Austin Adams

7 December 2023

CS-320

Professor Prasad

Project Two: Summary and Reflections Report

My approach to using unit testing for each of the three features for the software was to make use the testing techniques to ensure that each feature correctly met each of the requirements that had been laid out. Having a test for each class and function within those classes was an important part of this. I know that the JUnit tests that I ran for the three features were of good quality since I took time to check that everything ran properly with the expected inputs but also to make sure that the code knew how to properly handle situations when incorrect inputs were provided. For example, if there was a specific limit to the amount of characters that should be allowed for a certain variable in the requirements then I would test that it dealt with inputs of greater length as intended. I ran similar test to make sure that any null inputs could also be handled in an expected way.

In order to make sure that my code was technically sound I followed good coding practices and tested every part of the code afterwards for edge cases. This helped to keep the readability of the code and minimize any possible issues. To ensure that my code was efficient I tried to create a function anytime certain tasks got repeated to minimize the amount of code needed and that the steps in order to accomplish that tasks properly was kept to a minimum.

The primary software testing technique that I used in this project was JUnit testing. This type of testing consist of testing every piece of a software individually from every class to all of the different functions within each of those classes in order to make sure that they are functioning as they are supposed to. This testing technique was crucial in making sure that each part of the code worked as intended and in confirming that all of the requirements for the three different features were all implemented.

There are several different software testing techniques outside of those that I used to complete this project. Static testing is a software testing technique that I did not use for the project but it can be an excellent way to find errors within your code early on. This software testing technique does not require the code to be built or ran which is part of what makes it a very helpful tool in finding any errors early on in a development process.

Static testing is typically done in the early stages of development to find errors within complex code in place of other software testing techniques since it can be done without running the code, can be done very quickly and is generally a less expensive method when compared to testing techniques that require the code to be built and ran in order to work. Software testing techniques such as JUnit testing are typically used later in the development process or as different sections of the software are completed. JUnit testing is used once different larger sections are completed since it requires that the code is built and ran for it to be carried out. It is used to make sure that the completed piece of the software runs as expected and that there are no issues with each of the individual pieces.

The mindset that I adopted when working on this project was to meet of all of the requirements that had been laid out for the three different features in the most simple methods I could and to do so with as few errors in the code as possible. The extent to which I exercised caution was in my software testing techniques. Before this course I had not used typical testing techniques as were shown in this courses material so I took extra time in making sure that the techniques that I did use were properly implemented and covered as many different possible cases as I could. I made sure that all of the code I wrote followed proper coding practices as well to make it readable and easier to debug if any issues did arise when carrying out my tests.

The way I tried to limit the bias in the review of my code was to plainly look at how well my code fulfilled each of the requirements that had been laid out. Keeping a black and white view of whether those requirements were met and affirming all of that they were met correctly through testing is an important part of making sure that the amount of issues in ones code is kept to a minimum and that there are no assumptions being made on how well your code fulfills certain requirements.

Being disciplined in your commitment to quality as a software engineer is incredibly important. Cutting corners in both writing the code and conducting tests can lead to issues popping up later on that could possibly take considerably more time to resolve than if the proper steps had been taken in the beginning. This keeps your clients happy with a well written quality piece of software that needs minimal maintenance and saves both time and money for both the developers and the client.