Dogo Cyrille Yelibi

CS 320

Project 2

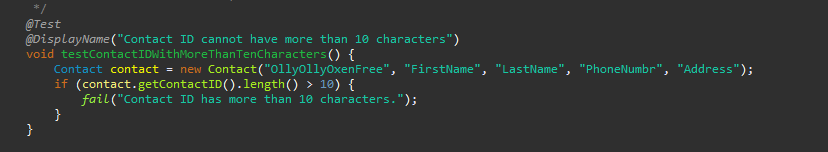
03/03/2024

The JUnit test I used throughout this project was considered in the requirement. Following the requirement and basic rules of the JUnit testing and simply applied them. The classes would run as test cases seem sound. For example one requirement was that “The contact object shall have a required firstName String field that cannot be longer than 10 characters. The firstName field shall not be null”

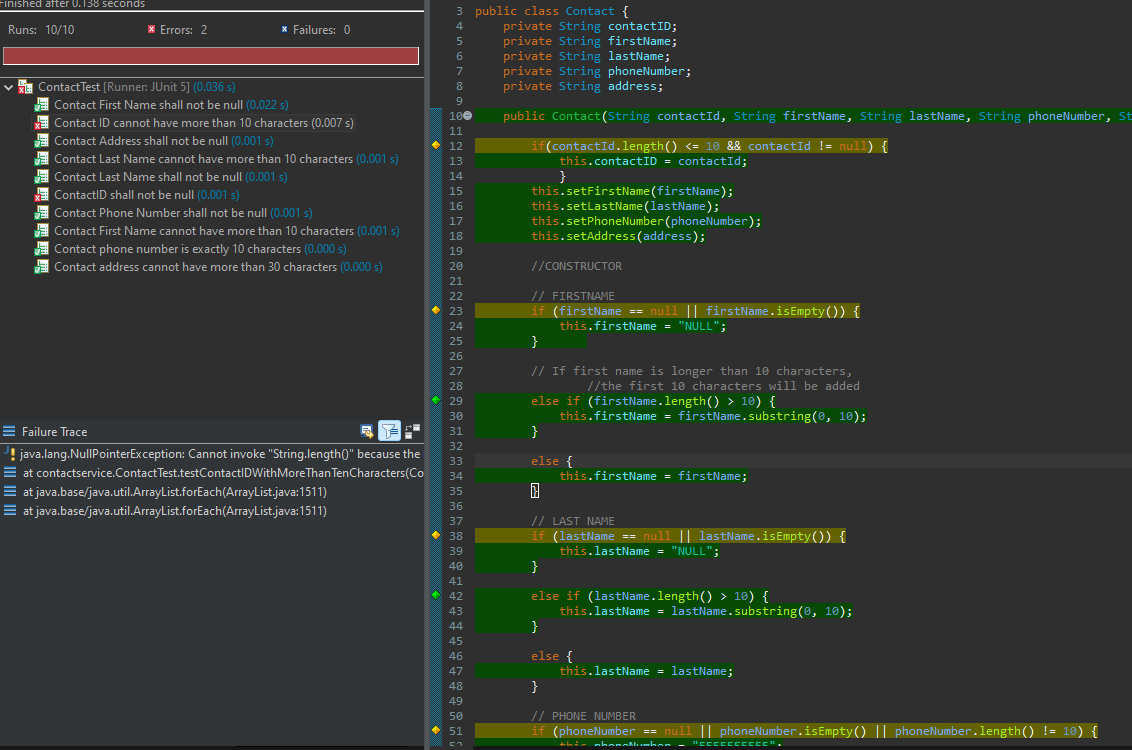
A computer screen shot of a code

Description automatically generated

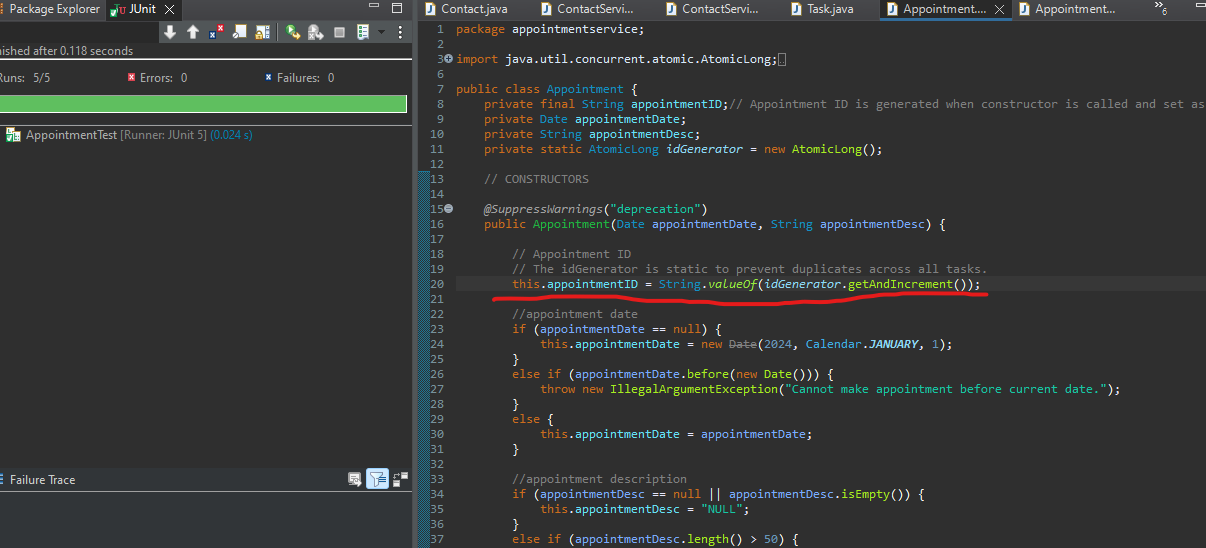
I then use the JUnit testing technique like @Test and @Displayname for the contact class. After some time I was thinking that maybe adding a @order to the conatctTest used case would help make my test better but I wasn’t successful there.



This was the test I used but was not getting that much result. I ran the tests a couple of times and although my test case seemed sound the test failed every single time. I think it was linked to the fact that I wasn’t able to have my contactID to be not null and not limited to 10 characters.



