Jerod Goode

CS320 – Journal Entry: Contact Service Review

Professor Gudivada

05/05/2025

The Junit tests that I conducted checked for the creation, deletion, and updating to a contacts first name, last name, phone number, and address. Each contact had unique ID that was not able to be updated or manipulated. I believe that my Juint testing aligned with the software requirements and that there was sufficient coverage in ensuring the software functioned as it should.

I created two Junit test cases to test the two main classes, Contact.java and ContactService.java. I wanted to avoid duplicate logic and have purpose-driven test cases. I kept my assertions minimal to help keep my code clear and understandable, helping me avoid unnecessary code. The Junit tests were to check if the classes created, deleted, or updated the contact’s first name, last name, phone number, and address. For example, I made a variety of tests, such as **testAddDuplicateContactThrowException()** that would throw an error if the user tried making a new contact with an ID that existed.

**assertThrows(IllegalArgumentException.class, () -> {  
 contactService.addContact(“firstName”, “lastName”, “0808”, “address”, “phone”)  
});**

I did encounter an error in my code that I didn’t realize until after I received feedback. The feedback I received was not using **private final String Id**. However, I didn’t include a setter method for the ID string, which prevents external code from modifying it—but since it’s not declared as **final**, the ID still isn’t truly immutable. Declaring the ID string as **private final** is a safer approach that helps enforce data integrity and prevents accidental changes.

I also ran tests to throw errors if any of the input fields are left null or empty. For example:

**assertThrows(IllegalArgumentException.class () -> {  
 contactService.deleteContact(null);  
});**

So, if the user tried to enter in nothing for the input for first name, last name, address, or phone number, the program would throw an error to the user saying that the input is invalid. Additionally, to keep code efficient and avoid redundant code, I used **@BeforeEach** to set up a new instance of ContactService before each test.

This project has shown me just how important key software testing principles are and why they are valuable to the performance of program as well as the security of a company. It’s important to make instance variables immutable when their values should never change, as failing to do so can lead to potential issues and unintended side effects. Conducting these test cases gave me a deeper understanding of how to build and test software so that it not only meets functional requirements but also performs reliably, handles edge cases, and maintains long-term quality.