1)/(x2-x1))));

//the degrees is equal to the arctangent(Δy/Δx), but remember,

//this depends on which quadrant we are in.

//we figure out which quadrant we are in by comparing the x

//and y values

totaldegrees = degrees;//setting the initial degrees

if(x2>x1&&y2>y1){//if the x and y values get larger

quadrant = FIRST\_QUADRANT;//we are in the first quadrant

//we adjust the total angle by adding nothing

}

if(x2<x1&&y2>y1){

//if the x values get smaller and y the values get larger

quadrant = SECOND\_QUADRANT;//we are in the second quadrant

totaldegrees+=90;//we adjust the total angle by adding 90 degrees

}

if(x2<x1&&y2<y1){//if the x and y values get smaller

quadrant = THIRD\_QUADRANT;//we are in the third quadrant

totaldegrees+=180;//we adjust the total angle by adding 180 degrees

}

if(x2>x1&&y2<y1){

//if the x values get larger and y the values get smaller

quadrant = FOURTH\_QUADRANT;//we are in the fourh quadrant

totaldegrees+=270;//we adjust the total angle by adding 270 degrees

}

magnitude = Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));

//sets the magnitude o the line by using the distance formula

}

/\*\*

\* redefines the position of the line

\* @param x1 the first x coordinate in the line

\* @param x2 the second x coordinate in the line

\* @param y1 the first y coordinate in the line

\* @param y2 the second y coordinate in the line

\*/

public void setPoints(int x1, int x2, int y1, int y2) {

this.x1 = x1;//sets the value of the first x coordinate in the line

this.x2 = x2;//sets the value of the second x coordinate in the line

this.y1 = y1;//sets the value of the first y coordinate in the line

this.y2 = y2;//sets the value of the second y coordinate in the line

}

}

/\*////////////////////////////////////////////////////////////////////////>>Main

./src/universals/ChineseMakeDictionary.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;//package

// <editor-fold defaultstate="collapsed" desc="imports">

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;// </editor-fold>

/\*\*

\* this class was used to make the database from the copied list of the most

\* frequent Chinese characters

\* @author Kieda

\* @since forever ago

\*/

public class ChineseMakeDictionary {

static int popularity = 31;//the starting point to add characters

static int line = 0;//the line number

public static void main(String[] args) throws IOException {

File loadfile = new File("lol.txt");

//loadfile: the file with the list of the most common chinese

//characters

FileWriter fstream= new FileWriter("new1.txt");

//the file being wrote to

BufferedWriter out = new BufferedWriter(fstream);

//the tool ues to write to the file

Scanner in;

//the tool used to read the loadfile

System.out.println(loadfile.exists());

//to check if the file exists

while(popularity<=3000){//main loop to the 3000 most common characters

in = new Scanner(loadfile);//makes a new scanner of the loadfile

fast:while(in.hasNextLine()){//reads through the entire document

String s = in.nextLine();

if(popularity==858)//mistake in the html file

popularity++;//skips the line

if(s.contains(popularity+"")){

//sees wether if the line inspected has the number in it

//(the documetnt had a format of

//popularity character [pinyin] UNNEEDED\_STUFF

s = s.split(popularity+"")[1];

//splits up the line

s = s.split("]")[0];

//extracts the important information

out.write(s+"]");

//writes the important information and a mreak (']')

//which is replaces later using the Ctrl+f and replace

//feature

out.newLine();//makes a new line

break fast;

//breaks the search for a line with the popularity

}

line++;//the line number goes up one

}

popularity++;//go to the next highest popularity

in.close();//close the scanner

}

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/ChineseStroke.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

import java.util.ArrayList;

/\*\*

\* This class is simply made for coding purposes. Instead of calling

\* an ArrayList<Integer[]>, one can simply call a ChineseStroke.

