Southern New Hampshire University

CS 320: 7-2 Submission

Project Two

Aeriel Sacia

04/17/2021

Throughout the course we have learned the reasoning and ways of testing code, when it comes to the way that I approached the testing of the software requirements it most definitely aligned with the software itself. This is because the testing methods used where within the base code. Starting with the contact class, one of the requirements was the length of the first and last name not be longer then ten characters. Usinging “assertThrows” to the actual test input making anything longer then ten characters be invalid and notifying of the length being too long. Next looking at the task class, a very similar situation is required dealing with the length of the task ID. Knowing this “testTaskIdTooLong” is in the base code to help flag anything longer then the requirement specifies.

With advances in JUnit tests, the resources withing each week of the course wear really went along way, witht the back knowledge that it is really ever growing. Using the knowledge I have gained in this class I learned that the coverage of actual contact test was of a higher percentage. Looking at service tests they had an even higher percentage then the rest. The evidence of a positive in percentage shows that the tests were working and covering all of the functions each one had to offer. This knowledge allows me to implement these types of test often for piece of mind knowing they do the job they are asked to do and do it successfully.Technically sound code is a very important job of this field, without it you are left open to flaws and problems later down the roade within a software. The steps taken to help make sure this software was sound started in contact class using things like “private static” and “array list” to help build a list for strings. I also attempted to use algorithms I have seen before like length and add when it came to taskId or ContactService.Knowing that in all the knowledge I have gained throughout this degree, with the numerous amount of class I have taken coding, admittedly is one of my weaker points. That being said I try my best to take as many mesures to ensure what I code is successful. I spnd time, almost to much on the provided material, reading and rereading until I retain as much as I can.

I take advice from mentors and professors as the course goes on and I take my time with things. Often resulting in me taking longer with projects then most of my peers but getting them done to the best of my abilities. When it comes to JUnit I try my hardest to declare clear variables everytime before I implement them into the code. Doing so making sure the code flagged anything that was invalid and proceeds with correct inputs. One of the techniques that really spoke to me in this course is the black box, white box type testing. This is what I tried to use in each milestone, the reason for this is because it has a rhyme and reason to it. Based off structure and specifications black box techniqes I relate to my own world, in the military there are blackworld missions.

This means no prior knowledge just what we are suppose to do. This is kind of how black box works, testing without knowledge of the internal workings. White box testing is testing with full insight on how the internal workings are built and work. The black box testing gives you functional, nonfucntionasl and regression testing. While white box covers more of that security sideof things just from the internal workings. Another player to point out is structure based testing, this helps us to look over things like if, then statements and breaks down the testing into small sections allowing for deeper exploration. The one I avoided is the experience based testing, to state the obvious I dont have the experience to use this type of testing. This type of testing leaves the tester to develop test cases on the fly or before the test the software.

What I have learned throughout the course is that having the right mindset for software development plays a huge role in the success of a developer. The way I look at each section of code is a list of different ways. First it is from a security mindest. How can this affect other people if someone where to gain access. That will always be the first thing I think of when it comes to building something. After that I enter a broader mindset, the first part is thinking analytically, using a very analytical mindset helps to be detail orriented and cautious as I run through different parts of the required code.

The next is with lack of experiance, is the trial and error. Experamental using things like class resources and different tutorials to really learn the best way to implement a certain class or task id. Along with this running through errors the best I can by trying different things to mitigate the errors I am getting.

WHen it comes to limiting bias and discipline within my code I dont find either of them hard. To start I have very little biass on my own code because I know my experience isnt there to be biased of the techniques or things that I use. Knowing it very well may not work I was open minded to learning a new way of implementation never hurt me in anyway.

WHen it comes to disapline, it is very important withing software development. There are many moving parts in a process of development and having strict discipline to make sure that every part works together and everything is clear is very important. The quality and integrity of ones work stands out in this field of study and work, it seems to be one of those things that developers pride themselves on and I always try to abide by that. Cutting corners as we have learned throughout history have lead to problems and mass histari