1. NUnit-Handson

| using NUnit.Framework; using CalcLibrary;  namespace CalcLibrary.UnitTests {  [TestFixture]  public class CalculatorTests  {  private SimpleCalculator calculator;   [SetUp]  public void SetUp()  {  calculator = new SimpleCalculator();  }   [TearDown]  public void TearDown()  {  calculator.AllClear();  }   [TestCase(1, 2, 3)]  [TestCase(-1, -2, -3)]  [TestCase(0, 0, 0)]  [TestCase(-1, 1, 0)]  [TestCase(1.5, 2.5, 4)]  public void Addition\_ReturnsExpectedResult(double a, double b, double expected)  {  var result = calculator.Addition(a, b);  Assert.That(result, Is.EqualTo(expected).Within(0.0001));  }  } } |
| --- |

Moq-Handson

1. Write Testable Code with Moq

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommTests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private CustomerCommLib.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 CustomerCommLib.CustomerComm(\_mockMailSender.Object);

}

[TestCase]

public void SendMailToCustomer\_ShouldReturnTrue()

{

var result = \_customerComm.SendMailToCustomer();

Assert.IsTrue(result);

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

}

}

}