**Task 2**

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

I installed the project Fibonacci into the Eclipse, and then check the source code, unit test code and the generated JavaDocs. I found the Junit 4.12 was already loaded into “lib” folder. So I should be able to use it directly.

Then I ran the FibonacciTest.java. A red bar came out in Junit window, indicating an error existing. It was shown that ”expected (0), but was (1)”, so I realized that the error was in the Fibonacci.java. I checked the code and found the bug in “public int fibonacci(int n)”. In case of 0, the return should be 0. So I corrected the code and ran the test again. The test passed successfully with green bar.

The updated code has been loaded to the remote Github repository.

**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.***

After I installed the project Rectangle, I realized there is no JAVA lib in this project. So I use Build Path and input same JAVA JARs. This made the program runnable.

When ran the RectangleTest.java, errors took place in testGetArea() and testGetDiagonal, indicating something wrong with the code in Point.java or Rectangle.java. Because expect (10), but was (25) for the area calculation, I carefully check the codes in them and found the constructor of Point has problem. This.x should be x, not the y. So I corrected it and rerun the Junit test. The test passed successfully with green bar.

As an improvement, I think we should clarify the concepts used to calculate. I setup length=P2.x-P1.x , width=P2.y-P1.y, and final result as area or diagonal. In this case, we have some concept variables in the method, easy for understanding and design of Unit test. So I updated some codes in the getArea() and getDiagonal(). I have added a new class with name of RectangleModified.java showing the source codes of improved getArea() and getDiagonal(), in this case, the original project code is not changed.

**Task 4**

***For this task, briefly describe any bugs that you found. You should also upload your Vending Machine***

***project to your GitHub account.***

The array item VendingMachineItem[] should be public, elsewise its value cannot be tested by Unit test. Therefore, I changed the “private” to “protected”. This bug came out because I wanted to test the values in the arrays. This is an internal member of project. I also noticed that the access of non-public members is not suggested by Unit Tests, so if you don’t test this internal array, there should be no bug existing in the project.

The code including added JUnit tests has been loaded to the remote Github repository.

**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:***

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

***testing***

A

I heard the Unit test before, but don’t really know how to develop the Unit test. This project, step by step, let me understand what the Unit test is doing and how to implement the Unit test. Upon the practice in this project, particularly task 4, I got some experience in designing the test, running the test, and solving the problem when test fails.

One point I realized is that individual Unit test just examines single aspect of a function. Other aspects or functions should be placed into other Unit tests. Additionally, once the test for one function is passed, this function can be used in other Unit test behind this test. This means that there is an order of Unit tests and need to be designed in advance.

I had a difficulty in push 3 projects into one Github repository. Fortunately, I figured out the solution. I placed 3 projects into common parent folder, then use this parent folder as a local repository. I found when one project is committed or pushed, all 3 projects are committed or pushed.

B

What I like: Normally, we code according to the specification and need. The use of the JUnit test could verify whether the code will run as we expected and how much it deviates from the intended goal. In addition, the JUnit test will make you sure that the modifications we may do later on in the code will not break your system without your knowledge.

What I don’t like: It takes time and effort. Sometime take more time than coding those you are testing. Therefore, doing Junit test at the same time as coding maybe a good option, because at that time you remember everything and easy to develop a test unit.