University of Puerto Rico
Department of Electrical and Computer Engineering

ICOM 4015 Laboratory: Advanced Programming

Laboratory 3: Introduction to Java Graphics

Completed by:
ID:
Date:

1 Introduction

In this laboratory we will learn how to open a window and use various of the Java drawing functions.

2 Opening a window

Create a class of name **Lab3Exercises** and make it so that it contains the following code:

```
import javax.swing.JFrame;

public class Lab3Exercises {
	public static void main(String[] args) {
		JFrame myFrame = new JFrame("Hello, world!");
		//myFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		myFrame.setLocation(200, 200);
		myFrame.setSize(400, 100);

		myFrame.setVisible(true);
	}
}
```

Observe that the second line of code inside the **main()** function is commented out and thus will not be executed. Run the program. When the window appears, close it and wait several seconds.

Is the program still running? Hint: Click on the Console tab in Eclipse and check the Terminate button (red square) to see if it can be pressed.

//
//

Now uncomment the commented line of code by removing the two slashes (//) and run the program again. When the window appears, close it and wait several seconds.

Is the program still running?

//
//

From now on, make sure to close the window that appears before you modify the code.
Change the title to your name, and the size to 200 x 200. Also, adjust the location so that the window shows up somewhat approximately in the center of the display when the program is run.

When you are done, copy and paste your program below.

//
//

3 Painting the background and drawing a border

Create a new class in this project and call it **MyPanelClass**. Then make it so that it contains the following code:

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Insets;
import javax.swing.JPanel;

public class MyPanelClass extends JPanel {
	public void paintComponent(Graphics g) {
		super.paintComponent(g);

		//Compute interior coordinates
		Insets myInsets = this.getInsets();
		int x1 = myInsets.left;
		int y1 = myInsets.top;
		int x2 = getWidth() - myInsets.right - 1;
		int y2 = getHeight() - myInsets.bottom - 1;
		int width = x2 - x1;
		int height = y2 - y1;

		//Paint the background
		g.setColor(Color.CYAN);
		g.fillRect(x1, y1, width, height);
	}
}
```

If your version of Eclipse displays an icon to the left of the terms "**public class**" such that when you hover over it with the mouse it gives the warning "**The serializable class MyPanelClass does not declare a static final serialVersionUID field of type long**", then click on the icon and from the menu that appears double click on "**Add generated serial version ID**". Eclipse will add a few lines of code which you can ignore. You should then save the file. (If you do not get this warning, you can simply ignore this paragraph.)

Now add the following two lines of code to the main **Lab3Exercises** class. They should be added just before the one that says **myFrame.setVisible(true);**

```
		MyPanelClass myPanel = new MyPanelClass();
		myFrame.getContentPane().add(myPanel);
```

Now run the program. (If you are asked to "**Select a way to run 'Lab3Exercises'**", choose "**Java Application**".)

Now, close the window with your name and change the background color. To do so, put the cursor right after the word **CYAN** and delete it. Also delete the dot. Now type the dot again, and you will see a list of color options. Choose any color name that you want. (By convention, newer code should use uppercase names because they are symbolic constants. You may use lowercase names if you wish, however, it is recommended that you stick to convention and use uppercase names.) Run the program again to see the change.

Observe that the background is not fully painted: the right edge and the bottom edge are not fully painted. (*If you do not see this, change the color again until you do. They are very thin edges.*) To make it paint fully, change the **fillRect** line so that it looks like this:

```
g.fillRect(x1, y1, width + 1, height + 1);
```

Run the program again to confirm that it now paints the background in full.

Copy and paste the few lines of code after the comment that says "**Paint the background**". (The chosen background color should be different from cyan.)
//
//

Now add the following lines of code after the **fillRect** one:

```
//Draw a border
g.setColor(Color.YELLOW);
g.drawRect(x1, y1, width, height);
```

Note: Make sure that the background and the border are painted in fairly different colors or you may not see the border!

Run the program to see the background and border painted in the colors you've chosen. Also, try resizing the window and maximizing it to see how the program paints as the window size changes.

Whenever precision is important, note that whenever you use a function that fills (i.e., a function that begins with the word "fill" such as in **fillRect**) that you should add 1 to both the width and the height, and that whenever you use a function that draws (i.e., a function that begins with the word "draw" such as in **drawRect**) that you should *not* add 1 neither to the width nor the height. To see this, modify the **drawRect** line of code so that it looks like this:

