Jad Alrehaoui

SNHU

JUnit testing

Summary and reflection report

In the Software requirements, we had to implement 3 classes and their services

* Contact class & Contact Service
* Appointment class & Appointment Service
* Task class & Task Service

Each of these classes has a constructor, that validates each field as it was required in the software requirement sheet. For example the description field in the Task class has to be less than 50 characters long and is required. The constructor of the class validates this input and throws an exception if it was not satisfied.

Each class in my solution is tested from different perspective.

Text

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated with medium confidence

Figure - Test classes

In figure 1, we can see that I have tests as much as I have functional classes, which means all the classes I am implementing have their respective tests.

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

Figure - Testing date

In figure 2, we can see my Appointment Test, which checks for required field, field length, setters , getters and invalid date. The Test Invalid Data function checks if an exception is thrown when a date is passed being not in the future.

Text

Description automatically generated

Figure - Date Generator

Talking about effectiveness and efficiency, I believe that the date generator function was effective when testing my code as it was used many times in the tests as I pass a string to it and it returns a well formatted date to be checked later. Also this function was efficient, because it saved me time writing it all over again and again.

In addition to that, there are some way that I can improve efficiency of the code by joining all similar tests together in a more broad test class where all the classes get tested through it, and the more specific test be kept specific in the respective test class.

For the most of my testing, I employed the black box and the white box testing techniques, as I checked for valid input and tested that it works, and checked for invalid input and tested that an exception is thrown. It was more of a static test as the application is not being executed in whole.

The unit testing technique was also employed as the solution was divided to the function and the function is being tested.

In this project I did not implement:

* Grey Box Testing: where the security of the application is being tested, and a mix of white and black box testing.
* Regression Testing: where we test the software to make sure that an update did not break existing functionality.

Working on this project made me see the code and the software in a different perspective, as I code I am thinking about the tests, what should work and what should not. It made the code more directed towards the requirement instead of it being open ended.

As I am the one coding and testing the software, my tests can produce bias results. To limit this bias, I tried my best to focus on the requirement and how they are described as I am testing the software that I coded.

Being committed to quality as a software engineer is like serving a dish in a restaurant with a smile and with love. It is the standard a developer should have when working on a project. Afterall when quality code is produced, time and money are saved later. It is important not to cut corners when writing code and tests in order to make sure the software works efficiently and effectively when it’s deployed and live. As a tester or a developer it is good to see the software from a different perspective, and consider inputs and states of the software and where it is being used. This way we can produce a list of tests customized to the software itself and be more efficient testing it.

**References:**

The QA Lead and Boog, J. (2023) *9 types of software testing in software engineering*, *The QA Lead*. The QA Lead. Available at: https://theqalead.com/test-management/types-of-software-testing/ (Accessed: February 15, 2023).