1.00/1.001

Introduction to Computers and Engineering Problem Solving

Spring 2010 - Quiz 2

Name:	
MIT Email:	
TA:	
Section:	

You have 80 minutes to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary packages have already been imported. Good luck!

Question 1	**/ 20**
Question 2	**/ 40**
Question 3	**/ 40**
Total	**/ 100**

Question 1 – Recursion (20 points)

Write a recursive method to calculate the sum of all squares from 1 to n. For example, sumSquares(2) would return 5 = $2^2 + 1^2$ and sumSquares(3) would return 14 = $3^2 + 2^2 + 1^2$.

You can assume that n will be greater than or equal to 1. You do not need to check to see if this condition is met.

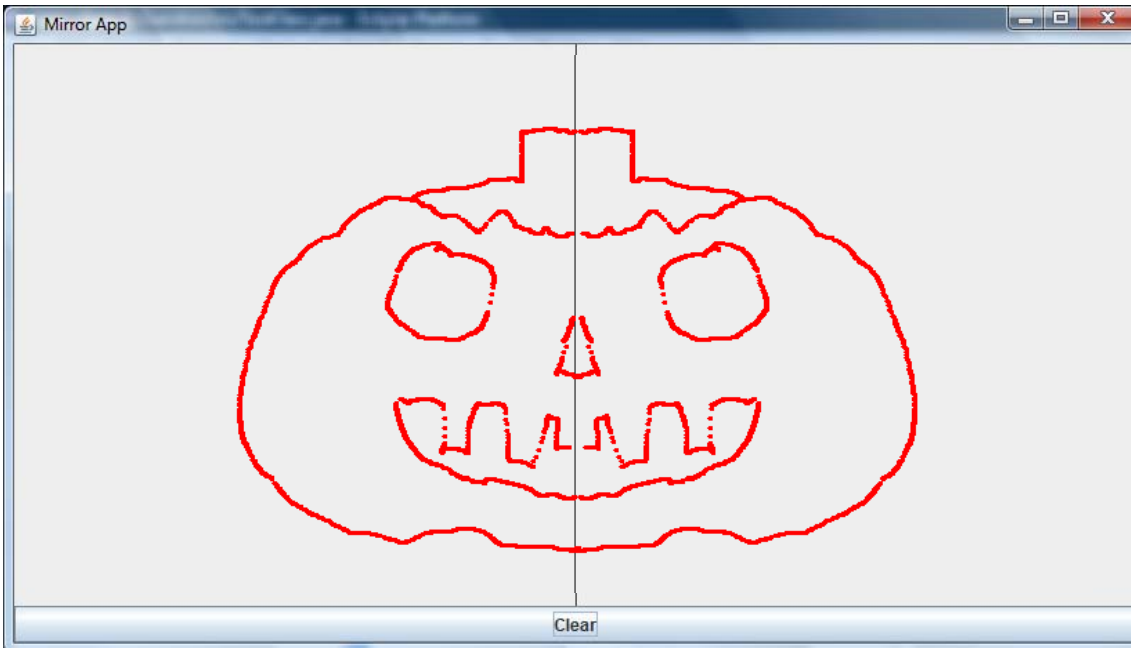
```
public int sumSquares(int n){
```

```
}
```

Question 2 - Swing (40 points)

You are entering the Swing portion of this quiz. You will be required to understand the code provided to you and write some yourself.

The application we are building is a simple painting application. Whenever a user draws a point on the panel (by dragging the mouse), a reflected point is drawn on the opposite half of the panel as well. Here is a screen-shot of what the panel would like if your TA was trying to draw a pumpkin for Halloween:



Part A:

The panel that draws these symmetric images is an extension of `JPanel`, and is called **`MirrorPanel`**. `MirrorPanel` has:

- A `private ArrayList<Point2D>`, called `paintPoints` that contains all the points created by dragging the mouse. `paintPoints` does NOT contain any of the symmetric points.
- A `public` method `addPoint(Point2D pt)` that adds a new point to the `paintPoints` ArrayList.
- A `public` method `clearPainting()` that clears the `paintPoints` ArrayList.

Look over the provided code. Two parts are missing in the `paintComponent()` method. You will be asked to fill them in the following question.

```
public class MirrorPanel extends JPanel{

	//List that holds all the paint points
	private ArrayList<Point2D> paintPoints;

	//Constructor for 800X400 pixel Mirror panel
	public MirrorPanel(){
		setPreferredSize(new Dimension(800, 400)); // Sets size of panel
		paintPoints = new ArrayList<Point2D>();
	}

	public void paintComponent(Graphics g){
		super.paintComponents(g);
		Graphics2D g2 = (Graphics2D) g;

		int panelHalfWidth = this.getWidth()/2;   // JPanel method

		//*************************************************************
		//
		// PART 1: draw the line that divides the panel in half
		//
		//*************************************************************

		g2.setColor(Color.RED);

		//Draw all the points and mirror points
		for(Point2D pt: paintPoints){

			int x = (int) pt.getX();
			int y = (int) pt.getY();

			//*******************************************************
			//
			// PART 2: draw the point and mirror-point as an ellipse
			//
			//*******************************************************

		}
	}

	public void addPoint(Point2D pt){
		paintPoints.add(pt);
	}

	public void clearPainting(){
		paintPoints.clear();
	}
}
```

The panel should have a vertical line drawn through its midpoint. Create a `Line2D` object that starts at the panel's top midpoint and ends at the bottom midpoint (like the line shown in the screenshot). We have provided a variable called `panelHalfWidth` that represents the midpoint (on the x-axis) for the panel. Use the `g2.draw()` function to draw the line.

