Kristle Morales Figueroa

Module 7 - Project 2

CS 320

December 15, 2024

**Summary and Reflections Report**

**Summary**

After completing the assigned development of a testing application for the Contact, Task and appointment services, this report will endeavor to summarize and reflect on the approach and testing techniques used to complete the project. To discuss the extent of alignment of the project with the software requirements we should list those requirements. Three classes were needed in the project named Contact, Task and Appointment. Each class had specific requirements for the objects and each class required a corresponding Service Class that would perform adding, updating and deleting operations of the objects created. The requirements of each class are listed below.

* Contact Class

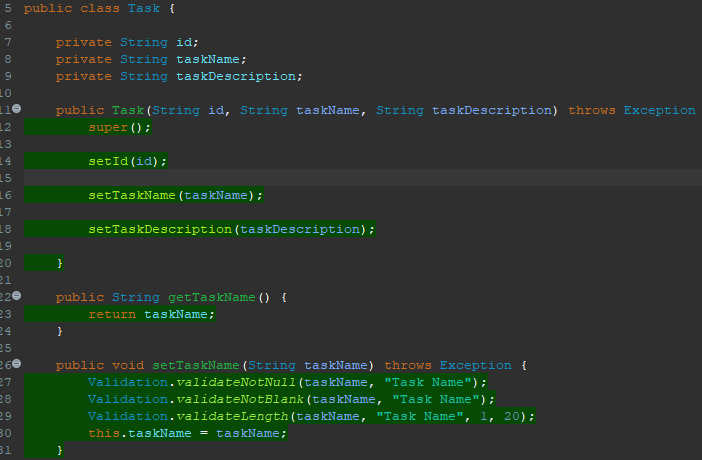
The Contact Class would create five objects. A unique ID object that would not be null or updatable. First Name, Last Name, Phone and Address objects would also be created not null or blank with character limitations and in the case of the phone object only numeric characters would be acceptable.

* Task Class

The Task Class would require an ID, task Name and task Description. As in the Contact Class ID, Name and Description objects cannot be blank or null. The ID object must be immutable and character limitations would apply for both the task name and task description objects.

* Appointment Class

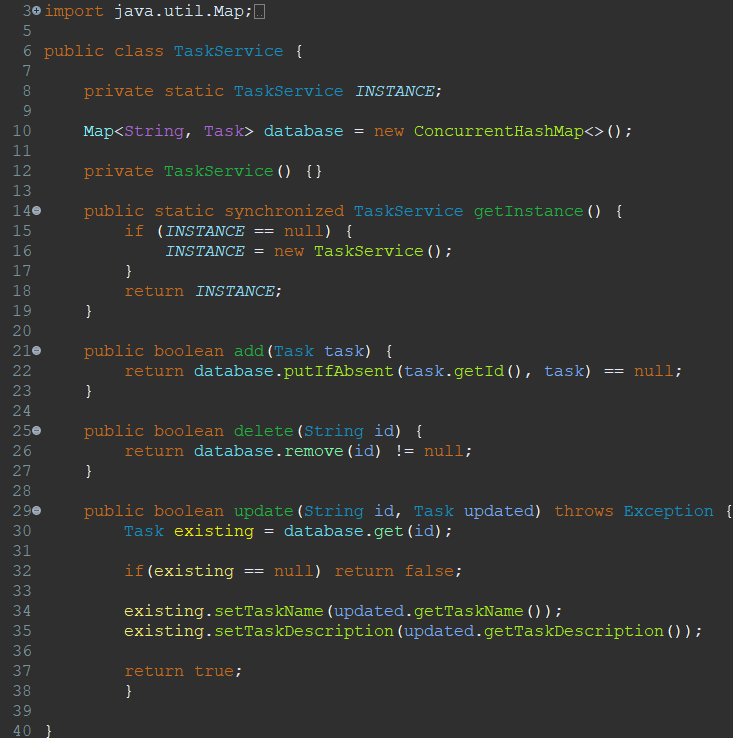
The appointment class would require a unique ID string, appointment date and a description that could not be blank or null. Character Limitations would apply to all 3 fields.

