Project Two

Jeffrey Sparks

Southern New Hampshire University

CS 320

Prof Kalysa Wilson

My experience of designing the JUnit tests for this project was one of relying on the software requirements, much like I did while designing each section of the code. I knew what the restrictions were for each class and knew how it should behave. With that mindset I continued to create the JUnit tests, mainly with white-box testing in mind. For example, when writing the Task class, I was asked to make sure that the task ID wasn’t longer than 10 characters, and that it wasn’t null. I created private variables as well creating setters and getters, and then lastly a constructor that ties everything together. With the setters, I incorporated an if statement that if the input is null, then it will show an error message or past a certain number of characters, if it isn’t null then it should add the requirement, throwing similar messages in the other classes I’ve designed as well.

I would say the quality of the JUnit tests were high. In my code, I tested for functionality of my code, ensuring that it would function as it should, but that it would also fail if the parameters that the software requirements have described were not met, such as having too many characters or null variables.

*@*Test

public void testDelete() {

TaskService ts = new TaskService();

Task test1 = new Task("1234567890", "Test one", "Test One");

Task test2 = new Task("2222222222", "Test Two", "Test Two");

Task test3 = new Task("3333333333", "Test three", "Test Three");

cs.addTask(test1);

cs.addTask(test2);

cs.addTask(test3);

assertEquals(true, cs.deleteTask("1234567890"));

assertEquals(false, cs.deleteTask"123456789"));

assertEquals(false, cs.deleteTask("1234567890"));

}

This is a test is testing the delete function for the taskService class. Three tests/tasks were created, invoking a true response for deletion of a correct task ID but invokes a false response if the task ID doesn’t match. Here Task test1 was successfully deleted, where as due to incorrect task IDs the other two tests failed, as expected.

*@*Before

public void setUp() {

appointmentService = new AppointmentService();

}

*@*Test

public void testAddAppointment() {

Appointment appointment = new Appointment("A123456789", new Date(), "Test Appointment");

assertTrue(appointmentService.addAppointment(appointment));

}

This section of code depicts setting up a test appointmentService object and adding a new test appointment within it.

*@*Test

public void testUpdate() {

ContactService cs = new ContactService();

Contact test1 = new Contact("1413252", "Jane", "Doe", "4444444444");

Contact test2 = new Contact("1309403", "Malleus", "Draconia", "2187123404");

Contact test3 = new Contact("9752319", "Vil", "Schoenheit", "9215501793");

cs.addContact(test1);

cs.addContact(test2);

cs.addContact(test3);

assertEquals(true, cs.updateContact("9752319", "VilFirst", "SchoenheitLast", "9215501793", "Land of Proxynee"));

assertEquals(false, cs.updateContact("9752322", "VilFirst", "SchoenheitLast", "9215501793", "Land of Proxynee"));

}

The above test depicts how to update a contact information, by using the contact ID.

I had to adjust my mindset for this project and this course when in consideration of how to properly write and execute tests. I have limited experience within the real world of coding/programming for starters, then adding a new hat to my collection was difficult and intimidating. I knew that as a coder, I had to be critical of my own work. I cannot think it is perfect or immaculate just because I wrote it. I had to be able to think about the bigger picture and the fine details at the same time. Though after the first milestone assignment, I feel like the assignments got progressively easier due to increased understandings of the roles being in play within the course.