Asa Brown

SNHU | Project Two

CS-320

December 13, 2024

Project Two JUnit Testing Reflection

For this course, we were given the task of creating three different features for a complete program package. The package included services for Contacts, Tasks and Appointments. For each of these features we were also required to test the functionality of the program and hopefully achieve the desired results through this testing. My approach to testing these three features was to test the class and its own functions like getters and setters and to also ensure that the initialization of the objects was correct and would not produce an error under certain circumstances. It was also crucial to test the Service and its functionality itself, which included adding, updating/changing information for an object within the collection of objects, or deletion of an object from the collection of objects. Each feature was tested utilizing JUnit testing within the Eclipse IDE, which is an open-source framework that allows Java developers to write and run tests to find bugs in the code.

Prior to this course, my experience writing JUnit tests was non-existent and through these projects I have come to a greater understanding for how to make tests while also making them technically sound and efficient. To ensure my code was technically sound, I used assertions to verify the expected output of various functions. In the TaskServiceTest code, the following line of code is meant to verify that an exception is thrown when attempting to update a task that doesn’t exist. An example line of code can be seen below:

Exception exception = assertThrows(IllegalArgumentException.class, () -> { taskService.updateTaskName("1", "This should fail"); });

Efficiency was also a significant consideration since each of these services can involve working with a large amount of data. To keep things efficient, I utilized hashmaps to represent a collection of various objects for each service. The code found within each of my tests performed as expected with no errors or exceptions thrown in cases where inputs were valid and failures occurring when invalid inputs were submitted.

The testing techniques I used for testing were manual, functional, and unit testing which aimed to test different aspects of the program. Since the testing was done by myself, I utilized manual testing and sought to look for potential errors in the code’s syntax. Functional testing was done to ensure the program met the functional requirements. In the last assignment, the Appointment Service had a list of requirements needed for the program to work as intended. Each appointment needed to have an ID, a date, and a description of the appointment. There also needed to be a means of adding, updating, and deleting appointment information. Additionally, Unit testing was done to test individual aspects of the program. This includes the set functions for setting the Appointment Date and the description, initialization for the Appointment objects, and testing the behavior of the AppointmentService java class.

Of the types of testing, I wasn’t actively trying to incorporate or didn’t include, performance and security testing come to mind. The program, as is, doesn’t accept user input; however, if they were to input a day that is less than 1 or greater than 31, it could cause some issues with the Calendar class and how it processes information. Additionally, while using a hash map is very efficient for organizing large amounts of information, other portions of the program could be more time-consuming or more demanding of a computer’s hardware.

To work on these projects, it was important that I understood what the requirements were for the program requirements and that I implemented functions properly to ensure the program met the demands of the consumer/client. Consequentially, I also gained an appreciation for the complexity of the programs and the tests that needed to be conducted to make sure everything worked as intended. The critical point of feedback I received was for the Contact program, where I was instructed to have the service update/manipulate the Contact objects by using the ID. Prior to that feedback my program wouldn’t have been as efficient in retrieving data, let alone manipulating the information within it.

During the coding process, I also looked to remain open-minded in my approach to solving problems and find valuable solutions that were appropriate. Like for Appointment and AppointmentService files, the functions within each class serve a different enough purpose that they required thinking outside of the box to test the functionality through JUnit testing. Once I found a suitable solution though, I relied on its application across my other programs, which means I certainly had a bias towards using that solution for test which works out well for the purposes of these program, but that’s certainly not to discount the possibility that a more efficient and effective solution exists. By pursuing those alternative and possibly more efficient solutions, a software engineer improves and solidifies their commitment to code quality, a cornerstone of success in creating reliable and trustworthy code and reputation.