Gabriel Romero

Module 7 Project Two

When going over the requirements of each feature I planned of action to go piecewise for each of the validation points. I wanted to ensure what was being implemented was also being tested. For example, in the contacts features I ensured each validation point was tested individually before moving onto testing the next portion. I separated the tests for length parameters and null values. For the tasks portion, there were fewer tests, so I ensured to include the ones that were needed. For this case I created methods within the task class that were only going to be used by the taskservice class. This in turn created a lesser coverage on paper but in fact the coverage was well over 80% for both test cases. Leading to why I feel my approach aligned with the requirements. Looking at all my test cases and going through them one by one, I had well over 80% coverage on each. This meant that for every method and error throw I setup I was hitting the needed portions of my code. This meant that my tests were also covering a large portion of the code base and I felt confident that no validation was left untested. In fact, on a second pass I was able to determine that there was a method in my appoinments class that wasn’t being utilized properly or tested. I was able to rectify this in the last project and get it corrected increasing the coverage to over 90%!

Going into the process of writing the JUint tests took me some time to wrap my head around fully. I was able to find a lot of examples of what could be done and started implementing assertTrue and assertThrow methods to ensure what was intended to happen did happen. A good example of this was made in my appointments testing, one of the parameters that were to be passed was supposed to be a date. To simulate a date, I had to convert a string in the format of what would have been implemented in the actual UI of the application. You can see an example below:

Text

Description automatically generated

I created a method to convert a string into a date just in case what was being passed to the service was just a string. This added to the number of tests I had and was not actually one of the validation points, but I believe was an important part to test and validate. I also made sure to keep my tests and code as DRY as possible. Keeping that mentality and taking my time with each validation helped me create an efficient testing document for the requirements asked from the stakeholders. That tied in with the coverage percentage I was achieving leads me to believe that not only was it effective but efficient.

I used a combination of static testing and dynamic testing. With static testing I took my time to write out code and I would revisit with a fresh pair of eyes the next day. I would then review the code over and over without running anything and ensure there were no errors. I would follow the flow of the code mentally as if I were making an instance of the classes and then using the service to manipulate what I wanted. Following this I would setup the Junit tests and go step by step. I would start with the main structure and work my way through testing each validation step, like in contacts with the name, phone number and address. Each one of those having a null and length test attached. This meant that each field had at least two tests per.

Well, when I said Dynamic testing, it wasn’t the full scope of dynamic testing. The code base had no UI or backend truly, so it was not a fully fledged run of the application in the specific definition of dynamic testing. The Junit testing was more of testing in a vacuum. On that note there were some static techniques that were not implemented either, for example code reviews. There was no one I went over the code with. This means that my bias and blind eye to my mistakes could have cost me if I missed something. There was also no exploratory testing or heavy edge case testing.

On a much larger scale having a full-on QA session with the code in parts would be the way to go. Each component that is created for the grand scheme should be tested thoroughly both individually and when combined with the larger project. This would ensure the best quality and cover the most ground when testing. These straightforward Junit tests are great for the cases you can define, but what about that large amounts that you might miss? That’s where use case testing and defining decision tables come in handy. Having a team in generally to work through the validations and requirements would allow a broader spectrum of issues to be tested. Now we don’t have infinite time to work on projects and concessions would have to be made. The proper approach would be to focus on what can be done and use the best case test scenarios that fit within the timeline and also provide the completion of what is being asked for.

My mindset going into this project was one of caution and focus. I knew what was being asked but, in the beginning, I must admit, that I was a bit lost. I was uncertain of how best to test the requirements and ended up with something that was repetitive and inefficient. I went to the drawing board more times that I can count but eventually I realized I was overcomplicating and being over cautious with what was being asked. That simple tutorial that was provided really helped shine a light in the direction I needed to go, and I was able to take off from there. From there I began looking into what validation methods I could use and how best to navigate the Junit test cases. I decided that for every method I create, I will create a corresponding test for it. Within that test if there were multiple clauses for validation to fail, those were also added. This led to a few encompassing tests like my appointmentServiceTest.java file. Within I wrote a removal test code that basically tested every aspect previously as it had to make sure to correctly create entries then remove them, I then tested to make sure they were in fact removed by attempting to remove them again and getting the return of false if they no longer existed.

Looking at how the service and structure class worked in tandem I ended up creating some methods that the structure class never used. This can be seen in my task.java where I have some setter methods that were not being used by the class call. These methods were primarily for the taskservice file to use to change out the fields that were required. This gave me an appreciation of how the two work together and how a service could really be leveraged to complete complicated tasks within an application. It was a learning experience for me to get it to a point where I was able to get all my tests running, and once I got most of them in the green, it felt like I accomplished something.

One way I can see bias interfering with my review of my own code is through believing you know exactly what you wrote and what is going on. I feel like it can be easy to lose track of what is being done to what variable if you end up with a lot of similar naming conventions or terrible naming conventions. This can in turn lead to some disastrous mistake that you might not catch believing what you are writing is good. I could also be overconfident in my testing and thing I covered all the bases when I left out several important edge cases that could be used to compromise the application. The opposite could be true as well, where you are spending so much time on the code due to lack of confidence that deadlines could pass without the product being close to ready. I am strong believer in having another set of eyes look at the code before going live, it is only helpful.

When it comes to being disciplined with code writing, I believe it must become somewhat of a tenet to write clean, effective, efficient, and safe code. In today's world its too easy to slip up and risk releasing data into the open. Data breaches and escapements are far too common nowadays to take it lightly. You do not want to be the person responsible for leaking someone’s personal information or causing and entire applications database to be wiped. These are all actual possibilities and it’s important to keep that in forefront as you write code. Especially anything that is being released to the public. Best practices must be kept up to date and security is a must. Security and correct code go hand in hand within my book. With everything I plan on writing, I will make sure to come up with those edge cases and put my code through grueling tests before sending it off to go to production.