Lab Exercise 10
Java Graphics
CS 2334

October 27, 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 Pikachu.

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.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.zip file. Click *Finish*

Image

Below is the illustration that you will be mimicking.

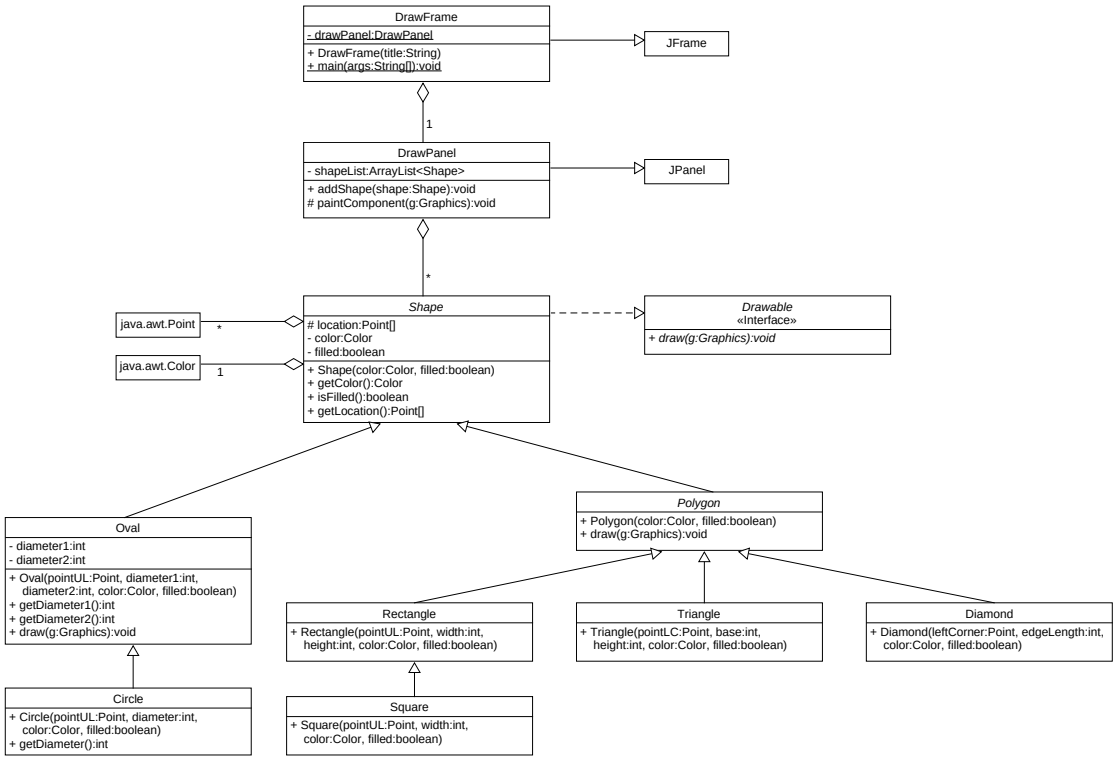
This is only an image and does not provide any way for a user to interact (other than closing the window).

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

- Two instances of **Circle** or **Oval** for the body and head
- Two instances of **Circle** for the eyes
- Two instances of **Circle** for the cheeks
- Two instances of **Triangle** for the ears
- Two instances of **Diamond** for the tail
- Two instances of **Rectangle** for the sky and ground

You do not need to worry about matching the exact locations shown in the example picture – 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



Lab 10: Specific Instructions

All of the classes shown in the UML are provided in lab10.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 the non-drawing aspects of the Shape classes. For example, creating a **Circle** instance should result in an object with the correct center and radii.

Notes

- DrawPanel is the **only** class that provides a *paintComponent()* method. The shapes are drawn because this method calls their *draw()* methods.
- If our code has a bug, fix it. If our code is missing JavaDoc, fill it in. If there are style problems (as identified by Web-Cat), fix the code. In other classes, as well as in industry, the code you are asked to add to and edit will not always be perfect.

Final Steps

1. Generate Javadoc using Eclipse.
   - Select *Project/Generate Javadoc...*
   - Make sure that your project is selected, as is the DrawFrame
   - 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 28th.

- Method 1: Submit through Eclipse

  1. From the *Window* menu, select *Preferences/Configured Assignment*.
  2. Select your project.
  3. From the Project menu, select *Submit Assignment*.
  4. Under *Select the assignment to submit*, select *Lab 10: Shape Drawing*.
  5. Click *Change Username or Password....* Enter your Web-Cat username and password. Click *OK*. You should only need to do this step once per session.
  6. Click *Finish*.
  7. Your browser should automatically open a Web-Cat page that shows your submission being graded. After a short wait, the page will show a report of your submission. See the main class web page for a link that describes the Web-Cat output.

- Method 2: Submit directly to the Web-Cat server

  1. From the File menu, select *Export*.
  2. Select *Java/JAR File*. Click *Next*.
  3. Select and expand your project folder.
  4. Select your *src* and *doc* folders.
  5. Select *Export Java source files and resources*.
  6. Select an export destination location (e.g., your *Documents* folder/directory). This file should end in *.jar*
  7. Select *Add directory entries*.
  8. Click *Finish*.
  9. In your web browser, login to the Web-Cat2 server.
  10. Click the *Submit* button.
  11. Browse to your jar file.
  12. Click the *Upload Submission* button.
  13. The next page will give you a list of all files that you are uploading. If you selected the correct jar file, then click the *Confirm* button.

14. Your browser will then open a Web-Cat page that shows your submission being graded. After a short wait, the page will show a report of your submission. See the main class web page for a link that describes the Web-Cat output.

Rubric

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

Correctness/Testing: 45 points

The Web-Cat server will grade this automatically upon submission. Your code will be compiled against a set of tests (called *Unit Tests*). **NOTE:** The tests used for this lab will focus on button interactions and values inside textFields - a style of testing different than what we have done before. These unit tests will not be visible to you, but the Web-Cat server will inform you as to which tests your code passed/failed. This grade component is proportional to the fraction of tests that your code passes (so 22.5 points means that your code passed half of the tests)

Style/Coding: 20 points

The Web-Cat server will grade this automatically upon submission. Every violation of the *Program Formatting* standard described in Lab 1 will result in a subtraction of a small number of points (usually two points). Looking at your submission report on the Web-Cat server, you will be able to see a notation for each violation that describes the nature of the problem and the number of subtracted points.

Design/Readability: 35 points

This element will be assessed by a grader (typically sometime after the lab deadline). Any *errors* in your program will be noted in the code stored on the Web-Cat server, and two points will be deducted for each. Possible errors include:

- Non-descriptive or inappropriate project- or method-level documentation (up to 10 points)
- Missing or inappropriate inline documentation (2 points per violation; up to 10 points)
- Inappropriate choice of variable or method names (2 points per violation; up to 10 points)
- Inefficient implementation of an algorithm (minor errors: 2 points each; up to 10 points)
- Incorrect implementation of an algorithm (minor errors: 2 points each; up to 10 points)

- Incomplete coverage of your Unit Tests. We expect that your unit tests will test all lines of your code (up to 15 points)
- Non-visible components (2 points per violation; up to 10 points)
- Unrecognizable Pikachu - all elements are present but the picture is not easily identified as Pikachu (5 points)

If you do not submit compiled Javadoc for your lab, 5 points will be deducted from this part of your score.

Note that the grader may also give *warnings* or other feedback. Although no points will be deducted, the issues should be addressed in future submissions (where points may be deducted).

Bonus: up to 5 points

You will earn one bonus point for every two hours that your assignment is submitted early.

Penalties: up to 100 points

You will lose ten points for every minute that your assignment is submitted late.