CSC171 — Lab 5

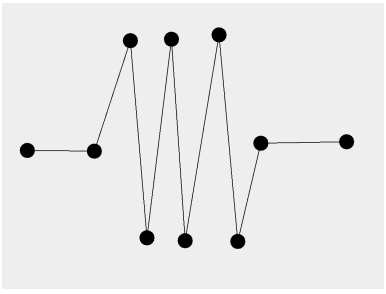
Events and Graphics

The goal of this assignment is to give you experience with handling events in graphical applications.

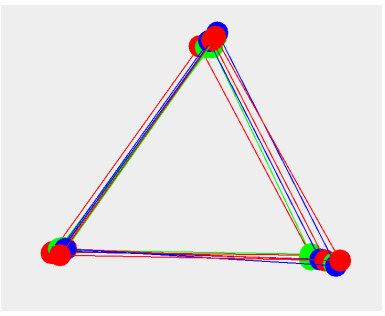
Regarding event handling, remember it's a question of how to get a signal from hardware (like a mouse, keyboard, ...) into your program. A message from the operating system to the JVM, which your program want to know about. In Java, that means implementing listener interfaces and attaching listeners to the objects that might generate events. That means this lab also covers implementing multiple interfaces. Ultimately, you will build an interactive graphical program for creating connected, colorful diagrams as in the figure below. There are some hopefully helpful links on the last page of this document.

For the following questions, you need to define a "canvas" class that extends `JPanel`. You also need a main method that tests your canvas by creating a `JFrame`, adding the canvas to it, and preparing the frame (set size, set window closing behavior, make it visible). You should put all your work into a file named `CircleEvents.java`, with a corresponding public class and the required main method. You should write your event handlers and subclasses as non-public classes in the same file. You may wish to use nested classes, but that is not required. What is required is that you submit a single source file with a public class named `CircleEvents` with a main method that demonstrates your working program. You must be sure to include the usual comments at the top of the file indicating your name, lab section, and the assignment. This is an individual assignment.

Submit your work to blackboard before the deadline of 1159pm Thursday November 10th.



(a) Connected



(b) Colorful.

Figure 1: Examples of connected and colorful lab results.

Questions

1. **Printing click coordinates.** (15pts) Build a custom subclass of `JPanel` which listens for `MouseEvents`, after observing a click, it should just print the coordinates, as in "Click detected at (X,Y)." (but with actual coordinates for X and Y). This means your class should also implement the `MouseListener` interface, and register itself as a listener in the constructor. Write a main method which creates a new `JFrame` and adds an instance of your class to the frame. Name your custom class `EventLabCanvas` (or anything else you prefer.

2. **Drawing circles centered on clicks.** (20pts) Add code to make your canvas draw a small filled circle on itself when it receives a `mouseClicked` event. The circle should be centered on the location of the click. You must determine how to communicate the click location to the `paintComponent` method. Hint: recall the discussion of the model-view-controller paradigm.

3. **Connecting circles with lines.** (20pts) Modify your canvas so that it draws lines between the centers of the circles as you create them. See Figure 1 for an example. Hint: use instance variables to store most recent click coordinates.

4. **Make circles draggable** (20pts) Have your program draw small filled circles as the mouse is dragged, and make the connecting line move too. One approach is to use `Graphics.setColor` and set color to `Color.WHITE` to erase the previous graphic when dragging; however, this erases more than just the previous line. You should maintain state in your canvas class to facillitate a smooth dragging operation. You will need to implement the `MouseMotionListener` interface, and to call `addMouseMotionListener` as appropriate. Similarly, once a circle has been placed, you should be able to click and drag the circle to move it.

5. **Make things colorful.** (20pts) Add an implementation of the `KeyListener` interface to your class which sets the color to `Color.RED` when the user types 'r', `Color.BLUE`, when the user types 'b', `Color.GREEN`, when the user types 'g', and back to `Color.BLACK` when the user types 'l'. New circles should have the corresponding color stored as a part of their state. Connecting lines should always be black. Note: When handling keyboard events, you have to call `setFocusable(true)` on the relevant object, so add that line to your constructor.

Useful Links

- Listeners – these are all in the `java.awt.event` package.
  - MouseListener - tutorial - docs
  - MouseMotionListener - tutorial - docs
  - KeyListener - tutorial - docs
- Events – are are also all in the `java.awt.event` package.
  - MouseEvent - docs
  - KeyEvent - docs
- `java.awt.Graphics` - tutorial - docs
- `java.awt.Color` - docs