NUnit Calculator Test - Code And Output

1. **Test Code**

using CalcLibrary;

using NUnit.Framework;

using System;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator \_calculator;

[SetUp]

public void SetUp()

{

\_calculator = new Calculator();

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(0, 0, 0)]

public void Add\_ShouldReturnCorrectSum(int a, int b, int expected)

{

var result = \_calculator.Add(a, b);

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

}

[Test]

[TestCase(5, 3, 2)]

[TestCase(0, 0, 0)]

public void Subtract\_ShouldReturnCorrectDifference(int a, int b, int expected)

{

var result = \_calculator.Subtract(a, b);

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

}

[Test]

[TestCase(2, 3, 6)]

[TestCase(0, 10, 0)]

public void Multiply\_ShouldReturnCorrectProduct(int a, int b, int expected)

{

var result = \_calculator.Multiply(a, b);

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

}

[Test]

[TestCase(10, 2, 5)]

[TestCase(9, 3, 3)]

public void Divide\_ShouldReturnCorrectQuotient(int a, int b, double expected)

{

var result = \_calculator.Divide(a, b);

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

}

[Test]

public void Divide\_ByZero\_ShouldThrowException()

{

Assert.Throws<DivideByZeroException>(() => \_calculator.Divide(10, 0));

}

}

}

1. **Calculator.cs**

using System;

namespace CalcLibrary

{

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

public int Subtract(int a, int b)

{

return a - b;

}

public int Multiply(int a, int b)

{

return a \* b;

}

public double Divide(int a, int b)

{

if (b == 0)

throw new DivideByZeroException("Cannot divide by zero.");

return (double)a / b;

}

}

}

1. **Output**

