Cooper Brien

CS320

Module 7 Journal

For the task and contact service assignments I tried my best to align my testing approach to the software requirements. I conducted different tests to determine whether the code followed the requirements supplied by the assignment. For example, in the contact class, the name of the contact could not be null, nor could it be more than ten characters. I had set up two different tests to ensure that input a name with more than ten characters or inputting null would return an illegal argument exception, which they did.

I think that my Junit tests were quite comprehensive in covering every requirement. I conducted separate tests for individual requirements. When testing to see if the contact service could make a contact, I had to create a test that attempted to create a contact and then attempt to return that contact based on the given ID. I then conducted multiple separate tests to ensure that each field of the contact could be updated as needed. I tried my best to not group together issues into a single test, but rather, separated them out as much as possible.

How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.

My code is technically sound because it is very well organized and has clear names given for each function to provide readability and understanding to users. My code also accomplishes the goals outlined by each assignment.

I know my code was efficient because I was able to create a function within the Contact Service that was able to return a contact if it was found with the correct ID. This allowed me to avoid repetition in my code, specifically in the areas where the code would need to update a specific contact.

For the past few milestones, I employed unit tests on for each of the services and classes that I developed. Using JUnit test cases, I was able to systematically test each feature of my code to make sure that every part of it was working properly. I had a different test for each requirement to ensure that I was following the requirements set by the assignment. For the appointment service in this module, I tested to see if the service could add an appointment by calling the function to add an appointment within the test, and then checking to see if the appointment existed within the list. For each of the data fields in the appointment class, I ran separate tests to make sure that the fields followed the guidelines set by the assignment.

One testing technique that I didn’t use was non-functional testing. Non-functional testing includes testing things like usability, reliability, performance, and efficiency. Because there was no app interface that was developed, there was no way to test the usability of the app. Once the app interface begins to be developed, that is when this form of testing would come into play.

Unit testing is very important for testing specific functions of an application, especially one that does not have an interface built yet. The Junit tests that I employed for the past few milestones ensured that my code was functioning properly even without an interface. Non-functional testing is used to test how user-friendly and efficient the app is. This type of testing would be used much more heavily during the development of the interface. While the app may be technically functional, the app may prove to be difficult for users to use and that is where non-functional testing can come in to guide the developers towards a more user-friendly interface.

While working on this project, I had to employ caution when testing the software. It is important to use caution and take it slow when testing the software because you need to make sure that the tests are written correctly. There were a few times when I was testing the software for this project that my tests were producing failures. I went through my code over and over again to try and solve the problem, only to realize that the tests were written incorrectly. The way it was previously written tested to see if there would not be an exception if the ID was null, while I meant to test to see if there would be an exception if the entered ID was null. On the other hand, ,if the code is the problem, it is important to understand the relationships that the block of code has with other blocks of code. Changing one block of code to succeed in the test may have effects on other parts of the code so it is important to understand what the code does and where else it is used within the overall project.

When testing your own code, it is important to stay unbiased. It is important to make sure that you are thoroughly testing your code. There is an incentive to having the code pass all the tests because that would lead to less work for the programmer. In another sense, there are benefits to having someone else look at your code with a fresh pair of eyes. This can help catch bugs and other issues that the programmer may not have caught if they were testing on their own. The bias we have.

It is important to stay disciplined when working on the project. Cutting corners when testing or writing code can lead to a reduction in the quality of the product, which can frustrate users and lose potential customers. This can also cost the company time and money to fix any errors that may have been caused by cutting corners. In the field, I plan to make sure that my code is thoroughly tested, and to check with other members of my team to see what they think of the work I’ve done before continuing on.