WEEK 2 HANDS ON

Exercise 1:NUnit

- testing program to validate a calculator addition operation

CODE:

[1 . Production Code (already supplied in CalcLibrary)]

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; }

}

}

}

[2. Unit Test Project Code (created - CalcLibrary. Tests)]

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class SimpleCalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void Setup()

{

calc = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

calc = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(-1, 1, 0)]

[TestCase(0, 0, 0)]

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

{

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

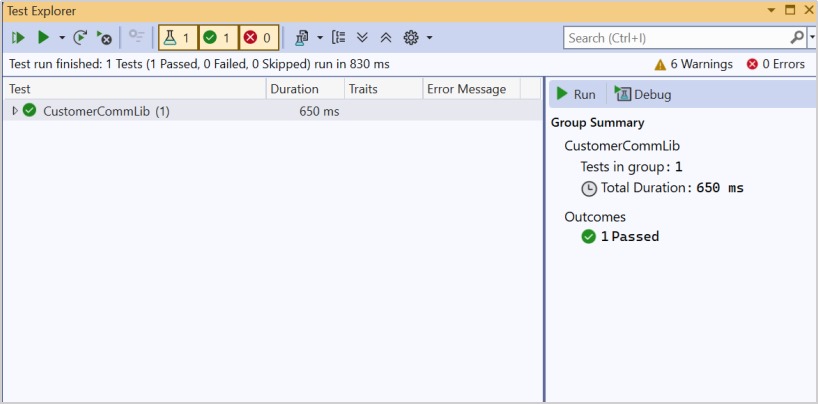
Assert.That(result, Is.EqualTo(expected));

}

}

}

OUTPUT:



Exercise 2:MOQ

- write testable code with MOQ

CODE:

[1a . Production Code (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)

{

MailMessage mail = new MailMessage();

SmtpClient smtp = 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;

smtp.Port = 587;

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

smtp.EnableSsl = true;

smtp.Send(mail);

return true;

}

}

}

[1b . Production Code (CustomerComm.cs)]

namespace CustomerCommLib

{

public class CustomerComm

{

private readonly IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

string email = "cust123@abc.com";

string message = "Some Message";

return \_mailSender.SendMail(email, message);

}

}

}

[2 . Unit Test Code (CustomerCommTests.cs)]

using NUnit.Framework;

using Moq;

using Assert = NUnit.Framework.Assert;

namespace CustomerCommLib

{

[TestFixture]

public class CustomerCommTests

{

[Test]

public void SendMailToCustomer\_ShouldCallSendMailOnce\_AndReturnTrue()

{

var mockSender = new Mock<IMailSender>();

mockSender

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

.Returns(true);

var comm = new CustomerComm(mockSender.Object);

var result = comm.SendMailToCustomer();

Assert.IsTrue(result);

mockSender.Verify(

m => m.SendMail("cust123@abc.com", "Some Message"),

Times.Once);

}

}

}

OUTPUT:

