Victor Sanchez

Professor Karl Lewis

CS-320-R1873

15 October 2023

Project Two

Before starting the assignments, I made sure to review the requirements to know what is expected. This allowed me to create tests that aligned with the requirements, by making sure that every requirement was met. For example, each class object had to meet certain requirements like that the entry should not be null nor exceed a certain length. I added the requirements in a method and then had an exception be thrown if one of the requirements is not met. Then in the JUnit test I verified that the exception was being thrown correctly by creating a new instance where the requirement is not being met. The JUnit tests proved to be effective since I was able to get over 80% of coverage throughout my project, this verifies that all the requirements were covered.

A screen shot of a graph

Description automatically generated

There are a few ways that I made sure that the code was technically sound. For example, in the AppointmentTest class I made sure to initiate the date with a time and then call for that time to be called again. Instead of calling for a new time twice which would cause the JUnit test to fail since the times would have not matched up. Also, by throwing exceptions if the requirements are not met and then testing to verify that the exceptions are working correctly helps the project be more technically sound. The test coverage also verifies that the code was technically sound.

A screenshot of a computer program

Description automatically generated

I ensured that my code was efficient by inserting comments throughout the code and following secure and efficient coding practices. I also made sure to follow standard naming conventions to make it easier to understand the code. Also, in my test classes I made sure to clear the arrays so that new instances could be added without the next test failing due to a similar name already being used. For example, in the TaskServiceTest class I began by clearing the array, to make sure that no previous entries would fail the test. Then I added a new instance and made sure I was able to delete it.

A screen shot of a computer code

Description automatically generated

The main software testing technique I used throughout the project was creating JUnit tests. A JUnit test is used to run tests on a Java project, a package needs to be imported and “@Test” must be used to run the test. If “@Test” is not used the test will not run and it will fail every time (speaking from experience, I was stuck for a couple hours trying to find the issue). For each of my milestones I added exceptions that would be thrown if the requirements for that field were not met. In the test classes I first started by creating a new item in an array and using assertTrue to verify that it is being added and that the fields match. I then tested that each of the requirements were met by using assertThrows to verify that the exception was being thrown if the requirements were not met. These techniques can be used in many other projects in which you need to create and add items to array/list. They will verify that the requirements are met and that the items are being added correctly.

Some techniques I did not use was to annotate any methods with the @Before & @After annotations. For example, in my ContactServiceTest class I cleared the array in each method, but I could have created a new method and added @BeforeEach to clear the list before each test, this would have reduced repetitive code. In other projects the @Before and @After annotations can be used if a method needs to be ran before or after running the tests.

I employed caution throughout the project by making sure that all the requirements were not only being met but that they were being met securely. For example, in the ContactServiceTest class I made sure to add a pass and fail method when adding a new contact. Since the contact should be unique, by adding a contact and then trying to add another contact with the same name, this tests that an error message is thrown and requires a new contact to be input.

A screenshot of a computer program

Description automatically generated

I tried to limit bias in the project by going to tutoring lessons and making sure that my code was working efficiently. I do feel like bias would be a concern if one was testing their own code. I believe it is better to have another person run the tests, not only to limit bias but also to look over the code and make sure that nothing is overlooked. For example, I went to tutoring because a test kept failing, after talking to the tutor I was able to find my mistake (@Test missing in the code). Having another person look at the code is very beneficial and can help the project in the long run.

It is important to be disciplined and not cut corners as a software engineer since coding now plays a large part in our everyday life. When writing or testing code it is beneficial to be disciplined and efficient so that time and money is not lost in the long run. Not only can code lose a company time and money, but it can also affect people’s mental and physical health. Hackers are constantly looking for ways to steal sensitive information and it is important that corners are not cut to prevent them from attacking your software. I plan to avoid technical debt as a practitioner by following secure coding practices and by testing the code effectively. It is important to make security a top priority and try to begin implementing security measures early in the software development lifecycle.