Michael Barbuzano

CS-320

4/16/2023

**Project Two**

**Summary**

When unit testing for each of the three features I first read each of the software requirements for each class. For example, the Task Service class requirements in module 4 specify that the task service class should be able to add tasks with a unique ID, delete tasks based on that ID, and update the name and description of each task. After developing the Task Service class, I created a Task Service Tester class to begin creating tests to validate each of these functions.

Here is an example of a test function in the TaskServiceTester class.

*@Test*

public void addTaskTest() {

TaskService service = new TaskService();

service.addTask(new Task("2", "Exercise ", "Go for a run"));

*assertTrue*(TaskService.*tasks*.contains(new Task("2", "Exercise ", "Go for a run")));

This method is used in the TaskServiceTester class to verify that the addTask method in the TaskService class is operating properly. I utilized the assertTrue Junit statement to check if the tasks array list contains the example class added to the array list in the line above it.

I will add another example of a Junit test I created for the Task Service class.

*@Test*

public void deleteTaskTest() {

TaskService service = new TaskService();

service.addTask(new Task("2", "Exercise ", "Go for a run"));

service.addTask(new Task("3", "Practice", "Practice piano"));

service.deleteTask("3");

*assertFalse*(TaskService.*tasks*.contains(new Task("3", "Practice", "Practice piano")));

This method in TaskServiceTester is called deleteTaskTest as it verifies that the designated task is deleted from the array list after the deleteTask method is called to delete the task with the ID 3. This method tests this by adding two new tasks to the array list, then calling the deleteTask method to delete task 3. Then I used the assertFalse Junit statement to make the test fail if task 3 was still in the array list.

By using Junit tests, I was able to efficiently check if the software was functioning properly. To ensure that my code was technically sound, I would execute the code and run the Junit tests to see if everything was working as it should, as well as manually review the code. To make sure that my code was efficient I made sure to follow industry best standards to the best of my ability, as well as create tests that thoroughly tested the code and its methods. To ensure that the tests I created were efficient, I looked into how Junit tests are generally used in the industry and used those examples as templates for my tests.

The main testing technique I used throughout these assignments was unit testing. Unit testing is a form of dynamic testing in which unit testing tools are used to verify that feature in the software is functioning properly. Dynamic testing is a type of software testing that requires the code to be executed. To use my example of addTaskTest from earlier in my TaskServiceTester. I had to execute code to run the test. When executed, the test creates a new task and then adds it to the array list, I then used the assertTrue Junit statement to verify that this new task was added to the array list. Although I have a lot more testing techniques to learn and practice with, I certainly intend to implement unit testing into all of my future projects, as it seems to be one of the most efficient ways to test software.

Another one of the testing techniques I used was static testing. I did numerous manual examinations when I suspected that some portion of the code was not functioning properly. Manuel examinations are a form of static testing in which the code is not executed, but is just analyzed, this is usually done by a tester in a scrum team. By manually reviewing the code for each of the assignments, I was able to correct many simple issues and errors.

One testing technique I did not use on this project was black box testing. Black box testing is “a software testing strategy where the design of the software system is unknown to the tester”. Black box testing can be very beneficial, because it gives the developers information of how the typical user with no knowledge of the application might use it. I think black box testing would be very beneficial for improving the interface of a software application, as it will be clear if the user is understanding the functions of the application in black box testing.

**Reflection**

While working on this project, I was able to better understand the mindset of a tester. It is important to appreciate the complexity and interrelationships of the code you are testing to better understand how the code functions and interacts with other portions of the code. By understanding the code’s functionality and interactions, the tester can create more effective tests that can assure accurate results and provide substantial feedback.

Although I can understand how some people could be biased when reviewing their own code. I find myself being stricter with my own code. I understand that I am very much a beginner and have a lot to learn, so I do not think I have much or any bias when reviewing my own code.

It is important to be disciplined when working to develop quality software because cutting corners when writing test code will likely lead to an error being missed in the code, which will probably create more issues down the line in the software development life cycle. To avoid technical debt while working in the field, I plan on communicating with my team frequently so that accurate goals can be made for each sprint. I also intend on using industry best practices and not cutting corners to avoid having to do more work down the road, further delaying a project, leading to technical debt.

Works Cited

The QA Lead, & Boog, J. (2023, February 6). *9 types of software testing in software engineering*. The QA Lead. Retrieved April 2, 2023, from https://theqalead.com/test-management/types-of-software-testing/

Parthiban, P. (2022, October 28). *7 reasons why software testing is important*. Indium Software. Retrieved April 16, 2023, from https://www.indiumsoftware.com/blog/why-software-testing/

Lutkevich, B. (2023, February 1). *What is technical debt?* WhatIs.com. Retrieved April 16, 2023, from https://www.techtarget.com/whatis/definition/technical-debt#:~:text=Technical%20debt%20is%20a%20common,is%20sometimes%20called%20code%20debt.