Software Engineering 2 BN509 **Assignment 2**

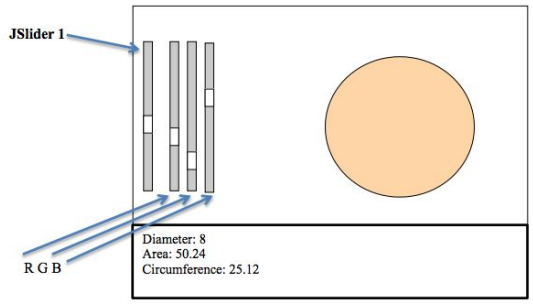
Student: Fiona Delaney | B00092671

Lecturer: Irene Murtagh
Date of Issue: 10th March 2016
Due Date: 27th March 2016

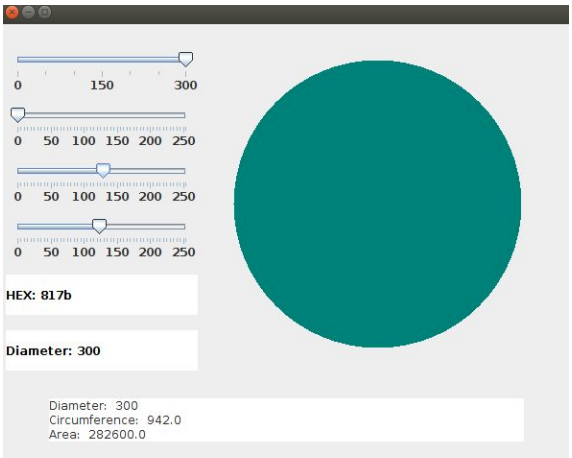
Requirements

You are required to design and create a Java Swing based application that implements Swing JSlider and a JTextArea.

Your application should consist of a Graphical User Interface, which draws a circle with a default diameter. To one side of the screen you should have four JSlider objects. Beneath all of this you should have a JTextArea. Use appropriate panels and layout managers to organise the GUI like this:



Assignment Diagram



Finished App: Fiona Delaney BN509

Specifics

1. JSlider 1 will be used to control the diameter of the circle - sliding will increase or decrease the circle size.
2. The read-only JTextArea will be used to display details about the diameter, areas and circumference of the circle.
3. Other JSlider objects will be used to control the RGB color ratio used to fill the circle:
4. JSlider 2 controls the amount of Red
5. JSlider 3 controls the amount of Green
6. JSlider 4 controls the amount of Blue
7. The user will use a mouse to move slider positions with an immediate screen response

Assignment Report | March 2016

Ref. java files: ColorSliderTest.java + .class file
ColorSlider.java + .class file

ColorSliderTest.java

Create Test file to drive the application.
Import relevant packages from API.

Define attributes of instance of ColorSlider and class TColor and to create an instance of ColorSlider displaying the interactive ColorSlider App onscreen.

Screengrab of ColorSliderTest file in Sublime Text

```java
/* Programme Description:Main ColorSliderTest file        *
/*********************************************************
//

import javax.swing.JFrame;

public class ColorSliderTest extends JFrame

  {
      public ColorSliderTest()
        {
          getContentPane().add(new ColorSlider());
          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          setSize(600, 600);
          setVisible(true);
        } // ends method ColorSliderTest

public static void main(String arg[])
  {
      new ColorSliderTest();
    }// end class ColorSliderTest

} // ends class ColorSliderTest

```

ColorSlider.java

Import relevant packages from API including layout, swing and awt managers.

```java
/* interactive colorwheel and report changes in JLabel
/* and JTextArea
/*********************************************************
//
import java.awt.BorderLayout;
import java.awt.Canvas;
import java.awt.Color;
import java.awt.GridLayout;
import javax.swing.JTextArea;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.Graphics;
import java.awt.Dimension;


class ColorSlider extends JPanel
  {
```

Create class ColorSlider which extends JPanel to define and implement layout.

