**CS 342 Report for (Project 1)**

**Code**

// Programmer: Bradley Golden

// Assignment: Project 1

// Date: September 15, 2015

// Description: A basic applet with the ability to display a rectangle, message, or UIC image

import java.awt.FlowLayout;

import java.awt.Graphics;

import java.awt.event.\*;

import java.awt.Color;

import javax.swing.\*;

import java.applet.\*;

import java.awt.Image;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

public class Project1 extends JApplet implements ActionListener

{

private static final String[] CHOICES = {" ", "Center", "Random"}; // User choices in combobox

private static final String[] COLORS = {" ", "Blue", "Red", "Green"}; // User choices in combobox

private JFrame parent; // Used for pop up windows

private int appletWidth; // Width of the applet

private int appletHeight; // Height of the applet

private int xCoord; // X coordinate location to draw in applet

private int yCoord; // Y coordinate location to draw in applet

private int borderHeight; // Border of the square to be drawn

private int borderWidth; // Border width of the square to be drawn

private int imageHeight; // Height of image to be drawn

private int imageWidth; // Width of image to be drawn

private int panelPadding; // Gives padding so drawings don't overlap the panel

/\*

\* GUI Front End Variables

\*/

// Row 1 contents

private JRadioButton drawSquareButton; // Radio choice for drawing square

private JRadioButton writeMessageButton; // Radio choice for writing a message

private JRadioButton drawImageButton; // Radio choice for drawing a picture

private JTextField messageField; // Message field for inputting a message

// Row 2 contents

private JLabel selectLabel; // Label to describe combobox

private JComboBox selectBox; // Provides the user with a set of options from

// CHOICES

private JLabel colorLabel; // Label to describe combobox

private JComboBox colorBox; // Provides the user with a set options from

// COLORS

private JCheckBox drawColor; // Option to draw a rectangle/message in color

private JButton drawIt; // Initiates the drawing

private Color textColor; // color to display message/rectangle

/\*

\* Back End Variables

\*/

private String message; // Used to store message

private Image image; // Used to store an image

@Override

public void init()

{

parent = new JFrame(); // Initialize new JFrame object

message = ""; // Initially draw nothing

image = getImage(getDocumentBase(), ""); // Initially draw nothing

imageHeight = 300; // Height of image

imageWidth = 300; // Width of image

borderHeight = 0; // Initially draw nothing

borderWidth = 0; // Initially draw nothing

appletWidth = getWidth(); // Get the width of the applet

appletHeight = getHeight(); // Get the height of the applet

xCoord = appletWidth/2; // Center width of the applet window

yCoord = appletHeight/2; // Center height of the applet window

panelPadding = 200; // Gives adequate padding for the panel

/\*

\* Row 1 Contents

\*/

// Initialize row 1 contents

drawSquareButton = new JRadioButton("Draw square", true); // Initialize draw square radio

drawImageButton = new JRadioButton("Draw image", false); // Initialize draw image radio

writeMessageButton = new JRadioButton("Write message:", false); // Initialize write message

// radio

messageField = new JTextField(20); // Initialize message field

messageField.addActionListener(this); // Get text from this field later

// Add contents to panel

JPanel contentPanel = new JPanel(); // Create new panel

contentPanel.add(drawSquareButton); // Add draw square button to panel

contentPanel.add(drawImageButton); // Add draw image button to panel

contentPanel.add(writeMessageButton); // Add write message button to panel

contentPanel.add(messageField); // Add message field to panel

// Logically link radio buttons

ButtonGroup rowOneGroup = new ButtonGroup(); // Add new group of buttons

// for linking

rowOneGroup.add(drawSquareButton); // Add draw square button to group

rowOneGroup.add(writeMessageButton); // Add write message button to group

rowOneGroup.add(drawImageButton); // Add draw image button to group

/\*

\* Row 2 Contents

\*/

//Initialize row 2 contents

selectLabel = new JLabel("Select where to draw:"); // Label for combobox

selectBox = new JComboBox(CHOICES); // Add new combo box populated from

// CHOICES

selectBox.setMaximumRowCount(3); // Set maximum combo box selections

// to 3

drawColor = new JCheckBox("Draw in color"); // Initialize draw in color checkbox

textColor = Color.BLACK; // Initially display black

colorLabel = new JLabel("Select a color:"); // Label for combobox

colorBox = new JComboBox(COLORS); // Add new combo box populated from

// COLORS

drawIt = new JButton("Draw It!"); // Initialize draw it button

drawIt.addActionListener(this); // Listen for click event

// add row 2 contents to panel

contentPanel.add(selectLabel); // Add select label button

contentPanel.add(selectBox); // Add select box

contentPanel.add(drawColor); // Add draw color checkbox

contentPanel.add(colorLabel); // Add color label for combobox

contentPanel.add(colorBox); // Add color combo box

contentPanel.add(drawIt); // Add draw it button

add(contentPanel); // Add content panel to GUI

} // end init

@Override

public void paint(Graphics g)

