Project 12

CS 1323, Fall 2015

# Learning Objectives

1. Construct objects from at least 3 classes from: Polygon, Color, Rectangle2D.Double, Arc2D.Double, Ellipse2D.Double, Line2D.Double, RoundRectangle2D.Double. (30 points)
2. Call at least one instance method on each of the three objects constructed above (30 points)
3. Use at least three different Color constants from the Color class. (15 points)
4. Use at least three methods from the Graphics2D class that are not used in the sample code. (15 points)

10 points will be awarded for the documentation of your program. That means using good names for variables, proper and consistent indentation of code, comments and meaningful use of whitespace.

Section 10: When your program is completed and running, have the teaching assistants check it to get credit for the lab. If you do not finish during the laboratory, this project is due by 11:59 p.m. on Monday, November 16.

Sections 1 and 995: This project is due by 11:59 p.m. on Monday, November 16.

# Description

Java has some wonderful facilities for creating graphics. This week we’ll use some simple graphics to explore creating, accessing and mutating objects. The purpose of this project is to give you a chance to learn to use new classes in the API. So read the API carefully. Remember that the API contains lots of details that are confusing to beginning programmers, so skip over the parts that don’t make sense.

The coordinate system for graphics is shown in the figure below. The origin is in the upper left corner of the screen. The x axis is horizontal. The y axis is vertical.

x

y

(0,0)

Width is measured in the horizontal direction. Height is vertical.

The code below will draw a diagonal line across the screen. This code uses some programming techniques that we’re not familiar with (like extends and paintComponent(Graphics g)). Don’t worry about these things as they do not impact your project. I’ve marked the parts of the code that you should not change with comments.

**// I’m using the \* notation here for imports since you may use many classes from**

**// these packages**

**import** javax.swing.\*;

**import** java.awt.\*;

**import** java.awt.geom.\*;

**public** **class** ExperimentWithGraphics **extends** JPanel

{

// You may change this constant

**private** **static** **final** **int** ***SIZE*** = 300;

**public** **static** **void** main(String[] args)

{

// You may change the parameters in the code below a little,

// but proceed with extreme caution. Do not reorder the methods.

JFrame frame = **new** JFrame("Line");

JPanel panel = **new** ExperimentWithGraphics();

frame.setSize(***SIZE***,***SIZE***);

frame.getContentPane().add(panel, BorderLayout.***CENTER***);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setVisible(**true**);

}

// Do not change this method name or parameters

**public** **void** paintComponent(Graphics g)

{

// Do not change the next two lines of code

**super**.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

// This is where your code should go

Line2D.Double line = **new** Line2D.Double(0, 0, ***SIZE***, ***SIZE***);

g2d.setColor(Color.***BLACK***);

g2d.draw(line);

// Don’t change anything after this

}

}

To draw a picture in Java, you have to construct an object of the proper type in 2D, then select a color for drawing, and then draw the object. The process is shown above in the last three lines of the paintComponent() method. The names of the classes that are appropriate to use are given in the objectives. Choose them strategically to make an interesting drawing.

The easiest way to draw something is to create a mock-up on paper first. Graph paper is ideal for this, if you happen to have some. If not, you can draw a picture on paper and figure out approximately where the objects should meet. Remember that the order you display things in can make things a lot easier. For example, if you want to draw a daisy, you can draw ellipses for the petals first, then put in the center over the top of the petals. This keeps the join between the petals and the center clean and nice and cannot be seen in the final product and avoids a whole bunch of tricky and unnecessary math.