Christopher O’Dell

SNHU – CS320

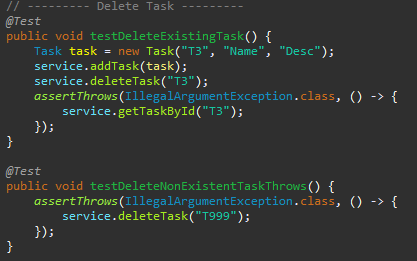
Module 7

Project Two

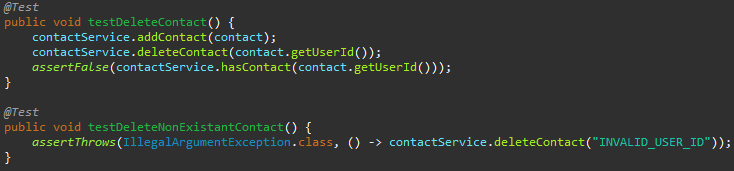
For this project, I implemented a comprehensive suite of JUnit tests for the ContactService, TaskService, and AppointmentService classes. Each test was strategically designed to validate specific functionalities such as adding, deleting, and updating records based on unique identifiers, with a direct alignment to the software requirements outlined for each service and its corresponding data model. For the ContactService, test cases verified the ability to add contacts with unique IDs, delete them by ID, and update individual fields like first name, last name, phone number, and address. The TaskService was similarly tested to ensure tasks could be added with unique identifiers, modified through name or description updates, and deleted accurately. The AppointmentService tests focused not only on basic CRUD operations but also on validating business logic such as disallowing appointments scheduled in the past and enforcing proper description length.

The quality of my JUnit tests can be assessed across several dimensions. Test coverage was a key focus, and I ensured that each software requirement was addressed by at least one test case, resulting in 100% code coverage. Assertions were used to verify expected outcomes precisely, ensuring the functionality met its design intent. I prioritized maintainability by writing modular and readable test cases, each accompanied by comments to clarify the test’s objective and expected results. Setup and teardown processes were handled using JUnit’s @BeforeEach annotation to guarantee a clean state for every test, thus preserving test independence and preventing cross-contamination between cases.

Efficiency was another principle I adhered to throughout the development of the test suite. I avoided redundancy and kept each test method tightly focused on a single responsibility. For instance, in the TaskServiceTest, the method that tests task deletion and non-existent task deletion utilizes their own specific method:

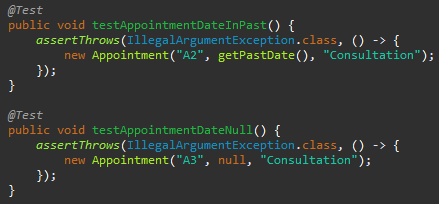


This approach made the tests easy to understand and maintain while ensuring they remained singular in their responsibility. The same standard of efficiency is reflected in the ContactServiceTest and AppointmentServiceTest classes, where test cases were constructed to provide meaningful coverage without unnecessary complexity.



Two key testing techniques were used in this project: unit testing and boundary value analysis. Unit testing served as the backbone of the testing process, allowing for the verification of isolated components early in development and facilitating iterative improvements. Boundary value analysis was implicitly applied to test how the system handles input at the edges of valid ranges, such as field length constraints for names and descriptions. These techniques provided a high degree of confidence in the individual components of the software. However, integration testing and regression testing were not formally applied. Integration testing, which validates interactions between system components, would be useful in more complex systems with interdependent modules. Regression testing, which helps ensure that new code changes do not break existing functionality, could be incorporated in future iterations to safeguard against unintended side effects.

In terms of professional conduct and mindset, I approached the testing process with caution and objectivity. Recognizing the complexity and interconnectedness of the code, I made a deliberate effort to test not only basic functionality but also edge cases and error conditions. For example, when validating appointment scheduling, I tested against invalid past dates and null dates to ensure the logic handled real-world constraints effectively:



I also minimized bias by adhering strictly to the requirements and evaluating the software based on expected outcomes rather than assumptions or personal preferences.

Commitment to quality was foundational to my approach. I understand that cutting corners in testing can lead to the accumulation of technical debt, which in turn increases maintenance costs and reduces system reliability. Therefore, I adhered to clean coding practices, structured my tests clearly, and refactored as needed to maintain clarity and performance. Through thorough validation efforts and methodical testing in my testing processes, I was able to contribute to the overall robustness and maintainability of the system.

In summary, my work on the JUnit test suites for the ContactService, TaskService, and AppointmentService demonstrates a comprehensive, technically sound, and quality-driven approach to software testing. Through high test coverage, effective use of testing techniques, and adherence to professional best practices, I was able to ensure that each component met its requirements and behaved reliably under various conditions.