CS 320

Project 2

Caleb Ewer

6/22/2024

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.

To start the project, I looked at each of the requirements and the criteria for each data type and created the data type and restrictions based upon those. As an example, the requirements for all the data types were that it could not be null. With this in mind, I declared the variables as before but, in the constructor, and setter methods I used an if check to make sure the length was not too long and that it did not equal null.

* + 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?

The way that I knew that my JUnit tests were effective was that they passed the tests. The testing criteria I used was based upon the requirements of the data types and the functions. In the previously mentioned data types that had length and could not be null. In the tests of these data types I checked for null arguments and verified that the data types could not be null.

* 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.

One of the ways that I ensured that my code was technically sound was by verifying that the creation of all the methods and the classes was compliable. After that I checked that all of the methods and data types worked the way that they were meant to. An example of this was the add and remove methods for appointment. In my creation of these two classes I then moved to the Junit tests and verified that these worked by creating test cases to check for the creation and deletion of the appointments.

A computer screen shot of a program

Description automatically generated

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

In my code I tried to minimize creating overly complex code. An example of this is the ID. As IO was writing the code for the first milestone, I started to overly think about the ID data type and requirements. The requirements stated that it was to be unique and not updatable. Where it stated unique I was thinking I was going to need to create a random string and a void setter method. After taking a step back I realized that I (and the code) would be better off making the string upon the creation of the appointment, contact, or task and not make it updatable. This can be seen in the contact service class where I imported “java,util.random” and commented it out.

A black rectangle with white text

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.

In my milestones the 2 main testing techniques I used were unit testing and static testing. In static testing the code is examined without execution. This was done by looking over the code and verifying that the conditions set met the conditions required. In unit testing the testing is carried out by testing individual units or components. In the testing we completed we made testing classes where we tested mock objects to simulate the behavior of real objects.

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

There are several different methods of testing that I did not employ in my testing. These are things like compatibility testing, ad-hoc testing, and security testing. In compatibility testing we would have verified that the software would work across other platforms. In our testing we only used eclipse, so we did not verify the testing in other platforms. The process for ad-hoc testing is that there is no formal process or documentation. For our testing we knew what criteria needed to be tested so we knew what process needed to be followed. Lastly the security testing is purely based on the security aspects. In other words, it tests to make sure your code is secure and unbreachable. For our purposes we made the data members private but did not focus on security too much because we are not releasing this to the public.

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

For compatibility testing the types of projects and uses would be things like mobile device compatibility and hardware compatibility. In both there can be varying types of devices with different systems that need to work together and this type of testing can help in that. In ad-hoc testing the best way to use this would be in exploration and last-minute testing. Because there is no formal process the tester can explore what works and doesn’t and adjust. Lastly, the security testing can work in things like data protection and compliance testing. This type of security testing can help when creating software where security is of very high importance.

* 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.

For much of the programming, I understood the datatypes and the programming I was creating but the biggest concern and difficulty I had was using the date datatype. In this type of coding, I was unsure how to use it and unsure how it worked. Because of this I was very scared to make changes to the code and concerned about what I was doing. The way I overcame this was to look up the class and documentation of it to understand how it worked. After doing this research I was able to test the data how I needed and feel more comfortable in using that data.

* + 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.

The two big ways I tried to limit bias were checklists and focusing on the standards. I worked from a checklist I created of the criteria of the program and worked down the checklist marking off when each of the points were met. As far as the standards I wrote the code as best to the security standards that I have been taught even though security was not a major concern here. I did these by using private variables when possible and only accessing them through methods.

* + 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.

As a software engineer maintaining quality is crucial for me for two reasons. One is reliability and stability; in my code I want to have reliable and stabile code as I program. I want this because without maintaining a high level of reliability and stability you are prone to more crashes and data loss that slows the workflow and creates more work. The other is user satisfaction, as a user of different software I understand the importance of keeping the user happy and satisfied with your program. If the user is unhappy, they leave negative reviews which leads to lower revenue and even deters future customers. By cutting corners you open yourself up to more bugs and issues and even the potential of accumulating debt either in workload or financial due to the corrected work. The best ways to avoid technical debt is by reviewing your code and using peers to review it, implementing rigorous testing practices, documentation, and refactoring as needed to keep the code up to date.