CMSC 256 – JUnit in Eclipse and Recursion Lab

Part 1 – Unit Testing in Eclipse

The first part of this lab involves analyzing the requirements and the design for a method and completing a test plan for it. Unit testing requires an understanding of the what the unit should accomplish and what results should be obtained from a correctly implemented unit.

After completing this exercise, you will be able to:

- Describe the basic concepts and techniques of unit testing.
- Analyze the requirements and design of a software unit in order to create a test plan for the unit.

Suppose you are part of a team developing a software unit that will convert an un-signed 16-bit binary number (in string format) to a Java int value. For example, the method *binaryToDecimal* does the following:

```
binaryToDecimal ("0000000000001111") returns 15
binaryToDecimal ("0000000010001100") returns 140
binaryToDecimal ("0000010000000000") returns 1024
```

In addition, another method exists that will convert an int (in base 10) to a binary number represented as a String.

For this lab, you are to use Eclipse. Create a new Eclipse project called JUnitLab.

Eclipse should have JUnit already included, but if not, download the JUnit testing framework from the following site, https://github.com/junit-team/junit/wiki/Download-and-Install

Add the **junit.jar** and **hamcrest-core.jar** libraries to your build class path by following these steps:

1. Right-click on your project, JUnitLab, and choose **Properties**.
2. Click on **Java Build Path**, then the **Libraries** tab.
3. Click on **Add External JARs...**. and navigate to **junit.jar**.
4. Select **junit.jar**, click on **Open**, click on **OK**. Repeat for **hamcrest-core.jar**.

Suppose that another programmer has implemented these methods in a utility class called **BinString**. Import or copy the code for **BinString** into a new Java class. For the first part of the lab, create a test class for the **BinString** class.

To create a test class:

1. Select the menu options **File > New > JUnit Test Case**
2. Click on the radio button **New JUnit test** (you might need to select Other first).
3. Type **BinStringTest** for the class name
4. For **Class under test:** type **BinString** and click **Next**
5. Select the methods to be tested (**binaryToDecimal**(), and **decimalToBinary**()) and click **Finish**.
6. Add code to the test class to create and run tests of the two methods. Include at least two test method calls for each one.

7. To run your test class, select your test class and choose **Run as > JUnit Test** from the **Run** drop-down menu in the toolbar. You can view the results of the test in the JUnit window. Red bar means the tests failed, green bar indicates that all tests pass.

The goal here is to write tests that will uncover any potential errors. Were you able to discover the error in **BinString**? Show your TA the corrected code and describe how you corrected the bug.

For more information on JUnit testing, you can refer to this Eclipse tutorial: **Help -> Help Contents -> Java Development User Guide -> Getting Started -> Basic tutorial -> Writing and running JUnit tests**

Part 1 – Unit Testing Recursive Methods

Download the `RecursiveMethods.java` and `RecursiveMethodsTest.java` files and import them into your JUnitLab project folder. Execute the test and they should fail.

Implement the **recursive** methods until all pass the provided unit tests. Once again show the green bar and completed `RecursiveMethods.java` file to your TA.