Connor Brereton

CS320 - Testing

Southern New Hampshire University

Professor Phillips

Project 2

The software requirements and the testing approaches aligned perfectly as it was perfectly implemented programmatically in the code base. One example of this was in the contact class. In this class the requirements clearly stated that the first and last name cannot be longer than 10 characters. Here is the line of code that implemented that perfectly tested use case of input that is too long:

```
Assertion.assertThrows(IllegalArgumentException.class, () -> {
        New Contact(“00112233445566778899”, "Connor William Connor William", "Brereton
Brereton Brereton Brereton", "40888888888888", "123 Main Street San Francisco, CA, 94109");
}
```

In this case the Task Test class, when being JUnit tested, would throw an illegal argument that the task_id is greater than 10 characters. The quality of the JUnit testing is pretty good and I would say that it covers the entire surface area of the functionality in the application. If I had to point out one area where the testing was not as good I would say that it is in the area that is not the contact service since I primarily focused on that area. One area that helped out a lot was the resources provided by the course instructor. These were great guidance for the rest of the program.

To ensure that the code was technically sound I took a few measures. The first was using really tight data structures to ensure that the fundamentals of computer science were followed. One area where I used them was in the contact class to store the list of contacts. The algorithms that I used throughout the functions were *assertEquals* which splits the string into characters in memory only to compare it against another parameter; *assertTrue* which does a logical comparison of two different values, and *assertThrows* which ensures that a test case is raised. The code I wrote was extremely efficient because it’s a pretty easy assignment. I used a few resources for getting the work done like coding tutorials, technical blogs, documentation, etc. There was a test case that was used to ensure that duplicate contacts were not added to the contact service.

Because the software testing techniques that I used were all based on specification and structure the software testing techniques are under the black and white box techniques. Black box testing techniques derive their test cases from the specification of what the system is supposed to do. The use of black box testing techniques is cool because they also include the ability to test for valid and invalid inputs, decision tables, along with state transition to test events changing. Finally, there are also boundary tests that are used to test the boundary of the software program. There are many elements to structure based testing. Structure based testing is encompassed as statement, path, and branch coverage. Structure based testing is used to compare components at different levels.

The testing technique that I largely ignored was the experience-based technique. These techniques see the users and the testers' experiences to figure out what the most important areas of the system are. They figure out the expected use along with the sites of the errors. There are two main components of this that are error guessing and exploratory testing. Error guessing is where the prior experience is used to determine which tests are best used to analyze the codebase. Exploratory testing is used to figure out which areas are lacking according to the specifications. These two techniques seemed like overkill for the overall project.

The mindset that I used when working on this project was focused on reviewing information that I had already learned since I had experience with testing prior to this course. I tried my hardest to limit my bias in this project by not forming hypotheses and making invalid assumptions. As a result, I tested the entirety of the software to make sure that I did not skip over anything due to bias. Another reason that I made sure to test the code prolifically was because it is vital practice as a software engineer. At the end of the day we're preparing for the real world.

Works Cited

Differences between Black Box Testing vs White Box Testing - GeeksforGeeks. (2018). Retrieved 17 August 2021, from https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/#:~:text=Black%20Box%20Testing%20is%20a,is%20known%20to%20the%20tester.

Software Testing - Definition, Types, Methods, Approaches. (2015). Retrieved 17 August 2021, from https://www.softwaretestingmaterial.com/software-testing/

Junit Assert & AssertEquals with Example. (2021). Retrieved 17 August 2021, from https://www.guru99.com/junit-assert.html