CS 112-01
Project 3: DrawPad: Graphics Editing Application.

Part 1

Part 1: Due Tuesday April 15th, 11.59pm (25 pts)
Part 2: Due Tuesday, April 22st, 11:59pm (25 pts)

For this project you will create a simple drawing program. The program should allow users to create several kinds of shapes (such as Circles, Ovals, Rectangles and general Polygons) on the drawing canvas, select them by clicking on them with the mouse, and then move them around the canvas and rotate them. There should also be an option to draw free-form strokes by dragging the mouse (the body of the whale in the image below was drawn using a "FreeDraw" mode). The user should be able to undo each operation. Finally, your application should be able to save and reload the drawings using a given format.

You will be using Java Swing to create GUI for your application. You are not allowed to use any GUI Designers. The user interface of your program should look similar to the image in Figure 1.

Part 1:
User Interface: this part of the project will allow you to be creative and use some advanced features of Java Swing. At a minimum, your application should have the following:

- canvas: a drawing area where the user can draw shapes,
- 3 buttons for creating 3 different shapes (such as a Circle or an Oval, a Rectangle, a general Polygon)
- 1 button for FreeDraw mode (drawing a free-form stroke while dragging the mouse)
- Erase button that allows the users to clear the canvas
- 3 Sliders for controlling the amount of Red, Green and Blue in the color. The sliders should be accompanied by 3 labels (Red, Green, Blue).
- 2 buttons for loading and saving a file. These buttons do not need to do anything for the first submission of the project.

