**NUNIT TESTING**

Follow the steps listed below to write the NUnit test cases for the application.

* Create a Unit test project(.Net Framework) in the solution provided.
* Add the CalcLibrary project as reference
* Create a class “CalculatorTests” to write all the test cases for the methods in the solution
* Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively
* Create a Test method to check the addition functionality

**Caluculator.cs**

using System;

namespace CalcLibrary

{

interface IMathLibrary {

double Addition(double a, double b);

double Subtraction(double a, double b);

double Multiplication(double a, double b);

double Division(double a, double b);

}

public class SimpleCalculator : IMathLibrary

{

double result = 0;

public double Addition(double a, double b)

{

result = a + b;

return result;

public double Subtraction(double a, double b)

result = a - b;

return result;

public double Multiplication(double a, double b)

{

result = a \* b;

return result;

}

public double Division(double a, double b)

{

if (b == 0)

throw new ArgumentException("Second Parameter Can't be Zero");

result = a / b;

return result;

}

public void AllClear()

{

result = 0;

}

public double GetResult

{

get { return result; }

}

}

}

CaluculatorTests.cs

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibraryTests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void Init()

{

calc = new SimpleCalculator();

}

[TearDown]

public void Cleanup()

{

calc.AllClear();

}

// ✅ Addition Tests

[Test]

[TestCase(10, 5, 15)]

[TestCase(-2, -3, -5)]

[TestCase(0, 0, 0)]

[TestCase(100.5, 200.25, 300.75)]

public void TestAddition(double a, double b, double expected)

{

double result = calc.Addition(a, b);

Assert.That(result, Is.EqualTo(expected));

}

// ➖ Subtraction Tests

[Test]

[TestCase(10, 5, 5)]

[TestCase(-2, -3, 1)]

[TestCase(5, 10, -5)]

public void TestSubtraction(double a, double b, double expected)

{

double result = calc.Subtraction(a, b);

Assert.That(result, Is.EqualTo(expected));

}

// ✖️ Multiplication Tests

[Test]

[TestCase(2, 3, 6)]

[TestCase(-2, 3, -6)]

[TestCase(0, 10, 0)]

public void TestMultiplication(double a, double b, double expected)

{

double result = calc.Multiplication(a, b);

Assert.That(result, Is.EqualTo(expected));

}

// ➗ Division Tests

[Test]

[TestCase(10, 2, 5)]

[TestCase(-10, 2, -5)]

[TestCase(9, 3, 3)]

public void TestDivision(double a, double b, double expected)

{

double result = calc.Division(a, b);

Assert.That(result, Is.EqualTo(expected));

}

// ❌ Division by Zero Test

[Test]

public void Division\_ByZero\_ThrowsException()

{

Assert.Throws<System.ArgumentException>(() => calc.Division(10, 0));

}

}

}

Output:

