

Latrelle Dickey

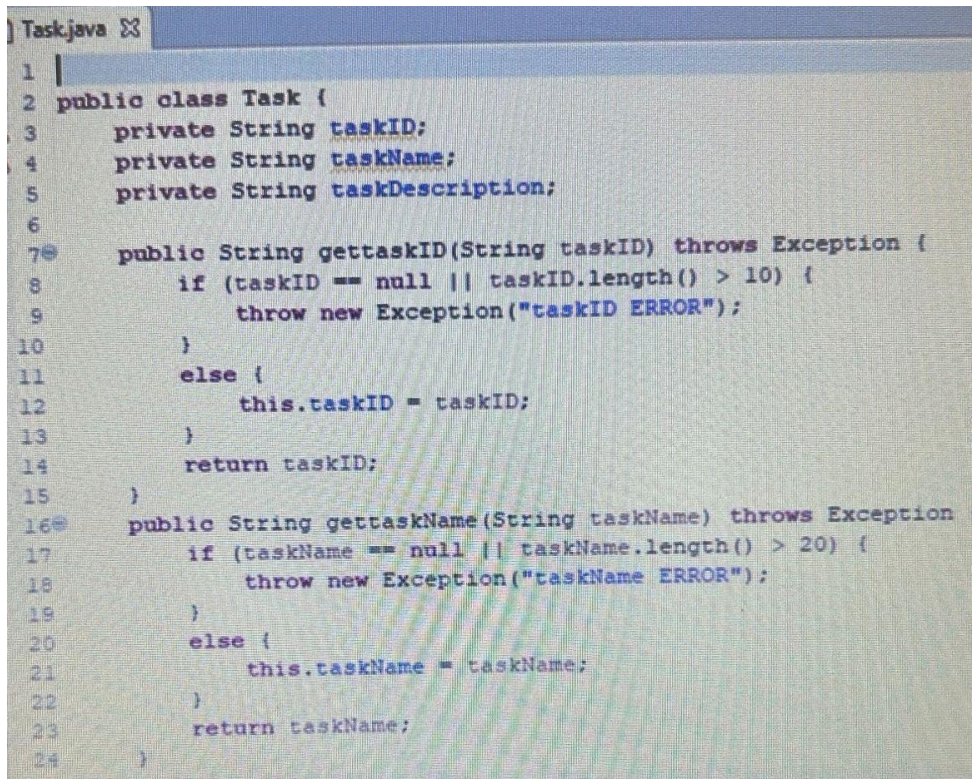
CS-320

20 October 2024

Final Project

My approach to completing the unit testing for each of the three features is adding all the user requirements that were needed for each feature and saving the testing for last. The type of unit testing I used for this project was the white box testing method which is where you test the behavior of the software application. From completing each module's Milestones, I seem to have added each software requirement for each feature but the only thing that was missing was the completion of the JUnit testing. The reason for missing the JUnit testing is that I didn't know how to complete the testing at first but with a few YouTube videos and a Google search I was able to complete the testing. My original submission for my modules Milestones is different from my final submission because I fixed a lot of the errors and warnings I had and I added more to each class.

This was my first experience with writing Junit tests and it was difficult. To ensure the code was technically sound, I tried to make sure I was typing the code in order it should be typed. For instance, the image below shows that I identify all the variables then define the tasks each variable should complete.



```

1 |
2 public class Task {
3     private String taskID;
4     private String taskName;
5     private String taskDescription;
6
7     public String gettaskID(String taskID) throws Exception {
8         if (taskID == null || taskID.length() > 10) {
9             throw new Exception("taskID ERROR");
10        }
11        else {
12            this.taskID = taskID;
13        }
14        return taskID;
15    }
16    public String gettaskName(String taskName) throws Exception {
17        if (taskName == null || taskName.length() > 20) {
18            throw new Exception("taskName ERROR");
19        }
20        else {
21            this.taskName = taskName;
22        }
23        return taskName;
24    }

```

After I finished typing all the requirements, I initiated the JUnit test. When the test was completed, the outcome was a fail. The message I received was “Not yet implemented”. I’m not sure if that made the code efficient or not because I’m still not familiar with JUnit testing. White box testing is the testing technique employed in the project. This testing method was used to test each statement and object of the user requirements individually. I didn’t use any other type of software technique for this project. Programmers use white box testing to ensure the program is performing efficiently and is secure. This type of testing is used to help write cleaner code and is transparent when identifying vulnerabilities. By using this type of testing within the early stages of the development process it is cost efficient and could possibly save the developer time on developing so they can spend more time checking the quality and integrity of the program.

Overall, by working on this project I was able to adopt the mindset of a software tester. Acting as a software tester was frustrating and difficult because I had a hard time figuring out and understanding the flow and the use cases. I also used bad programming practice by copying a piece of my code into other parts of my program. I employed caution when I first started working on the assignment. The directions and use cases were not clear so I wasn't too sure where to start for this program. I can imagine bias being a concern when testing my own code because I would overlook some warnings and errors especially if I know a piece of code is needed. Cutting corners as a programmer can lead to missing factors within the code.