Mathew Newton

CS-320

12/8/2021

Project 7-2

**Summary:**

Unit testing is important when it comes to building software resolutions. It helps reduce the risk of running into program errors and improving quality of the code by quickly finding defects and addressing them. The approach I used with unit testing for the three features were similar as I made it to align with the software requirements. An example was the Contact service which had some specific requirements like the Contact object shall require a unique contact ID string and it cannot be longer than 10 characters and it can’t be null or updatable. They wanted the contact object to require a firstName and lastName that cannot be longer than 10 characters and shall not be null, they also wanted a phone field which is a string and must be exactly 10 digits and should not be null. Lastly, the contact class should also have an address field which should not be longer than 30 characters ad should not be null.

The Contact object also has a service that does some crud (Create, Read, Update Delete) operations with the contact object. All of these were specified requirements for one of the three features, and all other features had similar requirements so the software must be implemented with those requirements to pass the unit test. Overall, the quality of my unit test was as good as I can get it to be. I tried to test each of the fields and the operation to check the validity of the test cases to ensure it makes proper use cases.

My experience with the Junit test was interestingly fun. I still have much to learn, and I have tried to implement what I have already learned to ensure the code was technically sound.

Text

Description automatically generated

The line of codes above was checking the validity of the parameters for the contact service test, and I also created a test case that was checking to see if the testDelete was too long. This was an effective way of checking if the code was efficient.

**Reflection:**

One of the software techniques I used was the Dynamic testing method which involves the testing of the dynamic behaviors of the software code. I used this technique to check the behaviors of the various dynamic variables which are not constant and find the weak areas during software runtime. The other software technique that I did not use was static testing, which is a way of testing the code without executing it. The difference between static and dynamic testing is that identification of defects cannot be easily found by dynamic testing, such as development standard breaches and the detection of dependencies and inconsistencies in software models.

Looking at the mindset I adopted on this project, I was able to capture all the test cases. I employed caution when it came to testing the various use cases to be checked. It was quite important to appreciate the complexity and interrelationships of the code I was testing because each of the class objects had requirements which must be fulfilled to certify the code as working correctly. All this requires the person to understand what each part of the code is doing.

Like areas where each unit of the class has some specific requirement to be able to function or the contact service that had some specific crude operations to be carried out effectively. I believe when it comes to reviewing your code, bias can be a major factor which has to be reduced, cause the person feels he is confident about the code he or she wrote and hence, won’t carry out proper testing of the units of the code. An example is when creating variables within a class, you’ll have the confidence since you were the one who created it, the code is effective.

When it comes to ensuring quality as a software engineering professional, discipline is vital, because that is what will ensure a smooth process of creating an application that contains fewer bugs and code errors. It is very important not to cut corners, especially when it comes to writing or testing code as millions of money loss can be avoided and even loss of life can be avoided by testing the codes properly and ensuring everything is working perfectly before shipping the code to production. Like using the Contact service for example when it came to ensuring each contact has a unique ID that is not updatable, this was done to ensure each contact is unique and can’t be interchanged with another contact to avoid errors.