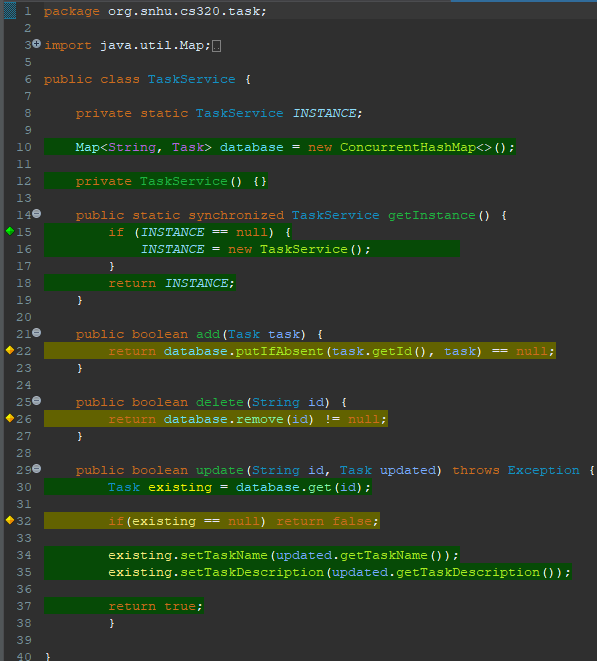
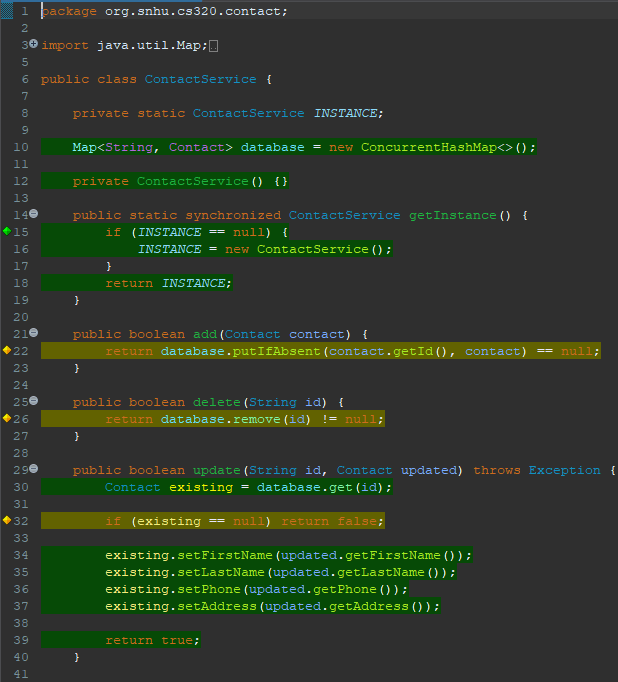
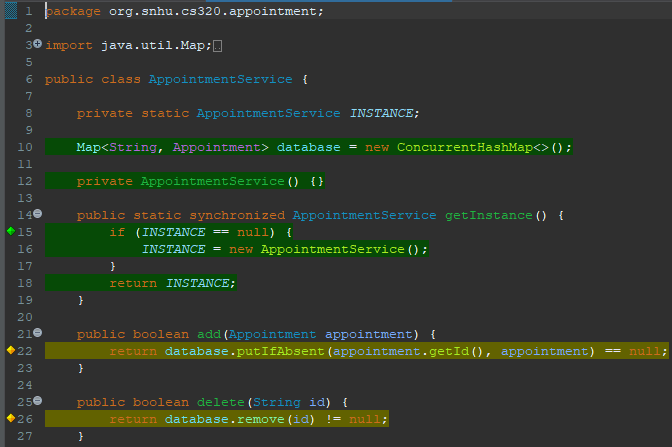
A screen shot of a computer program

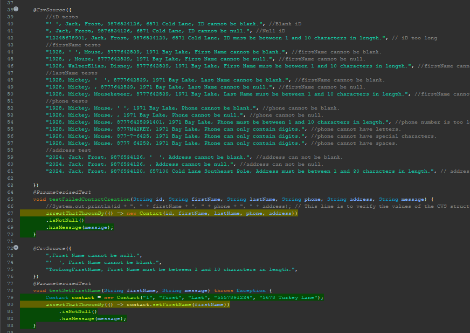
Description automatically generated My approach to testing was to take small steps into each requirement testing to keep the alignment between my code and the software requirements. My starting point were ***if*** statements in the setters to test out the setters of the Contact and Task Classes. My approach for both Contact and Task Classes was similar since their types and requirements were very similar with String types and Character limitations. Below is a portion of the Contact and Task code for comparison.

