Jewelia England

CS 320

Module Four journal Entry

November 19, 2023

In module three I submitted a Contact Service zip file containing four files in which I used JUnit tests to test the code that I had written. In module three I submitted a task service zip file containing four files in which I used JUnit tests to test the code I had written as well. I learned so much between these last two weeks. It really felt like hands-on learning and was exciting to run my tests and get them to pass.

In module three my testing approach was definitely unorganized and kind of random. This is how I missed some test cases. In module four I followed a specific order and ran tests from top to bottom to ensure that it aligned with the software requirements. This not only kept my code more organized but my mind as well.

When I was writing the JUnit tests for contact service I was not as confident because it was my first time going through this. I had written tests but came to find out when I got feedback that I did not run enough tests even though the ones I did run, passed. This is something I am going to fix in my program and ensure that I run more JUnit tests. WHen I was working on the task service, I was much more diligent. I actually used the JUnit Coverage test to show me what had been tested in my code and what had not been tested yet. This highlighted and showed me the percentage of my code that still required tests to be written. Anything that was highlighted in red, I knew I still had not covered. This was extremely helpful and something I will continue to use in the future.

I knew that my code was technically sound because of the mass amounts of tests that I ran on it. For example I ran tests to ensure the input was not null or out of the size range as shown below.

```
@Test
    void testAddTaskUniqueIdNullFails()
    {
        Assertions.assertThrows(IllegalArgumentException.class,() -> {
```

```
                TaskService service = new TaskService();

                Task task = new Task(null,  "Jewelia", "Jewelias Task");

                service.addTask(task);



        });
```

I knew that my code was efficient because I used less lines of code but was still able to pass all of my tests. Below is an example of part of my if loop.

```
Task(String uniqueId, String name, String description)

        {

if (uniqueId == null || uniqueId.length() > UNIQUE_ID_MAX_LENGTH)

                {

throw new IllegalArgumentException("Invalid UniqueId");

                }else
```