Luis Diaz

CS 320

16 August 2025

Project Two: Summary and Reflections Report.

During the creation of the service and model classes, I was focused on creating an application that would meet all the expectations of the client. These criteria as well as all basic functionalities and the possibilities of others trying to take advantage of any weaknesses in the code to access client data were what made me structure the code in such a way. The testing also needed to ensure that the separate parts of the application would work separately and together once the project was finished.

For the Contact Service class, it was required that the application would be able to handle data being entered, deleted, and altered. To make sure that all data was input correctly, the focus of one of the tests was to make sure that there were no null values.

@Test

**public** **void** testAddContactSuccess() {

Contact contact = **new** Contact("ID123456", "Jonathan", "Lane", "0987654321", "7890 Pine Rd");

service.addContact(contact);

*assertNotNull*(service);

The Task Service class was responsible for validating the functionality of adding, updating and removing tasks from the application. These tasks were required to have unique identifiers and specific restrictions for the descriptors of the tasks. The test will make sure that the task is properly identified conforming to the requirements previously established.

@Test

**void** testAddUniqueTask() {

String id = "0";

String fullName = "Robert Castillo";

String description = "This is a good description";

TaskService tempTask = **new** TaskService();

*assertEquals*(0, TaskService.*tasks*.size());

// Add a unique task and verify it's added correctly.

tempTask.addUniqueTask(fullName, description);

*assertTrue*(TaskService.*tasks*.containsKey(id));

*assertEquals*(fullName, TaskService.*tasks*.get(id).getName());

*assertEquals*(description, TaskService.*tasks*.get(id).getDescription());

}

The Appointment Service test focused on the addition, removal or changing of appointments. The test for this service makes sure that the information needed for appointments, such as date and time, is valid within the constraints set by the requirements. It will reject a null value.

@Test

**void** testAddUniqueAppt() {

String id = "0";

String description = "This is a good description";

Calendar c = Calendar.*getInstance*();

c.set(Calendar.***MONTH***, 11);

c.set(Calendar.***DATE***, 05);

c.set(Calendar.***YEAR***, 2025);

Date goodDate = c.getTime();

AppointmentService tempAppt = **new** AppointmentService();

*assertEquals*(0, AppointmentService.*appointments*.size());

tempAppt.addUniqueAppointment(goodDate, description);

*assertTrue*(AppointmentService.*appointments*.containsKey(id));

*assertEquals*(goodDate, AppointmentService.*appointments*.get(id).getDate());

*assertEquals*(description, AppointmentService.*appointments*.get(id).getDescription());

}

The JUnit testing was made more efficient by making sure that after every test is unique by ensuring that the testing environment is cleared after every test.

@AfterEach

**void** tearDown() **throws** Exception {

AppointmentService.*appointments*.clear();

}

These lines of code make sure that the residuals of past tests do not interfere with the results of the new test. This makes it so that our tests are of better quality and so is our code.

During this project I used static and unit testing. Static testing was used to review code before it is run, and we can detect any errors as soon as possible. Unit testing was used to test the implementation of each part of the application. There were also other techniques that I did not use such as system testing and integration testing.

While testing, I was focused on making sure that the requirements were met. As a developer it is critical to make sure that we deliver the product that a customer expects and that their requirements are met. In order to achieve this goal, it is necessary to test every aspect of the code to ensure that it works as intended. I tried breaking my bias as a developer by analyzing the project as an outsider would to see if it actually accomplishes its goal.

It is important that we remain committed to developing a quality code that has as few flaws as possible so that we can deliver on our clients’ expectations. As we keep testing further, we can uncover any possible errors that are not immediately apparent when doing static testing. As we uncover errors, we can correct them as we go and constantly improve our final product.