Assignment 7-2

Jarrod Thomisee

Jarrod.thomisee@snhu.edu

2023-08-13

**Summary and Reflections Report for Grand Strand Systems**

**Summary**

**Unit Testing Approach for the Three Features:**

For the contact, task, and appointment services, I adopted a white-box testing approach. This method allowed me to design test cases based on the internal workings of the application, ensuring that each logic path was thoroughly tested.

For example, in the ContactService, I had to ensure that each contact had a unique ID. The code snippet below illustrates this:



To test this, I used JUnit's assertThrows method to ensure that trying to add a contact with an existing ID would throw the expected exception.



**Alignment with Software Requirements:**

My testing approach was meticulously aligned with the software requirements. For instance, the requirement that the contact ID should be unique and not null was tested using assertions.



Similarly, for the appointment service, I ensured that the date provided was not in the past by comparing it with the current date.

**Quality of JUnit Tests:**

The JUnit tests I wrote achieved a coverage of over 80%. This high percentage indicates that the majority of the code paths were executed during testing. For instance, in the AppointmentTest, I ensured that the date provided for an appointment was not in the past:



**Experience Writing JUnit Tests:**

Writing the JUnit tests was both challenging and rewarding. It required a deep understanding of the application's logic and a meticulous approach to ensure that every requirement was tested. For instance, in the TaskServiceTest, I had to ensure that tasks could be added, updated, and deleted correctly. The tests also had to handle edge cases, such as adding a task with an existing ID.

**Technical Soundness and Efficiency:**

Ensuring the technical soundness of the code is paramount in software development. It's not just about making the code work; it's about making it work correctly, consistently, and efficiently. Here's a deeper dive into how I ensured these aspects:

**Assertions for Technical Soundness:**

Assertions play a pivotal role in validating the correctness of the code. They act as checkpoints, ensuring that the code behaves as expected in various scenarios. By using assertions, I could confidently verify that the logic within the services was functioning correctly.

For instance, in the TaskServiceTest, the use of Assertions.assertThrows was instrumental. This specific assertion ensured that the system would throw an exception when a task with a duplicate ID was added. This is a crucial check because allowing duplicate IDs would compromise the integrity of the data and could lead to significant issues down the line, such as data overwrites or retrieval errors.



**Efficiency Considerations:**

Efficiency in code is about ensuring that the application runs optimally, minimizing resource usage and maximizing speed. One of the primary ways I ensured efficiency was by being judicious in the use of external libraries. While libraries can provide quick solutions, they often come with overheads, and not all functions provided by a library might be used in the project. By minimizing their use, I ensured that the application was not bogged down by unnecessary code.

Another key aspect was avoiding nested loops, especially when dealing with data structures like maps or lists. Nested loops can exponentially increase the time complexity of operations, leading to performance bottlenecks, especially with large datasets. By structuring the code to minimize such constructs, I ensured that the application's performance remained at its peak.

In the ContactService and other services, the use of HashMaps ensured constant-time complexity for basic operations like add, delete, and retrieve. This choice of data structure played a significant role in maintaining the efficiency of the services.

**Reflection**

**Testing Techniques Employed:**

In this project, I primarily used white-box and boundary testing. White-box testing ensured that all logic paths were covered, while boundary testing ensured that edge cases, especially for input validation, were addressed.

**Other Software Testing Techniques:**

While I focused on white-box and boundary testing, other techniques like black-box testing, stress testing, and regression testing were not employed. Black-box testing involves testing the application without any knowledge of its internal workings, stress testing checks the application's performance under heavy load, and regression testing ensures that new code changes don't break existing functionality.

**Practical Uses of Testing Techniques:**

**White-box Testing**: Best for applications where the internal logic is complex and needs thorough testing, like financial software.

**Black-box Testing**: Useful for end-to-end testing of applications, especially when the tester is not familiar with the internal logic.

**Stress Testing**: Essential for applications expected to handle a large number of users, like e-commerce websites.

**Mindset Adopted**:

While working on this project, I adopted a cautious and detail-oriented mindset. Understanding the complexity and interrelationships in the code was crucial. For instance, ensuring that the TaskService didn't interfere with the ContactService required a deep understanding of how they both interacted with the underlying data structures.

Limiting bias was challenging but essential. As developers, we often become attached to our code, which can blind us to its flaws. By adopting the mindset of a tester, I could view the code more objectively. If I were testing my own code, it would be easy to overlook certain flaws due to inherent biases.

**Commitment to Quality:**

In software engineering, quality is paramount. Cutting corners might offer short-term gains but can lead to technical debt, where future changes become more challenging and time-consuming. By ensuring that every line of code is tested and every requirement is met, we can avoid this debt and ensure that the software remains maintainable and robust in the long run. As I progress in my career, I plan to always prioritize quality, continually update my knowledge, and employ the best testing practices available.

In conclusion, this project was a valuable learning experience. It reinforced the importance of thorough testing and the need to approach software development with a meticulous and unbiased mindset. As I continue in the field, I'll carry these lessons with me, always striving for excellence and the highest quality in every project I undertake.