Matthew Dziewiecki 7/12/24

CS320

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.

I believe that my approach aligned very well to the software requirements. I used JUnit testing to try and test out the boundaries of the specifications required in the software. For instance, the task specification says that the full name must be greater than 20 characters and must not be null. It will take no more than 4 branches to test this specification.



I support my claims as I tested a valid version of the name, a very long string, an empty string, and a null string.



* + 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 did not aim to cover the Tast and Contract classes in their entirety. The constructors had more than three parameters that needed to be set and I didn’t believe that it was worth the time to go and test each valid and invalid constructors of each of the parameters. Rather, I focused on getting 100% coverage on all the getters, setters, single invalid constructors, and single valid constructors.

A screen shot of a graph

Description automatically generated

* 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.

To ensure that my code was technically sound I made sure to test out multiple ways. For instance, the ContactService class needed to have a method that deleted an ID. There was a couple of ways of testing it out. One way to test it is to look through the entire collection to ensure that what was removed is removed. Another way is to add three objects to the collection, delete one object and then ensure that the collection now only has two objects. If you only perform one of these ways, then errors may still occur. The incorrect item may be removed, and the test would still pass if only the collection size is checked. Also, if all the items were deleted by accident, then the test might still pass if the item is looked for using only that method. To ensure my code was technically sound, I tested both.

A screen shot of a computer program

Description automatically generated

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

To ensure that my code was efficient, such as the example above, it was crucial to confirm that the collection was truly erased. It was not sufficient to look and check the collection’s size after an item was removed. Verifying that the remaining objects were still intact and functioning as expected could go taken a step further. I ran these tests shown below to ensure that the objects were only allocated upon request:



I had also verified that the string remained unchanged following an unsuccessful attempt at updating it:

A screen shot of a computer screen

Description automatically generated

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

The software testing techniques that you employed in this project was JUnit testing and static testing which are both types of white-box testing. Within the project, three modules called for JUnit and static testing. To find any bugs/errors, static testing entails examining the code and contrasting it with the specified requirements. When a JUnit testing fails then I would utilize static testing to help me identify and correct the faulty code. I was trying to match the logic of the class methods to the specified requirements. When I was creating the JUnit tests, I would go back to look over the specified requirements and then look at the methods to search for any errors/bugs I may have missed while writing the code. Once and a while, a Junit test would fail, and I would need to look back at the code to figure out what went wrong. For instance, I had accidentally used a wrong attribute within the code, so the JUnit test failed.

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

Some other software testing techniques that I did not use in this project were integration testing, system testing, and automation testing. Integration testing would test out the application. The specified requirements created a model class and a service class that interacts and drives the model class, so system testing was not needed to be used. Automated testing was not performed either, all the JUnit testing was performed manually.

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

According to a recent textbook by Fishpool, 2020 although units, functions, and modules can be tested on their own, they will be combined with others to form a complete software solution (Fishpool, 2020). I could have used integration testing on any of the classes and they would all function nicely together in the integration layer. For system testing, I could have technically tested the model class through the service class and treat it as a system. For automated testing, I could have created a service that would run the JUnit tests after each build. This form of testing is practically handy for any larger scale projects. A continuous integration pipeline should be included, especially if continuous delivery is being used. This testing can be performed in any stage of the software’s development.

* 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.

In acting as a software testing, I ensure myself to employ caution throughout each phase with a different perspective and look back into the specified requirements. It was important to appreciate the complexity and interrelationships of the code I was testing as it allowed me to be able to truly try and understand the requirement’s actual meaning through the coding phase rather than just was written. For instance, the appointment class required the following: “The appointment object shall have a required description String field that cannot be longer than 50 characters. The description field shall not be null”. This requirement indicates that a logic test for the length and whether the passed argument is null should be conducted. However, this also implies that a description is required no matter what. Therefore, it makes sense that the description shouldn’t be nether empty nor null and the validation test is as follows:

A screen shot of a computer

Description automatically generated

* + 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.

One the software developer side, I can imagine that bias would be a concern if I was responsible for on testing out my own code as I won’t believe that there would be any issues with the code I wrote as I believe in my abilities on writing professional standard working code. To combat this bias and limit it throughout the software’s development I made sure to test out code even if I believed it would perform without an issue. For instance, I utilized the JUnit tests to see if the right expectations would be thrown rather than any exception was being thrown.

A screenshot of a computer screen

Description automatically generated

* + 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 critical to not cut any corners when it comes to writing and testing out code in software development as it can lead to more errors/bugs and can cause setbacks in the development and overall, it is a bad coding standard. According to a recent article by Avgeriou (2024), It is concluded that the best path on combating technical debt and developing a better management is to continuously involve researchers, practitioners, and tool vendors to collaborate and aggressively in parallel to develop tools, and empower the culture change to embrace new coding practices (Averiou, 2024). This is what I aim for to reduce or avoid technical debt as a practitioner in the field. I will also plan on using coding reviews and JUnit tests to thoroughly examine the code and ensure that it is free of any errors/bugs and up to professional coding standards.

Works Cited

Avgeriou, P., Ozkaya, I., Chatzigeorgiou, A., Ciolkowski, M., Ernst, N. A., Koontz, R. J., Poort, E., & Shull, F. (2024). Technical Debt Management: The Road Ahead for Successful Software Delivery. https://doi-org.ezproxy.snhu.edu/10.1109/ICSE-FoSE59343.2023.00007

Fishpool Bernie, & Fishpool Mark. (2020). 10.2.1.4 Other Types of White Box Testing. In Software Development in Practice. BCS The Chartered Institute for IT.