\* @author Kieda

\* @since 3-13-2011

\*/

public class ChineseStroke extends ArrayList<Integer[]>

{/\*customized methods go in here\*/}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/CreatedCharacter.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

import java.awt.geom.Line2D;

import java.util.ArrayList;

/\*\*

\* This class is similar to the ChineseStoke class, by the fact that it is used

\* for coding purposes, so instead of calling ArrayList<ArrayList<Integer[]>>

\* each time, one only has to call CreatedCharacter

\* This class has one custom method, erase(ChineseStroke cs), which removes the

\* strokes off of the CreatedCharacter based on the intersections of a different

\* ChineseStroke

\* @author Kieda

\* @since 3-13-2011

\*/

public class CreatedCharacter extends ArrayList<ArrayList<Integer[]>>{

//ArrayList<Stroke>

//Stroke<PointsOnStroke>

//PointsOnStroke[x,y]

/\*\*

\* Erases lines within the CreatedCharacter based on a given erasing

\* ChineseStroke

\*

\* Call this method if an erase function is implemented

\* @param cs the eraser path that will erase ant ChineseStrokes in its path

\*/

public void erase(ChineseStroke cs){

/\*NESTED LOOPS, ADVANCED SELECTION\*/

if(size()>1){

//checks if there are more than one ChineseStroke in the

//CreatedCharacter; otherwise errors could occur

for(int s = 0; s<(cs.size()-1);s++){

//goes around all of the strokes

Line2D l1 = new Line2D.Double(cs.get(s)[0], cs.get(s)[1]

, cs.get(s+1)[0] , cs.get(s+1)[1]);

//

for(int z = 1; z<size();z++){

for(int q = 0; q<(get(z-1).size()-1);q++){

Line2D l2 = new Line2D.Double(get(z).get(q)[0], get(z)

.get(q)[1], get(z).get(q+1)[0] , get(z)

.get(q+1)[1]);

if(l1.intersectsLine(l2)){

remove(z);

}

}

}

}

}

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/FontFinder.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

import java.awt.Font;

import java.awt.GraphicsEnvironment;

import java.util.Vector;

import mechanics.database.DatabaseLoader;

/\*\*

\*

\* @author Kieda

\*/

public class FontFinder {

public static Vector<String> chinesefonts;

public static int fontcount = 0;

public static void findFonts(){

DatabaseLoader.j1.setText(" Finding Chinese Fonts");

chinesefonts = new Vector();

Font[] allfonts = GraphicsEnvironment.getLocalGraphicsEnvironment()

.getAllFonts();

String chinesesample = "\u4e00";

for (int j = 0; j < allfonts.length; j++) {

if (allfonts[j].canDisplayUpTo(chinesesample) == -1) {

chinesefonts.add(allfonts[j].getFontName());

}

fontcount++;

}

try {

Thread.sleep(500);

} catch (InterruptedException ex) {}

DatabaseLoader.f.dispose();

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/LogManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

// <editor-fold defaultstate="collapsed" desc="imports">

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;// </editor-fold>

/\*\*

\* This class manages output to a log. This is a "static method" class so

\* LogManager does not need to be instantiated to log things

\* @version 4.2

\* @author Kieda

\* @since 3-14-2011

\*/

public class LogManager {

public static final String logsPath = "./logs/";//default path to the logs

public static final String OLD\_ERR\_PATH\_PRE\_2\_0 = "./logs/ERR.log";

public static final String OLD\_ERR\_PATH\_2\_0 = "./logs/ERR\_V2-0.log";

//older error log path name

public static final String DEFAULT\_ERR\_PATH = "./logs/ERR\_V3-0.log";

//default error log path (for this version)

public static final String DEFAULT\_LOG\_PATH = "./logs/LOGS\_V3-0.log";

//default log path

public static final String DATABASE = "characters.DAT";//Database path

public static ArrayList<String> logPaths = new ArrayList<String>();

private static ArrayList<BufferedWriter> writers =

new ArrayList<BufferedWriter>();

/\*\*

\* simply logs an error and the time it occured.

\* @param e the error to log (from the catch)

\*/

public static void logError(Exception e){

logError(e.toString()+"; "+e.getCause(),TimeManager

.getCurrentTimeAndDate());

//logs error

}

/\*\*

\* This method logs an error, the time it occured, the description, and the

\* place where the error happened

\* @param e the error to log (from the catch)

\* @param description the description of the error that occured (can be user

\* defined)

\* @param lineNumber the line number of which the error occured

\* @param classOfException the class which called the logError

\* @param print whether or not to printStackTrace()

\*/

public static void logError(Exception e, String description, int lineNumber

, String classOfException, boolean print){

if(print)//if the error is to be printed

e.printStackTrace();//print the error

logError(e.toString()+"; "+e.getCause(),TimeManager

.getCurrentTimeAndDate() + "; "+description + "; at line "

+lineNumber+" of "+ classOfException);

//compiles data to string, passes to logError(String thing

//, String time)

}

/\*\*

\* similar to logError(Exception e, String description, int lineNumber,

\* String classOfException, boolean print),

\* except this does not print anything (only logs it)

\* @param e the error to log (from the catch)

\* @param description the description of the error that occured (can be user

\* defined)

\* @param lineNumber the line number of which the error occured

\* @param classOfException the class which called the logError

\*/

public static void logError(Exception e, String description, int lineNumber

, String classOfException){

logError(e.toString() + "; " + e.getCause(), TimeManager

.getCurrentTimeAndDate() + "; " + description + "; at line "

+lineNumber + " of " + classOfException);

//compiles data to string, passes to logError(String thing,

//String time)

}

/\*\*

\* similar to logError(Exception e, String description, int lineNumber

\* , String classOfException), except that an object is used and is logged

\* @param e the error to log (from the catch)

\* @param description the description of the error that occured (can be user

\* defined)

\* @param lineNumber the line number of which the error occured

\* @param classOfException the class which called the logError, in Object

\* form

\*/

public static void logError(Exception e, String description, int lineNumber

, Object classOfException){

logError(e.toString( )+ "; " + e.getCause(),TimeManager

.getCurrentTimeAndDate() + "; " + description + "; at line "

+ lineNumber + " of "+ classOfException.getClass().getSimpleName());

//compiles data to string, passes to logError(String thing

//, String time)

}

/\*\*

\* simply logs an error and has a choice to print

\* @param e the error to log (from the catch)

\* @param print whether or not to printStackTrace()

\*/

public static void logError(Exception e, boolean print){

if(print)//if the error is to be printed

e.printStackTrace();//print the error

logError(e.toString(),TimeManager.getCurrentTimeAndDate());

//compiles error to string, passes to logError(String thing

//, String time)

}

/\*\*

\* logs something into the default log path

\* @param thing String to be logged

\*/

public static void logThing(String thing){

logThing(thing,DEFAULT\_LOG\_PATH);

//passes to logThing(String thing,String location)

}

/\*\*

\* logs a string to a specific location

\* @param thing the String to be logged

\* @param location the location for the String to be logged

\*/

public static void logThing(String thing,String location, String time){

logThing(time+"--\t"+thing,location);

//passes to logThing(String thing, String location, String time)

}

/\*\*

\* logs a String to a specific time

\* @param thing the String to be logged

\* @param time the time recorded

\*/

public static void logError(String thing, String time){

logThing(time+"--\t"+thing,DEFAULT\_ERR\_PATH);

//passes to logThing(String thing, String location, String time)

}

/\*\*

\* adds a line to the end of a log

\* @param thing the String to be logged

\* @param location the location for the String to be logged

\* @param time the time recorded

\*/

public static void logThing(String thing, String location){

try {

if(!logPaths.contains((new File(location).getAbsolutePath()))){

addLog(location);

}

for(int i = 0; i<logPaths.size();i++){

if(logPaths.get(i).equals((new File(location)

.getAbsolutePath()))){

writers.get(i).write(thing);

writers.get(i).newLine();

}

}

} catch (IOException ex) {/\*DO NOT LOG ERRORS HERE

(INFINITE LOOPS SUCK)\*/}

}

public static void addLog(String s){

addLog(new File(s));

}

public static void addLog(File f){

logPaths.add(f.getAbsolutePath());

try {

// <editor-fold defaultstate="collapsed" desc="initialization">

ArrayList<String> originalLog = new ArrayList<String>();

Scanner in = new Scanner(f);// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="loading file to be logged

into memory">

while (in.hasNext()) {

//goes through all the previous lines in a log

originalLog.add(in.nextLine());