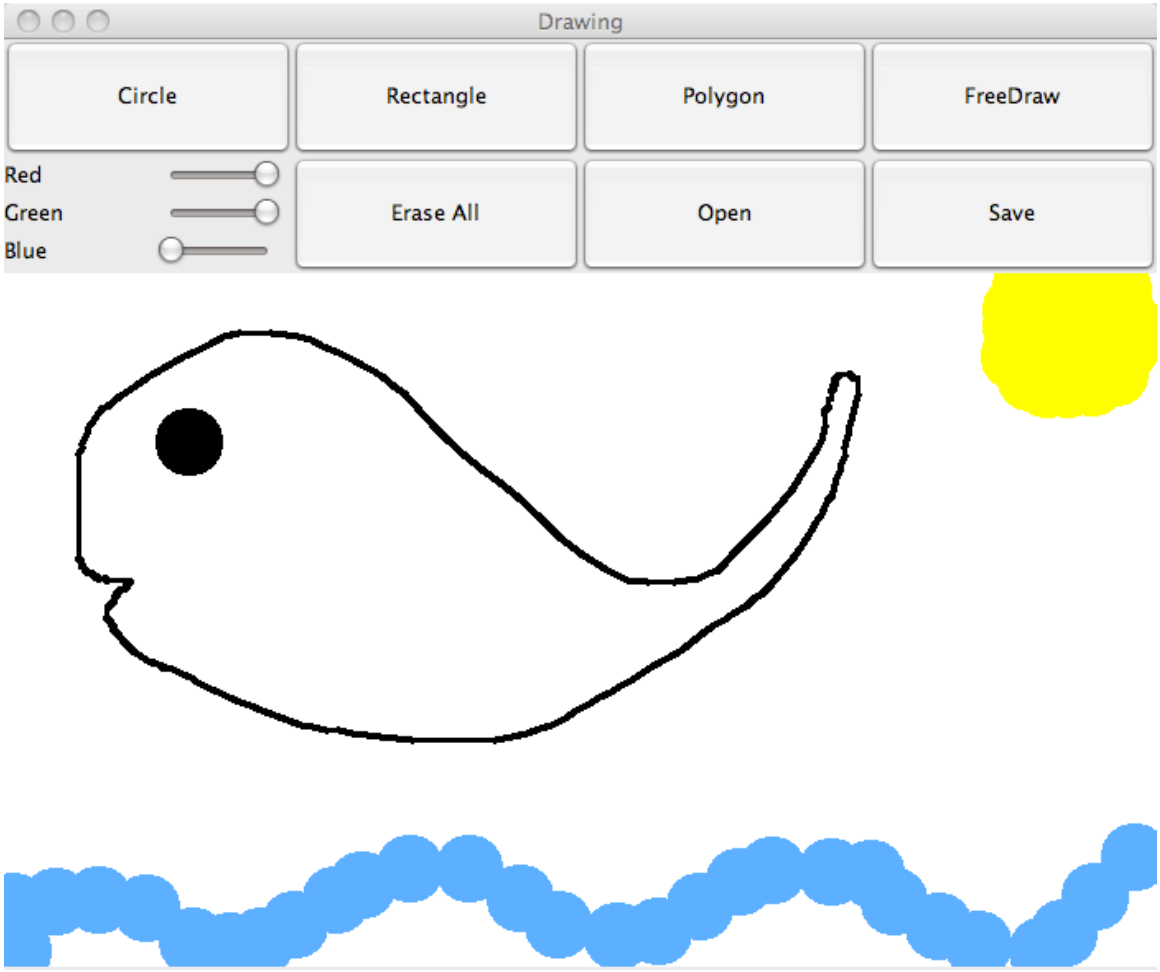


Figure 1: A snap shot of the drawing application.

Functionality: For the first submission, your program should allow the user to do the following:

- Choose the current color by selecting the amount of Red, Green and Blue using the sliders. After the R,G,B components of the color are selected, the color is used to draw all the "new" shapes on the canvas. The color of the previously drawn shapes should not change.
- Click on the Circle button and then click on the canvas. The program should draw a circle with the default radius (and the currently selected color) centered at the click point. You can replace the circle with an oval if you prefer.
- Click on the Rectangle (or whatever other shape you decide to add) and then click on the canvas. The program should draw a Rectangle with the default height and width.

- Click Polygon button and create a polyline out of line segments, where every two points clicked define one line segment. See RocketPanel code to understand what a polyline is. To create a polygon out of a polyline, simply connect the first and the last point.
- Click FreeDraw button and then draw on the canvas by dragging the mouse.
- Click EraseAll button to clear the canvas (erase all the shapes).

Implementation

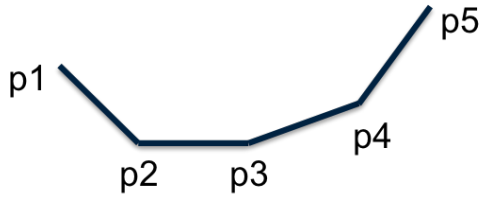
Look at the GUI demo code posted on the course webpage: Rocket.java, RocketPanel.java, RubberLines.java, RubberLinesPanel.java, Dots.java and DotsPanel.java. You are free to use this code as a starting point.

You are expected to use inheritance and polymorphism in this project: your program should have a class Shape and its subclasses such as Rectangle, Circle and Polygon. What do all these shapes have in common: what variables and methods? Each shape should know how to "move", to "draw", "rotate" etc. You should store all the shapes drawn so far in some data structure such as an ArrayList.

User Interface: You will probably want to use two panels: one for the canvas and one that will contain the buttons and the slides. In the `paintComponent` method of the canvas panel, you will need to loop over the shapes and draw each shape. When you call shape.draw(), the behavior will be polymorphic, since each shape has its own draw method.
Clearing the canvas (when EraseAll button is pressed) is just a matter of pointing your shapes array to the empty array, and calling the repaint() method.

Polygon Mode: Take a look at the RocketPanel example to see how to create a polygon in Java. From the UI point of view, the user would create a polygon by first pressing the "Polygon" button, and then clicking the mouse at several points on the canvas. You should create the line segments connecting the points, and connect the last point to the first one to form a closed polygon.

FreeDraw Mode: We can represent a free-form stroke (such as a body of the whale above) with a series of short linear line segments all chained together.



