CS 320 Project Two by Justin Leger

My unit testing approach for each of the three features; the testing approaches aligned with the software requirements as the contact class requirements and contact service requirements for example. The contact object shall have a required unique contact ID string that cannot be longer than 10 characters. The contact ID shall not be null and shall not be updatable. Additionally, there shall be a requirement for firstName String field, lastName, phone, and address fields. In Contact.java, a constructor would check that these fields are null or have certain length. Then in ContactTest, I’d use Assertions.assertThrows() to check that a field is null or has too long a length.

The overall quality of Junit tests can be determined by coverage. AppointmentServiceTest.java has 100% coverage (all green highlights), AppointmentTest has 76.6%, ContactServiceTest.java has 100%, ContactTest 91.4%, TaskServiceTest 100%, and TastTest 72.5%. Sometimes I’d write the test for testIdTooLong and two lines would be red.

My experience writing the Junit tests was complicated and difficult because it was very new to me. My code is technically sound because it meets with the requirements such as testDescNull() from AppointmentTest.java

Assertions.*assertThrows*(IllegalArgumentException.class, () -> {

new Appointment("123456789", date, null);

});

Which tests one of the fields as null. My code is efficient because I’d initialize important things up top and then have less things to write on the tests. For example, before all the tests I’d write:

*@SuppressWarnings*("deprecation")

Date date = new Date(2025, Calendar.***MARCH***, 2);

Appointment appt = new Appointment("123456789", date, "Buy cat food");

Upon reflection, the software testing techniques that I used was Junit testing which is an open-source unit testing framework for Java. Junit verifies that individual units of code behave as expected. It allows developers to write test cases for individual functionalities and check to see if actual outcomes match the expected results.

Other software testing techniques that I did not use for the project are Static Testing which is a software testing method that can examine a program without it being executed, which helps to find flaws early on in development.

The practical uses and implications for each of the testing techniques are automated tests allowing for repeated execution, early bug detection, simple syntax, support for various tests, ensures that code works correctly, and reduced cost.

The mindset that I adopted while working on this project was to develop all of the features and meet all of the requirements, starting with Contact, then developing ContactTest, then ContactService, then ContactServiceTest. I applied caution with testing techniques which are new to me. I did not consider the complexity of the code and was taken aback when it was not as simple as I had expected.

The way that I tried to limit bias in reviewing the code is to assume that almost all the code was wrong, and I had a lot of work to do. The code is written for a client and expected to be written as quickly and to the highest quality standards possible, so I asked for plenty of help.

The importance of being disciplined in commitment to quality as a software engineering professional is that clients are dependent on the code being written quickly and to quality standards. Not cutting corners provides better user experience and reputation and saves time and money. To avoid technical debt, I want to create proper documentation, use proper Agile practices, have a system of open communication, and regularly maintain the code.

GeeksforGeeks. (2024, September 19). *Introduction of JUnit*. GeeksforGeeks. <https://www.geeksforgeeks.org/introduction-of-junit/>

*What is Static Testing?* (n.d.). SearchSoftwareQuality. <https://www.techtarget.com/searchsoftwarequality/definition/static-testing>

Watson, M. (2021, June 11). *How to Avoid Technical Debt*. Full Scale. https://fullscale.io/blog/avoid-technical-debt/

‌

‌

‌