{

super.paint(g);

g.setColor(textColor); // change color as chosen

g.drawString(message, xCoord, yCoord); // Draw the current message

g.drawRect(xCoord, yCoord, borderWidth, borderHeight); // Draw the rectangle

g.drawImage(image, xCoord, yCoord, this); // Draw the image

} // end paint

public void actionPerformed(ActionEvent e)

{

if (selectBox.getSelectedIndex() > 0) // Check if the user selected a combo box option

{

if (selectBox.getSelectedItem() == "Center") // Center the drawing in the applet window

{

xCoord = appletWidth/2; // Get the x center of the applet

yCoord = appletHeight/2; // Get the y center of the applet

}

if (selectBox.getSelectedItem() == "Random") // Randomize the drawing location

{

xCoord = (int)(Math.random() \* appletWidth); // Randomize x coord

// Randomize y coord, gives padding so the square doesn't overlap the GUI panel

yCoord = (int)(Math.random() \* (appletHeight-panelPadding + 1)) + panelPadding;

}

}

else

{ // User didn't select something, prompt them to select from the combo box

JOptionPane.showMessageDialog(parent, "Please select where to draw"); // Prompt

return; // Do not process any further commands

}

if (drawColor.isSelected()) // Check if the user wants to draw in color

{

if(colorBox.getSelectedItem() == "Red") // User has selected to draw in red

textColor = Color.RED; // Set the color to red

else if(colorBox.getSelectedItem() == "Green") // User has selected to draw in green

textColor = Color.GREEN; // Set the color to green

else if(colorBox.getSelectedItem() == "Blue") // User has selected to draw in blue

textColor = Color.BLUE; // Set the color to blue

else // In case of error, set the color to Black

textColor = Color.BLACK; // Set to default black color

}

else // User doesn't want to draw in color :(

{

textColor = Color.BLACK; // Set the color to default black

}

if (e.getSource() == drawIt && writeMessageButton.isSelected()) // Write message upon clicking

{ // "Draw It!"

borderHeight = 0; // Clear the square

borderWidth = 0; // Clear the square

image = getImage(null, ""); // Clear the image

message = messageField.getText(); // Get text from message field

if (message.length() <= 0) // Prompt user if message not entered

{

JOptionPane.showMessageDialog(parent, "Please enter a message to display."); // Prompt

return; // Exit to stop futher processing

}

}

else if (e.getSource() == drawIt && drawSquareButton.isSelected())

{ // Draw square upon

// clicking "Draw It!"

message = ""; // Clear the message string

image = getImage(null, ""); // Clear the image

borderHeight = 100; // Set height of square

borderWidth = 100; // Set width of square

// Center the sqaure in the GUI

xCoord = xCoord-borderWidth/2; // Set x coord to left of square

yCoord = yCoord-borderHeight/2; // Set y coord to top of square

}

else if (e.getSource() == drawIt && drawImageButton.isSelected())

{ // Draw image upon

// clicking "Draw It!"

message = ""; // Clear the message string

borderHeight = 0; // Clear the square

borderWidth = 0; // Clear the square

image = getImage(getDocumentBase(), "uic\_flames.png"); // Set image to display

xCoord = xCoord - (imageWidth/2); // Center the image properly on the x axis

yCoord = yCoord - (imageHeight/2); // Center the image properly on the y axis

if(drawColor.isSelected())

{ // Set color border around image

borderHeight = imageHeight+10; // Set border height

borderWidth = imageWidth+10; // Set border width

}

}

repaint(); // Redraw items on screen

} // end actionPerformed

} // end Lab3 class

**Discussion**

This was a brief introduction to the IPO model and the process of developing applications on the front-end and back-end. One major concept that I learned in this programming assignment is that the various layouts for GUI applets can be fairly complex. In the case of this program, I learned that event listeners can be very tedious when not using proper interfacing methods. For example, there are many if statements that are required for all the various conditions that exist.