

7-2 Project Two Submission (Summary and Reflection Report)

CS-320 Software Test Automation& QA

Mihir Patel

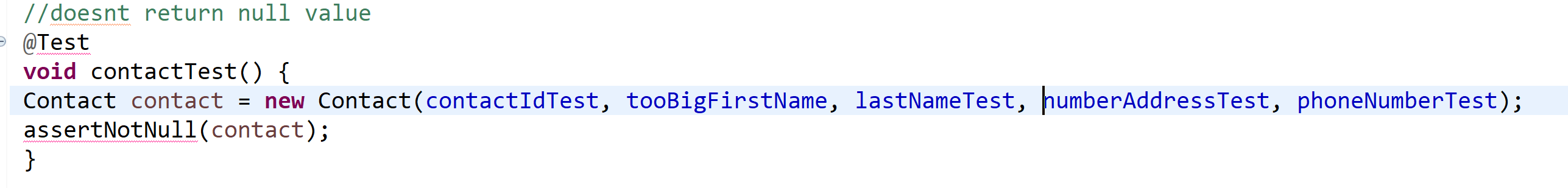
Southern New Hampshire University

February 26, 2023

**Summary**

When developing software solutions, unit testing is very crucial. By swiftly identifying flaws and fixing them, it lowers the chance of encountering program mistakes while enhancing the quality of the code. I used a similar unit testing strategy for each of the three features since I made it compatible with the software requirements. A notable example was the Contact service, which had a number of specific requirements, such as that the Contact object must have a unique contact ID string, which cannot be longer than 10 characters, cannot be null, and cannot be updated. They also wanted the Contact object to require a firstName and lastName, neither of which could be longer than 10 characters and must not be null. In addition, they requested that the contact class have an address field that should not be larger than 30 characters and should not have a null value. This address field is a string that must contain exactly 10 digits.

Moreover, the Contact object has a service that performs some basic contact-related tasks. All of these were particular requirements for one of the three features, and equivalent requirements applied to all the other features as well, thus the software had to be developed in accordance with these specifications in order to pass the unit test. My unit test's overall quality was good since I made an effort to test each field and operation to make sure the test cases were legitimate.



The line of codes above was checking the validity of the parameters for the contact class and I also created a test case that was checking to see we are getting NULL values or not. This was an effective way of checking if the code was efficient.

As same for the Appointment project, We are checking different possibilities as update the appointment date, appointment description, and did we receive the appointment or not.

**Reflection**

The dynamic testing method, which involves examining the dynamic characteristics of the software code, was one of the software strategies I employed. Using this method, I was able to identify the weak spots in the software during runtime and examine the behaviors of the many dynamic variables that are not constant. Static testing, which is a way to test the code without executing it, is the second software technique that I did not employ. The primary distinction between static and dynamic testing is the difficulty dynamic testing has in identifying problems, such as violations of development standards and the discovery of dependencies and inconsistencies in software models.  
 When I approached this project with the right perspective, I was able to guarantee that I captured all of the test cases. I was cautious when testing the numerous cases that needed to be checked. Understanding the intricacy and interdependencies of the code I was testing was crucial because each of the class objects had specific requirements that had to be met in order to vouch for the code's correct operation. To execute all of this, one must comprehend what each code element accomplishes. Like in regions where each class unit must meet a specified condition in order to work, or in the contact service, which required a set of precise primitive procedures in order to function.

When the person writing the code feels confidence in it and won't properly test the code's units, bias might be a significant problem that needs to be reduced when it comes to reviewing the code, in my opinion. Similarly when generating the variables in a class, you might feel confident that the code is efficient because you wrote it. When it comes to software testing, we should all make an effort to minimize that.

As a software engineering expert, discipline is crucial to assuring quality because it will provide a smooth process for producing an application that contains fewer flaws and code errors. Cutting corners while writing or testing code can result in millions of dollars in losses or even the loss of life. By thoroughly testing the scripts and making sure everything is working before releasing the code to production, these losses can be prevented. Use the Contact service, for instance, to ensure that each contact has a unique ID that cannot be updated. This is done to ensure that each contact is distinct and cannot be swapped with another contact in order to prevent problems.