```
g.drawRect(x1, y1, width + 1, height + 1);
```

Run the program and observe that the border is not painted at the right and bottom edges. That's because when we added 1 to the width and/or height to the **drawRect** function, it went beyond the visible area.

Now change the **drawRect** line of code so that it looks as before:

g.drawRect(x1, y1, width, height);

Run the program again to confirm that the background and border are painted correctly.

Now, add some code so that a second border is painted, this time it should appear inside the border we have already drawn and in a third different color. The size of the new border does not matter as long as it lies inside the outer border. Here is a sample picture:

You can use any three fairly distinct colors that you want. In this example, the outer border is in yellow and the inner border is in black. Hint: You may need to use **x1 + *some number***.

When you are done, copy and paste below *only the code you've added.*

//
//

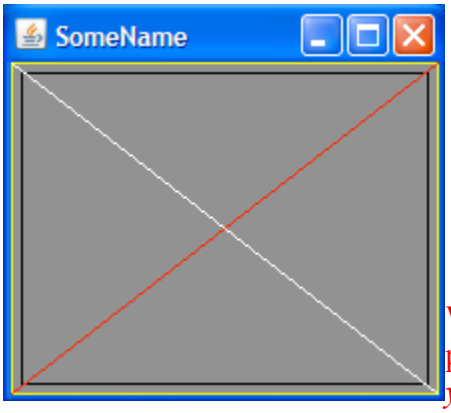
4 Lines and ovals

Add the following lines of code to the **MyPanelClass** class and run the program:

```
g.setColor(Color.WHITE);
g.drawLine(x1, y1, x2, y2);
```

If you're already using the color white, change it to a different color. If you're not, changing the color is optional. (Observe that when drawing lines, we use (x_2, y_2) coordinates instead of width and height.) Run the program and resize the window to observe the effect.

Now, make it so that another line is drawn, in a fifth distinct color, from the top right corner to the bottom left corner. Here is a sample picture:



When you are done, copy and paste below *only the code you've added.*

//
//

Now, comment out all of the lines of code that are used to draw the two borders and the two lines. To do so, highlight them (as if you were to do a copy and paste) and select "**Source → Toggle Comment**" from the Eclipse menu, or manually add the two slashes // in front of every line of code. Now run the program and verify that only the background gets painted.

Add the following lines of code and change the color if necessary:

```
g.setColor(Color.LIGHT_GRAY);
g.drawOval(x1, y1, width, height);
```

Run the program. You should see an oval. Again, try resizing the window and maximizing it to see how the program paints as the window size changes.

Make it so that the oval is filled. To do so, erase the word **drawOval** and the dot too. Type the dot again, and use the appropriate function. Run the program to verify.

Now make it so that the oval is always 55 x 55 in size. Run the program to verify.

Copy and paste below the line of code that draws the filled 55 x 55 oval.

//
//

Now, let's change the first two arguments (currently **x1** and **y1**) so that the oval is always at the center. Answer the following questions:

- Which variable holds the current width of the window? Answer: ///// /////
- What's the width of the oval we're using? Answer: ///// /////
- How much horizontal width is *not used* by the oval within the window? (The answer is a simple mathematical expression.) Answer: ///// /////

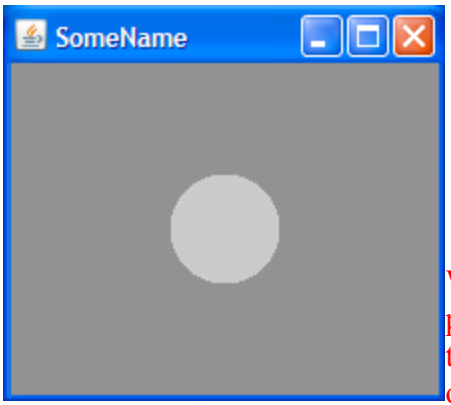
- How much horizontal width should be at each side (left, right) of a centered oval?

Answer: ///// /////

- Which *x* coordinate value should replace **x1** in order to have an horizontally centered oval? (Consider that **x1** is the leftmost *x* from which the empty left side begins.)

Answer: ///// /////

Replace **x1** with the answer to the previous question and run the program. Verify that the oval stays horizontally centered as you resize the window. Now, substitute **y1** similarly and verify that the oval also stays centered vertically. Here is a sample picture:



When you are done, copy and paste below the line of code that draws the centered 55 x 55 oval.

//
//

5 Polygons

Now comment out the lines of code used to draw the oval, and add the following ones:

