**Summary Reflections**

My mindset that I adopted while working on this project was that I used proper programming syntax and methodology that I would pass al my tests and not need to further test anything. I was hypnotized by the allure of developing an actual project for class that the idea of testing of looking for bugs was in the back of my head. I just wanted to code. However, once I began to start developing more and more classes and then their respective test cases I realized just how important a tool like Junit could be.

Junit made the testing process rather easy. Providing the annotations to identify test methods helped keep me extremely organized this final project where we were not just working with in one package. Something I wish I would have done from the beginning was organizing my Junit tests into test suits and test cases just to further my organization. I am of the mind that If I do not stay always organized it is extremely easy for me to fall of track. Getting the syntax down was easy as well because when using Junit, the syntax basically states what it wants to do. Ex @BeforeClass executes statements before all the test cases, After Class executes statements after all the test cases. It proved to be in invaluable tool for me. When at the beginning of this class the idea of software testing just was not appealing to me. (*JUnit - Overview - Tutorials point*, n.d.)

Staying disciplined to the quality of a software engineer is imperative to be successful. Not only for success but to have a good moral compass as well. In the future I could be developing an idea for some one’s business, innovative technology, dream. To be a careless programmer could lead to destroying a company’s reputation, a family’s finical situation, shattering someone’s dream. No matter who the employer is software engineers have a moral obligation to be sharp and skilled. These are peoples lives and hopes and dreams we will be developing for. A terrible bug, security exploit, wrongly implemented feature can ruin establishments or people. There are tons of stories on the internet about this. Along with discipline comes with looking at yourself in the mirror when you are reviewing your own code. Knowing that you are not perfect and can make mistakes no matter how talented of a programmer anyone if. Mistakes happen. Testing my code just as I would test someone’s code I did not trust is how I managed to not have a bias towards myself. I know that I am human, and I can make mistakes, so to test until the tests pass is the only option.