Lab Exercise 12: Graphics and Serialization
CS 2334

November 9, 2017

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 construct 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 and various mouse events
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.zip`

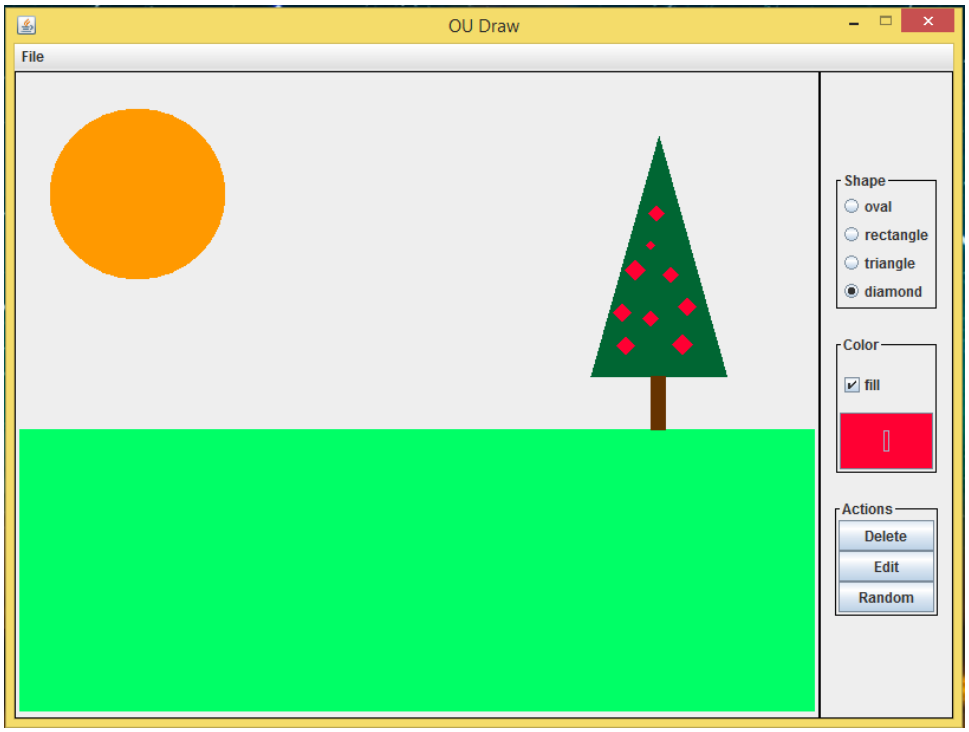
   (b) In Eclipse, select *File/Import*

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

   (d) Select *Select archive file*. Browse to the lab12.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 four shapes: oval, rectangle, triangle, and diamond
- Determine whether the shape is to be filled or not
- Select the color with which to be drawn
- Delete a shape
- Edit a shape's color and/or fill
- Create a randomly generated object

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

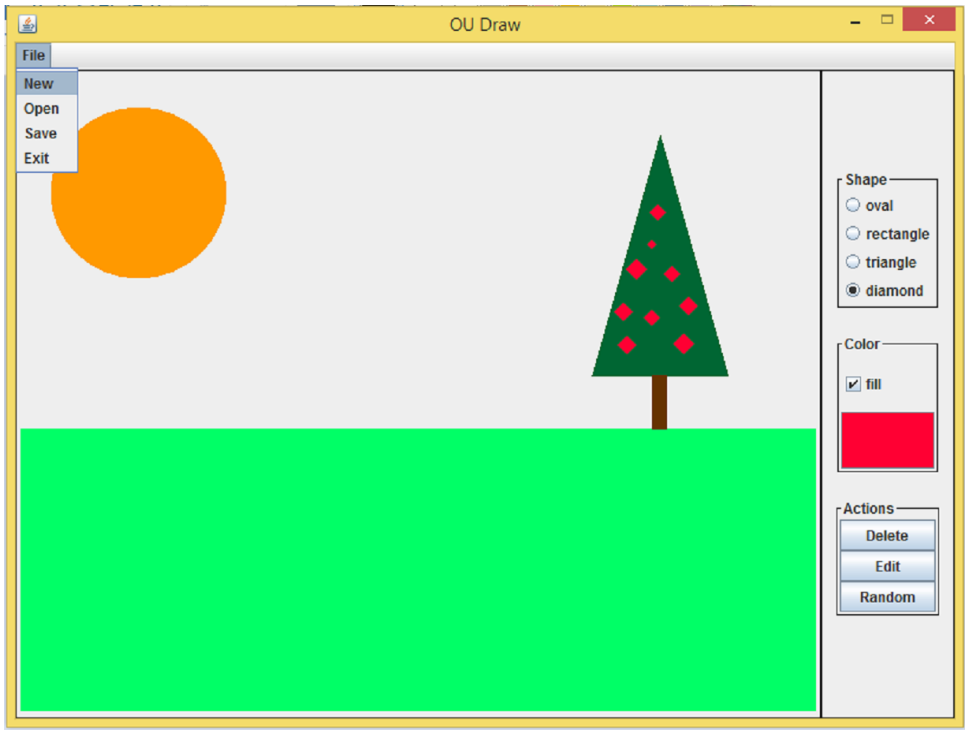
- Pressing mouse button down: the current cursor position determines: one of three things:
  1. if in drawing mode: the first corner of the object to be drawn
  2. if in edit or delete mode: the shape to be edited or deleted
- Moving the cursor while pressing down (in drawing mode): the current cursor position determines the second corner of the object. A temporary object is drawn to show the user what the object would look like if the choice is made final
- Lifting up from the mouse button (in drawing mode): 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 points selected by the user may have any relative relationship. In creating the shape objects, these two points must be transformed into appropriate parameters for the Shape constructors.

