

CS1322 Homework 7

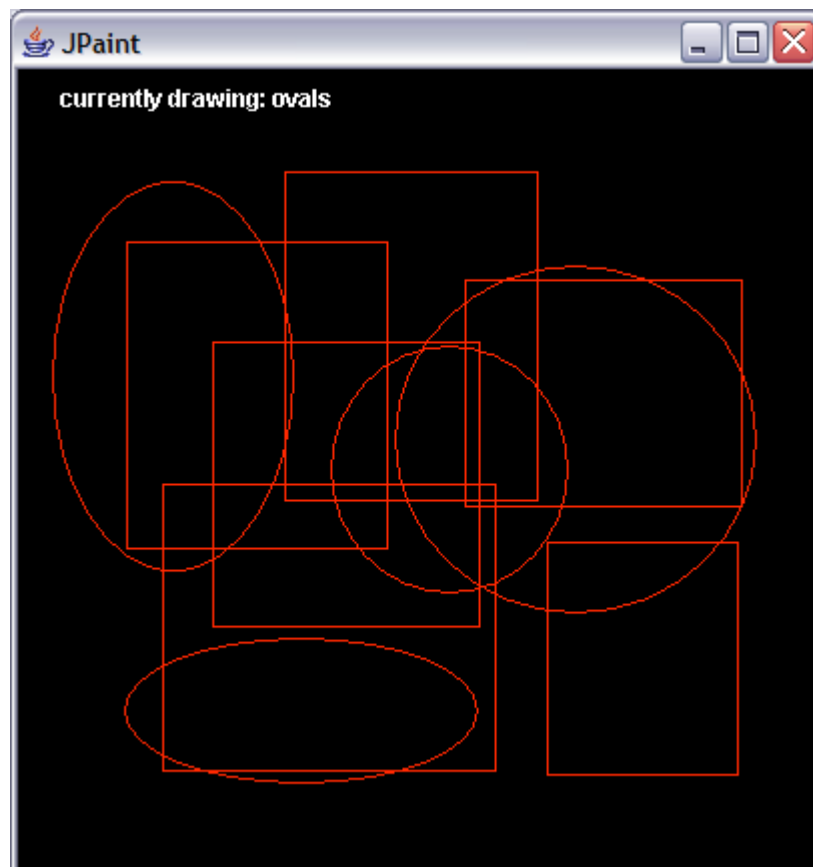
Due Tuesday, November 1st, 2005 @ 6:00 PM

JPaint

Up until now, you haven't had to worry about events when you've created your GUIs. Well, that's going to change. Your assignment is to create a simple Paint-ish program called JPaint. JPaint has two amazing features: drawing rectangles and drawing ovals! The way the user would draw those is the same you would draw them in paint:

- 1) First the user clicks and holds the mouse. That point becomes the shape's top-left corner.
- 2) While holding the left button, they drag the mouse to get the desired size of the shape. Your GUI will keep re-drawing the shape as they change its size.
- 3) Once the user lets go of the mouse button, the shape will be placed permanently on the screen.

The next figure shows a screenshots of a possible solution.



If this is hard to understand, open up Microsoft Paint and see how it draws ovals and rectangles. Also, in the book there is an example of doing a similar operation drawing lines (see RubberLines example).

GUI Requirements

We're going to allow you to be creative in your GUI design, but here are the basic requirements:

- 1) A customized panel, called `JPaintPanel`, which will be your drawing canvas.
- 2) The ability to switch between rectangle and oval mode using the keyboard. When the user hits the 'r' key, `JPaint` should switch to rectangle mode, and when the user hits the 'o' key, it should switch to oval mode.
- 3) An indication to the user of which mode is currently active (ovals or rectangles).
- 4) The ability to clear all of the previously drawn shapes off of the canvas. This will happen when the user hits the 'c' key.
- 5) A main method which will add `JPaintPanel` to a `JFrame` and display it (like you've done in all your other GUIs).

As you know, in order to create shapes and handle the keyboard input, you will need to define a few listeners. You will need to listen for key events and listen for the mouse events required to draw the shapes. Your listeners for the mouse events will behave differently depending on the current drawing mode (rectangles or ovals), which means that the drawing mode will have to be kept in some state variable.

Suggestions:

- See the `RubberLines` (mentioned above) and the `Direction` examples in the book for an illustration of how to implement mouse and keyboard listeners.
- To suitably implement the `paintComponent` method of `JPaintPanel`, you will need to keep track of all the shapes that have been drawn. To effectively store your ovals and rectangles, think about what information is needed to represent (1) where a shape goes and (2) what its dimensions are.
- Note that, when you call `super.paintComponent()` in your `paintComponent` method, this will have the effect of clearing the graphic contexts (i.e., your canvas).

Reminders & Deliverables

- Submit all of the files that are required to compile your program. Remember to download and test your homework solution a few minutes after you submit it.
- As usual, your source code should have proper Javadoc comments for classes, instance variables, and important methods.