*Max R Maysonet-Ramirez*

*2221003*

*10/14/2023*

**Summary and Reflections Report**

This is my project two submission based on the experience from building the programming set for project one and testing its capabilities and performance.

When approaching the software requirements of the product, I first started with an approach that was tailored to the client’s requirements, such as limited the input of “appointment.java” and “contact.java”. The requirements were carefully implemented into the programming of the product. After requirements have taken priority, I secondly focused on scalability, and overall goal and performance of code. In my appointment object below :

1public class Appointment {

2    private final String appointmentId;

3    private Date appointmentDate;

4    private String description;

5

6    public Appointment(String appointmentId, Date appointmentDate, String 7description) {

8        if (appointmentDate == null || appointmentId == null || description == 9null || appointmentId.length() >= 10 ||

10            appointmentDate.before(new Date() || description.length() >= 50) {

11           throw new IllegalArgumentException("Invalid, Please try again");

12

13      }

14        this.appointmentId = appointmentId;

15        this.appointmentDate = appointmentDate;

16        this.description = description;

17 }

In line 9, you see I implemented the limitation of “.length() >= 10”. This ensures that the appointment’s ID never surpasses more than 10 characters when being created. That along with creating the Appointment in an object container allows scalability to be applied by simply just editing the appointment object inside of the code, and the changes will be applied wherever the object may be used.

To ensure that all my code was technically sound, along with bug checks and other checks to ensure proper syntax was being used, I doubled checked my objects being used in the right location. I ran tests to create the appointment, contact, etc But also I created test files using the Junit program suite to build test code to make sure that the code is tested against it’s performance. This helped me portray situations and test said situations against the code. The situations include rare occasions, and “1 off” input types.

 @Test

    void testInvalidDescriptionLengthAtCreation() {

        String invalidDescription = "This description is more than fifty characters long, which is not allowed..";

        assertThrows(IllegalArgumentException.class, () ->

            new Appointment("12313", new Date(System.currentTimeMillis() + 10000), invalidDescription));

    }

In this code, you’ll see that I tested the description length of the appointment object. By Using a description that is more than 50 characters long. This tests the implementation of the description “.length()” implementation that was added to the appointmentservice.java file.

All this together, by making sure there wasn’t any unnecessary syntax in my applications, I ensured that my code was simplistic, performed well, and was able to be scaled by using Java professional standards that were stated in the documentation for Junit testing and Java programming language.

I also made sure that I created containers for all the mutators and accessors that were needed in the product for it to perform, and making my syntax simple to understand.

public class AppointmentService {

    private final ConcurrentHashMap<String, Appointment> appointments = ConcurrentHashMap<>();

    public Appointment addAppointment(String appointmentId, Date appointmentDate, String description) {

        Appointment appointment = new Appointment(appointmentId, appoinmentDate, description);

        if (appointments.putIfAbsent(appointmentId, appointment) != null) {

            throw new IllegalArgumentException("Invalid, that appointment has already been scheduled. Please try again");

        }

        return appointment;

    }

    public void deleteAppointment(String appointmentId) {

        appointments.remove(appointmentId);

    }

    public Appointment getAppointment(String appointmentId) {

        return appointments.get(appointmentId);

    }

}

Above, is an example of my appointmentService object. If you notice I placed the deleteAppointment mutator and getAppointment accessor in the object for the appointmentService. This allows me to create an object that holds the services for the appointment object that can easily migrate methods for the appointment wherever needed possible.

Now before this course, I didn’t have any experience writing JUnit tests, and I had trouble reading documentation on them. But I used “Speechify” in some parts where the language wasn’t too heavy. With that I was able to understand what it was used for and where. The code in the document was easy to understand, once I knew how to utilize its tools, it’s very simple to use.

Reflecting, I used a couple of techniques and tailored the JUnit tests to these methods such as testing the boundaries of the code, and it’s transitions as well. Such as deleting objects, mutating objects and creating objects, and changing the characters of these objects. One thing that I didn’t implement is stress testing. I’m not sure how the program will perform under public stress. I was more focused on the factor testing the boundaries and performance of transitions because this is an educational product and not designed to be used in enterprise networks and systems.

