Max Freeman

CS 320

Project 2 Submission

08/14/2022

## Summary

For each of the three functions of the application, I built a unique set of dynamic Junit tests to execute to assure quality in the software product. The contact service needed to be tested for several given requirements. I designed a series of tests that presented cases where the input should result in an added contact vs others where the contact service should refuse the request to add a new contact based on missing requirements. I have included an excerpt of code used to test some facets of this unit’s code here:

/\* testing the add method when input not valid\*/

@Test

public void testMethodAddFail() {

ContactService cs = new ContactService();

Contact c1 = new Contact("C001", "John", "Bush", "121 LA ST");

Contact c2 = new Contact("C002", "Jim", "Bush", "122 LA ST");

Contact c3 = new Contact("C003", "James", "Bush", "123 LA ST");

assertEquals(true, cs.add(c1));

assertEquals(false, cs.add(c1));

assertEquals(true, cs.add(c3));

assertEquals(true, cs.add(c2));

Next, I worked on the Task service, which was mostly an identical build to the appointment class. The requirements were similar in the sense none of the fields could be null and all had a character limit. I also needed to ensure the ID for this class was immutable, same as the previous one built. I also built dynamic Junit tests for this portion of the application that presented with a multitude of cases where the input either should or should not be accepted by the program and written to the system memory. I have included an excerpt of code used to test some facets of this unit’s code here:

public void testMethodDeleteFail() {

TaskService ts = new TaskService();

Task t1 = new Task("T001", "Task 1", "Complete task 1");

Task t2 = new Task("T002", "Task 2", "Complete task 2");

Task t3 = new Task("T003", "Task 3", "Complete task 3");

Lastly, I built the Appointment service portion of the application. This was a bit more complex due to a date field which I did not have much experience with validation for in Java programming language. I am actually still in the process of designing true validation and testing for that field now. The rest of the requirements mostly mirrored the logic for the other fields. I don’t have the code completed for the uniqueness of the date field for this report but will come back and update later. Here is a placeholder space for that code once it is completed early next week:

My experience with the Junit tests was that there is great documentation and resources in order to help guide building the tests. I found a lot of useful Reddit and StackOverflow threads from practicing industry professionals at software agencies on how the conduct Junit tests in Java, and I used this to shape my thinking on how to design these tests and get coverage from the testing that approaches or exceeds 80%. I still have a lot to learn on this subject, as it wasn’t something I’d been faced with before given I primarily write code in C++ or Python for work and personal projects. I look forward to honing my skills completely and being able to write Junit tests with even higher coverage levels and proficiency in the future. Here are some examples to illustrate technically sound and efficient Junit testing that were not included above:

The testing techniques used during this project were Junit tests conducted dynamically such that they executed lines of code in order to assess whether the software was behaving correctly given use cases that I designed within the code for the tests. This allows us to uncover problems that might exist in the code that would not be visible during other types of tests such as static tests where we are not executing the code for the entire program. I can certainly see a number of use cases for the techniques we used during this project. For instance, I am going to a conference in Orlando in September over automation of business processes, and these types of tests should be helpful in verifying that the automation software is effectively completing tasks given business use cases that either should or should not result in certain workflows completing in other programs through API integration.

As a software tester, I employed caution by first sketching out the basic logic for testing and the base cases I thought would be likely based on my knowledge of the program and the ways I was able to imagine it needed tested. Next, I researched other documentation and professional sources to get a sense of testing coverage I would need that I wasn’t able to think of myself. This allowed for me to get higher and higher coverage levels until I was satisfied that I was achieving the maximum test coverage that I could without getting too lost in the weeds.

Another concern I attempted to control for was bias in the design of the code. This is a concern especially if the same developer is responsible for both writing and testing the code. This is because the software engineer may believe certain things about the users and the possible inputs to the program that are either not true or not all-encompassing, and therefore it is possible that the software could be performing optimally in the eyes of the developer/tester while contain critical flaws in the eyes of end users. A prime example of this I used in this week’s discussion board post. Boeing designed a flight simulation software where they were biased to believe that experienced pilots would recognize that the real aircraft would behave slightly differently in takeoff due to having more powerful engines than the one they were actually using to simulate flight. This proved to be untrue, and ultimately resulted in two mass-casualty events. That was with a team of engineers and testers, so you can imagine how great a problem would be possible with just one person responsible for an entire project of this magnitude. This is also why it is important to be disciplined in your commitment to provide quality software to end users, as ripple effects can reach much further than the engineering team can possibly foresee in the development of that software.