Bee Best

10/20/24

CS 320

Module Seven Project

To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

My Junit tests for ever single one of the three aspects of the project were designed in such a way to cover as much as I could reasonably cover, I went through the assignment requirements for each part of the project and ensured the Junit tests covered those. In addition, I added more tests to cover the innate attributes and capabilities of the classes I made to ensure that everything involved was being tested properly.

Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?

Similar to what I stated above, I know my JUnits were effective based on the coverage percentage because I ensured that there were tests not only to cover specific requirements given to me but I also included tests to cover everything that the three aspects of the project could reasonably do.

How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.

I know my code was technically sound because I checked it over multiple times and some tests were so simple that honestly it’d be embarrassing if I made a mistake there (like this *assertThrows*(IllegalArgumentException.class, () -> { new Task("Task Name", null);}); to test for the description). All of my tests passed as well so I’m sure that having the tests pass and having looked over them all multiple times that my code is technically sound. Also I suppose I should mention that I took your feedback into account and used the put method instead of what I had In the last assignment.

How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.

I think the best example of efficiency is via my use of a hash map for the storage of data, a hash map is made to be efficient for quickly looking up a specific thing. There aren’t many more instances I can look to towards efficiency as the overall programs are quite small.

What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

During each milestone, the primary method of testing I used was with Junit tests, I made these tests by ensuring that everything I made as part of the milestone requirements was tested properly to ensure its function. To describe them specifically, we had a list of requirements in each milestone activity (or aspect of the project), for all the meaningful aspects of the code, I ensured that there were tests to go alongside them such that I could test for both the specific requirements given to me and also simply that the functions I created work. To manually list out the techniques used, I would say I used Manual Testing, and Unit Testing, (I don’t know if it would be right to say I used automated testing as I did need to run the Junit tests even if once running it automatically tested everything).

What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.

Some other kinds of testing that could be done might be Integration Testing, where you test if each Unit integrates with each other for the purposes of the project, Performance Testing, and Security Testing, these two are relatively self explanatory. They test performance, so efficiency, and security, which is… security. I don’t know how else to put it.

For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

All of the testing techniques I brought up have great practical use, going through them all one by one, Manual Testing is important simply for finding logical errors you might be unable to find through testing by running it, Unit Testing is practical because it ensures that each component of the system is running as intended according to the tests given, Integration Testing is practical because it allows for testing further than just Unit Testing, because with Integration Testing you can ensure that the system as a whole is cohesive and has no conflicts. Onto the last two, Performance and Security Testing, these two are extremely practical because for any business, both performance and security are imperative to a system running well (and for the business to not lose out on money and/or trust).

Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ **caution**? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.

I did ensure that for the project specifically I used caution while ensuring that everything was up to the standards I was required. Specifically the main thing I did was focus on organization and the implementation of each piece of feedback I was given for the individual pieces used to ultimately build the final project. It was important to pay attention to the relationships between my code because without doing so I wouldn’t have been able to effectively make my Junit tests cover everything that should be tested.

Assess the ways you tried to limit **bias** in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.

I built up my code over an amount of time so that I wouldn’t always be in the same mindset each time I was working on it, by taking breaks and continuing to integrate new information into what I was able to do for the project I ensured that my code was always sound and up to date. In addition I try to keep an open mind towards feedback when working on things as doing so generally results in me being able to actually learn stuff from doing a project and become better at making projects that are cohesive and well put together. I can imagine that testing my own code without any feedback whatsoever could be an issue because without another person looking at my stuff I could easily glance over issues that I simply do not know are issues that need to be corrected (for example when you told me in the feedback to use put, I then made sure to integrate that into all my future work).

Finally, evaluate the importance of being **disciplined** in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.

It’s important to not cut corners when working on things because cutting corners will result in a sloppy software that may be untested or unqualified to be released to the public in any way shape or form. I intend to make all my code designed in such a way that it can be easily added on to in the future or modified such that I don’t leave a large amount of work for myself in the future to fix. For example, the feedback I was given for the first aspect of the project I immediately implemented after I was told about it because I knew then I wouldn’t have to go back and change it later. Past that, I think I could do better by leaving my code more well commented so that it’s easier to understand if I were to go back to it years down the road.