Joshua Perez

SNHU CS-320-T1183

October 13, 2022

**7-2 Project Two**

1. **Summary**
   1. **Describe your unit testing approach for each of the three features.**
      1. **To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.**

The approach I chose for unit testing aligned to the software requirements by strictly adhering to the requirements listed in each module’s programming assignment. I even went a little further for some of the testing since I had designed my program slightly different than what was required. For the contact class, objects had to have a unique contact ID, couldn’t be longer than 10 characters, couldn’t be null or updateable. I tested for these specific things during the Junit testing. I used the same methodology to achieve the requirements for “first name”, “last name”, and “phone number” all of which needed to be shorter than 10 characters and not be null. For the contact service object, I fulfilled requirements by adding the ability to add contacts, delete contacts, and update contacts. I tested this in the JUnit tests by adding example contacts to add, delete, and update.

For the task class we were tasked with creating a unique id shorter than 10 characters, that is not null and not updateable. The name field must be shorter than 20 characters and not null. The description must be shorter than 50 characters and not null. I was able to test against these on the JUnit tests by creating scenarios where the length of these items was exceeded and tested for nullability on actual methods themselves. For the task service object, I fulfilled requirements by adding the ability to add tasks, delete tasks, and update tasks. I tested this in the JUnit tests by adding example tasks to add, delete, and update.

For the appointment class we were tasked with creating a unique id shorter than 10 characters, that is not null and not updateable. The appointment object also had a required appointment date field that cannot be in the past and cannot be null. The appointment description must be shorter than 50 characters and not null. I was able to create comprehensive JUnit tests for these by assigning data that would swing this in both extremes as well as a correct version to make sure that they functioned as intended. For the appointment services we had to be able to both add and delete appointments based directly on the unique appointment ID’s. I was able to test this by assigning appointments with unique IDs and both adding and deleting them.

* + 1. **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?**

I believe the overall quality of my JUnit tests were high and effective. While I was not able to get a concrete coverage number, my coverage map was working and showed that the functions were covered in an effective way.

* 1. **Describe your experience writing the JUnit tests.**
     1. **How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

I ensured that my code was technically sound by following tutorials on the maven and JUnit parasoft website as well as adhering to examples of JUnit pieces of code that I found online for other projects. A specific line I can cite would be lines 17-29 where I set up variables that would breathe life into the testing and allow for a multitude of situational testing. “@BeforeEach” was a technically sound idea I picked up from JUnit tutorials that were somewhat given to us during one of the lesson openings.

* + 1. **How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**

I ensured that my code was efficient by minimizing the amount of extra code that had to be written by somewhat copying what I was doing for one set of tests for another set of tests. The same area I cited in the question above, can practically be seen in use multiple times in the other classes.

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

A software testing technique that I first employed in this project is peer review, I did this by getting on a discord screen share with my friend and figuring out the issues I was originally having with my compiler since everything kept breaking. Together we worked out the kinks with my broken java library, maven install, and JUnit install. Another software testing technique I employed in this project was Unit testing. Essentially, I reviewed my own code, component by component, to make sure that some functionality was there. I also performed the system testing technique to some degree to ensure that the system requirements were met.

* + 1. **What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.**

There are many countless other software testing techniques that I did not use for this project, many because they were simply not needed or necessary in the context. For example, I did not need to do compatibility testing because I did not need to know if this piece of software would work in multiple environments. I also did not need usability testing since the software did not include anything being done with end users and therefore it would have been pointless to test for.

* + 1. **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

The practical uses and implications of the software development techniques I mentioned above are far reaching and encompass basic steps that we can take to improve the overall functionality and quality of the work we are producing. Doing these actions can lead to higher success rates and just plain better code. In the future, while working on larger projects, these will be outright basic steps that must be taken constantly and with each iteration to ensure consistency and avoid mistakes.

* 1. **Mindset**
     1. **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.**

The mindset that I adopted while working on this project is one of someone who is far more experienced than me. I stuck close to the documentation so I could try and figure out how these auxiliary programs work and make sure that I was utilizing them correctly. I employed very little caution when it came to the original writing of the code, but then I switched to a cautious mindset when it came to testing to ensure that I covered every area that needed covering and to make sure that the requirements were being met at every step. It was important to appreciate the complexity of the complexity and interrelationships between the pieces of code being tested because the system is one that’s built on itself. In order for one part of the program to function, a lot of other pieces would also need to function. Without one, you cannot have the other. This can be seen in the way all of the sections were structured, they each required a unique id within an array, meaning that all other parts of the program would have to be tied to said unique id and be able to be pulled up and manipulated, whether deleting, adding, or updating the underlying structure.

* + 1. **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 tried to limit my bias in the review of the code by first showing it to a friend of mine who I know isn’t afraid to say, “this is garbage, why did you do it this way?” and sometimes that’s exactly what I need to push myself. I imagine bias would only be a concern in the professional side when you are the direct sales leader software developer, AKA you’re selling a product that you made versus working on a product for a bigger entity. There would be no reason to be biased when reviewing your code if you are working for someone else.

* + 1. **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 is extremely essential that we be disciplined in the commitment to quality as a software engineering professional, as carelessness can cost the company and possibly you more money than is possible to recover from. Cutting corners ends up costing more money in the long run than just doing it right from the ground up. In order to avoid this technical debt, we should go for the better solution right away, even if it costs more, as it will save you and the company money overall. We can reduce technical debt by using agile methods and developing things on fly, working as a team and holding everyone accountable. A big solution, and one that I know a lot of companies use now, would be using automated software for deployment and bug fixes on top of the individuals writing the code.

**References**

*Junit - Test Framework*. Tutorials Point. (n.d.). Retrieved October 14, 2022, from https://www.tutorialspoint.com/junit/junit\_test\_framework.htm

*Software testing methodologies*. smartbear.com. (n.d.). Retrieved October 14, 2022, from https://smartbear.com/learn/automated-testing/software-testing-methodologies/

YouTube. (2020, September 11). *Junit tutorial*. YouTube. Retrieved October 14, 2022, from https://www.youtube.com/watch?v=KbXhK9HUng4

YouTube. (2022, January 18). *Java unit testing with junit - tutorial - how to create and use unit tests*. YouTube. Retrieved October 14, 2022, from https://www.youtube.com/watch?v=vZm0lHciFsQ&t=1s