Mary Peterson

8/15/2024

CS320

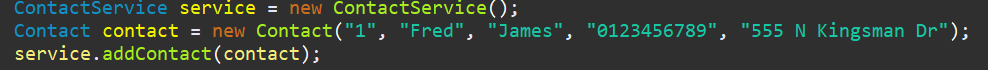
**Summary and Reflections Report**

As a software engineer at Grand Strand Systems, I recently completed a mobile application project for a customer, focusing on developing and testing back-end services for contact, task, and appointment functionalities. This report summarizes my unit testing approach, experiences writing Junit tests, and the overall quality of these tests, along with reflections on the testing techniques and mindset adopted during the project.

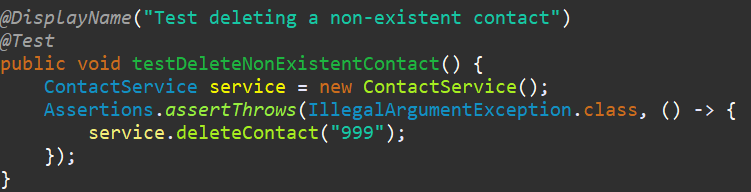
**Unit Testing Approach**

For each of the three features—Contact, Task, and Appointment services—my unit testing approach involved creating instances of the services classes, adding entities, and verifying their addition by checking retrieval. Additionally, I tested the deletion of entities to ensure proper removal from the services. This approach was in line with the requirements of CRUD operations (Create, Read, Update, Delete).

The alignment of my approach with the software requirements is evident in the way I structured the tests. For instance, in ‘ContactServiceTest.java’, the ‘testAddContact’ and ‘testDeleteContact’ methods directly support the requirement of managing contact information. In ‘testAddContact’, I created a new contact, added it to the service, and used ‘assertEquals’ to verify that the contact was correctly added:



Similarly, in ‘testDeleteNonExistentContact’ at a contact could be deleted and verified its removal with ‘delete’:



The same approach was applied to the Task and Appointment services. In ‘TaskServiceTest.java, I assessed adding and deleting tasks:

A computer screen shot of a program code

Description automatically generatedA computer screen shot of a program code

Description automatically generated

In ‘AppointmentServiceTest.java’, I ensured the functionality for adding and deleting appointments:

A screen shot of a computer

Description automatically generatedA screen shot of a computer program

Description automatically generated

**Quality of JUnit Tests**

The overall quality of my JUnit tests can be defended by their thorough coverage of critical functionalities. By testing the addition and deletion of operations, I ensured that the core functionalities of each service were working as expected. However, to further improve coverage and robustness, it would be beneficial to include more edge cases and exception handling in future tests.

**Experience Writing JUnit Tests**

Writing JUnit tests was a valuable experience that honed my skills in ensuring code soundness and efficiency. I ensured that the code was technically sound by using assertions to verify expected outcomes. For example, in ‘TaskTest.java’, I used:

A computer screen shot of text

Description automatically generated

These assertions helped confirm that the methods were functioning correctly.

To maintain code efficiency, I avoided redundancy by setting up the service objects before each test and clearing the tasks after each test. This setup and teardown method ensured that each test started with a clean slate:

A black background with blue and orange text

Description automatically generatedA blue and white rectangle with white text

Description automatically generated

This practice not only improved efficiency but also made the tests more reliable.

**Reflection**

Testing Techniques

The primary testing technique employed in this project was unit testing, characterized by testing individual units of code in isolation and using a mock environment to simulate interactions. This technique was crucial for early detection of issues within individual units of code. Other testing techniques not used in this project include integration testing, system testing, and acceptance testing. Integration testing ensures that different modules interact correctly, preventing integration issues. System testing validates the overall functionality and performance of the system, while acceptance testing confirms that the system meets user requirements and is ready for deployment.

Each of these techniques has specific practical uses. Unit testing is essential for detecting issues early in the development process. Integration testing is vital for ensuring that various parts of the application work together seamlessly. System testing is necessary for verifying that the complete system functions as intended, and acceptance testing is critical for ensuring that the software meets the end users’ needs and expectations.

Mindset

Throughout this project, I adopted a mindset of caution and thoroughness. As a software tester, I understood the importance of appreciating the complexity and interrelationships of the code I was testing. This understanding helped prevent issues such as data inconsistency and unhandled exceptions. For example, I carefully tested the deletion functionality to ensure that removing an entity did not leave any orphaned references or caused unintended side effects.

To limit bias in my code review, I wrote comprehensive tests covering different scenarios and edge cases. This approach helped ensure that the test was objective and thorough. Bias can be a concern when developers test their code, as they might overlook flaws due to familiarity. By writing a detailed test, I aimed to mitigate this risk and ensure a more objective evaluation of the code.

Being disciplined in my commitment to quality is crucial as a software engineering professional. Cutting corners in writing or testing code can lead to technical debt, making the codebase difficult to maintain and prone to errors. For example, neglecting to write tests for edge cases or not refactoring code can result in hidden issues that are costly to fix later. To avoid technical debt, I plan to follow best practices, such as writing comprehensive tests, refactoring code regularly, and documenting my work thoroughly. This disciplined approach will help ensure that the code remains maintainable and dependable overall.

In summary, this project provided valuable insights into effective unit testing approaches, the importance of thorough testing techniques, and the mindset required for high-quality software development. By adopting best practices and maintaining a disciplined approach, I ensured the delivery of robust and reliable software. This experience has reinforced the importance of thorough testing and the need for a cautious and objective mindset in software development.