Chase Carney

SNHU

06/19/2022

**PROJECT TWO**

For this project we were asked to create three features; a Contact, Task, and Appointments. After that we were instructed to develop unit tests for each, the is the approach I took. I wrote a Junit test for each attribute in the specific object making sure it would fail if the requirements were not met. Specifically, if we were to look at the Task class, the task ID requirement was that it couldn’t be null or longer than 10 characters and if it was it would throw the IllegalArgumentExpection:

Text

Description automatically generated

Though my coverage wasn’t 100%, I was able to achieve higher than the 80% coverage requirement to ensure the quality of the Junit tests.

To make sure my code was technically sound I intentionally wrote test to include bad inputs to make sure errors would be thrown. This example shows the test I created to check name and ensure an exception would be thrown for a null name field and a name with more than 10 characters. Creating these tests made sure I met the requirements which would ensure that my code would be efficient:

Text

Description automatically generated

The two main software testing techniques I used were unit/white box testing and integration testing. With unit testing, I made sure that I tested each and every testable portion of the code to make sure that I was meeting the requirements given to me. Every method of the class was written with the purpose of the requirements and also included its own Junit test to make sure coverage was acceptable. For the integration testing, after each portion of the project was tested to make sure it was acceptable it was necessary to test the project as a whole. This ensures nothing breaks when it’s put together.

There are many different testing techniques what were not implemented for this project. One of them is acceptance testing, this testing technique is performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. Acceptance testing is crucial for a product that will be sold to the customer, this testing ensures the end product will be acceptable for the targeted audience. Another one is security testing, with this project just being for a school project vulnerabilities were not a concern at this time. Though, some of the portions like mentioned above; unit testing and integration testing are both considered part of security testing. There was also no need for performance testing in this case either, the need for evaluating how the system preforms under a particular workload was not necessary. Performance testing could be crucial for projects that involve web pages, this testing will make sure the load of the website is handled correctly.

For this project I felt like you needed to be more in the mindset of a tester more than a developer. These exercises revolved around producing Junit tests that could pass with certain coverage percent plus adhering to the requirements that were given to us. That played a big role in the development process, keeping in mind that you needed produce not only the objects but the tests to prove your objects will pass the criteria provided. Limiting bias while developing AND having to test your own code is a tough one that I think shouldn’t be allowed. It seems almost impossible not to have some kind of bias when testing your own code. At times it felt like I was developing the tests around the code instead of developing my code around the tests which could put you in a predicament.

I think it’s pretty obvious the reason to write quality and not cut corners in your projects. The big reason is maintenance, all developed projects needs updates at some point and the person who develops the originally is not always the same person pushing updates. Writing clear and easy to read code with good comments explaining your intention is a must for all projects. In the example below I used clear and straightforward comments to tell what I was attempting to do.

Text

Description automatically generated