**Module 7 Journal**

Jesenia Roberts

Southern New Hampshire University

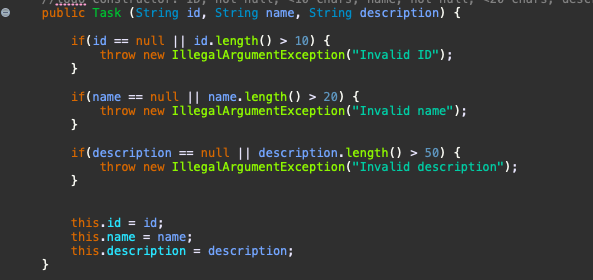
CS- 330

Professor Tanisha Jacks

December, 10, 2023

I made the necessary methods to complete the tasks outlined by the bulleted project requirements. After reading the requirements I made “todo” comments throughout the code describing the needed methods to accomplish the tasks outlined in the requirements. Then, I write each of the methods based on the todo comments that I left, and from there, I create the Junit testing file where I tested that exceptions were thrown when invalid data was input. For example, the Appointment class had methods that I outlined with todo comments, then after writing the code I tested that the given exceptions were thrown (such as calling the constructor with invalid information causing an error to be thrown).

For all of the Service classes I thoroughly tested the overall functionality of the class as well as the individual methods. I created the Service and data objects in their respective JUnit test classes and ensured that each object was successfully created with the correct data. Additionally, I tested that each piece of data could be updated by the mutator methods (except in the case of the Appointment classes where no mutators were required), and retrieved by the accessor methods. Finally, I checked that the exceptions I set within the methods would be thrown when needed. This tests each major aspect of the classes and their restrictions and thus warrants an effective coverage of testing.



In the above example, I have created multiple if-statements that check if the data abides by the requirements and is valid by those standards.

Regardless of the project, ensuring data integrity is important. The technique used throughout this course can be used in future academic and professional projects to prevent any unreliable data from being saved or used within the program.

One example of my code being technically sound is using a hashmap for the object lists in the Service classes. The use of a hashmap allows for tasks and contacts to be stored, retrieved, and deleted via their IDs:

private Map<String, Task> taskList;

private int next\_id;

This is an efficient way of storing these objects as retrieving information from the maps in this situation has a better runtime than looping through other data structures to find information:

public void addTask() {

Task task = new Task(next\_id + "", "", "");

taskList.put(next\_id + "", task);

next\_id ++;

}

public void deleteTask(String id) {

taskList.remove(id);

}

public Task getTaskById(String id) {

return taskList.get(id);

}

}

When programming for a client, the client’s requirements are immensely important, so having an understanding of the requirements at the helm of the work process is a good practice. Outlining these requirements and how they will transfer into code in notes or comments is a useful way to bridge the gap between the client’s requirements and the actual code itself. The testing element is paramount to creating sound code, and having an understanding of JUnit allows for the use of an efficient testing framework that will increase the speed and completeness of the testing process.

One way that I challenged my bias when reviewing my code was to test everything as if I was expecting that things would not work until proven otherwise. I made a test for each functionality aspect of the program, even if I was sure that what I had written would catch any discrepancies. An example of this is how I wrote lines in the JUnit tests that assured that an error would be thrown when data was invalid, even though the actual function that implemented that was straightforward and I was fairly confident that it would be successful.

However, on a larger scale and in different circumstances, it would be easy to fall into the “designing for myself” rutt that can sometimes happen with developers. I have, in the past, worked on web development projects that I think are foolproof from my perspective, only to watch them have bugs and errors when used by a friend. For this reason, it is immensely important to gain different perspectives when touching up code, if it is expected that there will be a variety of users, there should be a variety of testers as well.

The failure of a program due to poor quality and limited testing can result in consequences that range from money loss to death depending on the situation. Last week our class discussed defects, and the topics were as small in importance as buggy video games, to the deaths of a plane full of people, all as a result of poorly written/tested programs. As developers, it is our responsibility to complete the project to the best of our ability, and utilize the tools that we have gained through education, and may be available in our profession to create sound code. There is no long-term benefit to creating poor-quality code, regardless of whether it is faster to cut corners, the code will likely need to be repaired after the fact anyway.