CS 320

Rikki Xaysanasy

Professor Luo

4/24/24

Project 2 Reflection

The unit testing approaches for the implementation of ContactService was to be able to add, update, and delete contacts. An example of this would be testing the addition of a contact to ensure whether its ID was unique, and all the details are stored correctly. TaskService unit testing approaches were like ContactService, with the addition of tasks, updating tasks, and removing tasks. An example would be verifying the updating test would adhere to task descriptions. AppointmentService were similar as ContactService and TaskService, this included the addition of an appointment, being able to update its details, and then deleting the appointment when necessary while making sure the date is correctly implemented.

The overall quality between the JUnit testing was highly effective with the coverage percentage being an average of 76%. This coverage ensured that the right IllegalArgumentExceptions were thrown when contested with adding, updating, or deleting within the classes.

Writing these tests involved following the guidelines from the rubric that was given, this explained how each function’s behavior was expected to be. Ensuring each function behaves as expected under different scenarios was the focus. An example of using assertions like assertNotNull(service.getContact("1")) after adding a contact ensures that the add function works as specified. This was also efficient by directly testing each method's expected output. For instance, assertEquals("Ricky", updatedContact.getFirstName()) confirms the update functionality.

The testing technique that I employed was unit testing. This technique focuses on verifying individual methods or functions. The tests for all three classes checked individual methods for correctness which included adding, updating, and deleting in their respective fields. The technique not implemented was integration testing. This method focuses on combining units or components and assessing them as a group to expose faults in the interactions between these integrated units.

While unit tests are excellent for checking the functionality of individual components this makes it very practical with its early bug detections. A drawback would be that they cannot check how parts work together or how the system performs under load. This goes with the technique that I did not implement, which is integration testing. Integration testing identifies problems when units interact, such as data format not aligning, incorrect handling of valid input, and failure to manage error conditions between units. This makes integration testing more complex to set up and execute than unit testing. For different projects such as implementing an embedded system, unit testing would ensure that each module of the device's firmware performs its intended function correctly. While integration testing would be vital for checking the various parts of the system and making sure they are working together.

I took great caution in recognizing interdependencies, such as how deleting a contact might impact associated appointments or tasks. To eliminate bias when coding, it helps to assume that nothing is 100% bug free. I actively looked for potential failures even in straightforward code segments. As a developer, it is also a clever idea to let others test it and provide feedback, because it would be dangerous just believing in myself to be able to have 100% bug free code all the time.

The importance of maintaining discipline in code quality cannot be expressed enough. Ambitious standards and ethics prevent bugs, improve usability, and extend software’s life. Ensuring that keeping tests up to date is vital practice. For example, ensuring new functionalities are accompanied by comprehensive tests avoids future liabilities.

Sources

Thompson, G. (n.d.). Chapter 3. In *Software Testing : An ISTQB-BCS Certified Tester Foundation guide - 4th edition*. essay.

Hambling, B. (n.d.). Chapter 4. In *Software Testing : An ISTQB-BCS Certified Tester Foundation guide - 4th edition*. essay.

Morgan, P. (n.d.). Chapter 1. In *Software Testing : An ISTQB-BCS Certified Tester Foundation guide - 4th edition*. essay.