A Brief JUnit Introduction
CS 162 - Spring 2014 - Dr. Schwartz

Goals
- to learn about a helpful testing tool; this is not an assignment

Overview
JUnit is a tool that helps you test your program. You can learn more about it at [http://www.junit.org/](http://www.junit.org/). It's incorporated into Eclipse. Essentially, it helps you build automated test cases of input and expected output. It's a handy tool to easily run repeated tests on your code to uncover bugs.

We have JUnit installed on our machines.

Test Cases
In general, for any testing, you should create test cases that contain pairs of input and expected output. For example, with input of 15, 20, and 25, you would expect a mean of 20. You should think about your expected output before you run the program and say "well, that looks close" when you see the answer.

For instance, for the Statistician, I might create a list of three values as above, and write down what I expect the length, mean, sum, largest, and smallest to be. Then I'd write down another value (say -5) and write the answers I'd expect after that insert.

As in real life, bugs lurk in corners. They also are found at boundaries and edges. As you build test cases, check the limits. For example, for Statistician, there should be some negative numbers, zero, and some values with fractional parts.

Round numbers are easier to work with in hand-computing correct expected output. For example, a check of the mean is easy if all the numbers are the same. That shouldn't be the only check, but it could be one of them.

Using JUnit
JUnit lets you set up input, call the methods of your class, and compare the actual output to your expected output. I have included a test package in your statistician files.

Open Eclipse and the Statistician project. Open the test package. Create a new run configuration that is a JUnit configuration (not a Java application). Use the JUnit test runner. **Or just click on StatTest.java and choose Run / Run As ... and JUnit.**
When you run, the package explorer changes to the JUnit panel. Because the program you have does not pass the tests, you see a red bar. There were six runs, 1 error, and 5 failures. Let's examine one.

Double-click on the testConstructor name and it is selected in the code at right. At lower left, the failure trace states that the error is "Mean wrong". The first two calls worked because the sum and length functions are returning 0. But mean, largest, and smallest are not returning Double.NaN. Fix that in the Statistician code and run the test again. Now that test is passed. That doesn't mean that your code is "right". But it doesn't have the bug of returning 0 when there are no values.

There are other tests here in StatTest. **Your completed Statistician class should be able to get through them with no errors or failures.**

**Building a JUnit test.**
From the File menu, choose New and JUnit Test Case (near the bottom). Fill in the fields reasonably.

In the file that's created, import the class you are testing: `import model.Statistician;` Create a test case. The format will be different between the versions of JUnit, and I'm using the older version here. Create a function inside the class that declares and creates a Statistician, and inserts the value 42 into it, and then checks that the length is one. The function name must start with test.

```java
public void testOne( ) {
    Statistician s = new Statistician( );
    s.insert(42);
    assertTrue("Length should be one", s.length( ) == 1);
}
```

Run that test class. It shows red. Modify the Statistician to set count to zero to start, increment count on insert, and return count from the length method. Run the test again. You should see green. Insert some more values, think about the expected output, and create another assertion. Does that work?

This is a small example of using JUnit. I'll give you JUnit test cases for your assignments.