Assessing the mindset that I used for this code is that understanding interdependencies was vital factor when programming this to cover all the requirements needed from the client. A simple logical error could create backload in another section of the product and that is why we use JUnit testing techniques for this matter. Being cautious about how your code performs is a crucial piece of the full product.

With that being said, If I were testing my own code without actually creating the program, it is possible to overlook something because you are just learning how the program works and just reading the syntax. But it is also possible to think vice versa. The program works because I wrote the code is a mindset that can’t be persistently used in this industry because your product will reflect your work and not your ego. So having a bias is not useful in this case, and this is why we use testing to cover tests for all possible cases that seem to arise. Because of this, I did my best to cover 100% of possibilities when running the code in an environment.

Conclusively, I did my best to display discipline in my code and tried to show example of what proper product to client syntax and code should look like. As a software developer/engineer.. I have to ensure that my best work is always released to public hands because my name, date and reasoning is written in the code and having bad code in the world is unacceptable. With this, it’s really tempting to skip over “warning” or “caution” bugs. Having alarms in your code is bad practice and not uniform to proper appearance. Because of this I made sure to build elaborate testing for JUnit testing and take sufficient time to test code because of the number of possibilities and situations where programming can build faults.

class AppointmentServiceTest {

    @Test

    void testAddAndRetrieveAppointment() {

        AppointmentService service = new AppointmentService();

        service.addAppointment("03420", new Date(System.currentTimeMillis() + 10000), "Description");

        assertNotNull(service.getAppointment("03420"));

        assertEquals("03420", service.getAppointment("03420").getAppointmentId());

    }

    @Test

    void testDeleteAppointment() {

        AppointmentService service = new AppointmentService();

        service.addAppointment("0342", new Date(System.currentTimeMillis() + 10000), "Description");

        service.deleteAppointment("0342");

        assertNull(service.getAppointment("0342"));

    }

    @Test

    void testAddAppointmentWithDuplicateId() {

        AppointmentService service = new AppointmentService();

        service.addAppointment("0420", new Date(System.currentTimeMillis() + 10000), "Description");

        assertThrows(IllegalArgumentException.class, () ->

            service.addAppointment("0420", new Date(System.currentTimeMillis() + 10000), "Description")

        );

    }

    @Test

    void testAddAppointmentWithPastDate() {

        AppointmentService service = new AppointmentService();

        assertThrows(IllegalArgumentException.class, () ->

            service.addAppointment("0320", new Date(System.currentTimeMillis() - 10000), "Description")

        );

    }

    @Test

    void testAddAppointmentWithNullDescription() {

        AppointmentService service = new AppointmentService();

        assertThrows(IllegalArgumentException.class, () ->

            service.addAppointment("0320", new Date(System.currentTimeMillis() + 10000), null)

        );

    }

    @Test

    void testDeleteNonExistingAppointment() {

        AppointmentService service = new AppointmentService();

        // Try deleting an appointment which is not present in the service.

        // It should not throw any exceptions.

        assertDoesNotThrow(() -> service.deleteAppointment("03431"));

    }

    @Test

    void testRetrieveNonExistingAppointment() {

        AppointmentService service = new AppointmentService();

        // Retrieve an appointment which is not present in the service.

        // It should return null.

        assertNull(service.getAppointment("06724"));

    }

If you look into my appointmentServiceTest.java file. I have provided 7 tests to tests the possibilities of the appointmentService.java file and verify it’s implementations are working in this case.

Of course, we will need to discuss more exact cases on where more tests where specifically deployed, but this overall displays the robust efficiency and industriousness of the programming that was applied into the products through the project and modules 3 – 5 of this course. With a proper mindset, strive for perfection and cautious perspective, I was able to build 3 programs and apply JUnit testing to these applications.

Thank you,