EX-1 NUnit-Handson

Calculator.cs

```
namespace CalcLibrary
{
   public class Calculator
   {
      public int Add(int a, int b)
      {
         return a + b;
      }

      // You can add more methods later: Subtract, Multiply, Divide
   }
}
```

CalculatorTests.cs

```
using NUnit.Framework;
using CalcLibrary;  // Replace with your actual namespace

namespace CalculatorUnitTests
{
   [TestFixture]
   public class CalculatorTests
   {
      private Calculator _calculator;

      [SetUp]
      public void Setup()
      {
         // This runs before each test
         _calculator = new Calculator();
      }

      [TearDown]
      public void TearDown()
      {
         // This runs after each test
         _calculator = null;
      }
```

```
    [Test]
    [TestCase(2, 3, 5)]
    [TestCase(-1, 1, 0)]
    [TestCase(0, 0, 0)]
    [TestCase(100, 200, 300)]
    public void Add_ValidInputs_ReturnsCorrectResult(int a, int b, int expected)
    {
        int result = _calculator.Add(a, b);
        Assert.That(result, Is.EqualTo(expected));
    }

    [Test, Ignore("Subtraction is not implemented yet")]
    public void Subtract_ShouldBeImplementedLater()
    {
        // Placeholder for future test
    }
  }
}
```

Output

✓ Add_ValidInputs_ReturnsCorrectResult(2,3,5)
✓ Add_ValidInputs_ReturnsCorrectResult(-1,1,0)
✓ Add_ValidInputs_ReturnsCorrectResult(0,0,0)
✓ Add_ValidInputs_ReturnsCorrectResult(100,200,300)
⚠ Subtract_ShouldBeImplementedLater - Ignored

EX-2 Moq-Handson

IMailSender.cs

```
namespace CustomerCommLib
{
   public interface IMailSender
   {
      bool SendMail(string toAddress, string message);
   }
}
```

MailSender.cs

```
using System.Net;
using System.Net.Mail;

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");
         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;
      }
   }
}
```

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);
      }
   }
}
```

Program.cs

```
using CustomerCommLib;
using System;

namespace CustomerCommApp
{
   internal class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Sending mail to customer...");

         IMailSender sender = new MailSender();
         CustomerComm comm = new CustomerComm(sender);

         bool result = comm.SendMailToCustomer();

         Console.WriteLine(result ? "Mail sent successfully." : "Failed to send mail.");
      }
   }
}
```

CustomerCommTests.cs

```
using NUnit.Framework;
using Moq;
using CustomerCommLib;

namespace CustomerComm.Tests
{
   [TestFixture]
   public class CustomerCommTests
   {
      private Mock<IMailSender> _mockMailSender;
      private CustomerComm _customerComm;

      [OneTimeSetUp]
      public void Setup()
      {
         _mockMailSender = new Mock<IMailSender>();

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

         _customerComm = new CustomerComm(_mockMailSender.Object);
      }

      [Test]
      public void SendMailToCustomer_ShouldReturnTrue_WhenMockReturnsTrue()
      {
         bool result = _customerComm.SendMailToCustomer();

         Assert.That(result, Is.True);
      }
   }
}
```

OUTPUT

Sending mail to customer...
Mail sent successfully.

System.Net.Mail.SmtpException: The SMTP server requires a secure connection or the client was not authenticated.