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

- Rectangle and oval: the two points selected by the user determine the opposite "corners" of these objects. As in lab 10, the constructor for these shapes takes as input the upper-left corner and the width and height (or diameters).
- Triangle: we are only considering isosceles triangles with the base that is aligned along the x axis. The first corner specified by the user defines the location of one vertex along the base of the triangle. The difference between the first and second corners specifies the length of the base and the height of the triangle.
- Diamond: the point selected by the user defines one vertex of the diamond. The second selected point determines the orientation of the diamond relative to the first point, as well as the side length. The diamond constructor takes as input the leftmost point and the side length.

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

Representing Program State

The key instance variables used to represent the state of your program are as follows:

1. *shapeList* is a list of **Shapes** that have already been created

2. *tempShape* is a Shape that is currently being dragged. This is used to give the user a view of what their shape will look like once they release the mouse button

3. *shapeIndex* is an index of a shape in the *shapeList* that has been selected for editing

4. *drawingFlag* is set to **true** if the user is in the middle of drawing a new shape

5. *x0/y0* are the coordinates of the cursor when the user presses down on the mouse button

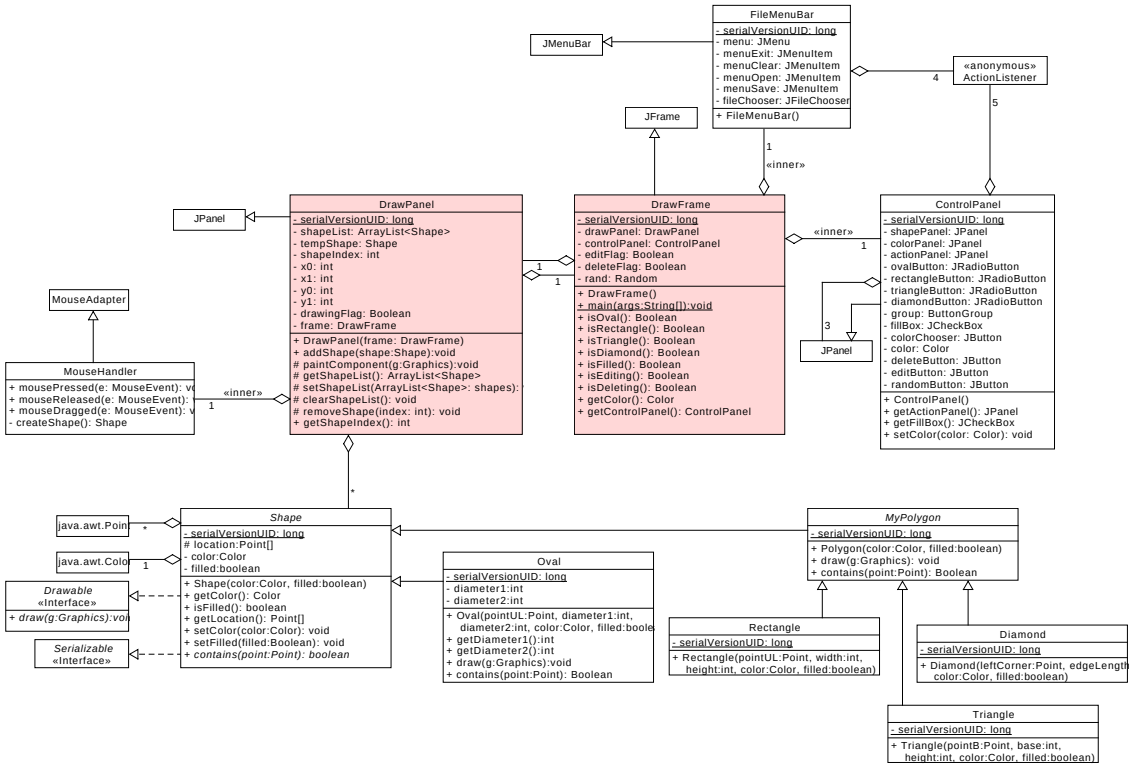
6. *x1/y1* are the coordinates of the cursor while it is being dragged (with the mouse button down)

7. *editFlag* indicates that the interface is in an object edit mode

8. *deleteFlag* indicates that the interface is in an object delete model

UML

The UML for this lab is given below. The classes in the lower half are only slightly modified from lab 10. They have been provided to you completely. The classes in the upper half of the UML are heavily modified from lab 10 or have been added for this lab.



Lab 12: Specific Instructions

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

1. Most of the classes have complete implementations. The key components that you must add are:

   - **DrawPanel**
     - paintComponent(): complete implementation

- **DrawPanel.MouseHandler**
  - mousePressed(): handle what a mouse press means when in the various modes - drawing, editing, or deleting
  - mouseDragged(): handle updating the temporary shape when in drawing mode
  - mouseReleased(): finalize the shape being drawn and add it to the shape list when in drawing mode
  - 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.
- **DrawFrame.FileMenuBar**
  - Complete the implementation of the menu
  - Add code for behavior of 3/4 menu items (file reading is implemented for you).
- **DrawFrame.ControlPanel**
  - Define the behavior for the color button

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

Testing

You only need to perform testing by interacting with your program. You do not need to provide your own JUnit tests. Note that we will provide JUnit tests, against which your code will be compiled.

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 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 7:00pm on Saturday, November 11th. You will not need JUnit tests for this lab. You will need to test your code manually through interaction with the GUI.

- Submit your code to *Lab 12: Graphics and Serialization* on WebCat using 1 of the 2 methods described from lab 1.

Rubric

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

Correctness/Testing: 40 points

The Web-Cat server will grade this automatically upon submission. Your code will be compiled against a set of tests (called *Unit Tests*). 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: 25 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)

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.