Landon Johnston Borboa

CS 320

SNHU

15 October 2022

**Summary**

In this project I created ContactService, TaskService, and AppointmentService for an application based on customer requirements. I then created JUnit tests for each class to test the requirements.

The approach I took aligns with the software requirements. I achieved this by using both comments and tests and grouping customer requirements. For example, all requirements for setting contact ID will be lumped into the same function. I put these requirements at the beginning of the code in the form of a comment. This ensured while I was coding, I was hitting all the requirements for each part as well as kept me in front of my code instead of flipping back and forth between code and documentation. The second way I assured it was by creating tests for each requirement stated separately. For example, I created a test that made sure nothing was set as null. The test would pass null to each variable separately and make sure a error was given to pass the test.

The overall quality of my JUnit tests was effective. I had no test under a coverage of 80%. The Appointment Service, Contact, and Task Test all had coverages of 100% in each individual class. This means that my JUnit test hit every line of code. The contact service class had a coverage of 89.5%, I was unable to reach full coverage for this class because I had added extra code that checked for duplicate contacts. It was not stated in the customer requirements, so I did not test for It. The Appointment class had the lowest coverage but still acceptable at 80%. The reason was for I created a constructor in case nothing was inputted, looking back this was not required so I would take this out. Finally, the task service class had a 85.9% coverage, in this just like the contact service I did not check for duplicates. Overall, my code had an average of coverage of 92% without any single class under 80% therefore my Junit tests were good quality and effective.

This was my first-time experiencing JUnit tests, and it surprised me how useful it is to a developer. Not only does it show that I kept to the requirements, but it allowed me to see how my code behaved without making a user interface or using a main function. It helped make my code technically sound and efficient

To ensure that my code was technically sound I used proper error handling such as “IllegalArgumentException” and “NullPointerException”. I realized that using the same generic error thrown would not work for all the input. Being flexible and observant led me to use proper tests to check for errors thrown. I also changed my code to “localDate” instead of “newDate” since it was depreciated. Using depreciated code can lead to errors and bugs in the future when it’s no longer supported. I believe that this led me to technically sound code.

To make sure my code was efficient I used in memory and limit extra coding. First, I did not add a lot of redundant code, I could have been better in not adding duplicate checker but I belive that it will be used if I furthered the project. I also used an arraylist by importing “java.util.ArrayList”. This dosent have to reach out to a database instead uses inline memory and can be accessed fast. I also added in constructors one default one and one to meet the requirements.

**Reflection**

To start in each class, I began with unit testing. I would test each function I wrote, I would code the constructor, then a getter and setter for a specific action such as setting contact ID. After writing that I wrote a test to make sure the input given was being used correctly and getting set. I did this for each string I needed to test. For the services class I had to do more of integration testing since the service class relies on the first class being done and ready to go. Having done it correctly the first time allowed for the integration to go smoothly. I also had functional test that made sure all requirements were being met.

There is a lot of other testing methods that I did not get to with my project. I think one of the many that would be beneficial for future testing is performance testing. This would show how my code handles with a lot of data getting thrown at it. It would be important to see this before the customer has access to it.

Since I was the software developer and tester it was important that I tested everything individually before moving onto putting it all together. Unit testing is important because not only does it ensure that the coding is correct and behaving correctly. If you don’t do unit testing it might be harder to pinpoint the issue as well as having to rewrite one portion might affect other code related. Once unit testing is done, I think its beneficial to both a tester and software developer to do integration test. Integration testing is important because it can be the entire code together, but also individual class tested together. This ensure that all moving pieces are cohesive and practical.

The mindset I adopted was keeping unbiased while knowing how the code worked together. This allowed me to make sure my tests were running correctly. Knowing how to declare local date and having to pass it through was vital. I also had to use arrays in my test to ensure it was saving to memory correctly. I think having written it. It made it a lot easier to know how to test it and how it all worked together.

I think the best way I limited bias was trying to only code to the requirements and not add anything unnecessary. I also didn’t code for tests to work I coded for the requirements then made sure it was working correctly. I was running into issues when I tried to go above what was expected and was getting errors in my testing. That’s when I went back to the requirements to see where I veered off.

It is important to stay disciplined and not cut corners because it will lead in errors that could cost the company more than money. It could cost the company a lot in money, time and reputation. To avoid technical debt I am going to make sure I don’t rush things or cut corners by using all the best standards. I will also make sure I have detailed comments for when I go back or if someone else reviews my code.