Jon Way

CS 320 Project 2

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 made sure to follow the given requirements when developing my program. After reviewing all the feedback on the past three milestones prior to the project, I made sure to adjust and correct aspects of each section of the software to meet the given requirements. One example of a change that I made is that I’m used to creating classes with “getters and setters” for the attributes of that class. I made the first Contact assignment with these getters and setters based on my previous knowledge and at first didn’t align them correctly with the requirements. I created a setter function for the contact ID when the ID was not meant to be updatable. Trying to keep consistency between my other classes, I made getters and setters for Task and Appointment classes as well. I realized after that not every attribute for these classes was supposed to be** **updatable and so I adjusted and removed the unnecessary setter functions.**

* + 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 think my JUnit tests were effective because I had 91.6% coverage for the entire project. The suggested coverage was to have above 80%. The only things that I think I missed were validating input to not be “null” as I generally checked to make sure the length of inputs was correct. Otherwise, I tested almost everything that I could have.**

* 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.

**Overall I think my code is technically sound. I again made sure to cover as much of my code through tests as I could. Looking back, I can already see where I could’ve improved and included more tests to cover all of the code.**

**Something that I did that I think helped ensure technically sound code was in one of my tests for the delete function for the Contact Service class. I don’t know why I didn’t replicate the test exactly or combine the two different tests I did on the other Service classes, which would have been sounder, but I think this one really verified the function. In the ContactServiceTest class I had my delete function test** **create two Contact objects and add them to the array list. The search function I created for these classes looked through the list and returned the array positional value or a value of –1 if the searched ID didn’t exist in the list. I deleted the first object in the list by matching the ID and then checked that the returned value of the originally second object in the list returned the value of 0 as it was now the first (and only) object in the list. On the TaskServiceTest and AppointmentServiceTest classes was search for the ID that was deleted and make sure the search function returned a –1. In hindsight, I should have had both of these tests on these classes to help validate that the delete function worked correctly. I do think these tests help verify that the functions work correctly and prove that the code is sound, but they would have been more thorough if I included both in each test class.**

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

**I did my best to make sure my code was efficient by removing as much unnecessary code as I could. I think my search function is one of those ideas that helped reduce some of the extra code. Because my search function returned the value of an object by matching its ID and returning the positional value, I was able to use one function to find existing and non-existing objects in the lists. I also used this function to store positions of objects to make it easier to update related attributes.**

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

**Since this project didn’t require an interactive interface, the only testing I was able to do was static testing. This required that I wrote out tests to verify that all of my functions were working correctly without running the code. My tests made sure to cover as much of my project as much as I thought I could, which, again, ended up being ~91%. Static testing occurs when the code isn’t running. By writing the JUnit tests, the code never actually functions but we create the tests to simulate the code running and check to make sure the results are what’s expected based on potential inputs.**

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

**In contrast to the previous answer, dynamic testing did not occur because there was no interactive interface for this part of the program. If we had a way to run the code and display the functionality of the program, we could have checked through running the code if it was behaving the way we intended it to. This dynamic testing would allow us to catch errors that we might not have been able to catch just by reading the code or through static testing. While the static testing might return correct results, it's possible that an error could still occur that wouldn’t be caught until the program is actually run.**

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

**Static and dynamic testing are both important and should both be used when developing software. As mentioned above, there are certain things that static testing will catch that dynamic won’t and certain things that dynamic testing will catch that static won’t. Combining these types of testing while developing a program will allow for the most coverage and removal of errors and defects, providing the best software with the fewest errors.**

* 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.

**I did my best to be cautious by trying to follow the requirements for the project as best as I could. As I completed the assignments, I realized I had errors and went and fixed them for the actual project. It’s important to reduce these errors because if they had reached a final product it could lead to software vulnerabilities. As I mentioned above, I had made some of the attributes for some of the classes updatable at first when they weren’t supposed to be. This could have led to software security issues later if they weren’t corrected.**

* + 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 know I wasn’t the most successful in doing this, but I wanted and tried to make my code across each class as similar as possible. For example, the code to update certain attributes in one class should have been nearly identical (other than the variable names) in the other classes. I think this would help other developers be able to read and understand the code more easily. In doing so, this would help remove some bias, because people naturally see their creations as being better than they might actually be because we may be proud of the work we’ve done. This could lead to oversight and errors. I tried to replicate as much code across classes as possible to reduce any errors. If a function worked correctly in one class it should work the same in another class with some small refactoring.**

* + 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.

**This summarizes the things I’ve mentioned above. By making sure to be cautious in code development and limit bias for my own work, I can be disciplined as a software developer. If I cut corners, it could lead to errors in the code which could lead to the product not working correctly or software vulnerabilities. By making sure that I write code to the best of my ability and to write tests that cover as much code as possible with the correct amount of variety for inputs, my programs will be professional and well written which will help avoid the potentially large issues that could come from poorly written code.**