Sam Maclean

April 15, 2022

CS-320

SNHU

**Summary and Reflections Report**

I was hired by Grand Strand Systems to create a mobile application for a customer who provided me with the requirements for this project. These requirements included code for task, appointment, and contact services which were coded as objects in the Java application. I approached the project by working on one individual object at a time which started with the contact service phase of the project. It was broken into three parts for each of the classes/objects and within each part, the code must align with the requirements and the classes must be tested with at least 80% coverage.

My code aligned closely with the requirements because I used them as a guide and made sure my program accomplished the tasks that were indicated in the requirements. For example, in the initial phase of the requirements, it was provided that a Contact object should have a first name field that is no longer than 10 characters and is not null. In my program, I created a verifyFirstName() method that checked if the length of the first name field is less than the maximum of 10 characters and is not null. If these requirements were not met for the first name field, an illegal argument exception was thrown and would have the user create a first name that aligns with the requirements.

The JUnit tests I used were highly effective in providing an efficient amount of coverage for the program because the functions and methods within each class were thoroughly tested by the use of assertions and other testing methods. The coverage percentage was required to align with 80% or greater coverage for the program because 100% coverage is difficult to achieve. JUnit tests told me whether or not the assertion proved to be effective by showing each case highlighted in green or red. I made sure my tests were of sufficient quality by ensuring the tests passed and the expected results were returned for the assertions.

While writing the JUnit tests, I made sure my code was technically sound by using the right coding practices that aligned with what I was trying to achieve. For example, I created objects for each class and tested the methods within the classes for these objects. In the Task program, I tested the TaskService class by creating a new instance of service in my JUnit tests and added a task to the TaskService list and then deleted that task. Finally, I made use of assertions to verify the task is not present in the list anymore as shown below.

//Verify deletion of Task

@Test

**void** deleteContactTest() {

//Create new instance of TaskService

TaskService service = **new** TaskService();

Task task = **new** Task();

service.addTask();

service.deleteTask(task.taskId);

*assertFalse*(service.Tasks.contains(task));

}

I ensured my code was efficient by cutting down excess lines of code that were not required and adding comments to ensure easy readability for other developers that look at the code. For example, in the Appointment program, the AppointmentService class contains an addAppointment() method that simply creates a new appointment based on the randomly generated appointment ID that is passed as an argument. In only two lines of code, the method creates a new Appointment object based on the appointment ID and adds the appointment to the list of appointments.

One software testing technique I used in this project was using assertions to create instances of objects for the classes being tested and using the objects to test the methods within the classes. This was the driving testing technique in this project and the AppointmentServiceTest class, I would test the verification of the description method which involved creating a new object with a description field that was not null and was not above 50 characters per the requirements. This technique helped me efficiently make use of JUnit assertions to verify all methods and functions in my project are returning results that align with the requirements. Another technique I used is using assertions to check if the result is equal to the expected result such as in the TaskServiceTest class where I tested the updateName() method to verify that the new name is equal to the name created in the system after the method is called. These techniques are commonly used in many development projects and situations that require thorough testing of various objects within classes to make sure the results properly align with the requirements.

I employed caution when coding the constructors properly since some calls to methods in classes required only one, two, or no parameters depending on the needs of the method. This required me to code multiple constructors for given parameters. For example, the Appointment class requires the fields ‘appointmnetId’, ‘appointmentDate’, and ‘desc’. It also has constructors for if the appointment ID is given, if all three fields are given, or if none of the fields are given. This way, an appointment can be created without worrying about if one piece of information is missing. It is important to make sure how one function, constructor, or method relates to the rest of the code within the program.

I limited bias in my review of the code by focusing purely on the technical aspects of the program and focusing on the requirements directly rather than letting my emotions get in the way of the construction of the program. For example, I was asked to add appointments with a unique ID and used a simple function to generate a random ID for the field instead of overcomplicating the requirements. It is important to stick to the requirements and remember how the user will interpret certain parts of your code. Discipline is also very important in software engineering because it is important to keep a clean structure within programs and file systems especially when working with a development team. Cutting corners when writing or testing code may be easier but in the long run, it will lead to vulnerabilities and sources of error that could have been prevented with the appropriate coding practices. I plan to avoid technical debt as a practitioner in the field by ensuring that my programs are readable, high quality, and user-friendly. It is also important to keep programs free of errors that could reak havoc on the user experience. When working as a team on a project, for example, it is a good discipline to update high-quality code to repositories and ensure the success of the whole team by maintaining efficient coding standards.