**MOQ Handson**

Calculator Code:

using System;

namespace CalcLibrary

{

public 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; }

}

}

}

NUnit + Moq Test Code:

using System;

namespace CalcLibraryTests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calculator;

[SetUp]

public void Setup()

{

calculator = new SimpleCalculator();

}

[Test]

public void Addition\_ShouldReturnCorrectSum()

{

double result = calculator.Addition(10, 5);

Assert.AreEqual(15, result);

}

[Test]

public void Subtraction\_ShouldReturnCorrectDifference()

{

double result = calculator.Subtraction(10, 5);

Assert.AreEqual(5, result);

}

[Test]

public void Multiplication\_ShouldReturnCorrectProduct()

{

double result = calculator.Multiplication(10, 5);

Assert.AreEqual(50, result);

}

[Test]

public void Division\_ShouldReturnCorrectQuotient()

{

double result = calculator.Division(10, 2);

Assert.AreEqual(5, result);

}

[Test]

public void Division\_ByZero\_ShouldThrowException()

{

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

}

[Test]

public void AllClear\_ShouldResetResultToZero()

{

calculator.Addition(5, 5);

calculator.AllClear();

Assert.AreEqual(0, calculator.GetResult);

}

}

[TestFixture]

public class MockInterfaceTests

{

[Test]

public void Mock\_Addition\_ShouldReturnExpectedValue()

{

var mock = new Mock<IMathLibrary>();

mock.Setup(m => m.Addition(5, 5)).Returns(10);

double result = mock.Object.Addition(5, 5);

Assert.AreEqual(10, result);

mock.Verify(m => m.Addition(5, 5), Times.Once());

}

}

}

**OUTPUT:**

