APS101 lab1 – week 2

This document contains the instructions for the week 2 APS101 lab (Lab1). To earn your lab marks, you must actively participate in the lab. *You do not need to finish in the time alloted, you just need to try hard.*

1 Objectives

1. Meet your fellow lab inmates.
2. Make sure you have email forwarding set up properly.
3. Practice reading and using common computer science terms: *variable*, *method*, *instantiate*, *execute*, *declaration*, *initializing declaration*, *syntax*, *semantics* and *expression*.
4. Understand the difference between a `void` method and a method that returns a value.
5. Practice using `JFrame`s.

2 Icebreaker

Here you will introduce yourself to the other students in your lab. Your TA will tell you what to do.

3 Using the lab PC computers

Sit down with your partner. The rest of these instructions call you two `s1` and `s2`. Pick which one is which. Execute the following set of instructions twice, once for `s1` and once for `s2`.

1. Log in: enter your user ID and password. Wait while the computer starts up.
2. We will sometimes send important email to everyone's APS101 account. Unless this is your primary email account, you should set up "email forwarding". After this, any mail sent to your APS101 account then will be forwarded to this email address (test this to make sure you have done this properly). Ask your TA how to do this if you do not know. Also, the following link has useful information about ECF (open the web browser and type the following link, scroll down and see topics like using emails, forwarding, webspace, printers, etc):

   `http://www.ecf.toronto.edu/ecf/`

3. Log out. Your partner should now complete the above email forwarding steps.

Throughout the term, we will use the terms *driver* and *navigator*. Here are the definitions of the two roles:

driver: The person typing at the keyboard.

navigator: The person watching for mistakes, and thinking ahead.

OK, let's start. Now the "driver" logins. Open a "terminal" (ask your TA if you don't know how to do it!). You can type commands here. To run Dr Java, simply type `drjava` and press enter. Wait while DrJava starts up; please be patient, and don't close the little window that pops up. Now exit Dr Java! There is an alternative way to run Dr Java from menus (GUI): find DrJava in menus and click on it. Again, wait for it to start up.

4 Using DrJava, playing with `JFrames`

Here is the most important rule for this lab:

> **The navigator must not touch the keyboard or mouse.** If the navigator does type or click when they are not supposed to, the navigator will get a zero for this lab.

4.1 Trying `JFrames`

This section is hard! Don't worry if you don't finish or don't understand everything. Remember that we will be repeating this information in lecture this week and next.

In particular, don't worry if there are several other students in you lab who finish this quickly. Grab them and make them explain it to you!

Also, if you are one of those fast students, please help the others around you.

`s1` should now log in again. Throughout the lab, you'll be switching back and forth between the driver and navigator roles.

In lecture you learned about `int`s and `double`s, variables, and how to create and manipulate `JFrame`s. In this lab you and your partner will experiment with those concepts.

- These statements create a `JFrame`, make it visible, set the size, and set the title:
  - `import javax.swing.*;`
  - `JFrame j1 = new JFrame();`
  - `j1.setVisible(true);`
  - `j1.setSize(400, 600);`
  - `j1.setTitle("Look!  A window!");`
- Start DrJava and type the above statements into the Interactions pane. See what (if anything) happens as you type them and hit the "Enter" key.
- Use three more initializing declarations to instantiate a total of four separate `JFrame`s (call the new ones `j2`, `j3`, and `j4`), and practice calling the methods you saw in class on those objects. Here are (some of) those methods:

`setVisible, setSize, setTitle`	`getWidth, getHeight, getX, getY`

- Discuss the syntactic and semantic differences between the methods in the left group and the methods in the right group. In this context *syntactic* has to do with form: how the method appears in a Java line, while *semantic* has to do with meaning: what does this method do to/with the JFrame you call it on. Make a **brief** list of the differences you can think of and show it to your TA.

4.2 Resizing `JFrames`

- Switch roles: `s2` drives and `s1` navigates.
- Reset the Interactions Pane.
- Use four initializing declarations to create four `JFrame` variables and corresponding objects, and call method `setVisible` on all of them. (Don't forget to call it with the `true` parameter - then again, you might want to experiment: call it with `false` and see what happens. What happens if you don't use a parameter? Is this a syntactic or a semantic fact about `setVisible`?)

- Set the size of windows 1 and 3 to 100x200 pixels, and the size of windows 2 and 4 to 200x100 pixels. (Which `JFrame` method should you call?)

  Does the horizontal or vertical coordinate come first in a call to `setSize`? Check your answer by calling `getWidth` and `getHeight` on one of the `JFrame`s.

4.3 Positioning `JFrames`.

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

Each `JFrame` has a method `setLocation(x, y)` that moves the upper left corner of the `JFrame` to the pixel with coordinates `(x,y)`.

- Using `setLocation`, move the first window flush against the left side of the screen, centered.
- Using `setLocation`, make the second window flush against the top of the screen, centered.

4.4 Spelling with `JFrames`

- Switch roles: `s2` drives and `s1` navigates.
- Reset the Interactions Pane.
- Write down `s1`'s initials.
- Use `JFrame`s to spell the initials on the screen by using a bunch of skinny and wide windows, placing them beside, above or below each other, or even corner-to-corner.

Show your `JFrame`s and your Interactions Pane to your TA.