1. NUnit-Handson

﻿using System;

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);

}

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; }

}

}

}

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void SetUp()

{

calc = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

// Optional cleanup

}

[TestCase(2, 3, 5)]

[TestCase(-1, -1, -2)]

[TestCase(0, 0, 0)]

public void Add\_WhenCalled\_ReturnsCorrectSum(int a, int b, int expectedResult)

{

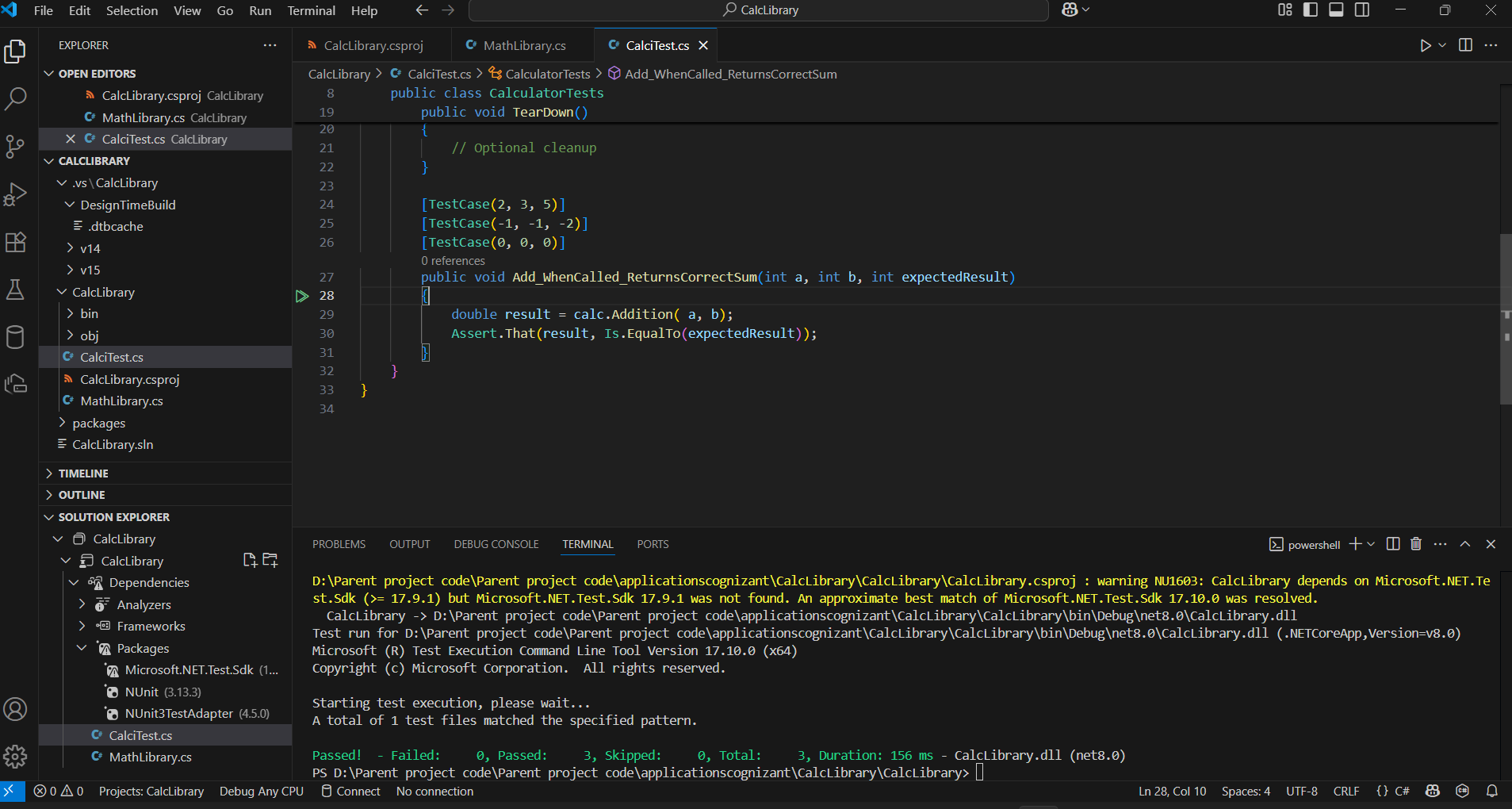
double result = calc.Addition( a, b);

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

}

}

}



|  |  |
| --- | --- |
| 1. Moq-Handson | 1. Write Testable Code with Moq |

namespace EmailService

{

public interface IEmailSender

{

bool SendEmail(string to, string subject, string body);

}

public class NotificationService

{

private readonly IEmailSender \_emailSender;

public NotificationService(IEmailSender emailSender)

{

\_emailSender = emailSender;

}

public string SendNotification(string userEmail)

{

bool result = \_emailSender.SendEmail(userEmail, "Welcome!", "Thanks for signing up!");

if (result)

return "Email sent";

else

return "Failed to send email";

}

}

}

