CSC108H lab – week 7

This document contains the instructions for the week 7 CSC108H lab. To earn your lab mark, you must actively participate in the lab. *You don't need to finish in the time alloted, you just need to try hard.*

In this lab, you will pick a new partner, learn about simple events in Java, and in particular you'll learn how to process button clicks.

1 Starting up

Sit down with your partner. The rest of these instructions call you two `s1` and `s2`. Pick which one is which.

`s1` should log in and start up DrJava, and be the first driver.

2 The task

To remind you: a `javax.swing.JButton` is a clickable button, and a `javax.swing.JTextArea` is a typing area. Both those things are *components*. Most GUI components go in the *content pane* of a `JFrame`. The content pane is a *container*, because it can contain components. It has five areas: north, south, east, west, and center.

An *event* in Java is something the user does to interact with your program: click a button, type a letter, move the mouse, and so on. When an event happens, Java notifies any object that is interested in the event. For example, when a `JButton` is clicked, every object that has registered with the `JButton` has their `actionPerformed` method called.

Download `ButtonJFrame.java` from here:

`http://www.cs.toronto.edu/~heap/108/labs/ButtonJFrame.java`

Open it in DrJava, compile it, and in the Interactions pane make a new `ButtonJFrame` and `show` it. There is only one button, called "Click me". Click it and watch what happens.

```
new ButtonJFrame().show();
```

Here are the important new concepts in `ButtonJFrame`:

- `class ButtonJFrame ... implements ActionListener`

  This tells Java that `ButtonJFrame` can listen to some events, and will have a method called `actionPerformed`.

- `b1.addActionListener(this);`

  This tells Java that `this` object (the `ButtonJFrame` window) is listening for clicks on `b1`.

- Method `actionPerformed(ActionEvent e)` (go read it now!)

  This method gets called by Java when `b1` is clicked. `e.getSource()` returns the memory address of the button that was clicked on –here, the button that `b1` refers to.

- `b.setText(s)`

  This sets the text on the button to `String s`.

Do the following:

- Change the initial button text from "`Click me`" to "`Click count: 0`". You may need to call `this.pack();` at the end of `actionPerformed` in order to make the button look right.
- Add an instance variable that keeps track of the number of clicks.
- Change `actionPerformed` to update the instance variable.
- Change `actionPerformed` so that it no longer asks the user for the new text, but instead changes the text of the button to "`Click count: i`", where `i` is the number of clicks so far. Be sure to rewrite the comment!

Compile it, test it, and fix any errors.

Demonstrate to your TA that it works.

3 Another task: `JTextArea`

Switch roles: `s2` drives and `s1` navigates.

Do the following:

- Change the button text back to "`Click me`".
- Add a `JTextArea` to the center and move the button to the north.
- Change the code so that on each click it appends "`Click count: i`" to the `JTextArea`, where `i` is the number of clicks so far. After three clicks, for example, it should contain

  ```
  Click count: 0
  Click count: 1
  Click count: 2
  Click count: 3
  ```

  Be sure to rewrite the comment!
- **Do not use any loops.**

Compile it, test it, and fix any errors.

Demonstrate to your TA that it works.