1. **Summary**
   1. **Describe your unit testing approach for each of the three features**
      1. The way that my approach aligned to the software requirements was to create test functions that corresponded to the requirements. A great example would be the Contact class. In its public constructor, I set up thee strings. I created multiple if statements stating that if the contact ID, name, phone number, and or address was empty (NULL) or its length in characters exceeded past 10 (30 characters for the address), the program was to throw a new illegal argument exception telling the user that the input is invalid. Another example would be in Contact service of which in its function, for loop would go over the entire contact list in search of the user’s input ID seeing if it’s unique or not. If it’s unique, then the illegal argument exception is thrown telling the user the Contact ID must be unique. If it’s unique, then the contact is then created and added to the list.
      2. The quality of my Junit testing was perfect, as in none of the functions of any Junit testing files came back with any errors nor failures. There’s a colored bar that would show green after the Junit test to make sure that my test cases worked the way that it was supposed to work. If the bar came back red, it would mean that the functions came back with testing errors or failures insinuating that I haven’t set up my code to run correctly as I had wanted to reinforce that the system’s requirements are met.
   2. **Describe your experience writing the Junit tests**
      1. The way that I made sure my code was technically sound by using Assertions.assertThrows(IllegalArgumentException.class, () -> {} for all of my testing classes as a way to test how my main classes respond when a user breaks the input rules for the software. Take for instance from my ContactServiceTest.java file. I would call the addContact function from the main class and have the incorrect input at play (whether null or exceeding character limit). If the Junit tester comes back with no errors in my testing, then that means my codes worked exactly as I had anticipated to help meet the requirements for the software.
      2. The way I made sure my code was efficient was paying attention to the results of the JUnit testing to see if any potential errors or failures for the testing functions that arose. If not, I know both my main and testing files are working just fine. From my contact service test, I use both assertTrue() and Assertions.assertThrows(IllegalArgumentException.class, () -> {} for all of my other testing classes using the same method helped make the testing process much simple and further assure that my codes are efficient.
2. **Reflection**
   1. **Testing Techniques**
      1. The software testing technique I used was Junit. It was the one I relied on to help verify whether I was on the right track with my codes or not. It gave me detailed feedback about the outcome of the test so that I can go back to certain locations of the code, whether test or main, and give it further touch up to fix the mistakes within my code.
      2. The software testing technique that I didn’t use was Static testing. The way that is described is that Static Testing checks the defects that lie within the software without the actual need to test the code. I was testing my codes all the time as I was developing the software hence reinforce the unnecessity of static testing.
      3. Since static testing is used to pinpoint all the errors located in the code, the testing method can be used by those that want to help others that are uncertain if their code contains errors or not. As far as Junit testing in practical situations can be used in agile projects that require testing the code before moving on to the next phase of the project.
   2. **Mindset**
      1. This is the first time I’ve had experience using Junit testing for Java, so I had some time learning the ropes of how the Junit test works and what to do to get it in to working order. That was my caution. It was important to appreciate the complexity of the code because now that I know how to write it, I can do so without any trouble at all. I put in the effort to make sure the requirements fit into the code and not have any errors along the way.
      2. From what I can recall, there were no biases within my code. Before I move on to the next part of the project, I want to make sure that the previous part is all well and good.
      3. It is important not to cut corners because doing so will only further the damage and costs that come from doing so. It is important to take extra precautions such as testing to make sure that everything is working the way it is supposed to with no errors in sight. The way I will avoid technical debt is through patience and staying determined by writing the utmost quality code possible.

Sources Used

GeeksforGeeks. (2024, September 19). *Introduction of JUnit*. GeeksforGeeks. https://www.geeksforgeeks.org/advance-java/introduction-of-junit/

GeeksforGeeks. (2019, May 13). *Static Testing Software Testing*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering/software-testing-static-testing/

‌