

CSCA08H lab – week 4

This document contains the instructions for the week 4 CSCA08H lab. The following two rules apply for the rest of the term, and will not be repeated in later lab handouts:

- To earn your lab marks, you must actively participate in the lab. *You don't need to finish in the time allotted, you just need to try hard.*
- The navigator must not touch the keyboard or mouse. For the rest of the term, if the navigator does type when they are not supposed to, the navigator will get a zero for the lab.

1 Objectives

1. Practice submitting assignment files (if you did not get a chance last week)
2. Practice writing constructors.
3. Practice with JUnit.
4. Practice using `Strings`.

2 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.

3 Account and AccountTester

Download classes `Account` and `AccountTester` (right-click on the links and choose "Save target as...") from the Labs page on the course website:

<http://www.cs.toronto.edu/~campbell/108/05w/labs.shtml>

and save them to your **G: drive**.

3.1 Practice submit

Log in to **fissure**. Change directories to your **pc** directory. Submit `Account.java` for assignment A0 by typing:

```
submit -N A0 csca08s Account.java
```

You won't be graded on the content of `Account.java`; this is just to practice submitting. Pay attention to case! Instructions on how to submit are also posted on the course website under campus-specific information.

Log off **fissure**.

3.2 JUnit: the first two constructors in Account.java

Open up `Account` and `AccountTester` in DrJava and read them and discuss them with your partner. All the `Account` method bodies are "stubs": they have just enough code in them to get the class to compile, but they're meaningless.

Click the `CompileAll` button.

Finish the first two constructors, compiling as you finish each one.

- One takes the owner as a `String`, the initial balance as a `double` and the minimum balance as a `double` (this has three parameters), and
- One takes the owner as a `String`, the initial balance as a `double` and has a minimum balance of zero (this has only two parameters).

3.3 JUnit: toString

Switch: `s2` is the driver, `s1` the navigator.

Once everything compiles click the `Test` button. It will have lots of failures. One of the problems is method `toString`: it currently returns `null`. Fix it so that it returns the expected value.

3.4 JUnit: String parsing

We have started the third constructor. This one takes all the information in a single `String` with the three pieces separated by commas. Finish it. You'll need to use `String` methods `substring` and `indexOf`. You will also need this: `Double.parseDouble(String)`, which converts the argument to a `double`.

Test it.

3.5 JUnit: testGetters

Switch driver and navigator.

`testGetters` fails because the getter methods in `Account` need fixing. Fix them so that `testGetters` passes.

Notice that the `assertEquals` method calls that involve `getBalance` and `getMinBalance` have *three* arguments. The last argument is a tolerance because `double` calculations are sometimes inaccurate. Here, `assertEquals` is checking whether the first two arguments are *exactly* the same.

3.6 JUnit: the rest

Finish working through the testers and writing `Account` code.

Switch driver and navigator at least once so that both `s1` and `s2` write tests and `Account` methods. Put the author's name in a comment above the method headers whose bodies you write in this section. Show your TA.