```
Polygon p = new Polygon();
p.addPoint(x1 + 5, y1 + 25);
p.addPoint(x1 + 20, y1 + 10);
p.addPoint(x1 + 35, y1 + 25);
p.addPoint(x1 + 25, y1 + 25);
p.addPoint(x1 + 25, y1 + 45);
p.addPoint(x1 + 15, y1 + 45);
p.addPoint(x1 + 15, y1 + 25);
g.setColor(Color.YELLOW);
g.drawPolygon(p);
```

Wait a few seconds. If you notice that Eclipse places an icon next to the line that says "**Polygon p = new Polygon();**", then click on it and then double click "**Import 'Polygon' (java.awt)**" so that Eclipse adds the following line of code close to the top of your class code. (If you do not see an icon, then manually add the following line of code next to all of the other **import** ones.)

import java.awt.Polygon;

Run the program.

Which figure do you see? (If you don't see a figure, change the color until you do.)

//
//

Now, replace the call to **drawPolygon** with a call to **fillPolygon** and run the program again. Make sure that you see that the figure has been filled before you continue.

Comment out the piece of code that draws the figure of the previous exercise, and then add the following to your program:

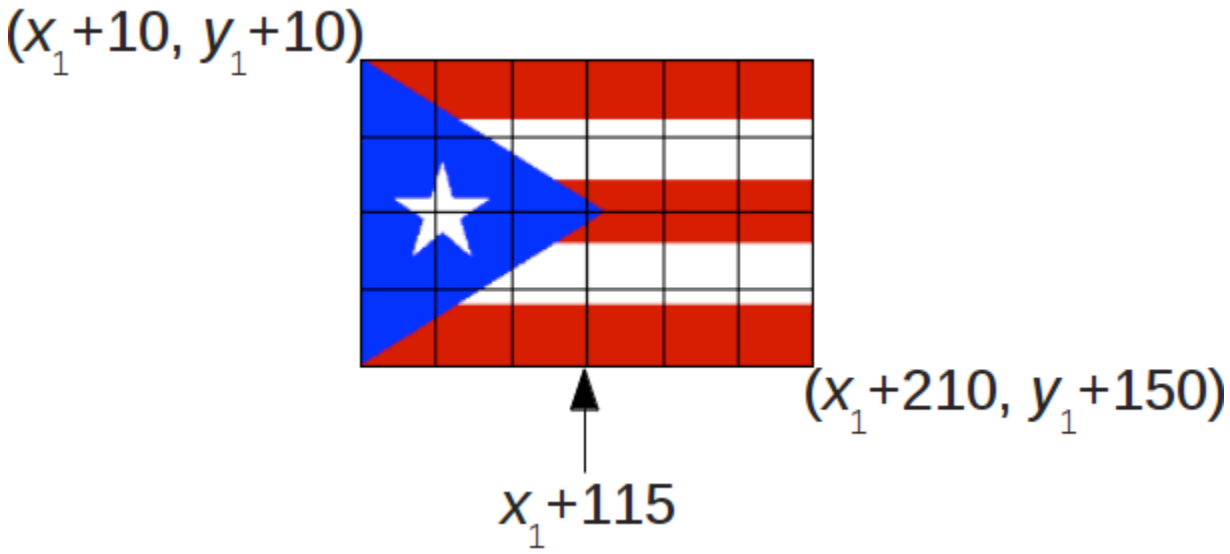
```
Polygon p2 = new Polygon();
p2.addPoint(x1 + 25, y1 + 73);
p2.addPoint(x1 + 41, y1 + 73);
p2.addPoint(x1 + 47, y1 + 58);
p2.addPoint(x1 + 53, y1 + 73);
p2.addPoint(x1 + 69, y1 + 73);
p2.addPoint(x1 + 56, y1 + 83);
p2.addPoint(x1 + 61, y1 + 98);
p2.addPoint(x1 + 47, y1 + 88);
p2.addPoint(x1 + 34, y1 + 98);
p2.addPoint(x1 + 38, y1 + 83);
g.setColor(Color.WHITE);
g.drawPolygon(p2);
```

Run the program.

Which figure do you see?

//
//

For the final exercise, add code so that when the program runs, it draws the flag of Puerto Rico. You can use the image below for reference, just keep using the figure from the previous exercise. Please note that coordinates are approximate. Which drawing order should you use? *The final drawing does not need to look perfect.* It should just look fairly similar. Change the window size if necessary by editing the **Lab3Exercises** class.



When you are done, copy and paste below *the entire code you have in the **MyPanelClass** class.*

//
//