APS101 lab 4 – week 6

This document contains the instructions for the week 6 APS101 lab.

1 Objectives

1. Review previously learned material — classes, objects, methods, and so on.
2. Practice writing JUnit test cases for your programs.
3. Practice using `String`s.
4. Practice using `if`-statements.

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 Book and BookTester

Download classes `Book.java` and `BookTester.java` from the Labs page on the course website (right-click on the links and choose "Save target as...") .

Open up `Book.java` in DrJava and then read it and discuss it with your partner. All the `Book` method bodies are just "stubs": they have just enough code in them to get the method to compile, but they are meaningless; they don't do anything.

Open up `BookTester.java` in DrJava. Read and discuss: as you can see, it contains a few tests for the methods you are to change in `Book.java`. The idea is that you change the methods (one or two at a time, as directed below), and then write the appropriate test cases for them in `BookTester.java`. You MUST test all the methods in `Book.java` thoroughly to complete this Lab!

Click the `CompileAll` button. If you haven't changed anything significant, all should compile.

3.1 JUnit: the first two constructors

Finish the first two constructors according to the definitions in `Book.java`, compiling your files as you finish each one.

3.2 JUnit: testGetters

The test cases in `BookTester.java` fail because the getter methods in `Book.java` need fixing. Fix them so that the two test cases pass. Then, complete the rest of the "getter" methods, adding new test cases for them as you go.

Notice that the `assertEquals` method call that involves `getPrice` has *three* arguments. As usual, the first is the expected value, the second is the actual value given by `getPrice` when it runs. The last argument is a "tolerance": the difference between the expected value and the actual value must be less than this "tolerance". This is needed sometimes, because calculations involving `double`s can be inaccurate. Here, `assertEquals` is checking whether the first two arguments are *exactly* the same.

3.3 JUnit: testSetters

Once you have completed all the "getter" methods, you can move on to the "setter" methods. Complete all of these (such as `setPrice`, `setBarcode`, and so on), adding new test cases for them as you go.

How do you check whether your "setter" methods are working properly? They don't return any values, so you need to use the "getter" methods you just wrote to test whether the "setter" methods are indeed changing the values of your instance variables.

3.4 JUnit: toString

Once that is done, you should write the `toString` method (scroll down: it's the last method). Right now it returns `null`, but what it should return is a String with the following contents: the title of the book, followed by "," and a space, then the number of pages, followed by "," and a space, then the author of the book, followed by "," and a space, followed by the price of the book.

Don't forget to write test cases for this method!

3.5 JUnit: String parsing

Now, you should move on to the third constructor. This one takes some information in a single `String` with the three pieces separated by commas and a space. Finish the constructor: you will need to use the `String` methods `substring` and `indexOf`. You will also need this: `Double.parseDouble(String)`, which converts the argument to a `double`. If you have questions, ask your TA.

`Compile` it.

`Test` it.

3.6 JUnit: isEqual and compareTo

Finally, you should complete the two remaining methods. These are a little more challenging, so ask your TA for help if you get stuck.

For method `isEqual`, you should write at least TWO test cases (why?).

For method `compareTo`, you should write at least THREE test cases (why?). Also, you will need to use `if`-statements to complete this method.