Nick Boodoo

Project Two

CS 350

4/17/22

Summary

My approach was aligned to the software requirements pretty well. For the contact class and service, I successfully completed all of their requirements which were “the contact object shall have a required unique contact ID string that cannot be longer than 10 characters. The contact ID shall not be null and shall not be updatable. The contact object shall have a required firstName, lastName, phone, and address Strings field that cannot be longer than 10 characters.” The firstName field shall not be null. These requirements were all successfully fulfilled.

Regarding the task class and service, I also created code that fulfilled the following requirements of “the task object shall have a required unique task ID String that cannot be longer than 10 characters. The task ID shall not be null and shall not be updatable. The task object shall have a required name String field that cannot be longer than 20 characters. The name field shall not be null. The task object shall have a required description String field that cannot be longer than 50 characters. The description field shall not be null.” In addition to this, functionality was added so that new tasks could add or remove tasks by a unique ID. I also added functionality to edit a name and description per task.

Regarding the appointment requirements, I have also completed those requirements being: “The appointment object shall have a required unique appointment ID string that cannot be longer than 10 characters. The appointment ID shall not be null and shall not be updatable. The appointment object shall have a required appointment Date field. The appointment Date field cannot be in the past. The appointment Date field shall not be null. The appointment object shall have a required description String field that cannot be longer than 50 characters. The description field shall not be null. The appointment service shall be able to add appointments with a unique appointment ID. The appointment service shall be able to delete appointments per appointment ID.”

The overall quality of my JUnit tests is pretty high since they all fulfill their function. In addition to this, the JUnit tests provide expected results and does so efficiently. Just for the sake of proving that my code is technically sound, I will insert a portion below and explain it.

“//checks date

if(date == null) {

throw new IllegalArgumentException("Invalid appointment date - null");

}

else if(date.isBefore(today)) {

throw new IllegalArgumentException("Invalid appointment date - past date");

}”

In this code, I am writing technical code that is correct and contains only what is necessary to function. This also makes my code efficient.

Reflection

I used a software testing technique called unit testing. This testing is where you would take units of software and determine if they are suitable for use in the code or not. In this project, I specifically used JUnit testing. Other forms of testing include integration testing, system testing and acceptance testing. Integration testing is the process of testing the interface between two software units.

I adopted the mindset of a professional developer while working on this project. I tried my best to employ time management and attention to detail skills in order to successfully avoid issues and complete the project on time. It is important to understand the complexity of the software and client’s needs so that you are successful. Bias can be eliminated when self testing code because at the end of the day, you still need to fulfill client requirements. It is very important to remain disciplined so that the coding process is always efficient and successful. I will personally employ all of these strategies to avoid technical debt.