Antonio Ocasio

8/15/2021

Software Test and Automation

Project 2

When designing the code, I made sure to hit all of the requirements that the program needed to have. For example, the Contact Service assignment had a few different requirements for each class. The contact service class needed:

* Unique ID no longer than 10 characters and cannot be updated nor null
* First name cannot be longer than 10 characters and cannot be null
* Last name cannot be longer than 10 characters and cannot be null
* Phone number that is exactly 10 digits and cannot be null
* Address no longer than 30 characters and cannot be null

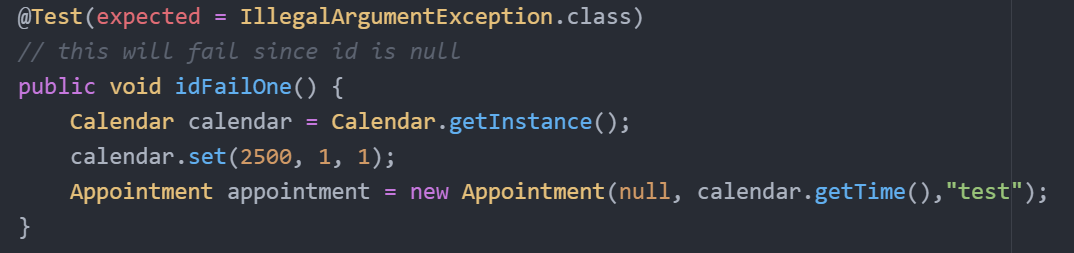
The contact service class also had requirements:

* Be able to add contacts with unique ID
* Be able to delete contacts per contact ID
* Should be able to update contact fields per id
  + First name
  + Last name
  + Number
  + Address

In my code, all of the programs that I have written have specific tests to make sure that each requirement is tested and does not allow or does allow certain changes based on the requirements.



Here, this is my ContactServiceTest.java and you can see that I have created contacts with different ID’s, names and addresses to make sure that I could add it in.



In this screenshot, you can see that I use junit testing and run tests that will fail to make sure that the code catches it and does not let it get added. The way that I made sure my code was technically sound was by making sure that I added comments on most lines of code explaining what each line is doing so that way if I forget what it does or another programmer is viewing the code they will understand as well. I made sure that my code was efficient by having methods in each class instead of rewriting the code. For example, inside of the contact program, instead of manually creating for loops to loop through the list of contacts to delete a contact, I created a method to do it for me so all that I have to do is type in removeContact() and pass in the unique ID. This saves time and memory and makes the code a lot more efficient. The testing techniques that I applied in this project is unit testing. I have never used Junit testing before but I found that it was rather easy. When doing Junit testing, I had to learn the annotations and the main one that I used was @Test. The @test method is only used for testing and creates a test class with methods that are needed to be tested. Junit testing was helpful since it is good for regression testing. It has an open-source framework so that it could be used anywhere and can be used over and over. Throughout this project I have learned a lot of testing techniques which I plan on using as I take on other classes and work. When working on the project I employed caution when analyzing the requirements for the code to make sure that I did not miss anything that was mandatory. It is important to be disciplined as a software developer and cannot be bias towards your code or anyone else’s. No matter whose code it is, it should all be tested and reviewed the same. When analyzing my code, I needed to make sure that I could understand it if I didn’t have any comments which would help me make sure that I wrote efficient code and it was easy to understand. For discipline, it is always good to have comments in the code everywhere listing what each line does and list the requirements at the top of the class so everyone knows what was checked in the code. It isn’t acceptable to cut corners in code because if you do that, the requirements are not all met which would leave the customer unsatisfied. Overall this project has taught me a lot when it comes to writing and testing programs and I plan on using Junit testing in the future to help me with my testing skills.