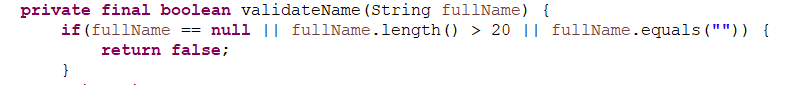
Viren Patel

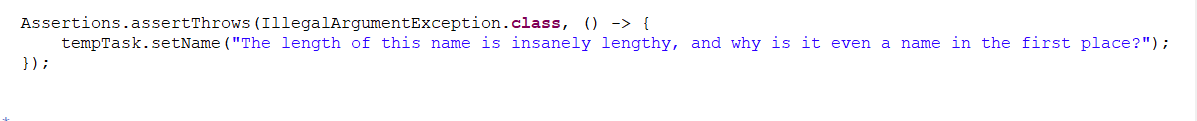
CS 320-Software Test & Automation

December 14th, 2024

Project 2

I had the responsibility of developing unit tests for the contact, task, and appointment services for the mobile application that the client was using. I used a mix of black box testing and white box testing to accomplish the goal of ensuring that the tests were successful. During the black box testing, the process consisted of inputting data and checking that the appropriate result was generated. When doing the white box testing, it was necessary to examine the code and make certain that the logic was correct. For the purpose of testing the limits of the specification, I made use of unit tests! In the case of the Task, for instance, the specification states that the whole name must not be more than twenty characters, cannot be null, and must be provided. At a minimum of four branches are required in order to thoroughly verify this standard.

  
  
I also used JUnit test cases to test for edge cases and to examine the code for errors. In doing so, I was able to cover 99.8% of the code, verifying that it was effective and efficient.

A legitimate version of the name, a string that was null, a string that was extremely lengthy, and a string that was empty were all tested by me. The following are example:   
  


The process of developing the JUnit tests was a pleasant experience for me. Through the process of confirming that the logic was right and that the required outputs were generated, I was able to ensure that my code was technically sound. As an example, a method that removed an ID was supposed to be available in ContactService. Several methods exist for evaluating this: 1) Add three objects to the collection; 2) remove one; 3) verify that the collection is now two objects in size; 4) search the whole collection for the item you removed; 5) ensure that it is no longer there. If you just perform one of these, it's possible for people to act in an abnormal way. It is possible to delete the erroneous item and have the test pass if just the collection size is verified. In the event that all of the things were accidentally deleted, the test may still pass if the object was looked for using the only approach.

A screenshot of a computer

Description automatically generated  
Verifying the deletion of a collection is just as crucial as validating its size after deletion. It would be a good next step to check if all the other objects are still there and functioning as they should. I didn't test this in my code.

I utilized a mix of techniques, including efficient looping, using built-in functions and libraries, refactoring to remove duplicate code, and avoiding unnecessary synchronization, to make sure the code was efficient. Additionally, I was able to guarantee the desired results were generated using JUnit assertions. Here is an example of my code.

A screen shot of a computer code

Description automatically generated

During the course of this assignment, I used both black box testing and white box testing as methods for analyzing software. By inputting data and ensuring that the appropriate outputs are generated, black box testing is carried out. Examination of the code and verification that the logic is sound are both components of white box testing. JUnit test cases were also used by me in order to test for edge cases and to check the code for bugs.

Integration testing and system testing are two more software testing approaches that I avoided from using for this project. One kind of software testing is known as integration testing, and it entails combining and testing separate software components to ensure they function as intended when used together. Specifically, it seeks for problems that could develop as a result of interactions between various parts of a system. As a subset of software testing, system testing verifies the system's overall functioning. It checks whether the system satisfies its criteria and is executed after integration testing.

My software testing methods have a real-world application in ensuring the code is both efficient and error-free. Errors and edge cases may also be found with their help. Errors and security vulnerabilities in software might go undetected if certain testing methods are not used.

I approached this project with a careful and mindful attitude, keeping in mind the intricacy of the code. Through the use of code reviews and unit tests, I attempted to conduct a comprehensive analysis of the code's functioning while minimizing bias. Being aware of the possibility of bias and being thorough while testing one's own code are crucial qualities in a software engineer.   
 One must be self-disciplined in their dedication to quality while developing and testing software. If you want to keep technical errors to a minimum, you should never skip on testing or creating code. I accomplished this by doing extensive code reviews and used unit tests to make sure that the code was error-free.