Jasmine D. Sadler

Southern New Hampshire University

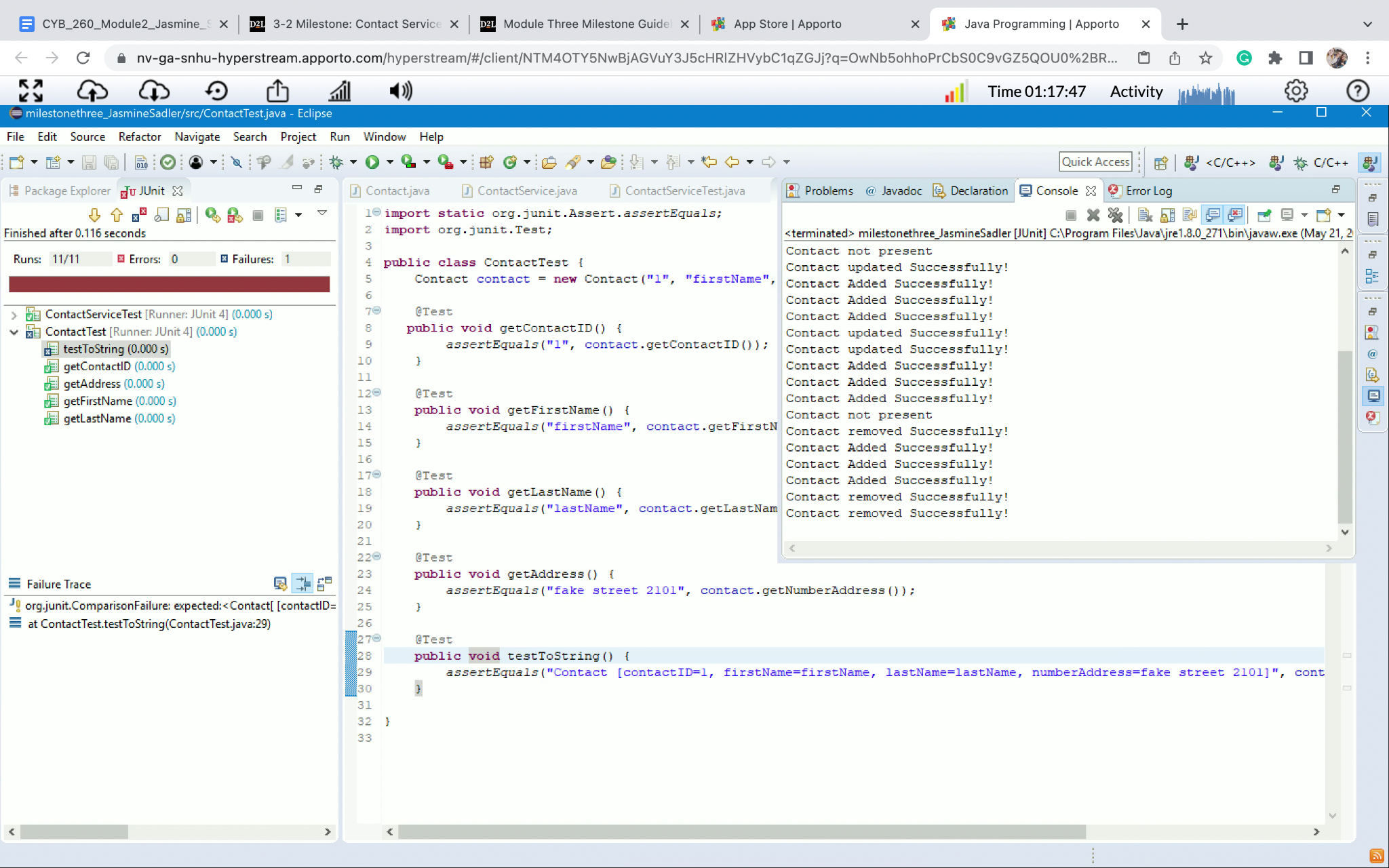
Project Two

June 18th, 2023

Tanisha Jacks

Alignment to Requirements and Effective Tests

In milestone one, the code allowed changes within the contacts. The changes to the contacts that were allowed were updating, deleting, and adding. Through JUnit tests, the contact service will be verified. Along with the contact service, the contact object is also present in the contact service. No database was needed for milestone two since the task service stores tasks in in-memory data structures. Through JUnit tests, it will validate the appointment service. Along with the appointment service, an appointment object is also included. Through JUnit tests, the task service will be verified. Along with the task service, the task object is also present in the task service. For milestones one and two, all the tests came back as passed. Milestone three was a little more tricky. After fixing some spelling errors the tests passed. I would say that this was a good coverage. This was my first time creating tests. They all matched the customer's requests and passed.

Summary: Technically Sound Code 

The image above is a screenshot from the contact test. As you can see all of the tests that were created to meet the customer's requirements were passed.

Summary: Efficient Code

This is a line from the file named ContactTest.java. It shows code that is simplified, clean, and well-organized. This practice was kept with all the code created for this project.

*‘import static org.junit.Assert.assertEquals;*

*import org.junit.Test;*

*public class ContactTest {*

*Contact contact = new Contact("1", "firstName", "lastName", "fake street 2101"); // making the object as a class member so that all the methods can access it*

*@Test*

*public void getContactID() {*

*assertEquals("1", contact.getContactID());*

*}’*

Reflection

For this project, I was able to test particular sections of my code using the JUnit testing approach to determine whether any mistakes needed to be fixed. I was able to test functions using the code's specifications to make sure nothing had been inserted wrong. Some testing technique that we didn’t use was functional testing and regression testing. Functional testing is where instead of testing small units of code, here the functionality of a module is tested for its behavior and output. Here the business requirements are validated. Based on provided input, the test case matches the expected output. Regression testing is performed to make sure that when the product is already deployed in a real-time environment with certain functionality and when new functionality is introduced or some bug is fixed, the existing functionality is not broken.

To make sure that the testing was going to be effective I kept a few things in mind. First, it’s important as a software tester to make sure that you are covering end-to-end and the smallest parts of the code while unit testing. This is going to help catch errors earlier. Also, as a tester, I had to make sure that I was testing for all possible inputs. This means testing for the ‘if’ function as well as the ‘else’ function to make sure everything is working correctly.

Bias is something that we all have when reviewing things. This can be an unconscious bias or a conscious bias. In any case, you have to be aware of these and make sure you're not excluding anything. This is why I did my best to test it and see what I left out and then try again. In most cases for me, it was forgetting to test for the ‘else’ cases. Sometimes I was so focused on trying to make sure that the correct thing they wanted was working I was forgetting about the other possible cases. This made me take a step back and reevaluate the code to make sure I was testing for everything.

It’s important to have a disciplined quality-focused mindset as a software engineering professional. Practicing this will lead to better product development. It will also help keep within the timelines for a project. After all, you are not having to redo work because you cut a corner. It will also help you start with a smaller amount of bugs that are being introduced to the product created. Practicing this can help create a stronger customer relationship.