**Handson-1**

using Collection;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using NUnit.Framework;

using System;

using System.Linq;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestFixture]

public class UnitTest1

{

EmployeeManger employeeManger;

[SetUp]

public void Setup()

{

employeeManger = new EmployeeManger();

var emp = employeeManger.GetEmployees();

}

[TestCase]

public void Test1()

{

var containnull = employeeManger.GetEmployees().Contains(null);

Assert.AreEqual(containnull, false);

Console.WriteLine(containnull);

}

[TestCase]

public void Test2()

{

var emp = employeeManger.GetEmployees();

int value = emp.Where(employeeManger => employeeManger.EmpId == 100).Count();

Assert.That(value, Is.EqualTo(1));

Console.WriteLine(value);

}

[TestCase]

public void Test3()

{

var emp = employeeManger.GetEmployees();

var unique = emp.Distinct().Count();

Assert.That(unique, Is.EqualTo(emp.Count));

Console.WriteLine(unique);

}

[TestCase]

public void Test4()

{

var cur = employeeManger.GetEmployees();

var pev = employeeManger.GetEmployeesWhoJoinedInPreviousYears();

Assert.That(cur, Is.EquivalentTo(pev));

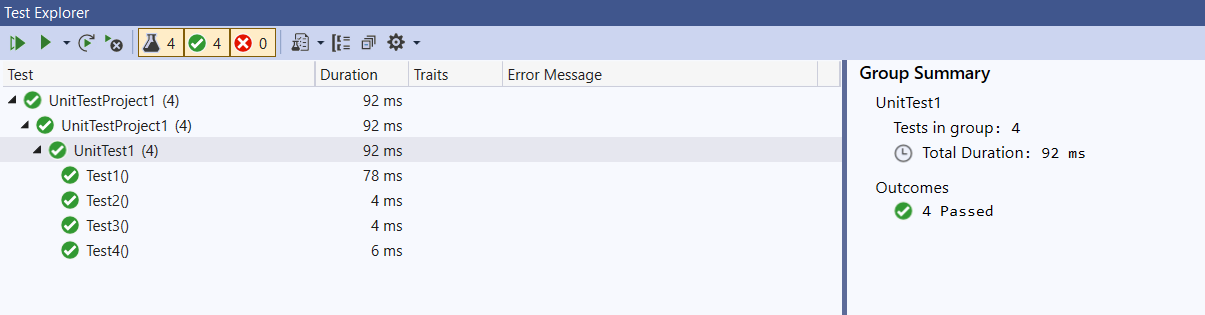
foreach (var i in pev)

Console.WriteLine(i.EmpName);

}

}

}



Handson 2

using Microsoft.VisualStudio.TestTools.UnitTesting;

using NUnit.Framework;

using System;

using Four;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

public class UnitTest1

{

[Test]

[TestCase("February", "Spring")]

[TestCase("May", "Summer")]

[TestCase("July", "Monsoon")]

[TestCase("November", "Autumn")]

[TestCase("December", "Winter")]

public void Test1(string a, string b)

{

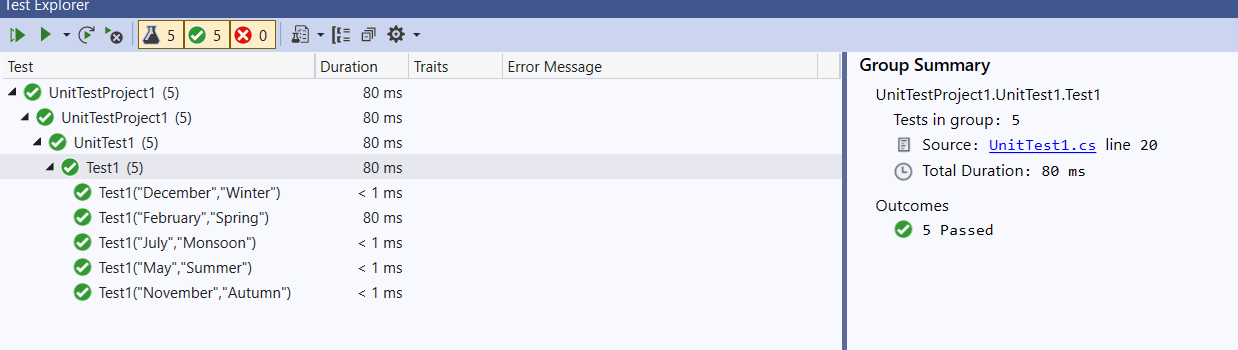
Season season = new Season();