//adds the file to be written to an ArrayList<String>

}// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="rewriting the file">

writers.add(new BufferedWriter(new FileWriter(f)));

//makes a new FileWriter to create the log

//makes a BufferedWriter to write to the log

for (String line : originalLog) {

//goes through all of the original lines

writers.get(writers.size()-1).write(line);//rewrites the log

writers.get(writers.size()-1).newLine();

//makes a new line for each line in ArrayList

}

// </editor-fold>

} catch (FileNotFoundException ex) {}

catch (IOException ex) {}

}

public static void closeAllWriters() throws IOException{

for(int i = 0; i<writers.size();i++)

writers.get(i).close();

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/TimeManager.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

import java.util.Date;

/\*\*

\* This class is soley responsible for giving

\* the time and date (as a String)

\* only one String available currently

\* @version 1.0

\* @author Kieda

\*/

public class TimeManager {

/\*\*

\* returns the current time and date in format

\* "Hours:Minutes:Seconds;Day/Month/Year"

\* @return the time and date in String format

\*/

public static String getCurrentTimeAndDate(){

Date d = new Date();//class date

String year = d.getYear()+"";

return d.getHours()+":"+d.getMinutes()+":"+d.getSeconds()+"; "+

d.getDate()+"/"+d.getMonth()+1+"/"+year.substring(1);

//these methods accessing the date are depricated (WHY WOULD SOMETHING AS

//USEFUL AS THIS DEPRICATE??? T.T)

}

}

/\*//////////////////////////////////////////////////////////////////////////////

./src/universals/UniversalDataStrorage.java

//////////////////////////////////////////////////////////////////////////////\*/

package universals;

import gui.mainframe.componentcreator.options.FloatingPartsCounterPanel;

import gui.mainframe.componentcreator.options.IntersectionStringInputPanel;

import gui.mainframe.componentcreator.options.StrokeComplexityPanel;

import java.util.ArrayList;

import mechanics.ChineseCharacter;

/\*\*

\*

\* @author Kieda

\*/

public class UniversalDataStorage {

public static int numberOfStrokes;

public static int totalNumberOfIntersections;

public static int numberOfFloatingParts;

public static int numberOfStraightStrokes;

public static int numberOfComplexStrokes;

public static int getXMouseDrawing;

public static int getYMouseDrawing;

public static ArrayList<Integer> intersections;

public static CreatedCharacter p = new CreatedCharacter();

public static ChineseStroke points = new ChineseStroke();

public static String character;

public static String meaning;

public static String pinyin;

public static ChineseCharacter[] database;

public static void updateOptions(){

FloatingPartsCounterPanel.setNumberOfFloatingParts(

numberOfFloatingParts);

IntersectionStringInputPanel.setIntersectionsString(

intersectionsToString(intersections));

StrokeComplexityPanel.setNumberOfComplexStrokes(numberOfComplexStrokes);

StrokeComplexityPanel.setNumberOfStraightStrokes(

numberOfStraightStrokes);

}

public static String intersectionsToString(ArrayList<Integer> intersections){

String s = "";

for(Integer i: intersections)

s+=i+",";

return s;

}

}

/\*////////////////////////////////////////////////////////////////////////>>Main

./test/CharacterPanelTest.java

//////////////////////////////////////////////////////////////////////////////\*/

import gui.mainframe.componentcreator.databasecreator.CharacterPanel;

import javax.swing.JFrame;

import mechanics.database.DatabaseCreator;

/\*\*

\*

\* @author Kieda

\*/

public class CharacterPanelTest extends JFrame{

public static void main(String[] args){

new CharacterPanelTest();

}

public CharacterPanelTest() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(500,500);

DatabaseCreator.init();

CharacterPanel cp = new CharacterPanel(DatabaseCreator.next());

add(cp);

setVisible(true);

}

}

/\*////////////////////////////////////////////////////////////////////////>>Text

./src/”note to anyone who is reading this.txt”

//////////////////////////////////////////////////////////////////////////////\*/

Hope you had as much fun reading the code as I had making it!

-Zachary Adam Kieda

-Candidate No. 00345-110