Project Two

Kyle B. Wucik

Dr. Albanie T. Bolton

CS320 - Software Testing, Automation, and Quality Assurance

20 February 2023

The way that I chose to approach the testing for the code I wrote during class and in Project One was by utilizing JUnit testing. I made sure to test each of the written requirements individually to ensure the functions were working properly. For example, for the taskID function, I wrote tests with valid inputs to determine if they could be added and with invalid parameters to make sure that it threw the proper error.

My approach was aligned with the software requirements, as I made sure to validate that each function had valid parameters and met specific length requirements. For example, a description was required to have a length of 50 characters or less.

I believe that the overall quality of my JUnit tests was good. I used code such as @Before and @After, as well as @Test(expected = xyz) to ensure that the error thrown was in line with what the function was supposed to do.

My experience writing JUnit tests was pretty straightforward for the most part. For each function, I made a valid and invalid test to ensure that it was working properly. For the valid tests, I made sure to check the input with the variable that was added to a list (the array list that held all of the tasks, appointments, etc)

To ensure that my code was technically sound, I compiled and ran the code after every function I wrote to avoid syntax errors and followed common coding practices, such as using pointers, setters, and getters. I also wrote comments to remember what each function did and why I wrote it.

Similarly, to make sure my code was efficient followed the general Object-Oriented Programming guidelines. For example, I made sure that there were no redundancies and used pointers to make sure the code wasn't overwhelming to look at. I also make sure to use setters and getters to allow updating specific parameters to be updated and returned for other functions.

The software testing technique that I chose to use was JUnit testing. I chose JUnit testing to make sure each function worked properly and, if they didn't, they would throw errors.

There are a lot of other software testing techniques that I chose not to use for this project, including integration, acceptance, and smoke tests. Each test makes sure a different part of the coding process is good to go to the client. For example, integration testing is used to check how well your program works with other programs, acceptance tests ensure that the code meets the client's requirements, and smoke tests are quick tests to ensure that the code is working properly.

When making tests for your program, it is important to be diverse with the tests, for project one I tested string length, made sure the input was not NULL, and that each parameter could be set. This ensures your code works properly.

When programming it's important to be cautious and aware of what you are doing inside the program because even the slightest mistake in the written code might ruin how the program functions; incorrect code will give an incorrect output; Incorrect outputs are discouraging. Writing comments for your code is important because it helps others understand what each function does and why it was written.

It is also important to limit bias in code review. Software development is a forever-changing field and it is important to be flexible and adapt as standards change to avoid ineffective code.

When testing your own code it's important not to be biased and make sure the program functions the way you want it to. It's good practice to make sure you are self-aware the developer, might prefer and instead, focus on meeting the client's requirements, even if you think otherwise.

Discipline is also an important thing to have when coding; Making sure you write code the same way consistently is an important factor in developing your coding skills. There are various examples of industry code and it's easy to tell when someone codes for their own fun or is writing for work (the working code will follow industry practices). Some of the practices include writing clean and legible code that utilizes comments so others understand what the program does. This will also help to prevent technical debt and maintain a commitment to quality.

**References**

Boni Garcia. (2017). *Mastering Software Testing with JUnit 5 : A Comprehensive, Hands-on Guide on Unit Testing Framework for Java Programming Language*. Packt Publishing. Retrieved Febuary 19th, 2023 from https://eds-p-ebscohost-com.ezproxy.snhu.edu/eds/detail/detail?vid=0&sid=9f2f8a47-3853-467e-92fe-96c6d5d6ccba%40redis&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#AN=1626950&db=nlebk

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.

Southern New Hampshire University. (n.d.). Software security: Module one through seven. Brightspace. Retrieved February 20th, 2023, from https://learn.snhu.edu/d2l/le/content/1236879/Home?itemIdentifier=D2L.LE.Content.ContentObject.ModuleCO-21978651