Project 2

Summary Report:

The assignments that we accomplished in this class show me a great deal in figuring out how to test code based on requirements. As developers we will be tasked which requirements of specific code and told just based of the requirements to make a product that does those things. It is important to make sure that our code meets the client’s requirements. For these projects I kept that in mind as I was developing the code and writing out the tests. Examples would be using the illegal argument exceptions in the code. To someone using the code they would just see that what they have entered is wrong. For us developing the code on the other hand it gives us a clear and testable way to check for the requirements. In the assignments that we had this term all the requirements were simple. We wanted to create objects that would later house inputs from the different people that would later use this application. Then we just are making sure that the lengths of these inputs need to set to a certain length. Input lengths requirements are very important in coding projects if people are able to input mass amounts of data without and restrictions it can lead to large DDoS attacks. After going through and writing all of the code I then wrote the Junit tests after checking on the final project I have a 91 percent total coverage for the project. Have 91 percent total coverage just describes the number of instructions the code can give against the number of instructions that you tested with your Junit test. According to the numbers the overall quality of my Junit tests was really good. While writing the Junit tests I had to think about how I was going to test the requirements. I decided to start by turning the functions into Boolean function that would spit out a true or false depending on if the requirements were exceeded or not. First was the tests for the Objects to make sure that they would only be entered if they were the write length. For those I would use the illegal argument exception. In ContactTest.java line 29 to 30 you see the first use of this exception. I start by using the Assertions.assertThrows this will check to make sure that the IllegalArgumentException throw out. Next in line 30 I would create a new contact then try and break the requirement that was set out for one of objects. This example was a test to make sure that the illegal argument exception was used if the name was longer then the assigned length of 10 characters.

Reflection report:

In this project we used the white box testing methods. White box refers to testing the code in a manner where you know what the actual code is trying to do and know the overall design of the code. While this testing method is useful in its own way it makes sure that the code you are producing does in fact work in the manner that you intended. It also tends to close the mind off to different testing ideas. Instead of testing all kinds of different inputs, you tend to focus more on just making sure that he requirements in the code are meet. White box testing is the overall approach I also used unit testing to make sure that the requirements that were laid out were met. Unit testing is the technic of testing induvial units within the software. A software testing technique that I did not use in this project would be black box testing. Black box testing is when you test the software with no knowledge of the code behind the software this type of testing would have been impossible since I was writing the software myself. There are multiple different testing techniques that I didn’t not use for these specific projects as they just would not have made sense in the context of the assignments. One would be system testing where in you attempt to check the entire system. The way that the code was factored this would have been quite difficult. For the practical uses of system testing is quite simple once you have the software nearing full development you would want to make sure that the entire system works together without lagging or adding to much time to commands. In fact, trying to resale a finished product without running a system test would be irresponsible. Black box testing is not a necessary in the development process, but I would say that it might be one of the best testing methods that one can use. Black box testing is one of the best ways to find bugs in the final versions of the software. I don’t know about all software, but games do this all the time. The do what is called alpha testing where people who have had no part in the development phase test their games. This is great because it allows for bugs that the developers where not really thinking of to be found. Using all these techniques will allow you to catch as many bugs as possible before releasing the final product.

Caution is an important trait for a software tester. You need to follow established test plans, identifying and reporting. As for my use of cation in this particular case I made sure that I was checking all of the code to make sure that there were no potential issue with in the code that could have caused any trouble. For our code it was crucial to make sure that the limits on the object lengths were working. It was important to understand the complexity and interrelationships within the code I was testing. For the code that we were working within worked in more than a couple ways. First it was important to make sure that we followed the clients’ requirements as it that is how they wanted it set up. The other reason that knowing how and making sure that the lengths that we put on worked was that if those objects didn’t not have set length limits it could be an easy attack point for people later. If the objects that we worked with in this project didn’t not have any limits it makes them a prime target for DDoS attacks. With any type of white box testing there is going to be slight amounts of bias in your testing process. I tried to eliminate this by just working on the requirements that were laid out in the assignment. As for this bias though I think that when I am testing my own code, I tend to focus to much on just making sure that it accomplishes the designated outputs. I know in previous classes on previous assignments I was very concerned with the outputs making sure that I was getting the results that I wanted for the requirements. The problem with this is I ending up missing some problems with certain inputs. My code ends up not meeting all of the requirements that it needed too. This is a bias that I think this class really helped me to see. After taking this class I have a much bigger appreciation for making sure that I stay disciplined in my software engineering career. I think that before this course I was unsure how important it was in coding to not cut corners. This course showed me that it is important to be diligent in all facets of coding from the development stage where you are making sure that you are correctly. To the testing stage where you need to make sure that things are working as intended. As well as check for any type of security breaches within the code. To avoid any technical debt as a practitioner of software development I plan on first doing lots of code reviews. Code reviews are a great way to catch all kinds of mistakes within your code. Also like a lot of developers these days I plan on making sure that I learn and get used to using some type of automated testing to catch any small mistakes in my code.

Reference:

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.