Justin McKinney

Project Two

CS-320

June 25, 2022

My goal when I was developing the different test cases for my code was to ensure that I tested each functionality so I could verify that the code addressed the requirements and was working as I intended. I realize now that my coding and testing approach wasn’t the greatest for the contact service, I made the mistake of implementing an entire user interface section that would require various inputs from the user. I know now this section wasn’t needed as per the requirements and I also realize it wasn’t covered by my Junit test cases. I did however test it by executing it and running through various inputs to ensure everything worked as I wanted. I fixed this mistake for the final project submission and removed the entire user interface portion of the code. I feel like I did a much better job with the task service and appointment service code overall and specifically in the Junit testing section. I made sure that each of the functions were all covered by at least one test case, and I verified that each of these cases passed. I also reviewed the requirements on multiple occasions to ensure that I captured all the desired functionality without any unnecessary additions. I also verified each of the test cases passed as they should and even went as far as editing a few just to ensure the test would fail if the parameters were wrong to make sure I wasn’t missing something in the logic that would enable the test to always pass

While developing the various test cases I tried to envision how the code would be used and tried to capture that within each test case to make sure all aspects of the program were covered. I was able to verify that my code was technically sound by both utilizing the Junit test cases and by writing various lines of code and then executing the program to see if the outputs were as I expected. I would write several lines of code in which I would add new objects and then use different methods to manipulate those objects and then I would send the output to the screen so I could verify each method did exactly as I wanted it to do. An example of this was when I wrote some code to add a new task and I would make sure the new task has one argument that was either null or too long and then I would run and verify that the exceptions I wrote on lines 17-42 of task.java would throw as I expected them to. I tested this further by using the setters instead of the constructor to set parameters so I could verify the checks for null and length would still throw exceptions via that route.

I tried to do my best to ensure my code was as efficient as possible by limiting unnecessary lines of code where I could. One example of this is in the TaskService.java on lines 31-35, when I was writing the code for the removeTask method I chose to implement a call to the findTask method instead of writing all the code out again to loop through the array and find the specific task. I also chose to not write a test case for the findTask method because it is essentially covered in the test case for testRemoveTask on lines 36-44 of TaskServiceTest.java. I figured since the removeTask method makes a call to the findTask method then it would be sufficient to only write a test case for one and keep things as simple as possible. Overall, I feel like I did a fairly good job of ensuring all the code and functionalities were captured inside of a test case. I also believe that the code I wrote fully address the requirements set forth in the module four assignment document.

In my work on each milestone the main test technique that I implemented was specification-based testing. I think the reason I used this testing technique the most is because it ensures that my testing covers the required functionality that is laid out in the provided requirements/rubric document. I like this particular form of testing especially when dealing with assignments because it gives me confidence in the fact that my code will address all the tasks laid out in the rubric. An example of this is my code would be the test cases that I developed that check to make sure exceptions are thrown when the value of a variable is either null or too long. I derived these test cases directly from the requirements document and used them to verify that my program was working as per the requirements.

One of the testing techniques that I didn’t use that often was static testing. Static testing is done by reviewing the code, or any of the provided documents to try and to find and remove errors before they become a bigger problem later. The main reason I didn’t really use this method is because the provided document in this instance isn’t something that can be reviewed and changed and most of my code was tested within the test cases I designed using the requirements document. I did however use static testing a small amount on the contact service milestone. In this milestone I erroneously added a user interface menu and much of that code wasn’t really covered inside the test cases that I designed as I was unsure how to optimally do so. In this section of the code, I went through line by line and tried to picture how the program would be functioning and ensure the logic was correct.

Specification-based testing techniques are a very important part of any software development project because they are a means to ensuring the developed software meets the functionalities outlined by the requirements document. As you can imagine any development team would want to make sure their software does exactly what it is intended to do, otherwise the stack holders would not be very happy with the end result. Static testing is equally as important because without it we could possibly have a requirements document that is overly vague which could lead to software being designed incorrectly and costly changes later down the line. Static testing is also important because it can be done very early in the life cycle and any errors that are found can be corrected early with minimal impact to the budget. Each testing technique is good on its own but to really have full coverage and ensure you develop a good product you have to implement multiple techniques and strategies.

While working on this project I tried to take on the mindset of someone who was writing this code for an actual company and not just for a school assignment. I tried to be very cautious in ensuring that I addressed all the needs of the customer with very efficient and error free code. I made sure to frequently check the requirements document against the code that I had written. I tried very hard to implement any of the feedback I received into the final project to make it as professional as possible. As I was writing the code for each of the services I tried to picture how each section of the code would work together and tried to keep things as simple as possible to make sure the interactions would not cause any unseen issues during normal operation. I also feel like my Junit testing had very thorough code coverage, I really wanted to be sure that each section did exactly what I wanted it to do.

I imagine that bias would be a big concern if I were to be responsible for testing my own code, it would make it harder to see areas that may fall outside of the requirements. An example of this would be when I created and tested the contact service code, as I mentioned earlier, I added an entire user interface section that wasn’t needed as per the requirements. If someone else would have been responsible for testing my code then this error could have possibly been caught earlier, therefore it is hard to efficiently test your own code. We often get tunnel vision when working on a project and carry that same tunnel vision over to the testing if we aren’t very careful as it is much harder to see your own mistakes. I think the best way to limit bias when reviewing my own code is to try to view the code as if it is someone else’s code and go over it thoroughly while checking it against the requirements documents.

It is extremely important for developers to be quality focused anytime they are writing code or developing software. Quality assurance must be on the forefront of every developer’s mind if they wish to deliver a product that addresses all the clients needs while being as issue free as possible. I feel like writing quality code is something that is learned and perfected over time, in the beginning we may make certain mistakes, but the key is to learn from these mistakes and to do everything we can to prevent ourselves from making that same mistake again in the future. The more we write code the better we will become at identifying issues earlier in the development life cycle because we will have more experience.