Brandon Lingenfelter

CS-320

12/12/21

**Summary and Reflection Report**

A product cannot be released without having been properly tested for defects and efficiencies. If a product is released without being properly tested it could lead to an expensive failure and a blow to the company’s reputation. This is why testing your software is one of the most important parts of the software development life cycle. When a project is properly tested you can be sure that you have met all the requirements and that the functionality in the program will work as intended. Two major methods are those of Static and Dynamic testing. Static testing is where we test the code prior without running it to locate any errors earlier on in the development cycle, while Dynamic testing is done near the end of the development cycle and is utilized while the code is running to ensure the code is free of defects and meets the clients’ requirements.

Unit testing is an important step in the testing of a project. Unit testing helps reduce errors and find and fix defects to ensure the program functions as intended. My approach to writing unit test were to align them with the given software requirements. An example of this is ensuring that the strings are the appropriate length and not null, which was a requirement present in all three of the services. Another example was to write a test to see if the element could be updated or not. I checked the features of the services in the program to be sure it met the requirements implemented with those that were required to pass the unit test. I tested the overall quality of my unit test by trying each of the test fields and operations to check the validity of the test case.

My experience with writing the JUnit tests was enjoyable. Before working on these projects I had never written any unit tests, so this experience allowed me to learn something brand new and something that will be very useful in the future. My code is technically sound. The code I put together is modular, syntactically correct and utilizes the appropriate data structures for the task at hand. Here is an example of my test to ensure that a Task was created successfully:

*@Test*

void testCreateTaskSuccess() {

Task task = new Task("123456", "Do Something", "This is a task to complete work from a backlog");

*assertTrue*(task != null);

*assertTrue*(task.getID().equals("123456"));

*assertTrue*(task.getName().equals("Do Something"));

*assertTrue*(task.getDescription().equals("This is a task to complete work from a backlog"));

}

These lines of code show that we are getting the task information from the task file and checking that that the string lengths are the appropriate size and not null. I ensured that my code was efficient by being sure that it worked properly. An example of this is seen in the Contact file:

public class Contact {

private String ID;

private String FirstName;

private String LastName;

private String number;

private String address;

public Contact(String ID, String FirstName, String LastName, String number, String address) {

if(ID.length() <= 10 && ID != null) {

this.ID = ID;

}

this.setFirstName(FirstName);

this.setLastName(LastName);

this.setNumber(number);

this.setAddress(address);

}

In the code above we can see that the elements and string length for ID is set.

The software techniques I employed to complete this project were JUnit testing and Coverage. The JUnit test is where we wrote test for each function and requirement to see if the function in our project passed or failed. The coverage allowed us to see what was being tested to ensure we were testing the whole program functionality. These techniques are extremely useful for creating properly functioning programs and detecting defects that need to be fixed.

My mindset for testing became meticulous in order to find the relationships between files and functions as well as designing test that would define a failure or a pass. I was cautious to not overlook functions that were not being tested by using the coverage testing. It is important to be committed to quality. The products that you produce could be used by many people and delivering a product of poor quality could lead to an expensive recall or patch and could ruin the company and your reputations.