Lab: Graphical User Interfaces

CS 162

Background

As the programs you write become more complex, the programs will depend more and more on external classes that you didn't write. These external classes are called APIs (application programming interfaces.) This simply means other code with which your code interfaces with to allow you to create full-fledged applications. You are already familiar with `String`, `Integer`, `Double`, etc. These are part of the Java Core API. Additionally, Java has an extensive graphical user interface API—otherwise known as the GUI API.

In this lab, you will gain more experience using the Java GUI API. Use chapter 14 in *Building Java Programs* (or equivalently sections 9.6-9.9 and 10.9-10.11 and chapter 18 in *Big Java*) as reference. Most of you should already have some experience with these APIs from CS161. Additionally, googling "`java 6 JButton`", for example, will provide you with a link directly to the JavaDoc API reference for the `JButton` class.

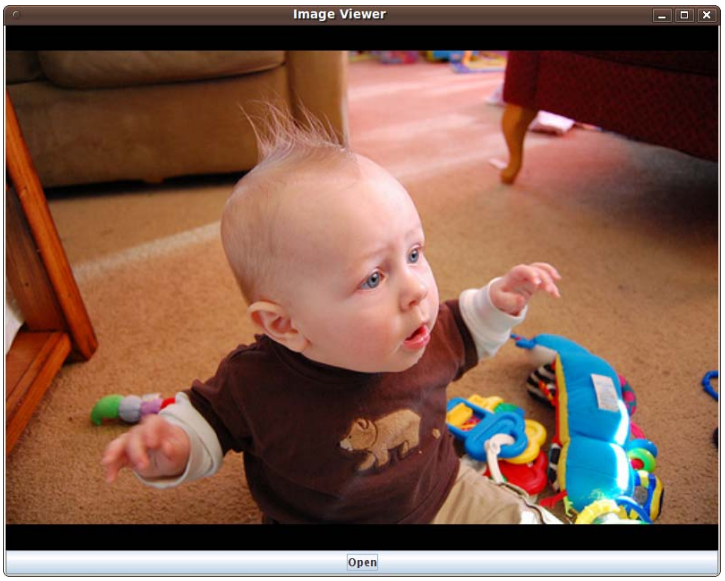
If you are not already familiar with Listeners, look at Page 835 of *Building Java Programs.* Listeners are excellent candidates for inner classes (look at the code I posted online for an example of using an inner class for an `ActionListener` or look at *Big Java* Section 9.7).

Getting Ready

Go to the class website and download the lab resources. You can find a link to the resources from the class schedule. You will need `ImagePanel.java` and `ImageViewer.java` for the first exercise. You may want to use the high resolution screenshots for reference.

You will be working individually for both exercises.

Exercise 1: Image Viewer



You will implement a simple image viewer. See the recitation screenshots for a high resolution example. The image viewer will have an area to display the image and an "Open" button at the bottom. When the "Open" button is clicked, it should launch a `JFileChooser` to allow you to select a picture to display.

You will need the `ImagePanel` class to display the image. In order to change which image is displayed, you should call the `setImage` method. Additionally, you will be using the `BorderLayout`, `JButton`, `JFileChooser`, and `JFrame` classes. Furthermore, you will need to create an `ActionListener` to receive the button press from the `JButton`.

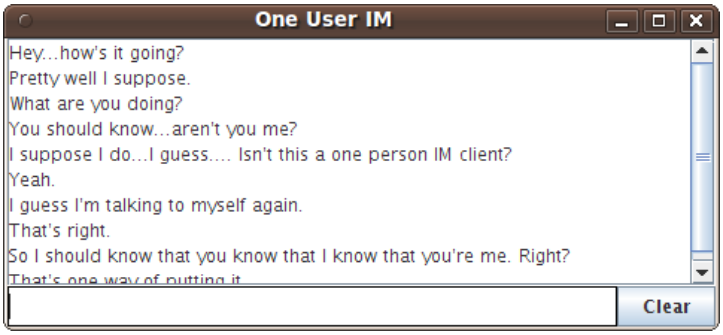
If you find it helpful, you can follow these steps. I would recommend running your program after each step to verify the program is working as you expect.

1. Set an appropriate title and size for the `JFrame`.
2. Create a new `ImagePanel` and add it to the `JFrame` in the “center”.
3. Create a new `JButton`, give it the label “Open”, and add it to the `JFrame` in the “south”.
4. Create a new inner class that implements the `ActionListener` interface. For the moment, have the `actionPerformed` method simply print “pressed” to `System.out`. Create an instance of this class and add it as an `ActionListener` on the `JButton`. Remember, you can consult section 9.7 in *Big Java* for an example. Run the program and verify that “pressed” is printed when you press the button.
5. Now modify the `actionPerformed` method to launch a `JFileChooser`. Google “`java 6 JFileChooser`” for example code. A `JFileChooser` will provide a `File` object for the file the user selects. Use the `File` object to get the path to send to the `setImage` method for your `ImagePanel`. If your `ImagePanel` is declared within a method, move it to be an instance field instead so it can be accessed by both your `ImageViewer` and your `ActionListener`.

When you are done, demo your image viewer to the TA. Also, show the TA the code you write to implement the viewer.

Exercise 2: One Person IM Client

You are going to write a simple, interactive GUI application. It won't do much—it's a one person IM client. Essentially, you will be posting messages to yourself. (You have to walk before you can fly, right?) See the screenshot for an example.



I am not providing skeleton code for this exercise, so use your code from exercise 1 as reference. Some important classes to note for this exercise are the `JTextArea`, `JTextField`, `JScrollPane`, `JPanel`, `JButton` classes.

Here's how the client should function. You will type a message in the text box and press return to post the message. When the message is posted, it should be appended to the “log” which is displayed above the text box. The text box should then be cleared out in preparation for a new message. To the right, there should be a clear button which allows you to clear the log.

General steps:

1. Layout the whole GUI first. Notice how the `JTextArea` is inside the `JScrollPane`. Also notice how the `JTextField` and the `JButton` both share the space at the bottom. You will need to put a `JPanel` at the bottom of the `JFrame` and a `JTextField` and a `JButton` inside that `JPanel`.
2. Implement the `ActionListener`, as an inner class, to handle the input from the `JTextField`. Remember, any variables the `ActionListener` needs to access must be declared as instance fields in the outer class.
3. Implement the `ActionListener` for the `JButton`.

Like exercise 1, when you are finished, demo your one-person IM client to the TA and show the TA your code.

Follow-Up

I highly recommend finishing both of these exercises after the lab if you do not have enough time within the lab to complete them. If you need help outside of the lab, you can visit the mentor office or come during my office hours.