**CIS 162 Lab 3**

**Simple Graphics**

**Objectives**

After completing this lab, you should be able to:

* *create* a class
* *invoke* methods
* *draw* simple shapes

**Practice Good Coding Style**

Elegant source code that follows the GVSU [Java Style Guide](http://www.cis.gvsu.edu/java-coding-style-guide/).

**X-Y Coordinate System**

Computers often use a coordinate system a bit different from the regular x-axis and y-axis. The x-axis increases from left to right like usual but the y-axis increase from top to bottom. Therefore, the origin (0, 0) is in the upper left corner.

(0, 0) x-axis

y-axis

**Draw a Rectangle**

Four numbers determine the location and size of a rectangle: 1) the x location of the upper left corner, 2) the y location of the upper left corner, 3) the width 4) and the height.

g.drawRect(x, y, width, height);

For example, the following statement will draw a 200 x 100 rectangle at location (40, 60).

g.drawRect(40, 60, 200, 100);

**Lab Activity #1 – Draw A Picture**

1. Start BlueJ
2. Create a new project called “Lab3”.
3. Create a new Class called “Drawing”
4. Replace all of the code with the template provided below.
5. Does it compile?
6. Right click and invoke the main() method.

**Starting Template**

Read the comments within the code to learn how to draw rectangles, ovals, lines and text. Copy and paste this template into a BlueJ project.

import javax.swing.\*;

import java.awt.\*;

public class Drawing extends JPanel{

public static void main(String[] a) {

JFrame f = new JFrame();

f.setContentPane(new Drawing());

f.setSize(600, 400);

f.setVisible(true);

}

public void paintComponent(Graphics g){

// this statement required

super.paintComponent(g);

// optional: paint the background color (default is white)

setBackground(Color.CYAN);

// display words

g.setColor(Color.black);

g.drawString("Picture Title", 130, 20);

// draw a solid and empty rectangle

g.setColor(Color.RED);

g.fillRect(100, 100, 70, 50);

g.drawRect(100, 170, 70, 50);

// draw a solid and empty oval

g.setColor(Color.GREEN);

g.fillOval(50, 10, 70, 50);

g.drawOval(20, 50, 70, 50);

// draw lines

g.setColor(Color.BLUE);

g.drawLine(190, 50, 190, 150);

g.drawLine(210, 50, 210, 150);

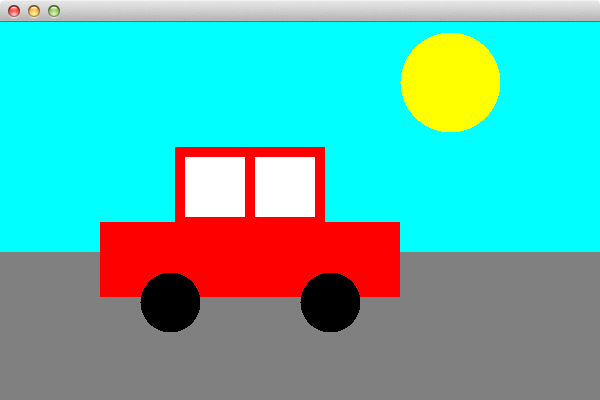
g.drawLine(230, 50, 230, 150);

}

}

**Lab Activity #2 – Draw A Car**

Modify the paintComponent() method to draw a car similar to Figure 1. It contains two red rectangles for the car body, two white rectangles for windows, two black circles, one yellow circle and a gray rectangle for the road.

****

**Figure 1. A sweet car**

**Lab Activity #3 – Reposition the car on demand**

Revise your code, if necessary, to use variables for the location of each element. You should be able to reposition the car easily by changing ONE variable. Show your instructor and be prepared to reposition the car as requested.

**Examples:**

g.fillRect(x, y, width, height);

g.fillRect(x + 2, y + 10, 80, 80);

g.fillRect(x - 5, x + 2, width - 3, height - 4);

**Lab Activity #4 – Add a second car**

Create a second car with unique variables so that you can repositioned each car independently.

**Lab Activity #5 – Animate the car's movement**

Ask your instructor about minor changes to your code for one, or both, of the cars to 'move' as the user moves the mouse.

**Grading Criteria**

This lab is worth a possible 10 points. Show your instructor or lab assistant your solution.