Screengrab from class ColorSlider

Create constructor class ColorSlider() to place components on panel using GridLayout. Components include four JSliders(D,R,G and B) two JLabels (Diameter and Hex Ref) and JTextArea (Details). Create four instances of public class getSlider. Create one instance of Wheel class. SliderD controls diameter of the oval object and Sliders R, G and B control red, green and blue colour values in the foreground of the oval object.

```
public ColorSlider()
  {
    //setLayout with GridLayout and add JPanel to the NORTH
    panel.setLayout(new GridLayout(6, 1, 16, 16));
    add(panel, BorderLayout.NORTH);

    //call getSlider to define attributes of each slider
    sliderD = getSlider(0, 300, 50, 150, 50);
    sliderR = getSlider(0, 255, 0, 50, 5);
    sliderG = getSlider(0, 255, 0, 50, 5);
    sliderB = getSlider(0, 255, 0, 50, 5);

    // add sliders to panel
    panel.add(sliderD); //add diameter slider
    panel.add(sliderR); //add red value slider
    panel.add(sliderG); //add green value slider
    panel.add(sliderB); //add blue value slider

    // add JLabel to display Hex Ref
    rgbValue.setBackground(Color.white);
    rgbValue.setForeground(Color.black);
    rgbValue.setOpaque(true);
    panel.add(rgbValue);

    // add JLabel to display diameter
    diamValue.setBackground(Color.white);
    diamValue.setForeground(Color.black);
    diamValue.setOpaque(true);
    panel.add(diamValue);

    // add oval to display colourwheel
    add(oval);
    add(detail); // add JTextArea to display dimensions
  } // ends ColorSlider method
```

Screengrab from ColorSlider() constructor

Create public class getSlider which declares outlines key attributes, adds a ChangeListener and returns the sliders to ColorSlider constructor.

Screengrab of public class getSlider from ColorSlider

Create Wheel class which extends JPanel with defined attributes and methods.

```java
    // class to define Circle panel
class Wheel extends JPanel
      {
        //declare variables
      int redValue, greenValue, blueValue;
      private int diam = 64; // default diameter

      // draw and fill circle of specified diameter
      @Override
      public void paintComponent(Graphics g)
      {
      super.paintComponent(g);
      g.fillOval(32, 32, diam, diam);
      }
      // validate and set diameter, then repaint
      public void setDiam(int newDiam)
      {
      // if diameter invalid, default to 64
      diam = (newDiam >= 0 ? newDiam : 64);
      repaint(); // repaint panel
      }
      // get diameter
      public int getDiam()
        {
          return diam;
        }
      // get prefered circle size
      public Dimension getPreferredSize()
        {
          return new Dimension(380, 380);
        }
      // set minimum size
```

Screengrab1 of Wheel class extends JPanel from ColorSlider

```java
      public Dimension getMinimumSize()
        {
          return getPreferredSize();
        }
      // set foreground colour with red, green and blue values
      public void setForegroundColor()
        {
          Color color = new Color(redValue, greenValue, blueValue);
          setForeground(color);
        }
      // get foreground colour and return as Hex Ref
      public String getForegroundColorAsHex()
        {
          Color color = new Color(redValue, greenValue, blueValue);
          return Integer.toString(color.getRGB() & 0xffffff, 16);
        }
    } // ends class Wheel
```

Screengrab2 of Wheel class extends JPanel from ColorSlider

Create class SliderListener with nested inner class StateChanged to handle slider events, repainting the object oval with new diameter dimensions and new color. It also calculates and updates the dimensions in the JTextArea.

```
ColorSlider.java

    class SliderListener implements ChangeListener
        {

            public void stateChanged(ChangeEvent e)
            {
              JSlider slider = (JSlider) e.getSource();

              if (slider == sliderD) {
                oval.diam = slider.getValue();
                displayDiam();
              } else if (slider == sliderR) {
                oval.redValue = slider.getValue();
                displayRGBColor();
              } else if (slider == sliderG) {
                oval.greenValue = slider.getValue();
                displayRGBColor();
              } else if (slider == sliderB) {
                oval.blueValue = slider.getValue();
                displayRGBColor();
              } // ends final if statement of stateChanged Listener

              oval.repaint();

```

Screengrab of SliderListener and stateChanged inner class from ColorSlider

```
  detail.setText("Diameter:  " + oval.diam + "\n"
    + "Circumference:  " + oval.diam * pi + "\n"
    + "Area:  " + (oval.diam * oval.diam) * pi);
} // ends stateChanged
```

Screengrab of setText for JTextArea called detail within stateChanged inner class

SliderListener also contains a nested class displayRGBColor to display slider RGB values as a string Hex ref. in JLabel “Hex Ref” and class displayDiam to display length of diameter as a string in JLabel “Diameter”. Outer class SliderListener ends. Class ColorSlider ends.

```
          public void displayRGBColor()
            {
              oval.setForegroundColor();

              rgbValue.setText("HEX: "+ oval.getForegroundColorAsHex());
            } // ends displayRGBColor


          public void displayDiam()
           {
              oval.setDiam(sliderD.getValue());
              diamValue.setText("Diameter: "+ oval.diam);
            }

        } // ends SliderListener
    } // ends class ColorSlider
```

Screengrab of displayRGBColor and displayDiam inner classes from SliderListener.

