**1. NUnit-Handson**

**Exercise: TestFixture & Test**

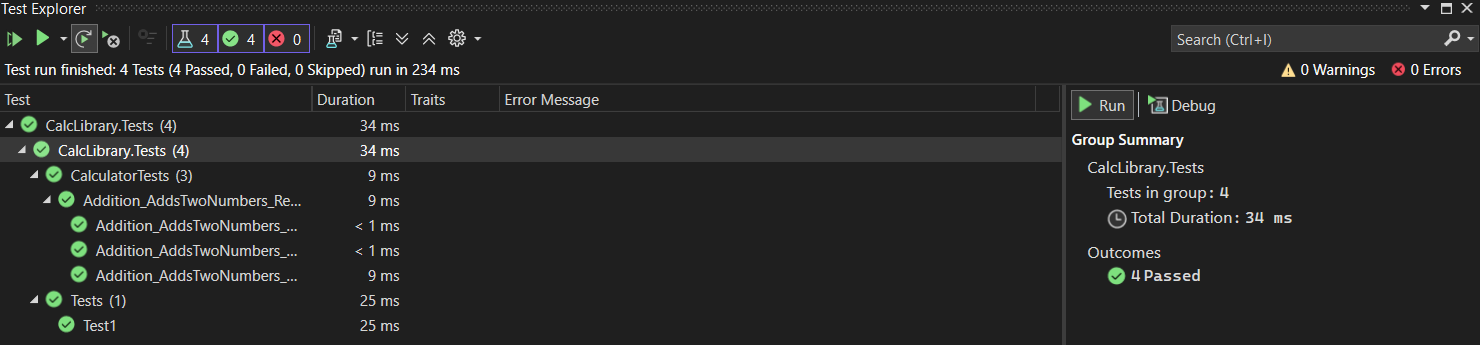
Code:

using NUnit.Framework;   
using CalcLibrary;

namespace CalcLibrary.Tests   
{   
 [TestFixture]   
 public class CalculatorTests   
 {   
 private SimpleCalculator \_calculator;  
 [SetUp]  
 public void Setup()  
 {  
 \_calculator = new SimpleCalculator();  
 }  
  
 [TearDown]  
 public void Teardown()  
 {  
 \_calculator = null;  
 }  
  
 [Test]  
 [TestCase(2.1, 3.7, 5.8)]  
 [TestCase(-1.3, -1.0, -2.3)]  
 [TestCase(0, 0, 0)]  
 public void Addition\_AddsTwoNumbers\_ReturnsCorrectResult(double a, double b, double expected)  
 {  
 var result = \_calculator.Addition(a, b);  
 Assert.That(result, Is.EqualTo(expected).Within(0.0001));  
 }  
}

}

Output:



**1. Moq-Handson**

**1. Write Testable Code with Moq**

Task 1:

Code:

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

namespace CustomerCommLib   
{   
 public class MailSender : IMailSender   
 {  
 public bool SendMail(string toAddress, string message)   
 {   
 MailMessage mail = new MailMessage(); SmtpClient smtpServer = new SmtpClient("smtp.gmail.com");  
 mail.From = new [MailAddress("your\_email\_address@gmail.com](mailto: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;  
  
 smtpServer.Send(mail);  
 return true;  
 }  
}

}

namespace CustomerCommLib

{

public class CustomerComm

{

private IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

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

return true;

}

}

}

Task 2:  
Code:

namespace CustomerComm.Tests   
{   
 [TestFixture] public class CustomerCommTests   
 {   
 private Mock \_mockMailSender; private CustomerCommLib.CustomerComm \_customerComm;

[OneTimeSetUp]  
 public void Setup()  
 {  
 \_mockMailSender = new Mock<IMailSender>();  
 \_customerComm = new CustomerCommLib.CustomerComm(\_mockMailSender.Object);  
 }  
  
 [TestCase]  
 public void SendMailToCustomer\_WhenCalled\_ReturnsTrue()  
 {  
 \_mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);  
  
 bool result = \_customerComm.SendMailToCustomer();  
  
 Assert.IsTrue(result);  
 \_mockMailSender.Verify(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>()), Times.Once());  
 }  
 }  
  
}  
  
Output:

