

Lab Exercise 12

CS 2334

November 12, 2015

Introduction

In lab 10, we learned about creating drawings using primitive shapes. To produce these shapes, we made use of low level graphics generation methods provided by the **Graphics** class. The final drawing was created in the *main()* method by creating a fixed sequence of shapes.

In this lab, we will introduce a Graphical User Interface that will enable a user to create a drawing using any list of shapes. For any given shape, the user will be able to select the type of shape, its color and whether the shape is filled. The position and size of the shapes will then be determined by the user's clicking and dragging in the drawing panel. In addition, the GUI will provide a *File Menu* that enables the user to load and save drawings, create a new drawing, and exit from the program.

Learning Objectives

By the end of this laboratory exercise, you should be able to:

1. Create a drop-down menu for a graphics frame
2. Attach actions to drop-down menu items
3. Interact with the user with appropriate pop-up dialog boxes
4. Open files for reading and writing object-level data
5. Draw a set of primitive shapes on a panel

Proper Academic Conduct

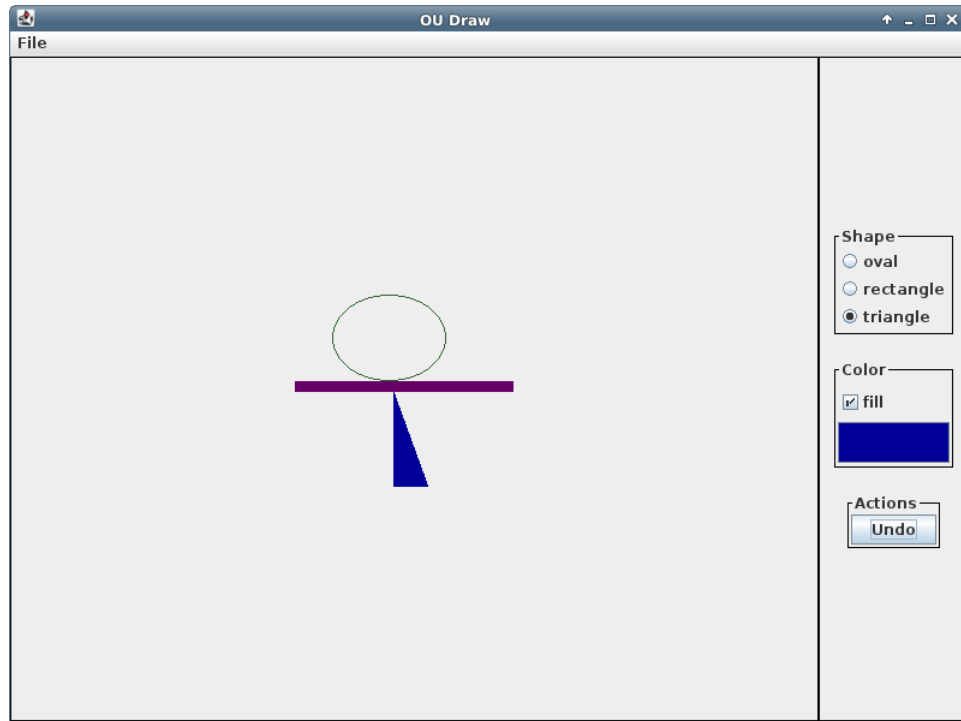
This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

1. Import the existing lab12 implementation into your eclipse workspace.
 - (a) Download the lab12 implementation:
`http://www.cs.ou.edu/~fagg/classes/cs2334/labs/lab12/lab12-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 lab12-initial.zip file. Click *Finish*

Drawing with Shapes

Below is a picture of the graphical user interface after a partial drawing has been completed:



The control panel to the right allows the user to do the following:

- Select between one of three shapes: oval, rectangle and triangle
- Determine whether the shape is to be filled or not
- Selects the color with which to be drawn
- Delete the most recently created shape

Mouse events within the drawing panel lead to the following behavior:

- Pressing mouse button down: the current cursor position determines the first corner of the object to be drawn

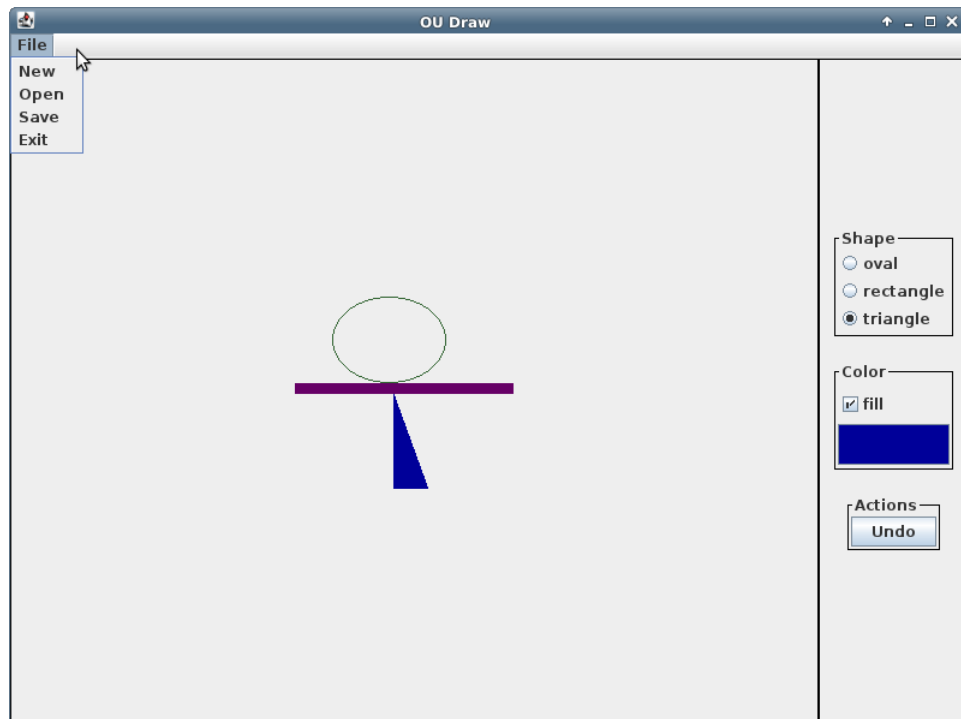
- Moving the cursor while pressing down: the current cursor position determines the second corner of the object. A temporary object is drawn to show the user what it would look like if the choice is made final
- Lifting up from the mouse button: the current cursor position determines the second corner of the object. A new object is added to the official list of objects

Note that the first and second corners may have any relative relationship. It is also useful to be able to describe the upper-left (UL) and lower-right (LR) corners.

The position and extent of the three different objects is determined as follows:

- Rectangle and oval: the position property of these shapes corresponds to the UL corner. The width and height (or diameters) of these shapes is determined by the difference between the UL and LR corners.
- Triangle: we are only considering right triangles that are axis-aligned (each of the legs of the right angle aligns with either the X or Y axis of the figure). The first corner specified by the user defines the location of the vertex with the right angle. The difference between the first and second corners specifies the width and height of the triangle.

The graphical user interface also includes a file menu:

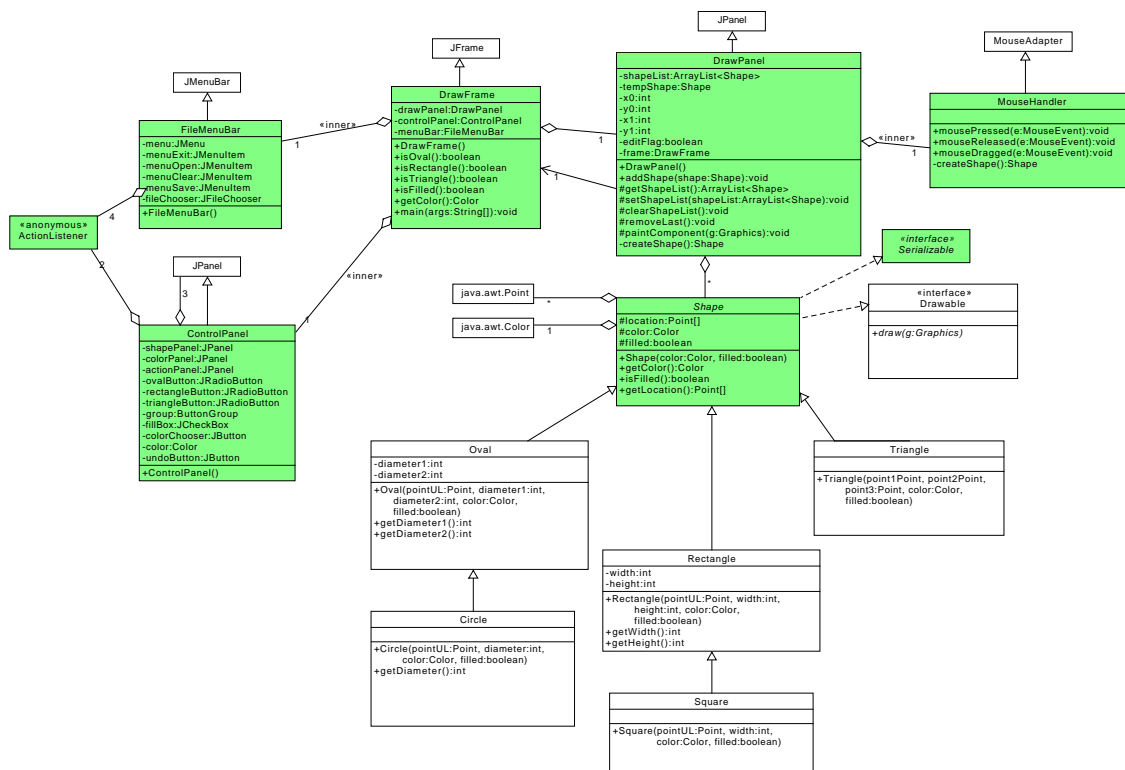


The actions for the file menu are:

- **New:** after confirming with the user, clears the drawing
- **Open:** after prompting the user for a file to open, a drawing is read from the file. Errors are indicated with pop-up dialog boxes
- **Save:** after prompting the user for a file to which to save, the drawing is written to the file. Errors are indicated with pop-up dialog boxes
- **Exit:** the program exits immediately

UML

The UML for this lab is given below. Half of the classes are from lab 10 and do not require changes. The green classes are new or have been modified to some degree.



Lab 12: Specific Instructions

All of the classes shown in the UML are provided in lab12-initial.zip.

1. Most of the classes are implemented. The key components that you must add are:

- **DrawPanel**

- createShape(): create the correct shape, depending on what what points have been selected by the mouse, and which shape type, color and fill that the user has chosen.
- paintComponent(): complete implementation

- **DrawFrame.FileMenuBar**

- Complete the implementation of the menu
- Add code for behavior of the four menu items

- **DrawFrame.ControlPanel**

- Define the behavior for the color button
- Define the behavior for the undo button

2. Do not add functionality to the classes beyond what has been specified
3. Don't forget to document as you go!

Final Steps

1. Generate Javadoc using Eclipse.
 - Select *Project/Generate Javadoc...*
 - Make sure that your project is selected, as well as all of the Java source files
 - Select *Private* visibility
 - Use the default destination folder
 - Click *Finish*

2. Open the *lab12/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 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, November 13th.
- 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: *lab12.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 lab12 folder on D2L.

Rubric

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

Implementation: 35 points

Program formatting: 10 points

- (10) The program is properly formatted (including indentation, curly brace and semicolon locations).
- (5) There is one problem with program formatting.
- (0) The program is not properly formatted.

Data types and method calls: 15 points

- (15) The program is using proper data types and method calls.
- (10) There is one error in data type or method call selection.
- (5) There are two errors in data type or method call selection.
- (0) There are multiple 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.

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.