Design decisions:

I chose JPanel and Gridlayout to ascribe components across the panel. I explored a few options including JFrame and GridbagLayout but I found JFrame too basic and GridbagLayout a bit more than I wanted.

I had a bit of trouble with GridLayout. At one point I lost my app! My code compiled and ran but displayed a white screen. Finally I realised that it was displaying but outside of the frame set-up in ColorSliderTest. The default origin is in the centre of the screen. I used the NORTH alignment of the panel and it displayed beautifully.

I found some code in Schaums Java Programming book which assesses the size of the user's screen and opens the app in the centre, displaying it at a ratio of screen dimensions. However, when I compiled it I learned that I was using a deprecated java.awt.toolkit and java.awt.dimension

```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.*;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.SwingConstants;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

public class Test
{
    public static void main(String[] args)
    {
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        int w = screenSize.width;
        int h = screenSize.height;
        JFrame frame = new MainFrame(w/2, h/2, w/4, h/4);
        frame.show();
    }
}
```

Sample deprecated getScreenSize()

As required, I used JSlider and JTextArea to create standard components and created class Wheel to define the circle object. I also used JLabels to display other information I was interested in highlighting.

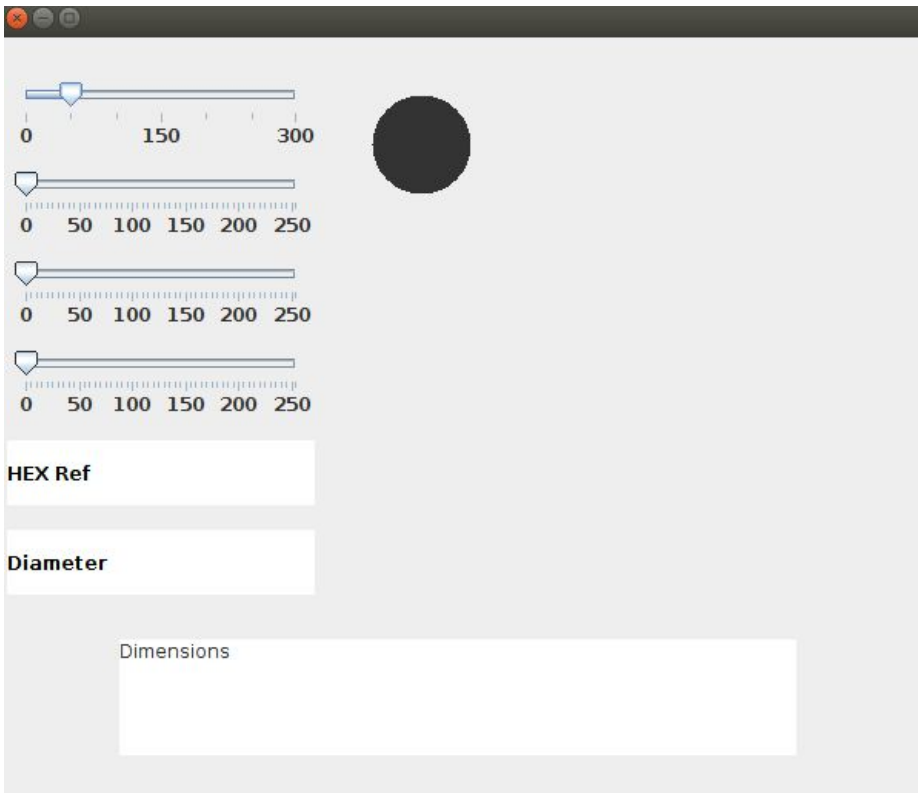
The sliders are all the same size. I have set the RGB minimum to 0 and maximum 255 - as with RGB hexadecimal color wheels used in digital displays. I set the Diameter slider to have a min of 0 and max of 300 to fit the panel.

At first, I used an anonymous class to handle events for sliderD. The RGB sliders were handled by the nested inner class stateChanged in class SliderListener. The effect however was disappointing. The changes in size viewed in a clunky manner and lacked the smoothness of the changes made in the color sliders. I learned that the nested inner classes are better for handling graphics than anonymous classes.

All components flow across the panel which is aligned to the NORTH. The JSliders appear HORIZONTALLY on the left side. The circle object, called oval, appears on the right.

