Clayton Smith

CS-320

Project 2

06/12/2023

Summarizing my testing approach on the latest project submission, I utilized a testing method called “JUnit” testing. To ensure my testing approach was aligned with the software requirements described in the project outline, I made sure the JUnit tests covered each requirement that was coded. For example, the project asked that a “contact” service was added that created a contact ID that could not be null, nor longer than 10 characters. To test if this function was working properly, I created a JUnit test to check what would happen if a contact ID was added the right way, but also if a contact ID was added incorrectly (too many characters). The JUnit test will then run the scenario of a properly added contact ID and an improper ID to make sure it will run correctly when an ID is added properly, or throw an exception if an improper ID is added. To make sure my JUnit tests were effective, I ran a coverage test on the entire project. My coverage came back at around 93%, which ensures that at least 93% of my code was tested as working properly.

My experience writing JUnit tests went fairly smooth after understanding the basic concepts of the testing itself. Finding out the proper way to write tests was tedious initially, but became easier the more tests I wrote. To ensure my code was technically sound, I manually reviewed each class checking for any obvious errors. The JUnit tests also helped to ensure the code worked as intended. As far as efficiency goes, adding the array list to my “service” classes was a great way to compile adding appointments, contacts, and tasks.

Reflecting on the testing techniques, I not only used JUnit testing, but also used some other methods of testing alongside the JUnit tests. As mentioned above, I utilized manual testing. As I wrote code, I would manually go through everything I wrote to ensure everything looked in order and nothing was mislabeled or spelled wrong. Manual testing ensured that there were no unnecessary errors in the future when I got to the JUnit testing portion of the project. It also prevented any time lost during the production portion of the project which allowed for more time to be spent on the JUnit portion.

Testing methods not used for this project or not the main focus of testing would include automated testing, functional/non-functional testing, system testing, acceptance testing, and regression testing. Automated testing was used during the testing period of my code, but was not a main focus compared to the main testing function we used which was JUnit testing. Functional testing sounds very similar to Unit testing, but is slightly different. Functional testing will test a piece of functionality in a system which could potentially interact with dependencies such as databases or web services. Unit testing tests individual units such as a function in a class, which is what we focused on. System testing focuses on how components of an application interact together in a completed, integrated system or application. Acceptance testing determines if an application meets the end-user’s approval. Regression testing basically re-runs functional and non-functional tests to make sure an application functions as intended after updates to the code. Even though I did not utilize some of the testing methods above, there are many benefits to utilizing those testing methods in the right scenarios. They each have their perks and can be efficient during proper coding environments.

As I worked on this project, the mindset I adopted was fairly calm. The testing became seamless as I practiced and implemented, as did the code I wrote. As far as employing caution, I would say I became cautious when I would add a test and it would not work immediately. This would make me go back through the code and double check myself that it was written properly and look over my test method to ensure everything was written and functioning properly. I appreciated the code that I wrote because it was a fairly large project to oversee compared to my previous projects I had worked on. Working on a project this large can become overwhelming fairly quickly if you don’t stay on top of everything and make sure to leave notes for yourself so you know where you left off. I found myself creating many drafts of code and going back and forth to ensure a proper outcome.

Bias is always going to be there regardless of what project you are working on. As a developer, there will always be better ways to do certain tasks, and there will always be a client that wants it the complete opposite way. When trying to limit bias, utilizing the JUnit testing to apply as much coverage as possible to test the functionality of the code helped eliminate any bias over certain parts of code compared to others. Being responsible to test your own code could create a bias fast, as you could accidentally write a test the wrong way and the code could come out as working properly, but in reality it would be flawed. Having another set of eyes for testing is always a good idea to eliminate bias and ensure the code is working properly for the end-users.

Having proper discipline when writing code is extremely important. Cutting corners when developing software can lead to many errors not only in the code, but could even cost you money or a job if done too many times. If you cut corners when testing or coding, this can lead to unnecessary or preventable errors that could’ve been caught if not for negligence. There have been documented tragedies that have happened due to lack of care when coding that have not only cost individuals money, but also lives lost. To avoid technical debt, I will ensure that the proper testing methods are implemented and help create software that is sound in development. If I do my part, I can only hope that others do their part as well to ensure sound software upon deployment to the end-user.

Sources:

Lutkevich, B. (2023, February 1). *What is technical debt?*. WhatIs.com. https://www.techtarget.com/whatis/definition/technical-debt#:~:text=Technical%20debt%20%2D%2D%20or%20tech,costly%20it%20becomes%20to%20rectify.

*What is the difference between unit tests and functional tests?*. Stack Overflow. (2021, October 26). https://stackoverflow.com/questions/2741832/what-is-the-difference-between-unit-tests-and-functional-tests