Lab Exercise 10
Java Graphics
CS 2334

October 29, 2015

Introduction

This lab will give you experience with creating graphics in Java. A knowledge of using graphics will allow you to customize your GUI with shapes and colors. Combined with what you have already learned about graphical components (**JButton**, **JLabel**, **JTextField**, etc.), there are infinite possibilities!

Your specific task for this lab is to put together a set of shapes that will create the image of a spooky pumpkin.

Learning Objectives

By the end of this laboratory exercise, you should be able to demonstrate a knowledge of graphics by:

1. Creating a window
2. Adding various graphical components to the window
3. Customizing the size, location, and color of those components to form an organized image

Proper Academic Conduct

This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

1. Import the existing lab10 implementation into your eclipse workspace.

   (a) Download the lab10 implementation: `http://www.cs.ou.edu/~fagg/classes/cs2334/labs/lab10/lab10-initial.zip`

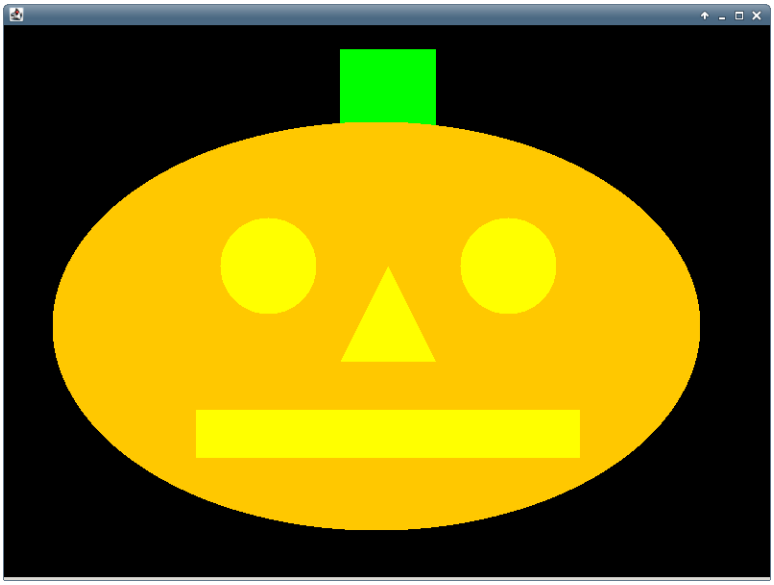
   (b) In Eclipse, select *File/Import*

   (c) Select *General/Existing projects into workspace*. Click *Next*

   (d) Select *Select archive file*. Browse to the lab10-initial.zip file. Click *Finish*

Image

Below is the illustration that you will be mimicking.



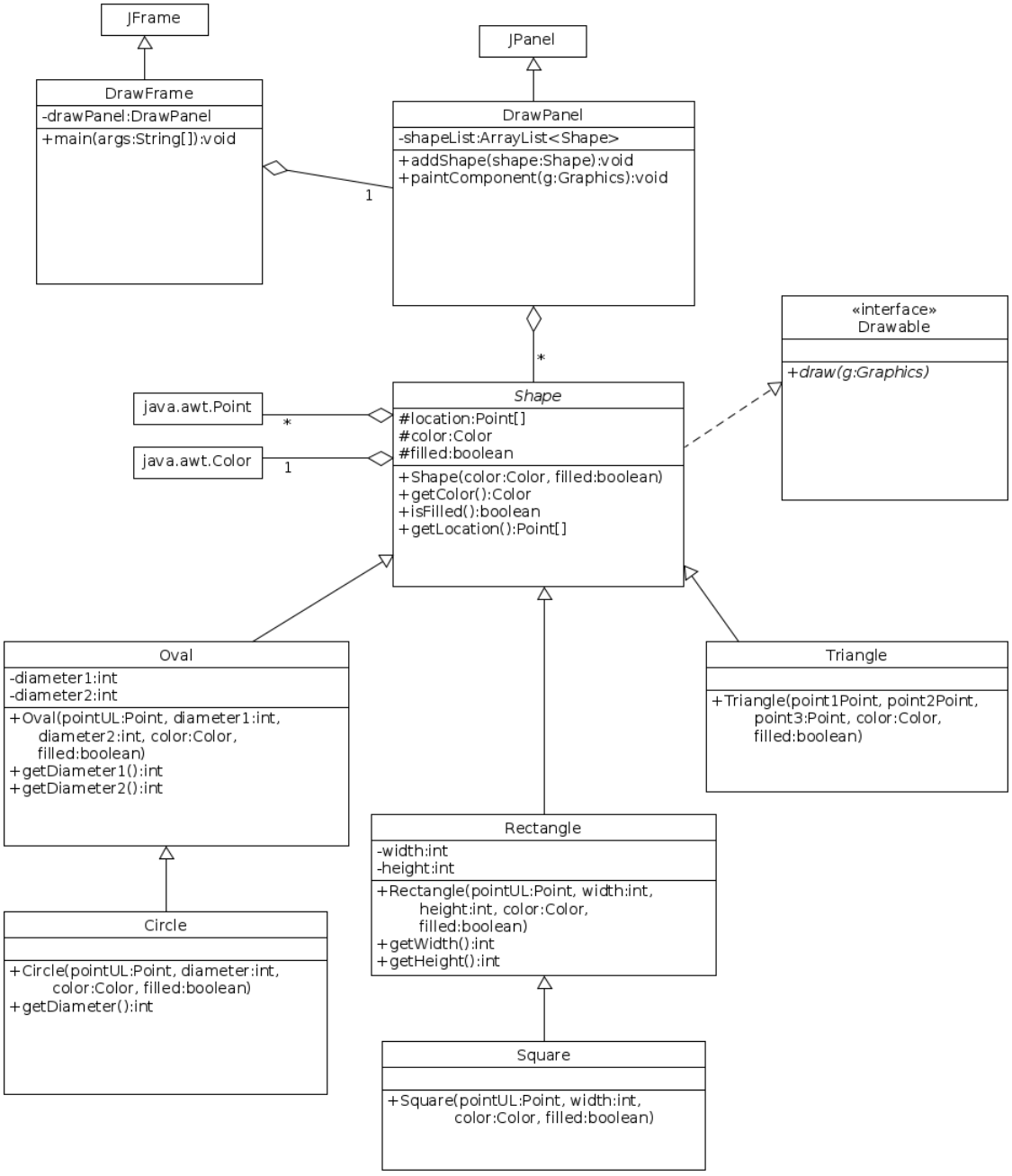
This is only an image and does not provide any way for a user to interact (other than closing the window).

Specifically, the picture shown is made up of the following components:

- A **Square** for the stem
- An **Oval** as the pumpkin
- Two instances of **Circle** for the eyes
- A **Triangle** for the nose
- A **Rectangle** for the mouth

You do not need to worry about matching the exact locations shown in the example picture, and you are allowed to use a small amount of creativity in your choice of sizes and colors. However, each of the shapes *must* be represented, and your final product *must* closely resemble ours. Be aware that some of these shapes do overlap, so the order in which you add them to the panel is important. You will not receive credit for a shape that is technically in the panel but not visible.

UML


JFrame
JPanel
DrawFrame
-drawPanel:DrawPanel
+main(args:String[]):void
DrawPanel
-shapeList:ArrayList<Shape>
+addShape(shape:Shape):void
+paintComponent(g:Graphics):void
1
«interface»
Drawable
+draw(g:Graphics)
*
java.awt.Point
java.awt.Color
*
1
Shape
#location:Point[]
#color:Color
#filled:boolean
+Shape(color:Color, filled:boolean)
+getColor():Color
+isFilled():boolean
+getLocation():Point[]
Oval
-diameter1:int
-diameter2:int
+Oval(pointUL:Point, diameter1:int,
diameter2:int, color:Color,
filled:boolean)
+getDiameter1():int
+getDiameter2():int
Triangle
+Triangle(point1Point, point2Point,
point3:Point, color:Color,
filled:boolean)
Rectangle
-width:int
-height:int
+Rectangle(pointUL:Point, width:int,
height:int, color:Color,
filled:boolean)
+getWidth():int
+getHeight():int
Circle
+Circle(pointUL:Point, diameter:int,
color:Color, filled:boolean)
+getDiameter():int
Square
+Square(pointUL:Point, width:int,
color:Color, filled:boolean)

Lab 10: Specific Instructions

All of the classes shown in the UML are provided in lab10-initial.zip.

1. Make sure that in your final implementation, all variables and methods shown in the UML are included in these classes
    - Be sure that the class name is exactly as given in the initial zip file
    - You must use the default package, meaning that the package field must be left blank
    - Do not change the variable and method names provided
    - You are responsible for making sure all method documentation is complete
2. Modify the code according to the TODO instructions given in the comments
3. Implement JUnit tests to check your shape constructors

Notes

- DrawPanel is the **only** class that provides a *paintComponent()* method. The shapes are drawn because this method calls their *draw()* method.

Final Steps

1. Generate Javadoc using Eclipse.
    - Select *Project/Generate Javadoc...*
    - Make sure that your project is selected, as is the CalculatorFrame
    - Select *Private* visibility
    - Use the default destination folder
    - Click *Finish*
2. Open the *lab10/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that that all of your classes are listed and that all of your documented methods have the necessary documentation.
3. If you complete the above instructions during lab, you may have your implementation checked by one of the TAs.

Submission Instructions

- All required components (source code and compiled documentation) are due at 11:59pm on Friday, October 30th.
- Prepare your submission file:
  1. Select the project in the *Package Explorer* window.
  2. Right-click. Select *Export*
  3. Select *General/Archive File*
  4. Expand your project and verify that both the *src* and *doc* folders are selected, as well as all of their contents
  5. Enter the archive file name: *lab10.zip* (note that you may want to browse to a different destination folder)
  6. Select *Save in zip format*
  7. Click *Finish*
- Submit your zip file to the lab10 folder on D2L.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 5 points

(5) The program is properly formatted (including indentation, curly brace and semicolon locations).

(3) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(7) There is one error in data type or method call selection.

(4) There are two errors in data type or method call selection.

(0) There are three or more errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(7) A component of one required method is missing.

(4) A method is not implemented or components are missing from two methods.

(0) Two or more required methods are not implemented or components from three or more methods are missing.

Unit Tests: 10 points

(10) A full set of unit tests has been implemented.

(8) One key component is not tested properly by the unit tests.

(6) Two key components are not tested properly by the unit tests.

(4) Three key components are not tested properly by the unit tests.

(2) Four key components are not tested properly by the unit tests.

(0) Five or more required methods are not implemented or components from three or more methods are missing.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.
(10) The program fails one test.
(5) The program fails two tests.
(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.
(8) The program executes, but there is one minor error.
(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used and has the correct contents.

(5) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.