CS1110 Spring 2015, Assignment A0
The Java assert statement and Eclipse Due on the CMS on Tuesday, 29 January

Introduction

This assignment is designed to help you learn more about Eclipse and Java, and it is in your best interest to do it very soon. We encourage you to do it as soon as Eclipse is working on your computer. The assignment serves two purposes:

1. Introduce you to the Java assert statement, which you must use in assignment A1.
2. Help you get started with Eclipse —create a new project, run method main, insert a "VM argument" (VM stands for *Virtual Machine*") into the run configuration, and look at error messages.

Getting help

If you don't know where to start, if you don't understand, if you are lost, etc., please SEE SOMEONE IMMEDIATELY —a course instructor, a TA, a consultant. Or, ask a question and look for answers on the Piazza for the course. Do not wait. A little in-person help can do wonders. See the course homepage for contact information.

The Java assert statement

The Java assert statement has the form

```
assert <boolean expression> ;
```

To execute it, evaluate the <boolean expression>; if it is true, do nothing else, but if it is false, "throw" a java.lang.AssertionError. Throwing that error means that your program will stop with an error message that indicates on which line of your program the <boolean expression> of the Java assert statement evaluated to false.

For example, suppose you have the following on lines 17 and 18.

```
17  x= 5;
18  assert x == 6;
```

and execute the program (we see later how to do that). You get output like the following:

```
Exception in thread "main" java.lang.AssertionError
at Bee.main(Bee.java:18)
```

Creating an Eclipse project and a class to go in it

Do the following:

1. In Eclipse, start a new project by using menu item **File —> New —> Java Project**. In the window that opens:
   - give it project name a0
   - check that it is using execution environment JavaSE-1.7 or some other version 7 JRE
   - we tend to create separate folders for sources and class files
   - we don't add the projects to Working sets
   - click button **Finish**
2. Add a new class to the project using menu item **File —> New —> Class**. In the window that opens:
   - give it the name A0

- under "Which method stubs would you like to create?" check only the box "public static void main(…)"
- click button **Finish**
- you will see class A0 appear in the main pane of the Eclipse window.

3. You see a definition of a method called main. This method will be called, resulting in its body (the stuff between { and }) being executed, when a certain menu item is used. We'll do that in a minute. First, copy the following lines and paste them into the body, in place of the comment "// TODO Auto-generated method stub" —you can remove that comment:

```
System.out.println("Executing method main.");
int x= 5;
System.out.println("x is now " + x);
assert  x == 6;
System.out.println("The assert statement was not executed");
```

4. If the lines are not indented well —for example, each } should appear under the **p** of the word **public** above it— select all lines by using control-A (pc) or command-A (mac) and then using control-i (pc) or command-i (mac). The class should now have no errors in it and the program can be executed.

5. Use menu item **Run —> Run**. This will cause method main to be executed, and you should see three lines of output:

   Executing method main.
   x is now 5
   The assert statement was not executed

   This indicates that the assert statement was *not* executed. We next show you how to fix it so that the assert statement is executed.

Making sure assert statements are executed

A nice thing about assert statements is that their execution can be turned on or off. Thus, after testing a program thoroughly using assert statements to help test and debug, when you want to actually use the program to get something done, you can leave assert statements in the program but not have them executed during program execution. Then, if an error is detected later on, or changes have to be made in the program, you can turn on assert-statement execution to again help in testing and debugging.

Here is how to turn assert-statement execution on:

1. In Eclipse, choose menu item **Run —> Run Configurations**
2. In the window that opens, click tab **Arguments**
3. In the field titled **VM arguments**, type: -ea
4. Click button **Apply**, near the bottom of the pane
5. Click button **Close** at the bottom of the window.

Having done that, run the program again using menu item **Run —> Run**. The output should now be:

```
Executing method main.
Exception in thread "main" x is now 5
java.lang.AssertionError
    at A0.main(A0.java:11)
```

or

```
Executing method main.
```

```
x is now 5
Exception in thread "main"
java.lang.AssertionError
    at A0.main(A0.java:11)
```

The last three lines indicate that an "exception" was "thrown" in method main; it was an *AssertionError*, and it happened on line 11. You will learn about exceptions and throwing them later on in the course.

Fixing Eclipse so that new JUnit run configurations always have argument -ea

In assignment A1 and perhaps other projects, you will be creating new "JUnit test classes" to help you test and debug your program. It is good to have assert-statement execution turned on —to always have argument -ea present in JUnit testing run configurations. Do the following to set up Eclipse so that assert-statement execution is always turned on:

1. Choose menu item **Eclipse —> Preferences**
2. In the window that opens, choose item **Java —> JUnit**
3. Near the top of the window, check the box "Add '-ea' to VM arguments when creating a new JUnit launch configuration.

What to Submit

On the CMS for the course, submit source file A0.java by the due date. The due date is shown on the CMS.