Your class Stroke can store an ArrayList of line segments. The program should allow the user to draw multiple strokes. You need to decide what data structure to use to store all the strokes.

Whenever the button is pressed, your program should create an empty new stroke and store a reference to it since you need to access this stroke when the mouse is dragged. When the mouse is dragged, the program should create a line segment starting at the previous mouse point and ending at the current mouse point. This line segment should then be added to the current Stroke. When the user releases the mouse, he or she is done drawing the current stroke. When the user presses the mouse button again, the program will create a new stroke.

Submission:

Your code needs to be submitted to svn, to a subdirectory called project3 inside a cs112 directory. Go to https://www.cs.usfca.edu/svn/<username>/cs112/project3/src and double check that your java files show up in the correct directory. If your code is not in svn by the deadline, you will get a 0 on the project.

CS 112-01
Project 3: DrawPad: Graphics Editing Application.

Part 2

Part 2: Due Saturday, April 26th, 11:59pm
10 pts for the required part + up to 10 pts of extra credit

In this part of the project you will add some extra functionality to your DrawPad application. The following operations are required for this part of the project:

- Saving all the shapes drawn on canvas into a text file (see below for the suggested format). 5 pts
- Loading the shapes from the text file onto the canvas. 5 pts

The operations below are also fun to implement and will allow you to earn extra credit:

- Moving any shape around the canvas by pressing and dragging the mouse while holding the Shift key. 5 pts
- Resizing /scaling the selected shape by pressing up and down buttons. 5 pts

Saving shapes

The user should be able to save everything that was drawn on the canvas by clicking the "Save" button. You can either go the easy route and just save everything into the file "shapes.txt" in the project directory, or allow the user to select a file (look at JFileChooser)) after he or she clicks the "save" button. The format of the file should be the following: first, the number of shapes should be printed, then the name of the shape ("Circle" or "Rectangle" or "Polygon" or "FreeDraw"), then the color: red, green, blue values on one line separated by white spaces.
What we save for each shape will vary depending on the type of the shape:

- For the Circle
    - Write the x and y coordinates of the center (on one line, separated by white space)

    - Write the radius
- For the Rectangle
    - Write the x and y coordinates of the center
    - Write the width
    - Write the height
- For the Polygon and FreeDraw
    - Write the number of points in the polygon
    - Then write all the points of the polygon (each on one line): the x coordinate and y coordinate of each point separated by white space.

Below is the example of the shapes.txt file (the corresponding shapes that were drawn on the canvas are shown in Figure 1). Everything that is shown in orange is NOT part of the file, these are just my explanations for each line in the file.

```
4                    <-- Number of shapes
Circle
255 0 0              <-- Color
141 70               <-- x and y coordinates of the center
10.0                 <-- radius
Rectangle
0 255 0
233 117
40.0                 <-- width
20.0                 <-- height
Polygon
0 0 255
5                    <-- Number of points in the polygon
339 27               <-- First point
342 198              <-- Second point
537 199              <-- . . .
585 97
490 15               <-- Fifth point
Circle
0 234 255
181 177
10.0
```

Using PrintWriter is probably the easiest for option for writing to the

file. Do not forget to call writer.flush(); in the end after you are done writing.

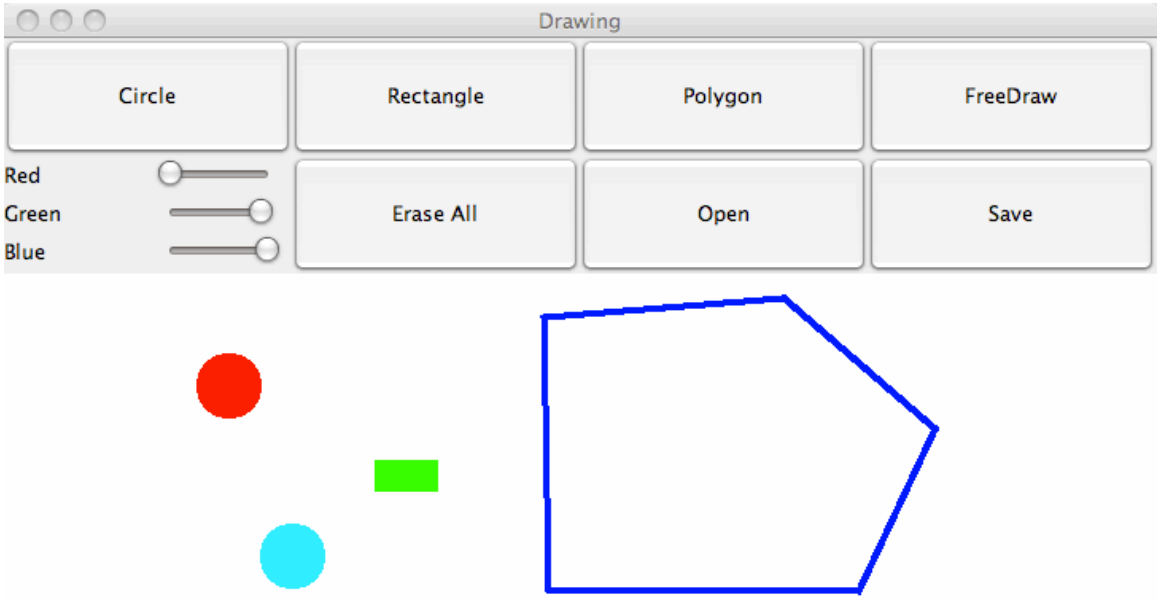


Figure 1: Shows the shapes that were saved to the "shapes" file shown above.

Loading the shapes from the text file

When the "Open" button is clicked, the program should open the file with the shapes and load all the shapes to the canvas panel ("Load" is probably a better name for this button).

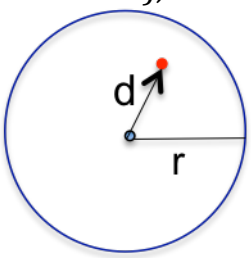
Again, you can read from "shapes.txt" or you can bring up the menu and let the user choose the file. I recommend getting it to work with "shapes.txt" first.

The format of the text file is described above, you can read shapes one by one, adding the appropriate object to the array of shapes. For example, when you read all the information for a given circle, create a new instance of the Circle class with the color, center and the radius that you read from the file, add it to the array of shapes and call repaint. The circle should show on the canvas.

Selecting and moving the shape

The user should be able to move the shape on canvas by clicking "on the shape" and dragging the mouse while holding the shift key. To check whether the shift key is pressed, you can call `event.isShiftDown()` inside the mousePressed and mouseDragged method.

- **Selecting a shape:** the mouse cursor should be "inside" the shape to select it. How can we check if the mouse cursor is inside the shape? Let us take an example of a circle. Consider the figure below: the clicked point is shown in red, the center of the circle – in blue. Let d be the length of the vector connecting the center of the circle with the clicked point. If d <= r (where r is the radius of the circle), then the clicked point is inside the circle.



Deciding whether the point is inside the polygon is a bit trickier. What you can do for this assignment is assume that the point is inside the polygon if it's inside the circle whose center is at the center of the polygon, and whose radius is the maximum distance from the circle to all the points of the polygon. This is not a great solution, but it will be simple to implement.

- **Moving the shape**

Once you click inside the shape and drag it while pressing the Shift key, it should move on canvas. By how much do you move the shape? When you drag the mouse, save the previous and the current points. Change the coordinates of the shape by adding the vector that connects the previous and the current point.

Submission:

Your code needs to be submitted to svn, to a subdirectory called project3_Part2 inside a cs112 directory. Go to https://www.cs.usfca.edu/svn/<username>/cs112/project3_Part2/src and double check that your java files show up in the correct directory. If your code is not in svn by the deadline, you will not be able to get any credit for it.