David Robey

CS-320

SNHU

8/9/2022

Project 2: Summary and Reflections Report

During each of the milestones, I created a unit test at the same time as creating the class file. The reason I did this was to allow me to write code, and then immediately test to see if it was functional within the parameters of the assignment. For instance, I created a constructor and tested that the constructor worked, after that I built parameters into the constructor to make sure that it would follow the milestone guidelines. An example of this was creating a test that would make sure a description held no more than 50 characters. Or that the string ID’s were unable to be duplicated. After testing the class, I would then create service class that went along with it. As I tested the service class, I would make sure it fit the parameters of the assignment as well. One of things I made sure to do was check to see if it didn’t break any code such string lengths, adding and deleting ids, and any other parameters of the assignment.

At first, I was still new to using junit tests, and made sure that everything was a Boolean method. This way I can check to see if it was true or false that the method worked. In the next milestone assignment, I had learned more about how to use the Junit methods to help create tests that were a lot more efficient and didn’t need to return a Boolean to test if it was effective or not. The more I used the Junit tests, the shorter time it took me to complete the milestones. It had started with taking 6 hours to complete and by the final milestone, it had only taken me 1 hour to complete the assignment. My goal to completing the test cases was to reach an 80 percent coverage rating at the very least. I tried to always go higher, and by project one, I had done 87 percent coverage with 0 failures in the test.

The reason I ran the tests as I was developing was to find the failures early rather than later, during the first milestone, I had a failure that allowed for multiple ID’s to be the same. When I found this through a failed test, I was quickly able to fix it because it was the code, I had just got done writing. Each method created had its own test, some of them I had lumped together for instance, adding and deleting methods. This was to not have redundant code within the test. One of the best things I had learned to do was lambda functions. Before this class, I had no idea how to utilize them. When I had learned how to do the assertThrows method found in appointment and task tests, it was game changing and made creating tests easier and more efficient. What this did is change it from trying to check a value with true or false to stopping the code from an exception being thrown. This way it was easier to test and easier to write the code to see where something went wrong.

During this class I did not perform much on static testing. Static testing is done through running a static test that performs a probe of the code without ever running it. This is helpful to help find OWASP top 10 security threats. One of the benefits to this is it allows for you to know if there is a security threat before it is ever released and can save time, money, and protect the company’s image.

What this class focused on was dynamic testing through Junit. I employed a boundaries testing where I test the boundaries of both upper and lower bounds. This is seen when I performed a null test, as well as tests that made it so it went past the amount of designated characters. I also performed tests to make sure that input that was supposed to work would work. This allowed for me to not have to test every 50-character choice for description in both task and appointment classes. This would be called a boundary value analysis which is a form of black box testing. I also utilized integration testing. This is found in each of the service classes where I would test the integration of contact class, appointment class, or task class with their respective service classes. I made sure that integrating them did not break the previously written code and that they were compatible with each other. Finally during the project one, I had to integrate them all together under java project and run a coverage test that found I had an 87% coverage.

My testing mindset has changed a lot since beginning this class and performing the work for the project. I was use to testing each line of code I created to see if it worked, but only by running the code, after running the code and coming up on error, I would have to sit their and try to constantly recreate it, adding command line prompts to narrow down where it had went wrong and then performing a debug on that area and stepping through each line of code. Now I am more understanding that you can create a test bench that you can test multiple values at the same time and then narrow it down faster and more efficiently.

The caution I employed was mostly to trying something new. I was at first afraid of how to start utilizing the Junit testing tools and was a bit worried that using it would be hard to understand and figure out. I tried to limit bias by looking at it in an objective way. I tried to create tests not to prove that it would work, but to prove that the boundaries of the assignment were reached. Creating null tests, add and delete tests, unique ID tests, while not creating assumption of “This worked in the last set of tests so it works in this one automatically.” I feel like there would be a lot of concern for bias when testing your own code. When someone spends a lot of time developing something, and after weeks of developing, they find that a certain set of events makes it not work, there is that desire to try and sweep it under the rug like its just a glitch for that one time. Along with being biased, this would also be cutting corners which should never be done. I found during my testing that a particular line of code that was fixed before being turned in should have worked. It was suppose to validate the character length of input. I thought it worked, and unfortunately didn’t. Little did I know that I the logic incorrectly coded and it would cause the program to run, but it would not have followed any of the criteria for the assignment. If I had just assumed that it would work, or even knew it didn’t work and shrugged it off like it was just a glitch, then I would be both biased towards my code and also be cutting corners in the testing procedures. If this application was being developed for an actual company, I would have failed to perform what was needed due to a lack of discipline.

Citations

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015). *Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition).* BCS The Chartered Institute for IT. Retrieved from   
<https://app.knovel.com/hotlink/toc/id:kpSTAIST01/software-testing-an-istqb/software-testing-an-istqb>

Agarwal, T. (2020, February 11). *Testing techniques : Types, advantages and disadvantages*. ElProCus. Retrieved August 9, 2022, from https://www.elprocus.com/what-are-testing-techniques-types-advantages-disadvantages/