APS101 lab 8 – week 10

1 Objectives

1. Learn about `JButton`s and Event Listeners.
2. Learn how to get `JFrame`s to accept keyboard input.
3. Learn about `GridLayout`.
4. Practice reading a file.

2 The content pane of a `JFrame`

In lab week 8, you wrote code to put a `JTextArea` in the *content pane* of a `JFrame`. A `javax.swing.JTextArea` is a typing area, which is a Graphical User Interface (GUI) *component*. Most GUI components go in the *content pane* of a `JFrame`. The content pane is a *container*, because it can contain components.

A `JButton` is a component with a label that can be clicked. Go get files `ButtonWindow.java` and `ButtonExample.java` from the Labs page on the course website.

Open these files in DrJava and make sure that you both understand every line. Look up things that you don't understand in the Java APIs.

Method `main(String[])` is special in Java. You can call method main in class `ButtonExample` like this in the Interactions Pane:

```
java ButtonExample
```

2.1 Event listeners: MouseListener

In Java, events like mouse clicks and keystrokes are handled by objects that are *listeners*: they listen for those events. A `java.awt.event.MouseListener` is something that listens for mouse events.

Go to the Java APIs and look up `java.awt.event.MouseListener`. There are five methods listed. You will only deal with the first one for this lab.

2.2 Event listeners: MouseAdapter

A `java.awt.event.MouseAdapter` is similar to the `java.awt.event.MouseListener`, and in fact they both have the same methods. The difference between them is that the `MouseListener` only describes what the methods should do, and the `MouseAdapter` can actually respond to them. The `MouseAdapter`'s implementation isn't particularly useful: its reaction to mouse events is to do absolutely nothing, which you will change when you customize it.

Get file `Alphabet.java` from the Labs page, and open it in DrJava. This class extends `MouseAdapter`, which means that it can listen for mouse events. After you compile this class, write the following line in the Interactions Pane, click on the JFrame, and type some stuff on the keyboard:

```
Alphabet a = new Alphabet();
```

Nothing happens, right? Uncomment the line near the end of the `Alphabet` constructor, and try it again. That commented line tells the `MouseAdapter` which window to listen to, which is why the `JButton` doesn't change if you click on a different window and start clicking.

Still nothing, eh? Look at the `mouseClicked` method, which specifies what to do when a mouse is clicked. Uncomment the code in that method and try again.

3 GridLayout

Switch roles: the navigator becomes the driver and the driver becomes the navigator.

By default, the content pane uses a `BorderLayout` to lay out its contents in the North, South, East, West, and Center. You can use other layouts. Add this line to `Alphbet.java` right after you get the content pane:

```
  c.setLayout(new GridLayout(2, 3));
```

Now add more `JButton`s to the layout and "pack" it so that it looks nice:

```
  c.add(new JButton("A"));
  c.add(new JButton("B"));
  c.add(new JButton("C"));
  c.add(new JButton("D"));
  c.add(new JButton("E"));
  capsFrame.pack();
```

Compile and make a new `Alphabet` object. Notice that there are now 6 items arranged in a grid. In what order are they arranged?

4 JFileChooser

Switch roles: the navigator becomes the driver and the driver becomes the navigator.

Create a class called `FileStuff`. Have it import `javax.swing.*`. Write a `static` method called `readFile` that has this as the body:

```
    JFileChooser chooser = new JFileChooser("");
    int returnVal = chooser.showOpenDialog(null);
    if(returnVal == JFileChooser.APPROVE_OPTION) {
      System.out.println(chooser.getSelectedFile());
    } else {
      System.out.println("Cancel was selected.");
    }
```

That code will prompt you to click on a file and then print the name of that file. Call your method at least twice from the Interactions Pane. The first time, select any file you like and click button `Open`. The second time, click button `Cancel`.

5 Reading a file

Switch roles: the navigator becomes the driver and the driver becomes the navigator.

`chooser.getSelectedFile()` returns a `File` object. You can read from the file by making a new `BufferedReader` wrapped around a new `FileReader` with the file as the argument.

Modify method `readFile` to print the contents of the file that was selected. You will need to import `java.io.*`, and deal with exceptions, of course.

When you are done, try it out in the Interactions Pane on `FileStuff.java`.