# Project 3

*COSC 603 Software Testing*

*Sigrid Berkebile-Stoiser*

*3/23/2016*

## Task 2

*Question: What was the error in the Fibonacci project?*

The first Fibonacci number was defined as 1 instead of 0. I corrected the error by changing the first case in the switch statement of the fibonacci() method to return 0 instead of return 1.

## Task 3

*Question: What was the error in the Rectangle class?*

The Rectangle class was correct, the error originated in the Point class where the x-coordinate of a Point was mistakenly instantiated with the value for the y-coordinate. I corrected this in the constructor of the Point class.

## Task 4

I did not find any bugs in VendingMachine.java with my unit tests. In the VendingMachineItem.java, however, I tracked down some potentially unwanted behavior. The name of a vending machine item can be an empty string or set to null. Although I did not find any documentation requiring that the names not be empty or null, I thought that the tracking of vending machine purchases would be very hard if some items do not have names. Therefore, I added a check in the constructor to throw an exception when the name is not specified.

## Task 5

What I learned from this project: It requires a lot of time to design tests for all known error conditions, even in a small project and small functions such as the vending machine project. For complex applications, testing must require a huge amount of test cases. However, the management of the test cases and classes seems to be fairly easy with Junit test classes and test suites. I also suspect after this project, that successful testing mostly depends on the ability and experience of the tester to design good test cases which needs a lot of thought.

I also begin to understand how unit tests can increase the developers’ confidence in the program correctness significantly, as once the test cases are set up, they can be run in a matter of seconds and test a large amount of code without additional effort by the tester. Also, tests can be run at any time without changing the code. When I learned to program, I used print statements and inserted test variables to test my code which might introduce errors and the test statements cannot be kept in the final program. I also found the unit test process very useful to quickly pinpoint where a bug was located, as opposed to traversing the program with a debugger when the program output is different than expected.

What I liked about Junit: I think that Junit is a great tool to create and preserve and manage test cases. The process itself is very easy to learn and thus reduces the risk that the test cases themselves contain bugs that are hard to track down. The process is very efficient. Once the test cases are written, retesting can be done almost instantly.

In my opinion, the support in Eclipse to generate test classes is very nice too, by providing an intuitive interface and support for easy test case generation. I also liked that it is possible to integrate test classes in test suites which provides a nice way of running several test classes at once or selecting a subset of existing test classes of a project. However, I wonder how unit tests are written for application which require a lot of user input, e.g., in web applications. I imagine those test cases are harder to write.

The only thing I maybe did not like so much about the process was that it was not more automated. I imagine that code for testing exceptions or conditional statements can be generated automatically.