Jordan Stelivan

CS-320

Professor Wilson

12/12/2021

Summary and Reflection

Up until this project, I’ve had no interaction with JUnit Testing. I have done some static testing in the previous semester for my System Securities course. That being said, I believe that I did a good job implementing JUnit testing into my code. The code I’ve produced for both assignments meets the requirements listed. You can see this with my ‘addContact’ and ‘addTask’ functions especially. Both of these functions were required to make certain each contact/task had a unique ID. Using the JUnit tests I set up for each file you can see that it tests for duplicate IDs being added to the contact/task list data storage. I think the coverage percentage for the contact service and task service was also well done as I’ve tested each and every function, including the branches within the function. The code I’ve produced is technically sound and this can be seen by the lack of errors within the code. I’ve made sure that any libraries called are used, and that each class interacts with its parent class correctly. As for efficiency, I originally wrote the ‘addContact’ function very differently. While this did work, it didn’t account for unique IDs and was not particularly well written. However, the final lines that I’ve ended up writing seem to be much better at achieving the required goal of the function.

The main testing technique that I had used for each milestone was the JUnit testing. For each milestone I set up several use case scenarios in order to test the capabilities of the code. For example, in each milestone, I set up a test that consisted of creating a new instance of the type class. This would be of the classes ‘contact’, ‘task’, or ‘appointment’ depending on the code being tested at the time. The test would add information to fill out the class parameters and then run a check to make sure that the saved class item was saved properly with all of the information attached to it. I ran checks for variables like ‘id’, ‘name’, and ‘description’ for each of the different classes. What’s excellente about using JUnit testing is that I can check for errors when running the code without actually needing to run the code each and every time. These tests were also crucial in pointing out various errors that I was able to correct.

JUnit testing is a type of dynamic testing, and that was the primary software testing technique that I had used. I say this because I didn’t really use any static testing over the course of each milestone. Static testing focuses more on prevention within the code. Pointing out dependency issues and error codes to better prevent any long term issues within the application. However, I haven’t made much use of this as of now because I’m not really using any libraries and am not all too concerned with dependency issues at this time.

For other projects, and maybe in the future development of this current project, I would look to better make use of both dynamic and static testing. Both of these testing techniques, when used together, make very powerful tools that allow developers to find various types of issues and errors within their software’s code.

All in all, I’d say I’m rather happy with how both assignments have turned out. Especially considering my weakness when using Java as a coding language. I’ve not had a ton of experience coding within Java. I ran into some issues because of this, but was able to overcome them given a bit of time.