COSC 603

Software Maintenance & Testing

Spring 2016

William Cosulich

**Project #3 – Unit Testing with JUnit**

Task 2 - ***For this task, briefly describe what the error was and how you corrected it.***

In task 2, FibonacciTest.java failed because the “*assertEquals*("0",0, tester.fib(0));” in FibonacciTest.java was expecting a 0 for the result. In Fibonacci.java always returns a 1 for that particular case. The solution was to returning a 0 instead of a 1 when n = 0 in Fibonacci.java.

Task 3 - ***For this task, briefly describe what the error was and how you corrected it. Also, provide the source code for your improved getDiagonal() and getArea() methods.***

In task 3, RectangleTest.java failed because of a bug in the Point.java class, the constructor was initializing all x values of the points the y values that were entered. An exampling of this was with a point with the values x = 2.0 and y = 4.0. The point constructor would result in creating a point with the values x = 4.0 and y = 4.0 instead of using x = 2.0 and y = 4.0. In order to correct this problem, I just had to make the point constructor in Point.java to initialize x values of points with x not y.

/\*

\* This calculates the area of a rectangle and returns it as a double.

\*/

**public** Double getArea() {

**double** width;

**double** height;

**double** area;

width = Math.*abs*(p2.x - p1.x);

height = Math.*abs*(p2.y - p1.y);

area = width \* height;

**return** area;

}

/\*

\* This calculates the diagonal of a rectangle and returns it as a double.

\*/

**public** Double getDiagonal() {

**double** widthSquared;

**double** heightSquared;

widthSquared = Math.*pow*((p2.x - p1.x), 2);

heightSquared = Math.*pow*((p2.y - p1.y), 2);

**double** diagonal;

diagonal = Math.*sqrt*(widthSquared + heightSquared);

**return** diagonal;

}

Task 4 - ***For this task, briefly describe any bugs that you found.***

*When writing the TestAddItem test case I needed use of the itemArray within the instance of theVendingMachine Object but it was a private itemarray so I had to change it be protected so that it could be called outside its class. Therefore I needed to change the line“private VendingMachineItem[] itemArray; “ to “protected VendingMachineItem[] itemArray;”.*

Task 5 **-** ***Upon completion, each student is to submit a short report that includes your write-ups from the previous tasks (clearly labeled) as well as:***

* ***A description (2-3 paragraphs) of what you learned from this project (particularly Task 4)***
* ***A description (2-3 paragraphs) of what you liked and didn’t like about JUnit’s support for unit testing***

What I learned from this project:

This project was to help us learn to use the JUnit Framework for testing. There were 3 projects within this assignment which were Fibonacci, Rectangle, and Vending Machine. The tasks related to the Fibonacci and Rectangle projects were fairly easy to do. The goals were to debug the code and find any errors. After correcting the errors that occurred the test cases turned out successful. This helped with my understanding how to debug code using JUnit and also aided in giving a basis for reviewing the reports that were generated by JUnit.

The Vending machine project task was a little more difficult and required a lot more work on my part to complete. This time we had to write the test cases from scratch and find any bugs within the system. I have to say this was relatively hard for me as I had never written any JUnit test cases before and also was not familiar with exactly had to written one. It was very import to understand exactly how the code worked in order to be able to write any of the test cases. It required me to spend a time searching the internet on how exactly to write the test cases for different assertions. I learned that there are a variety of assertEquals statements as well as many others that I needed to use for the various test cases. This project did teach me a lot about writing test cases for JUnit. I did have a hard time with finding bugs while I wrote the test cases as I was not exactly sure I was writing the test cases exactly right. I only found one bug while creating the test case but I know I should probable have found more. I just had a hard time finding more. I had to use the test cases from the projects for the two tasks to help as a guid as again I had never written any test cases before. This helped me immensely.

What I liked about JUnit’s support for unit testing

There was not a whole lot that I did not like about JUnit. I am not really familiar with unit testing tools as I have never really used any in the past when code various projects so I have nothing to compare it to. I can see the usefulness in writing JUnit test cases for your code. I like that I was able to be able to create a test and run it to test if it was working properly slowly build on the tests for each class. The JUnit suite was nice with being able to run all the tests cases in each test class that was created.

JUnit framework was simple to use and understand. I found the API documentation to be really well written. It was very clear and easy to understand. This helped in writing test cases. It helped me able to quickly understand how to write test cases and what the proper syntax was. I was also able to see that there were deprecated assert statement that no longer should be used. It was a very good experience with writing test cases using JUnit. After this I know I’ll need to learn how to create the test cases more efficiently and effectively. Even though it is easy to understand how to use JUnit it is not as easy to write the test cases that cover all cases.