

CSC108H lab – week 10

1 Objectives

1. Learn more about `JButtons`.
2. Learn more about `ActionListener`.
3. Learn `GridLayout`.
4. Learn `JFileChooser`.
5. Practice reading a file.

2 The content pane of a JFrame

Back in lab week 7 you wrote code to change the text of a `JButton` when it was clicked, and learned that components like `JButton` and `JTextArea` go in the *content pane* of `JFrames`.

Go get files `JButtonWindow.java` and `JButtonExample.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 `JButtonExample` like this in the Interactions Pane:

```
java JButtonExample
```

2.1 Event listeners: `ActionListener`

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

Go to the Java APIs and look up `java.awt.event.ActionListener`. There is one method called `actionPerformed`. `ActionListener` only describes what the method should do, so to use that method you need to implement it.

Get file `Alphabet.java` from the Labs page, and open it in DrJava. This class implements `ActionListener`, which means that it can listen for button events. After you compile this class, write the following line in the Interactions Pane, click on the button in the window (labelled with “A”):

```
Alphabet ac = new Alphabet();
```

Nothing happens, right? Uncomment the line in the `Alphabet` constructor, and try it again. That commented line tells the `ActionListener` to listen to the button.

Still nothing, eh? Look at the `actionPerformed` method, which specifies what to do when an event occurs. Uncomment the code in that method, and try it again.

Now it's working!

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 `Alphabet.java` right after you get the content pane:

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

After you add the button, add a few more `JButtons` to the layout and “pack” it so that it looks nice:

```
c.add(new JButton("B"));
c.add(new JButton("C"));
c.add(new JButton("D"));
c.add(new JButton("E"));
c.add(new JButton("F"));
this.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?

(For A3, you will declare a 2D array of `JButtons` the same size as the `GridLayout` to show the contents of the board.)

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.*`, deal with exceptions, of course.

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