**JUnit Test Documentation for Calculator Projects.**

**UNIT Testing:**

UNIT TESTING is a type of software testing where individual units or components of a software are tested. Unit Testing is done during the development of an application by the developers i.e. individual units of source code, such as functions, methods, and class are tested to determine whether they are functioning as expected. Intuitively, one can view a unit as the smallest testable part of an application.

For the calculator Project the different ways and techniques used as perform unit testing is stated below.

**Explanation for Calculatortest:**

The Calculator class contains 8 methods which perform different functionality.

It Contains functionality of addition, subtraction, multiply and Division for a basic calculator.

In order to test these functions, I have used **Whitebox testing technique** where code is checked for the functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyse and fix the defects at initial stages itself.

Using junit5 Unit test Framework created a implemented the test class.

This Test class contains 8 tests corresponding to 8 different methods in calculator class.

In order to attain maximum code coverage, tried to implement different code coverage techniques like

**Statement coverage:** Have written a testcase so that maximum number of statement in a method is executed .

**Branch coverage:** Used this technique especially for Divide() which have if condition. Wrote the testcase with data which make execution of the method to traverse at least once inside the if loop.  The purpose of branch coverage is to ensure that each decision condition from every branch is executed at least once.

**Decision coverage:** The goal of decision coverage testing is to cover and validate all the accessible source code by checking and ensuring that each branch of every possible decision point is executed at least once.

* **testAdd(),testSubtractionTest :** Using a assertions provided by the Junit checked that it accepts integer values and give expected results.
* **testMultiply()** : In this test method, by using assertions, tested for different integers like when two positive, two negative, alternate positive and negative integers are multiplied results in correct output as expected without giving any false output.
* **TestDivide():** Divide() method has a loop so to check that loop work as expected , passed the data in test method so this block of the code gets executed as expected.
* tested the setValue and getValue methods to ensure they accepted the same values which were pass by the user in application.

**Explaination for Inspirationtest:**

Inspiration method has two methods

**Quote():** In order to test this method I used same unit testing techniques as used in Calculator class. Here, I am checking whenever the callcounter is incremented to 1. If incremented then some string is printed out of the switch case. On every entry inside the method random statement is picked and printed on console .In other words, I am testing that string is never null.

**getCallcounter():** here I am testing that this method is returning the call counter which is set 0 initially.

The Data provided for both Calculator Class and Inspiration Class uses **Boundary values analysis.**

Integers are vast set of data. Instaed of testing for all values I have tested for 0,-1,1 i.e one positive, negative values which validates the function to work properly for all other set of values