# **Week-2 Hands-On ( Nunit and MOQ )**

***Nunit***

***Simple calculator.cs***

*namespace CalcLibrary*

*{*

*public class SimpleCalculator*

*{*

*public double Addition(double a, double b) => a + b;*

*public double Subtraction(double a, double b) => a - b;*

*public double Multiplication(double a, double b) => a \* b;*

*public double Division(double a, double b) => b != 0 ? a / b : throw new DivideByZeroException();*

***CalculatorTests.cs***

*using CalcLibrary;*

*using NUnit.Framework;*

*using System;*

*using NUnit.Framework;*

*using CalcLibrary;*

*namespace CalcLibraryTests*

*{*

*[TestFixture]*

*public class CalculatorTests*

*{*

*private Calculator calc;*

*[SetUp]*

*public void Init()*

*{*

*calc = new Calculator();*

*}*

*[TearDown]*

*public void Cleanup()*

*{*

*calc = null;*

*}*

*[TestCase(10, 5, 15)]*

*[TestCase(-3, 3, 0)]*

*public void Add\_TestCases(int a, int b, int expected)*

*{*

*var result = calc.Add(a, b);*

*TestContext.WriteLine($"Addition Result: {a} + {b} = {result}");*

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

*}*

*[TestCase(10, 5, 5)]*

*public void Subtract\_TestCases(int a, int b, int expected)*

*{*

*var result = calc.Subtract(a, b);*

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

*}*

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

*public void Multiply\_TestCases(int a, int b, int expected)*

*{*

*var result = calc.Multiply(a, b);*

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

*}*

*[TestCase(10.0, 2.0, 5.0)]*

*public void Divide\_TestCases(double a, double b, double expected)*

*{*

*var result = calc.Divide(a, b);*

*Assert.That(result, Is.EqualTo(expected).Within(0.0001));*

*}*

*[Test]*

*public void Divide\_ByZero\_ThrowsException()*

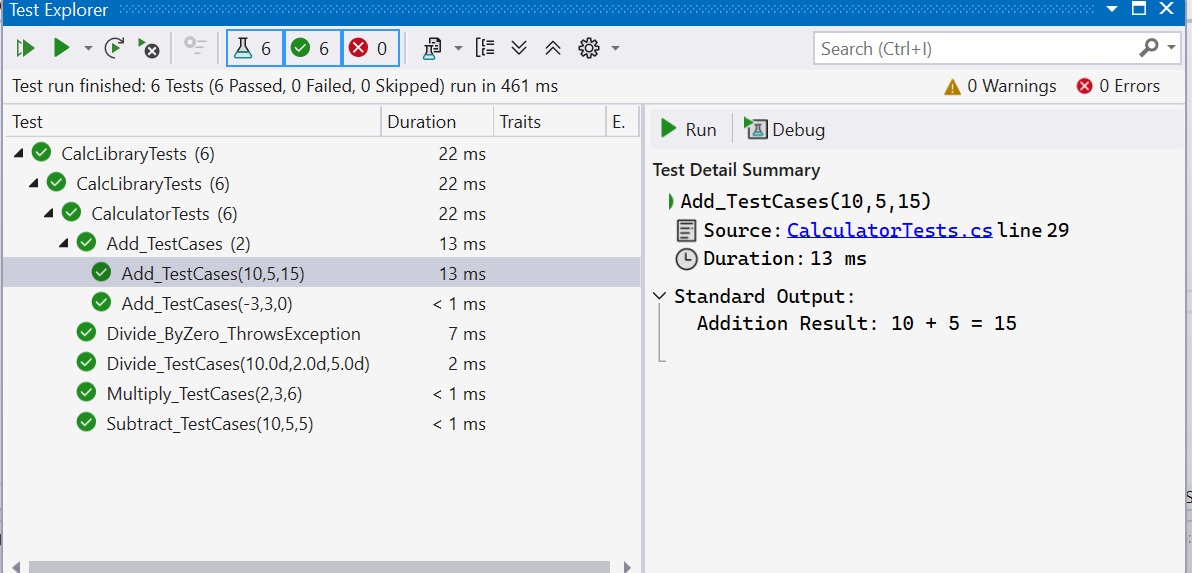
*{*

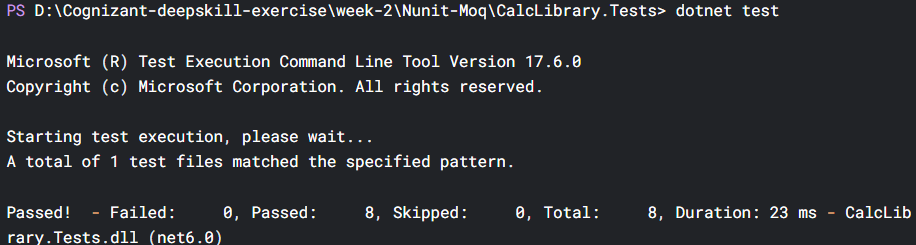
*Assert.Throws<DivideByZeroException>(() => calc.Divide(10, 0));*

*}*

*}*

*}*





**MOQ**

*Task-1: Write Testable Code with Moq*

**CustomerComm.cs**

***namespace CustomerCommLib***

***{***

***public class CustomerComm***

***{***

***private IMailSender \_mailSender;***

***public CustomerComm(IMailSender mailSender)***

***{***

***\_mailSender = mailSender;***

***}***

***public bool SendMailToCustomer()***

***{***

***return \_mailSender.SendMail("cust123@abc.com", "Some Message");***

***}***

***}***

***}***

**MainSender.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomerCommLib

{

public class Class1

{

}

}

**IMailSender.cs**

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

***Task-2***

**CustomerCommTests.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommLib.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> mockMailSender;

private CustomerComm customerComm;

[OneTimeSetUp]

public void Init()

{

mockMailSender = new Mock<IMailSender>();

mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>()))

.Returns(true);

customerComm = new CustomerCommLib.CustomerComm(mockMailSender.Object);

}

[TestCase]

public void SendMailToCustomer\_ShouldReturnTrue\_WhenMailIsSent()

{

// Act

var result = customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.True);

mockMailSender.Verify(m => m.SendMail("cust123@abc.com", "Some Message"), Times.Once);

}

}

}