Part 1 code goes here:

Now draw the `paintPoints` and the points that correspond to their reflection around the middle dividing line. We've provided code that iterates through the list of points. You should represent each point as a small circle. This can be done by filling in a small `Ellipse2D` object with a height and width of 4 pixels for each point. Each `Ellipse2D` object should be centered on the corresponding `pt`. You'll need to calculate the x-coordinate for the reflected point. To test your math: if the panel width is 800, and a point's x value = 150 then the mirror x value = 650.

Part 2 code goes here:

Part B:

Now we are going to build a small `JFrame` class that uses the panel. The class is called `TestClass` and it extends `JFrame`. It contains:

- A `MirrorPanel` called `mirror`.
- A `JButton` called `clearButton` that is responsible for clearing the `mirrorPanel`'s current `paintPoints`.

Read over the provided code. You will be asked to add the `mirror` and `clearButton` objects to the `TestClass`'s `ContentPane` to make the application look like the screen shot above. You will also be asked to add an `ActionListener` for the `clearButton` object.

```
public class TestClass extends JFrame{

	MirrorPanel mirror = new MirrorPanel();
	JButton clearButton = new JButton("Clear");

	public TestClass(){
		super("Mirror App");

	      //************************************************************
	      //
	      // PART 1: add the mirror and clearButton objects
	      //
	      //************************************************************

		//MouseMotionListener is a type of Listener that captures mouse drag
		//events on the mirror panel, and adds each mouse position point to the
		//mirror's list of draw points.
		mirror.addMouseMotionListener(new MouseAdapter() {
				public void mouseDragged(MouseEvent e) {
					Point2D mouseDragPoint = new
					     Point2D.Double(e.getX(), e.getY());
					mirror.addPoint(mouseDragPoint);
					repaint();
				}
			});

	      //************************************************************
	      //
	      // PART 2: add an ActionListener for clearButton
	      //
	      //************************************************************
	}


public static void main(String[] args) {
		TestClass frame = new TestClass();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.pack();
		frame.setVisible(true);
	}
}
```

Write code so that the `mirror` and `clearButton` objects are added to the `ContentPane` of the `TestClass`. You should write code so that the resulting layout would resemble the layout of the screenshot at the beginning of this problem.

Part 1 code goes here:

Using an anonymous inner class, add an `ActionListener` to the `clearButton`, so that the `mirror`'s `paintPoints` are erased and the frame is redrawn every time the button is clicked.

Part 2 code goes here:

Question 3 – Inheritance and Interfaces (35 points)

A physical analysis program written in Java has a library of materials available to the engineer to model physical entities. The materials are organized in a hierarchy of classes. Each material is eventually represented by an object holding the material properties, such as the density, electrical conductivity and elasticity. You do not need to worry about the units when using the material properties in this problem.

All material classes are derived from the `Material` superclass.

```
public abstract class Material
{
     private String name;

     public Material(String name){this.name = name;}

     public String getName() {return name;};

     public abstract double getDensity();
}
```

Interfaces are used to describe specific material properties. The `Conductor` interface must be implemented by all materials that can conduct electrical current.

```
public interface Conductor
{
     public double getConductivity();
}
```

The `Structural` interface must be implemented by all structural materials.

```
public interface Structural
{
     public double getElasticity();
}
```

Part A

Metals are materials conducting electrical current. Complete the `Metal` class below by filling in the boxes you need.

```
public class Metal
```

[ ]

```
{
      // data members
```

[ ]

```
      // constructor
      public Metal(String name, double dens, double conduct)
      {
```

[ ]

```
      }

      // other methods
```

[ ]

```
}
```

Part B

Steel is a metal widely used as a structural material. It has a density of 0.28 (pounds per cubic inch) and an elasticity of 29,000 (kilo-pound per square inch). Conductivity varies by type of steel. Complete the `Steel` class below by filling in the boxes you need.

```
public class Steel [                                    ]
{
      // data members
      [                                                 ]
      [                                                 ]
      [                                                 ]

      // constructor
      public Steel(String name, double conduct)
      {
            [                                           ]
            [                                           ]
            [                                           ]
      }

      // other methods
      [                                                 ]
      [                                                 ]
      [                                                 ]
}
```

Part C

The program also has classes to represent the physical entities made from the materials. You will write one such class to represent plates. A plate can only be made of a structural material and has an area and a thickness. In addition, the `Plate` class has a `printResistance()` method, which should do the following:

- If the plate is made of a material conducting electrical current, print out the electrical resistance of the plate, equal to: *thickness / (area x conductivity).*
- If the plate is not made of a material conducting electrical current, throw an `IllegalArgumentException`.

Complete the `Plate` class below by filling in the boxes you need.

```
public class Plate
{
     private [          ] material;

     private double area;
     private double thickness;

     public Plate([          ] m, double a, double t)
     {
          material = m;
          area = a;
          thickness = t;
     }

     public void printResistance() [                    ]
     {
          [                                        ]
     }
}
```

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012