Southern New Hampshire University

CS 320 Software Test Automation

Project Two Submission

George M Harrison Jr.

**Summary**

To meet the requirements for the customer there were three classes which were written Task Services, Contact Services, and Appointment Services. For each class, to ensure their functionality to meet the customer’s specifications, J-Unit tests were designed and written to test the code against the specifications. The approach to the J-Unit testing was direct in each of the classes. First, the code was written in accordance with the specifications.

For example, in the AppointmentServices.java file, there were requirements were to be able to add appointments with a unique appointment ID and be able to delete appointments per appointment ID. This class also had requirements for the length of the characters used in the appointmentID was limited to no longer than ten characters. Date field was limited to the present and could not be a past date. Also, the description string could not be longer than fifty characters.

A picture containing company name

Description automatically generated

The above line of code is written to ensure that the appointmentID is not empty OR the length of the appointmentID is not longer than ten characters and if the appointmentID does not fall into these parameters there will be an IllegalArgumentException thrown. This was hard coded into the method. This is just one example of how the code testing was approached and requirements were ensured.

Text

Description automatically generated

For the J Unit testing, It was decided to test for the thrown by using the asset.True method from the Assertions api built into java for J Unit testing. The idea is that if the class is but correctly there will be no exceptions and the methods, when passed the correct parameters is true. There was also a method that tested whether the exception was thrown by passing the method incorrect data outside of the requirements. If the method works properly there should be an exception thrown. To ensure that all J Unit testing covered all as much code as possible the tests were also run with coverage to track the path during runtime. The trace indicated that eighty percent of the code’s line were covered. I would prefer that one hundred percent was cover so there will be some code revision in the future to ensure that goal.

**Reflection**

The reflecting on techniques and testing conducting in this project, there were small bits of code that were tested for specifications. The approach was based on the boundary value analysis principles which is based on testing at the boundaries of the data partitions.  It includes maximum, minimum, inside or outside boundaries, typical values, and error values. It seemed to be he most appropriate method to use for this project. I could’ve also used an equivalence class partitioning approach dividing the rest conditions into partitions which could be considered the same.

When approaching this project, the mindset was aimed at accomplishment of the customer requirements and writing test code that ensure the functionality of the code written. The appreciation of the tester is necessary. This is the person who makes sure the code is functional and to specifications. He also ensures that no faulty product leaves the facility which protests the integrity of the company’s brand, saves potential losses monetarily and saves time in production. The tester’s role is essential.

Testers must approach testing with limited bias when reviewing code. Which is why we should always test our code but, it much better to have a tester who didn’t develop the code to test its functionality in relation to the specifications when at all possible. It is easy to be biases and egotistical when testing your own code as its developer. There is always that idea of “I know it’s corrected”.

Therefore, it’s important to develop good coding habits as a professional software engineer. Adherence to coding standards is always recommended for the best outcome. For example, if testing code is not written to code all the possible correct and incorrect scenarios there could be a need to rewrite code which is costly to the company. Recalling software already released to a customer could be quiet embarrassing and not only costly in money but in reputation also.

Citations:

*Software Testing Techniques with Test Case Design Examples*. (2019, December 4). Guru99.com. https://www.guru99.com/software-testing-techniques.html