**Hands-On: Stage 3 – .NET Advanced Features** ï **NUnit Day 1 - Session-2**

Parameterized test cases

1. Create test case to verify the subtraction feature of the calculator with various input types.

· Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

· Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

· Use Assert.Equal to check the actual and expected results

2. Create a test case to verify the multiplication concepts of calculator

3. Create a test case to verify the division logic of the calculator

· In one of the inputs, provide the divisor value to be 0

· Use Try Catch block to catch the ArgumentException

· Use Assert.Fail to notify the user that the test case has failed. Give the message “Division by zero” in the Assert.Fail, which will be notified to the user. This message will be seen in the test explorer.

Test void methods

In the MathLibrary class there is a property “GetResult”. The result of every operation is stored in a variable ‘result’. This value is accessed by the property.

The class also has a method “AllClear” that sets the value of the result variable to 0.

· Create a test method ‘TestAddAndClear’

· Invoke the Addition method of the math class library

· Verify if the expected and Actual results match using Assert.AreEqual

· Invoke the ‘AllClear’ method

· Use Assert.AreEqual to check if the result is 0 or not

**PROJECT CODE:**

SimpleCalculator.cs

using System;

namespace CalcLibrary

{

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; }

}

}

}

IMathLibrary.cs

using System;

using System.Collections.Generic;

using System.Text;

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);

}

}

SimpleCalculatorTest.cs

using CalcLibrary;

using NUnit.Framework;

using System;

namespace CalcLibrary\_Tests

{

[TestFixture]

public class SimpleCalculatorTests

{

private SimpleCalculator \_simpleCalculator;

private double \_result;

[SetUp]

public void Initialize()

{

\_simpleCalculator = new SimpleCalculator();

\_result = 0;

}

[Test]

[TestCase(10, 20, 30)]

[TestCase(1, 2, 3)]

public void Addition\_ValidInputs\_ExpectedResult(double a, double b, double expected)

{

\_result = \_simpleCalculator.Addition(a, b);

Assert.AreEqual(expected, \_result);

}

[Test]

[TestCase(10, 20, -10)]

[TestCase(10, 2, 8)]

public void Subtraction\_ValidInputs\_ExpectedResult(double a, double b, double expected)

{

\_result = \_simpleCalculator.Subtraction(a, b);

Assert.AreEqual(expected, \_result);

}

[Test]

[TestCase(10, 20, 200)]

[TestCase(10, 0, 0)]

public void Multiplication\_ValidInputs\_ExpectedResult(double a, double b, double expected)

{

\_result = \_simpleCalculator.Multiplication(a, b);

Assert.AreEqual(expected, \_result);

}

[Test]

[TestCase(10, 2, 5)]

public void Division\_ValidInputs\_ExpectedResult(double a, double b, double expected)

{

\_result = \_simpleCalculator.Division(a, b);

Assert.AreEqual(expected, \_result);

}

[Test]

[TestCase(10, 0)]

public void Division\_InValidInputs\_ExpectedException(double a, double b)

{

Assert.Throws<ArgumentException>(() => \_simpleCalculator.Division(a, b));

}

[TearDown]

public void CleanUp()

{

\_simpleCalculator.AllClear();

\_result = \_simpleCalculator.GetResult;

}

}

}

OUTPUT :

