Exercise 1: Nunit

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 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; // Return true after sending

        }

    }

    public class CustomerComm

    {

        IMailSender \_mailSender;

        public CustomerComm(IMailSender mailSender)

        {

            \_mailSender = mailSender;

        }

        public bool SendMailToCustomer()

        {

            // define message and mail address

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

            return true;

        }

    }

}



Exercise 2: Moq

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

    [TestFixture]

    public class CustomerCommTests

    {

        private Mock<IMailSender> \_mockMailSender;

        private CustomerComm \_customerComm;

        [OneTimeSetUp]

        public void Init()

        {

            // Create mock of IMailSender

            \_mockMailSender = new Mock<IMailSender>();

            // Configure mock: SendMail accepts any string arguments and returns true

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

            // Inject mock into CustomerComm

            \_customerComm = new CustomerComm(\_mockMailSender.Object);

        }

        [Test]

        public void SendMailToCustomer\_ShouldReturnTrue()

        {

            // Act

            var result = \_customerComm.SendMailToCustomer();

            // Assert

            Assert.IsTrue(result);

            // Verify that SendMail was called exactly once with any string parameters

            \_mockMailSender.Verify(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>()), Times.Once);

        }

    }

}



