**NUNIT-HANDSON**

**MathLibrary.cs**

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; }

}

}

}

**CalculatorTests.cs**

using NUnit.Framework;

using CalcLibrary;

namespace CalculatorTests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void Init()

{

calc = new SimpleCalculator();

}

[TearDown]

public void Cleanup()

{

calc = null;

}

[Test]

[TestCase(5, 3, 8)]

[TestCase(10, -5, 5)]

[TestCase(0, 0, 0)]

public void AdditionTest(double a, double b, double expected)

{

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

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

}

[Test]

[Ignore("This test is intentionally ignored.")]

public void IgnoredTest()

{

Assert.Fail("This test is ignored.");

}

}

}

**UnitTest1.cs**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CalculatorTests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

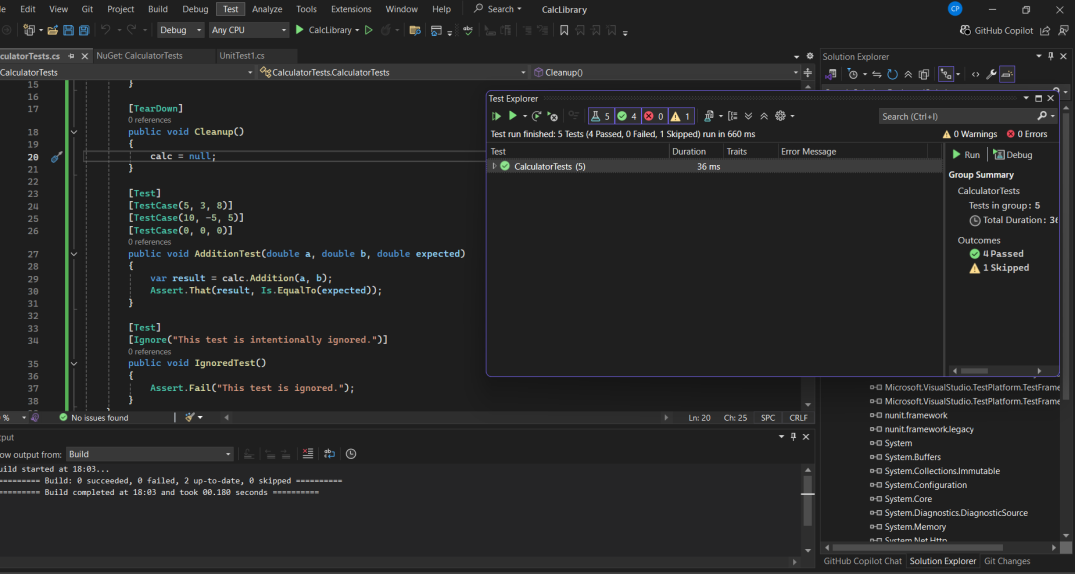
{

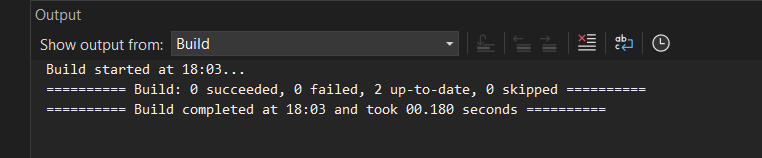
}

}

}

**OUTPUT**

****

****

**Moq-Handson**

CustomerComm.cs

namespace CustomerCommLib

{

public class CustomerComm

{

IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

// Simulate sending mail

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

return true;

}

}

}

MailSender.cs

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

public class MailSender : IMailSender

{

public bool SendMail(string toAddress, string message)

{

// Simulate sending an email

MailMessage mail = new MailMessage();

SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress("your\_email\_address@gmail.com");

mail.To.Add(toAddress);

mail.Subject = "Test Mail";

mail.Body = message;

smtpServer.Port = 587;

smtpServer.Credentials = new NetworkCredential("username", "password");

smtpServer.EnableSsl = true;

// For real email: smtpServer.Send(mail);

// For test purposes, just return true to simulate success

return true;

}

}

}

AssemblyInfo.cs

using System.Reflection;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

[assembly: AssemblyTitle("CustomerCommLibTests")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("")]

[assembly: AssemblyProduct("CustomerCommLibTests")]

[assembly: AssemblyCopyright("Copyright © 2025")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

[assembly: ComVisible(false)]

[assembly: Guid("e11c57f0-5cd4-499f-b6b3-003a322b1dbd")]

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.0")]

[assembly: AssemblyFileVersion("1.0.0.0")]

UnitTest1.cs

using CustomerCommLib;

using Moq;

using NUnit.Framework;

namespace CustomerCommLibTests

{

[TestFixture]

public class CustomerCommTests

{

[Test]

public void SendMailToCustomer\_ShouldReturnTrue\_WhenMailIsSentSuccessfully()

{

// Arrange

var mockMailSender = new Mock<IMailSender>();

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

var customerComm = new CustomerComm(mockMailSender.Object);

// Act

bool result = customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.True);

}

}

}

OUTPUT

