Alexandra Meitz

CS 320

Final Project Part Two

12/13/2020

Summary

During this course we had the opportunity to write our own code and create Junit test to make sure everything worked properly. This was my first course creating code from scratch as well as my first experience with the Junit test. I am very grateful that I have learned to understand these types of tests as I found them extremely helpful and easy to use.

For both of my codes, in some cases I had multiple tests to make sure the software requirements were met. For example, in both assignments we needed to make sure the ID was less than 10 characters. For this test I created a code that would test to make sure there was a scenario where the name was the appropriate length, one where the name was null, and one where the name was too long. Creating all three test ensures that your code does not only work when it meets the requirements but also fails or gives the appropriate error messages when it is supposed to as well. I created these multiple tests for all of the software requirements.

As for the quality of the JUnit test in both the contact service and task service assignments, I made use of the “coverage as” tool and was able to go through the code section by section. Both assignments had two different Junit tests for each class. When I would run the test for one specific task, I made sure that I was seeing 100% coverage meaning that all the code for that specific class was tested and was working as it should.

One example of how I made sure my code was technically sound was by choosing to include multiple IDs. For example:

tClass tClass1 = new tClass("Alexandra", "12345", "descriptiondescriptiondescription");

tClass tClass2 = new tClass("Alexandra", "11111", "descriptiondescriptiondescription");

I only used one specific ID for my test, but I wanted to show that my code had the option to pick from multiple IDs and made sure that it ran properly despite this. Additionally, I was very meticulous with the software requirements making sure that I was including the right character lengths in the right places as well as keeping all requirements in order so that they were easy to trace back to my tests.

The efficiency relates back to my use of the Junit test as I made sure I was seeing 100% coverage in all categories before I deemed it successful. Also, I made sure to complete the test for the service classes last because they had to pull from the “class” class. Another way to make sure I was being efficient with my code as well as the tests.

Here is an example from my service code where you can see it is calling from “class”:

void testserviceClassAddContact() {

tService TService = new tService();

tClass TClass1 = new tClass("Alexandra", "11111", "descriptiondescriptiondescription");

tClass TClass2 = new tClass("Alexandra", "44444", "descriptiondescriptiondescription");

TService.addContact(TClass1);

TService.addContact(TClass2);

TService.setContactName("11111", "Chris");

TService.setContactDescription("44444", "Description");

TService.deleteContact("11111");

TService.deleteContact("00000");

}

This was another way to make sure I was being efficient with my code as well as the tests.

Reflection

We have learned there any many type of software testing techniques that are utilized to make sure that your software is working correctly, efficiently, and can operate in multiple environments across different platforms. For your assignments in modules three, four and five we utilized Junit testing. This was my first experience with this type of test, and I found it to be incredibly helpful and intuitive. Junit tests are a unit testing framework for Java and are a very important part of the development process. These types of test allow developers to test software for expected results.

In each of our three milestones, there were cases where the program needed to produce an exception and generate an “invalid” message. In each assignment I wrote code to test the program for the output if the requirements were followed correctly, but I also had to write code to show that if the criteria was not met I was presented with the proper error message. This is important because although a program may run, it does not necessarily mean everything is going to have the desired output. This is especially true for exceptions and error messages.

For example, in our last assignment we had to make sure the appointment date was not in the past. I used this section of code to make sure that if given a previous date, my error message would pop up:

void testclassrequirementsDateInPast() {

Date date = new Date(5678945L \* 1000);

Assertions.assertThrows(IllegalArgumentException.class, () -> {

new classrequirements("12345", date, "descriptiondescriptiondescription");

});

}

By adding in this section to the test I found an error in my main code as well as made sure that I could test the exception. These types of test are critical to developing a fully functioning program.

There are other useful software tests, many of which were not utilized for this project so far. Software tests are broken up into functional and non-functional test. Some of the functional tests include unit testing, integration testing, system testing and acceptance testing which are usually conducted in order. Functional testing involves testing the application or program against the business requirements to make sure all parts of the software are behaving as expected. The non-functional testing methods focus on the operational parts of the software and include performance testing, security testing, usability, and compatibility testing.

Mindset

When working on this project I tried to have the mindset I would as if I was working on this project professionally. I used caution when developing this code the same way I would if I was developing code for a company and tried to act as though any mistakes would not be acceptable. It was important to appreciate the complexity and interrelationships of the code I was testing to understand how minor details can drastically affect an entire program if not checked thoroughly. During my testing I did not have much bias in the sense that I was not thinking my code would be correct the first time around. Perhaps if I were a more experienced programmer, I would have the mindset that my code is correct and would have been more relaxed in my testing. Being inexperienced helped me in this particular situation since I was under the impression I would have many errors to fix each time I tested and did not have any bias that my code would always be accurate.

When developing any software, it is extremely important to make sure that each step of the process is done thoroughly and with caution. Small errors could cost a company time and money and if you are the developer responsible it could cost you your job or reputation. This was the first time I used Junit testing and I appreciate being taught this method as it was nice to be able to see exactly how my code worked and what parts were not correct or producing the output that I intended. Ven if I perceive my code to be correct I will always remember to not cut corners and test my work before moving on or submitting a final project.