The reason I didn't label the sliders is that I didn't like the cluttered look of the interface. The sliders in particular are well-understood as commonly used components. I think that once a user engages with the interface it becomes evident what each component does and how the user can interact with the app.

JLabels appear below the sliders on the left side and display Diameter and Hex Ref (because it's a useful additional facility).
JTextArea appears at the bottom and displays the details about oval's size changes via sliderD.

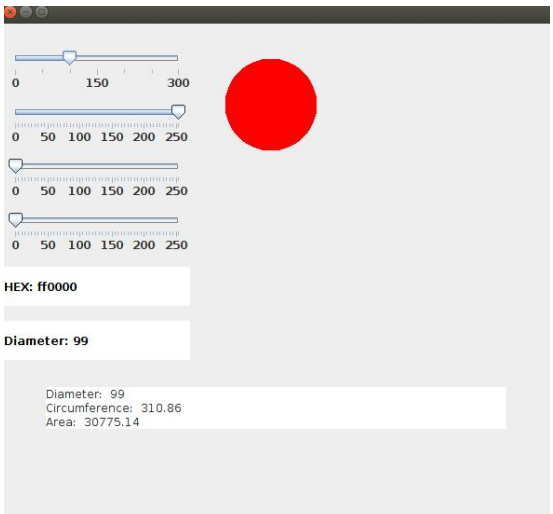


Screengrab of initial ColorSlider interface.

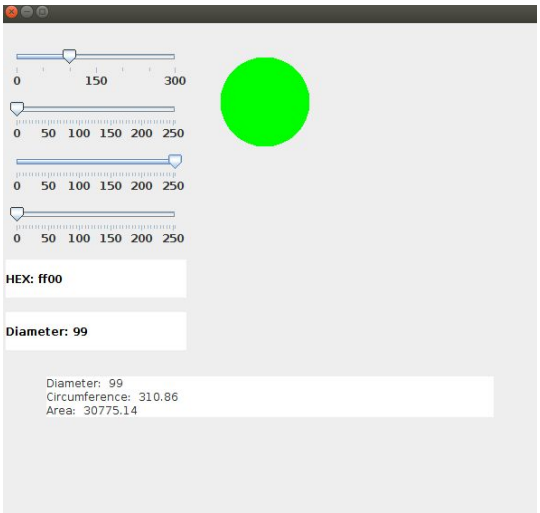
More images of App in action below.

Screengrabs of ColorSlider interface in action:

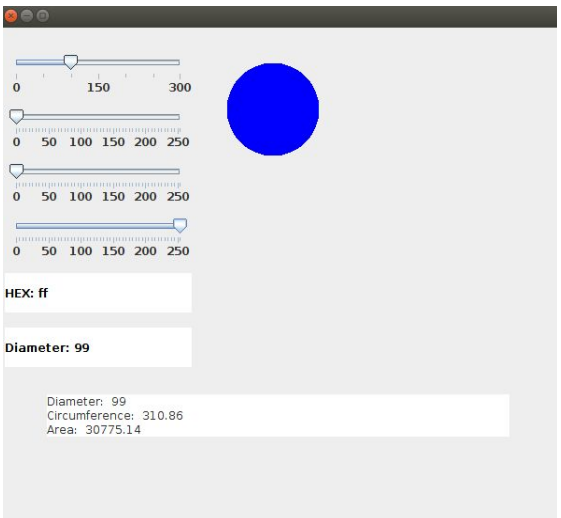
Red Slider: 255



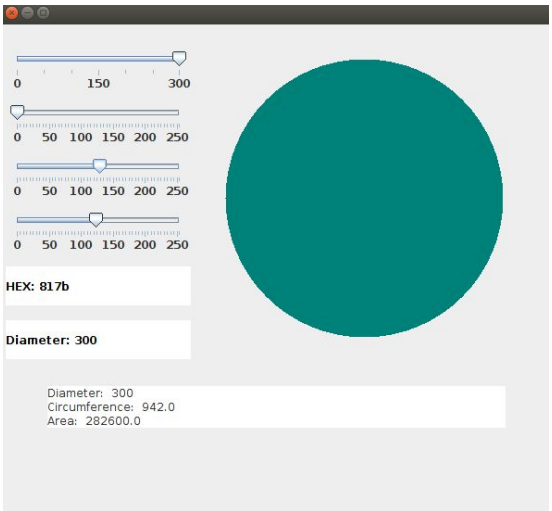
Green Slider: 255



Blue Slider: 255



Diameter Slider: 300



Fiona Delaney: B00092671 | BN509