| **1. NUnit-Handson**  **TestFixture & Test** |
| --- |

**SimpleCalculatorTests.cs**

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcTests

{

[TestFixture]

public class SimpleCalculatorTests

{

private SimpleCalculator \_calc;

[SetUp]

public void Setup()

{

\_calc = new SimpleCalculator();

}

[TearDown]

public void Teardown()

{

\_calc.AllClear();

}

[TestCase(2, 3, 5)]

[TestCase(-1, 1, 0)]

public void Addition\_WorksCorrectly(double a, double b, double expected)

{

var result = \_calc.Addition(a, b);

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

}

[TestCase(5, 3, 2)]

public void Subtraction\_WorksCorrectly(double a, double b, double expected)

{

var result = \_calc.Subtraction(a, b);

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

}

[TestCase(4, 2, 8)]

public void Multiplication\_WorksCorrectly(double a, double b, double expected)

{

var result = \_calc.Multiplication(a, b);

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

}

[TestCase(10, 2, 5)]

public void Division\_WorksCorrectly(double a, double b, double expected)

{

var result = \_calc.Division(a, b);

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

}

[Test]

public void Division\_ByZero\_ThrowsException()

{

var ex = Assert.Throws<ArgumentException>(() => \_calc.Division(5, 0));

Assert.That(ex.Message, Is.EqualTo("Second Parameter Can't be Zero"));

}

[Test]

public void GetResult\_ReturnsCorrectValueAfterOperation()

{

\_calc.Addition(2, 3);

Assert.That(\_calc.GetResult, Is.EqualTo(5));

}

[Test]

public void AllClear\_SetsResultToZero()

{

\_calc.Addition(10, 20);

\_calc.AllClear();

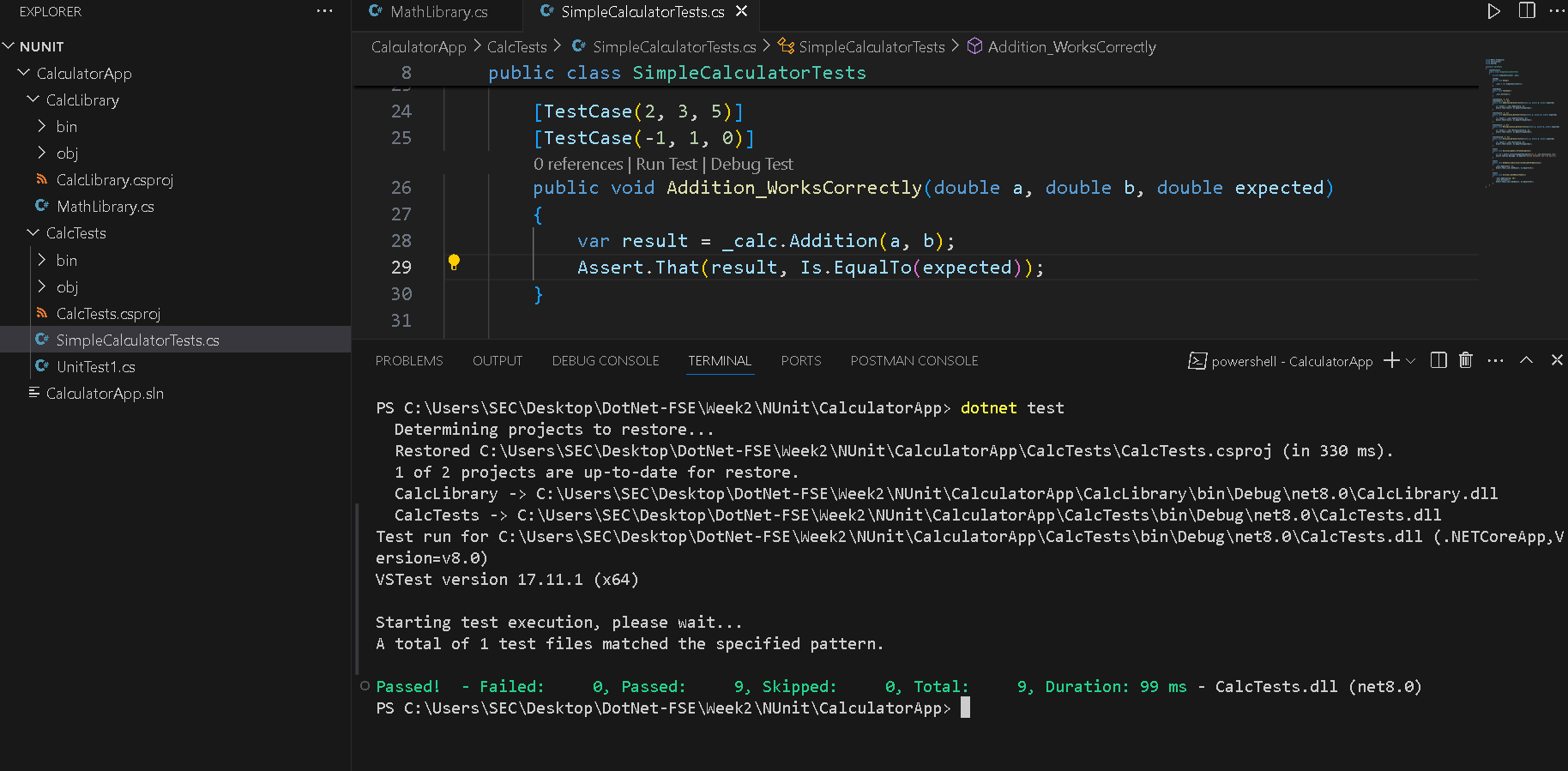
Assert.That(\_calc.GetResult, Is.EqualTo(0));

}

}

}

**OUTPUT:**

****

|  |
| --- |