A computer screen shot of a program

Description automatically generatedThe appointment class was a special challenge for me since my knowledge of the ***Date()*** type object was very limited. To achieve a successful testing, I used localDate instead of Date. Below is a copy of the code.

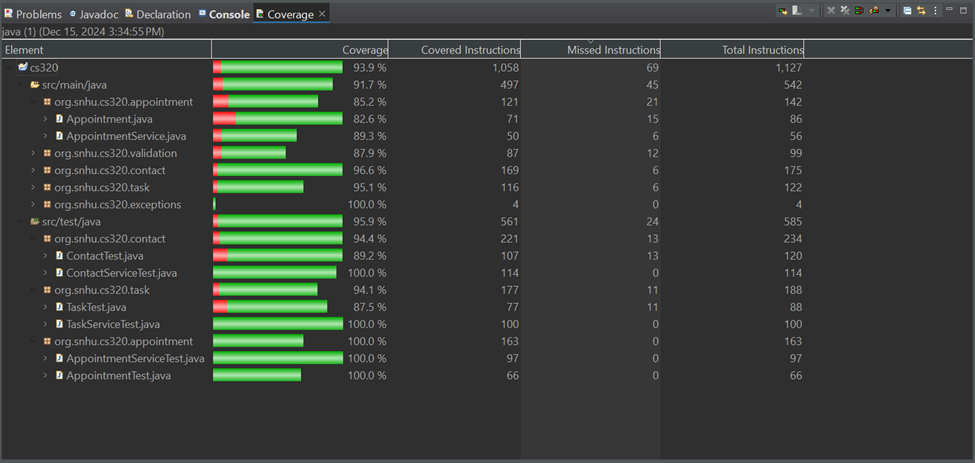
A screen shot of a computer program

Description automatically generatedThe Service Classes for the Contact, Task and Appointment Classes required operations of adding, updating and deleting objects in each class. Copies of the code are posted below.The JUnit tests were made technically sound by sectioning each test as well as adding a CSV source structures for specific case testing. The overall code was made more efficient by adding a A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedValidation class to limit code repetition. Examples are shown below.

The overall efficiency of my code as measured by the Coverage of eclipse is shown below. The lower percentage of Coverage applies to the Appointment tasks which I think is a direct result of my struggles with the date type variable.



**Reflection**

Thinking back through the development of my code, the most obvious technique that was used is Unit Testing. Unit Testing is the name given to when testing is done in induvial units of code, which I think aligns perfectly with the encapsulation principle of software development. A technique that I used but had to remind myself to use every time was Regression testing. I began my Contact and Task testing with ***if*** statements to test out the requirements of null and character limitations, as I progressed my code into using a validation class to avoid repetitive code, I would apply regression testing to make sure that changing my code was not introducing errors to the code. Another technique used was boundary values specifically for Contact and Task Classes. Some other technique that I would like the opportunity to use is Ad-hoc testing, trying to break an application to find bugs that may have been missed sounds a lot like stress testing, which I think is very interesting. For this technique the most practical use is for fully developed applications. As an example, in my current job I work closely with a software development team that is testing un updated version of an application we are currently working with. Though I do not take part in the testing myself it has been a fantastic learning experience which has allowed me to see the practical side of our readings of testing techniques.

In reflecting about how I employed caution during the development of my code for testing, I don’t know that I paid more caution to once specific task. Since my experience in software development and testing specifically is still extremely new, I believe I applied the same level of caution through my code. In terms of bias, I believe that a bias may be present in testing my own code, though I do believe that I would carry a negative bias rather than a positive bias. The tendency to believe I made a mistake rather than got everything right would more than likely color my bias. I do believe that a risk of repetition fatigue would be more likely to affect my testing of my own code rather than bias. To develop testing techniques I would have to constantly look at my code which I think would eventually cause me to miss a necessary detail.

Finally, a disciplined approach to code building and code testing should be at the forefront of every developer’s mind. Discipline in software development would mean a detailed breakdown of the requirements of the code and an equally detailed breakdown of the planned testing. In general, is never a good idea to cut corners but when applied to software development this becomes amplified since delivering a product that is not up to standard translates in lost confidence from the client in both the product and the developer.