CS-320: 7-2 Project Two Submission

Meixin Zhang

Department of Computer Science, Southern New Hampshire University

CS-320-H7021 Software Test Automation& QA 23EW2

Professor Angelo Luo, M.S

December 10, 2023

Summary

In terms of unit testing, I applied what’s need according to what the customer requires and made them testable to see if I reached their requirement. To ensure the codes reach the client’s requirements I follow their criteria in steps. For example, when asked to create an unique ID that cannot be update, not null, and no longer than 10 characters; I made the ID variable into final so this field won’t be updateable. At the same time, I also made sure when the object is created there are validation to ensure the length of the ID doesn’t exceed 10 characters and not null. As for the unique part, I let the system decide the ID rather than the user to ensure its uniqueness. The codes are shown below on how it was written for the Contact Class.

private final String contactId; // <=10 char, !null, !updateable

public Contact(String firstName, String lastName, String phoneNum, String address) {

//Check if id meet requirement before setting to contactID

if(String.valueOf(id).length() > 10) {

throw new IllegalArgumentException("ID overflow");

}

//Contact ID auto generated Accepted

this.contactId = String.valueOf(id);

id++; //update id for next contact

...}

By implementing the codes as the client requires, the next step will be with the JUnit testing which shows us the code coverage if we run that. With code coverage we can ensure that the codes are effective based on the percentage of the codes that are tested. We know that the JUnit testing were effective based on the coverage percentage as long as its 80% or higher coverage it will be a success since the main parts of the codes are tested.

During writing the JUnit tests I ensure that my codes are technically sound by adding comments to the codes for readability. This way any programmer who reads over the code can understand the purpose this section of the code is to the program. For example in one of my code for the taskService class there is a method that adds tasks objects into a list. I would know this by the comments shown on top of the method as shown below.

//Add task to taskList

public void addTask(String name, String description) {

Task task = new Task(name, description); //create task object

taskList.add(task); //add task to taskList

}

This way when the test coverage shows that it works, and we will also know the code is good to go when tested. To ensure my code was efficient I try to make sure no codes are repetitive, and if they are repetitive, I will just create a method to take care of it. This holds when it comes to the all the Service classes. For example, for contactService class to find the object if they exist, I create this method to run rather than rewriting the codes.

//Using contact ID to return contact object if found

public Contact getContactId(String contactId) throws Exception {

//Check if contact object exist in list with given id

for(int i = 0; i < contactList.size(); i++) {

if(contactList.get(i).getContactId().contentEquals(contactId)) {

return contactList.get(i);

}

}

//If id not found throw error

throw new Exception("No Id found");

}

Reflection

The software testing techniques that I employed on this project are mostly JUnit testing and code coverage. They are the most common used when writing Java projects for unit testing. With JUnit testing we are checking if the codes will compile correctly and as expected. While code coverage helps ensure most of our codes had been run and tested. If there are any that aren't tested, we can cover them in the unit testing. These are the two major testing techniques I used on this project. Some other software testing technique I didn’t use for the project are automated testing. That’s probably because automated testing is usually for larger project compared to the short project as this one. Another technique that’s not used was performance testing. “Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload.” (Gillis, 2023) But for this project we mainly test the function of the codes rather than its performance, so this testing wasn’t done.

When writing test cases I try to be cautious when looking over them. I do that by trying to test the code from a third party. Because often times when it’s our own code we will produce some bias and let errors pass easily the code function won’t work for the client’s needs. I done that towards the end when putting all the codes together to rewrite any redundant codes and test certain aspects of the code that code coverage missed to test. One example is in the contactService class of finding the object with a given ID. The part that was never tested was if the ID wasn’t found there should be an error thrown. So as the tester I wrote a unit test case for that part as shown below.

@Test

@DisplayName("Contact Id not found")

void testContactIdNotFoundByDelete () {

ContactService service = new ContactService();

service.addContact("Johnny", "Anderson", "1234567890", "123 Grand St, NY 12345");

Assertions.assertThrows(Exception.class, () -> {

service.deleteContact("23");

});

}

So, when employing caution, we must try our best to distance ourselves from the code, especially when we wrote it ourselves. It’s important to appreciate the complexity and interrelationship of the code as it was testing because all the codes that are written interact with one another. If one of the codes failed, then the other functions of the code will do so too. Especially, when we are calling methods from other classes and objects that are being used. One example for this is when testing the service class to update one of the contact’s addresses. When doing this test, we will have to call the updateAddress method which calls the setter method for address in the contact class. If code that connects one to the next are not working as expected then the output if any, will also be wrong. As a software engineering professional, we all must be disciplined when it comes to writing and testing our codes and not cut corners. Because if we do, the codes may run unexpected, or it may also lead to many vulnerabilities for attackers to hack into the system and do harm to the client with our code. For example, in this project if an object aren't tested because the tester thought it was to long or that they were to lazy to test. Then this can lead to a bigger problem down the line when the variable causes issue due to it being empty or null because it hasn’t been tested to handle it before. Plus, this is also programmer’s professional ethics they should have for their future employers. If there are carelessness then it shows the employers of their big flaws that can lead them unemployed and skilled over during job qualifications. The best way to combat this problem is practice on through coding and not cut any codes for short term pleasure.

Gillis, A. (2023, March). What is performance testing and how does it work?; TechTarget Definition. Software Quality. Retrieved December 9, 2023, from https://www.techtarget.com/searchsoftwarequality/definition/performance-testing