This was the same case of the Task class too. I think I understood by the end how to correct this mistake in the Appointment class. I underlined on the image below where I believe was the correction in my code that I needed to apply to the other classes.

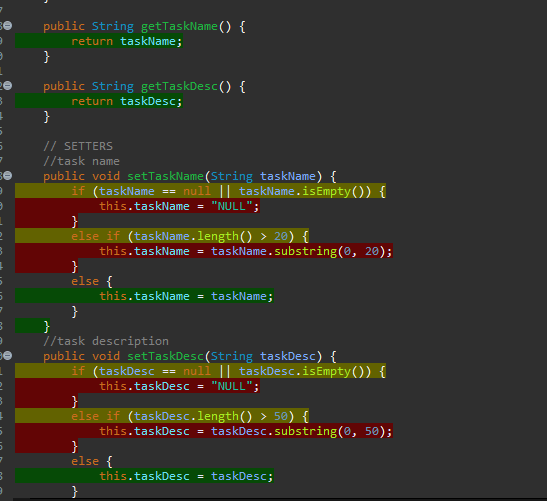


After fixing that my code ran perfectly with a coverage of more than 80%.

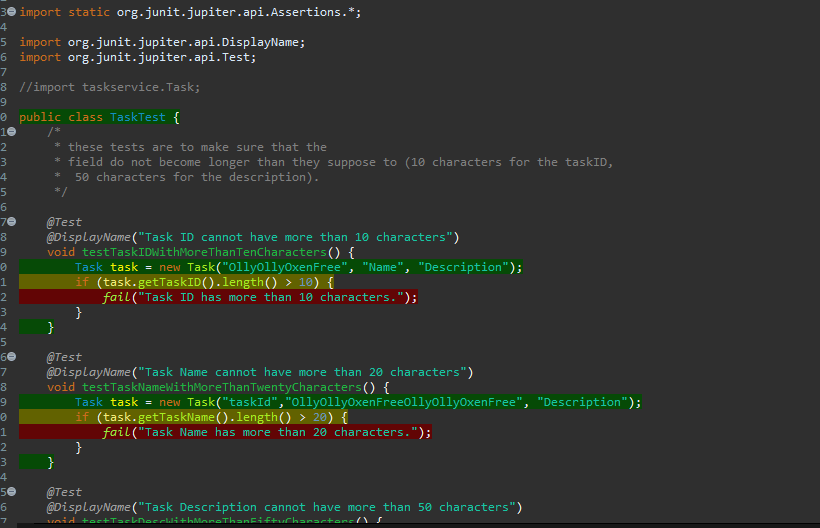
A screenshot of a computer program

Description automatically generated

I didn’t try to make a it more complicated at first and wrote a lot of “if “and “else” statement because of the object oriented nature of the program. Applying the setter and getter was not that complicated and the service class was here to assure that the requirements were followed, like updating the list of tasks and deleting them if they were null. My task class was made in a way where it was not doing what I wanted it to do. I did follow the rules and regulations and the requirements of the program.