Assert.That(b, Is.EqualTo(season.DisplaySeasonBy(a)));

}

}

}



**Handson 3**

using Leap;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using NUnit.Framework;

using System;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

LeapYear leapYear;

[SetUp]

public void Setup()

{

leapYear = new LeapYear();

}

[Test]

[TestCase(1600, -1)]

[TestCase(2012, 1)]

[TestCase(1995, 0)]

public void Check(int x, int y)

{

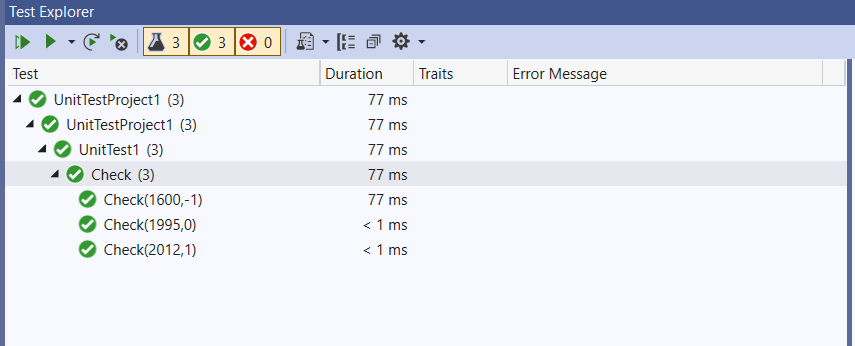
int value = leapYear.IsLeapYear(x);

Assert.AreEqual(value, y);

}

}

}



**Handson-4**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using NUnit.Framework;

using System;

using User;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

[Test]

[TestCase("ABCDE4417Q")]

[TestCase("ZXQRQ7890M")]

[TestCase("NTYLK6792Y")]

public void validpancard(string number)

{

UserManager user = new UserManager();

try

{

user.CreateUser(new UserManager { PANCardNo = number });

}

catch (NullReferenceException e)

{

Assert.Fail(e.Message);

}

catch (FormatException e)

{

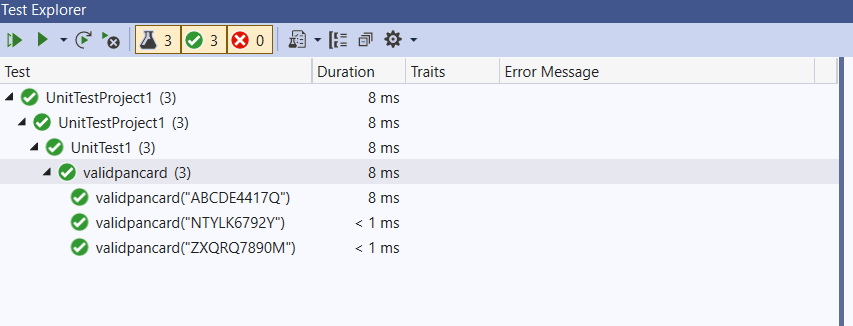
Assert.Fail(e.Message);

}

}

}

}



**Handson-5**

using ConverterLib;

using CurrencyConverter;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Moq;

using NUnit.Framework;

using System;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestFixture]

public class UnitTest1

{

Mock<IDollarToEuroExchangeRateFeed> mock;

[SetUp]

public void Setup()

{

mock = new Mock<IDollarToEuroExchangeRateFeed>();

}

[Test]

public void Test()

{

double usd =8.5;

double x = 1000;

mock.Setup(t => t.GetActualUSDollarValue()).Returns(usd);

Converter converter = new Converter(mock.Object);

var euro = converter.USDToEuro(x);

Assert.AreEqual(euro, 8500);

}

}

}

