ASSIGNMENT 4 : TEST REFLECTIONS

EMAIL: GREGOIRE.COUSIN@UCDCONNECT.IE;
GREGOIRE COUSIN STUDENT NUMBER: 18204188; COMP 47480 PRACTICAL

Introduction:

In our lab today we dove into a few concepts of software testing that helped me realize how valuable it can be. In software testing, many different methods and approaches chosen are based on the code at hand. In our lab, we used standard tools such as java Junit but could have as well used TestNG which are both platforms that provide annotations to identify test methods, assertions for testing expected results, automatic testing to give you accuracy in your code and generate quality reports of your test results based on UNIT TESTING. Unit testing is where you test specific parts of a system to verify that the behavior is what you want it to do. The JUnit API is loaded with multiple assert type functions such as checking for an equal object, variable, boolean values to make sure you are passing the right information.

A Bit of Theory and Methods:

There are many different types of software coverage methods, and each one can work affectively when testing alongside another. To start, we can provide our code with a lot of results when testing the functionality of a program through branch coverage. Branch coverage is all about the decisions making of an application. Making sure that each possible branch is checked and ensuring that reachable code is executed. We can also test code with Path Coverage where unlike branch coverage, Path Coverage will cover every possible control flow paths through a program. These are some of the most popular methods of testing code.

The Program:

In the assignment, we were asked to provide knowledge on how to test if the given output of a class was precisely the same String in the test class. In the main class, I wanted to test most of my outputs based on a large if statement. Checking if a given number was divisible by 3 or 5 independently or if they were both divisible by 3 and 5. We were asked to simulate a test-driven development. This meant that the tests were needed to be given to the professor one by one; reanalyzing the code at each interval, making sure they pass and committing them on a git branch. It was a good exercise that sets you up for the real world when testing. We also were given the task of creating some a sort of mutation based testing from the previous code written. Mutation testing is checking the polar opposite of what you intend to write.

Emma and a lot more:

Since it is a good practice to use different technics of testing, we used Emma to give us an overview of the coverage achieved by our test cases. This was a graphical view with percentages and line highlights of where the test reached your code in specific lines and where it did not. I found this to work well with Maven, so I took the liberty to learn to implement Emma, jUnit and generate tests based on Mavens architecture all in one module.