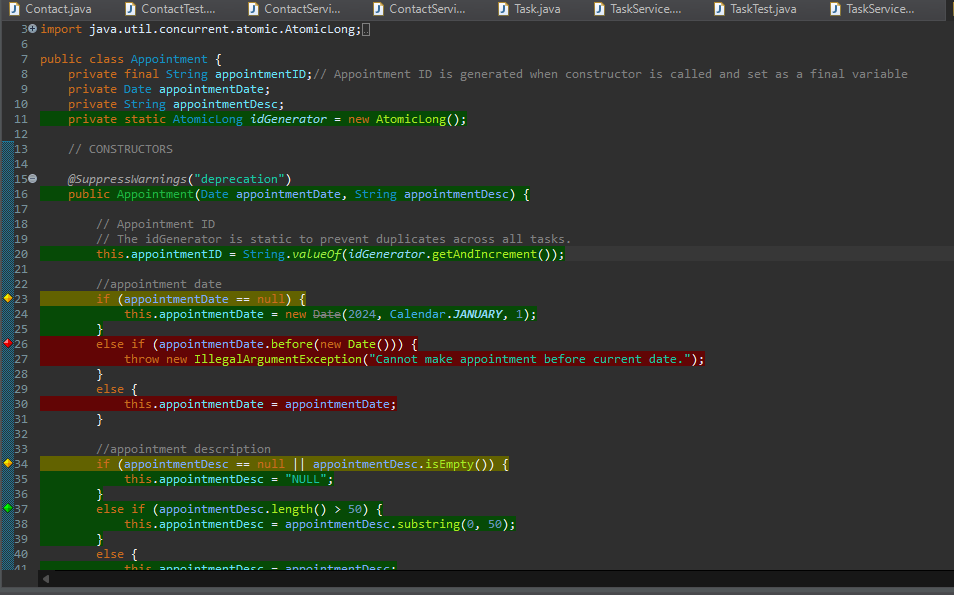


I think that I realized that my code was sound while writing the tests. I did make sure that what I was testing was what the requirement was asking. Which is basically a bit backward I feel like because It was the test that confirmed to me that my code was sound.



I did a few coverages runs before fixing all my issues and realizing that I had made a mistake on it. I then spent 2 weeks trying to find a way to fix the mistake I made. Fortunately, I was able to reach the 80% coverage but not after thinking that I could increase my coverage by lowering the number of branches in my code. It was a disaster because I was breaking my code non-stop.

I used an object-oriented technique to code each class and made sure that each function was working as intended in the service version of the class. To do so I use JUnit test. I ran the coverage run to check my code and assure that the functions were correct. For example, in the appointment part of the project I did a run to see if the appointment date was going to be current and added a condition that if this parameter failed it would show me an error message.



The good thing with Eclipse is that it shows you which branches are not covered with colors highlighted (red for branches that are missed and green for branches that were covered). Since my test was successful it was highlighted green. This is dynamic testing. It involves the execution of the software to validate its behavior against the specified requirements. This testing technique is concerned with the runtime aspects of the software, evaluating how it performs under different conditions. The primary objective of dynamic testing is to ensure that the software functions as intended and meets the specified requirements. It involves the creation of test cases, execution of these cases, and comparison of the actual results with the expected results.

A screenshot of a computer

Description automatically generated

For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.

Static testing is a testing technique that does not involve the execution of the code. Instead, it focuses on examining the code, design, and documentation to identify defects and improve the quality of the software. The primary goal of static testing is to detect issues early in the development process, reducing the likelihood of defects progressing to later stages. Common static testing activities include code reviews, inspections, walkthroughs, and requirements analysis. Static testing serves as a proactive approach to identify and rectify defects at an early stage. This is how I did my first testing and thought everything was working fine because from the look of it everything was fine. I was wrong. Which makes me feel like dynamic testing is a more accurate way to test a software. Or at least the way to be the most confident about the functionality of a program.

I thought that the object-oriented technique would assure that my code would work no matter what the test I threw at it. I was wrong. I think that I should be testing my code as I rewrite it. But as a person that is not a genius coder, I tend to write easy algorithms. If I don’t pay attention won’t be as efficient as the other technique. I try to always remember the requirement and end up focusing too much on it.

A screenshot of a computer

Description automatically generated

This mistake would have been avoided if I noticed that the logic of that statement does not prevent the contactID from being null. It just set one condition and not all scenarios will cover it. I don’t feel like I cut corners, but I don’t think I did a great job at the beginning. Hope fully I was able to find the solution and get a good Run coverage.