Project Two: Summary and Reflection

Jeremy Leuck

This course has been my first encounter with any sort of testing other than a manual review to look for simple mistakes. Learning about the Junit tests and how to deploy them has expanded my knowledge of testing and how code works to a new level. Being able to develop the tests for code that I wrote allows me to know exactly what to test for and ensure the code is fluent. When given the requirements for the first assignment, Contact Service, the code was decently simple. I was able to create protected variables, with getter and setter methods to ensure that. Within the setter methods, we were able to ensure the variables were meeting the length and non-null requirements. This is the same with the ContactService.java, where we created code to add, delete, and update each part of the contact. This assignment had the longest list of requirements out of the three assignments for project one, but it was not difficult to create the code. Every single requirement was met for the Contact Service, which made it easy to simply modify the code to fit Task and Appointment requirements. The more difficult part for me was writing the Unit tests for the first time because I had never had too before. Knowing what to test for helped a lot, but the actual syntax of the tests caused initial trouble. I was able to get the test to run without error, as well as get 100% coverage when running the Coverage Junit test. Although that does not mean the project is perfect, the unit tests were perfect for the assignment requirements.

Having touched on the trouble of writing Junit test initially, I will go into more depth here. Seeing as I never wrote tests before, I wanted to make sure that I spent a solid chunk of time learning the ins and outs for these requirements. There was an announcement post earlier in the course that essentially said to keep things simple and solely hit requirements, which helped me digest the information better to match what we were learning and not go above and beyond so that the program was not confusing.

To discuss sound and efficient code, I will be referring to Task.Java and TaskService.Java. The code I wrote for this assignment was created to be simple and easy to modify and test. For example, in Task.Java, I created setter methods that implemented that each module validated input. That made it easier to read as well as made sure that each variable is validated correctly. On top of that, it makes it easier to test since I can call on each method individually. Another example of this is in TaskService. Java where I created separate methods to update the task created instead of creating a method filled with lines of code (Lines 47-75). It may add a few extra lines of code, but this ensures that each method is doing exactly what it is being asked to do.

Next up is the discussion of efficient code. Even though there were more lines due to many methods, they are simple methods that were able to be transported and modified to different assignments with different variable names which is good to have when working on projects. By creating clear comments for which each one does there is no clutter. It also only takes a simple test for the method and saves time.

For the assignments and project, I only used two types of tests. The first one that I used was unit testing with Junit tests, and I manually tested it by looking over each line. By reading over it I ensured that the project was clean and by writing tests classes I was able to back up my findings. Combining these two is perfect because it provides a great base foundation for sustainable and clean code. After unit testing, I used the technique of Acceptance and System testing to ensure all requirements were met and the code was functional. Even though I wrote each line of code to fit the requirements, ensuring the requirements are met saves time. We do not have to go back afterwards to create or fix something to match requirements.

Even though our techniques were perfect for this assignment, there are others that could have been used, and need to be used for a larger scale program. An example of this is Integration testing. There was no need to fully use this technique due to the simplicity of the programs developed. It did not make sense to spend time on this since there were not many pieces to the project. Another example of testing that was not used was performance testing. This is another technique that is intended for a larger project that does more than just take an input and sets a variable. Our program should not stress a computer or the program since it is simple and does not deal with large amounts of data input or code that is being run at once. This means there is no need to run a stress or spike tests. The last technique I will discuss that was not used is Security testing technique. Again, since the program was simple to code, it was simple to protect. I made sure to privatize variables being used which does add a form of encapsulation and protection, but the program itself was not complicated enough to need to run a security test or develop levels of security. Had I been required to, or if it was to be released to the public, I would have added more security techniques and then tested them.

In all honesty it was hard to initially change my mindset from developer to tester. I like to work in bunches so trying to go from writing to testing was difficult. I found it easier to write the program, walk away for a little bit, and then come back to the project and write the tests. This allowed me to be more cautious when writing the tests to make sure I was not just writing tests for the code, but also for the requirements. It is easy to get lost and write tests to mirror the code, but to really test a program, you need to think with a different mindset. For example, in my test classes, I wrote the test to see if I could add a contact with an identical ID as another contact. If I was writing a test just to get the checkmark, I could have written a test to add a contact, but instead I sat back and thought about the requirements.

Another problem I found was being biased. Seeing as I wanted my code to work, and my test to say it is right, the first test I wrote was not acceptable. I made sure I passed instead of making sure it met requirements. I can see how initially this is a problem for people new to testing their code because I fell into the same trap. Writing a test that gives you a confirmation bias saying your project is correct can create all sorts of disasters in security and program integrity.

In software development and engineering, testing is without a doubt the most important part when looking at the big picture of a project. This is the reason why being disciplined in your testing is so vital. While you need the code to run the application, testing will give the final okay. Not only is testing done when the project is first being created, but a good program is tested repeatedly while it is being used. Testing ensures that the program runs without errors, protects the data, and has no room for a hacker to breach. Between manual, functional, and non-functional testing, the program will be as safe and secure as it can be from the get-go.

References

García, B. (2017). *Mastering Software Testing with JUnit 5*. Packt Publishing.