INTCDE21ID008

STAGE-3

Day 2 - NUnit

91648 - Naveen S

Hands-On 1:

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally, add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

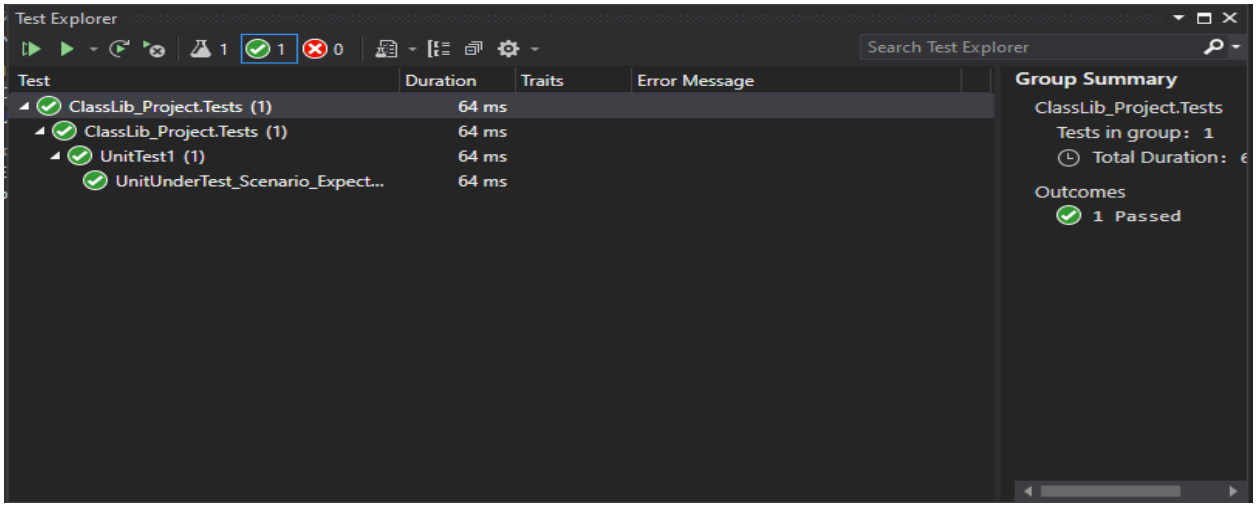
9) Observe the test result.

IMPLEMENTATION:

```
using NUnit.Framework;
using CollectionsLib;
using System.Linq;

namespace ClassLib_Project.Tests
```

```
{
    [TestFixture]
    public class UnitTest1
    {
        [Test]
        public void UnitUnderTest_Scenario_ExpectedOutcom()
        {
            EmployeeManager em = new EmployeeManager();
            var contain_null = em.GetEmployees().Contains(null);
            Assert.AreEqual(contain_null, false);
            var empid = em.GetEmployees();

            int id = empid.Where(e => e.EmpId == 100).Count();
            Assert.That(id, Is.EqualTo(1));


            var unique = empid.Distinct().Count();
            Assert.That(unique, Is.EqualTo(empid.Count));

            var geo = em.GetEmployees();
            var gepyo = em.GetEmployeesWhoJoinedInPreviousYears();
            Assert.That(geo, Is.EquivalentTo(gepyo));


        }
    }
}
```

OUTPUT:



Hands-On 2:

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the ConverterLib project to the test project.

4) Additionally, add the reference of NUnit, NUnit3TestAdapter and Moq in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

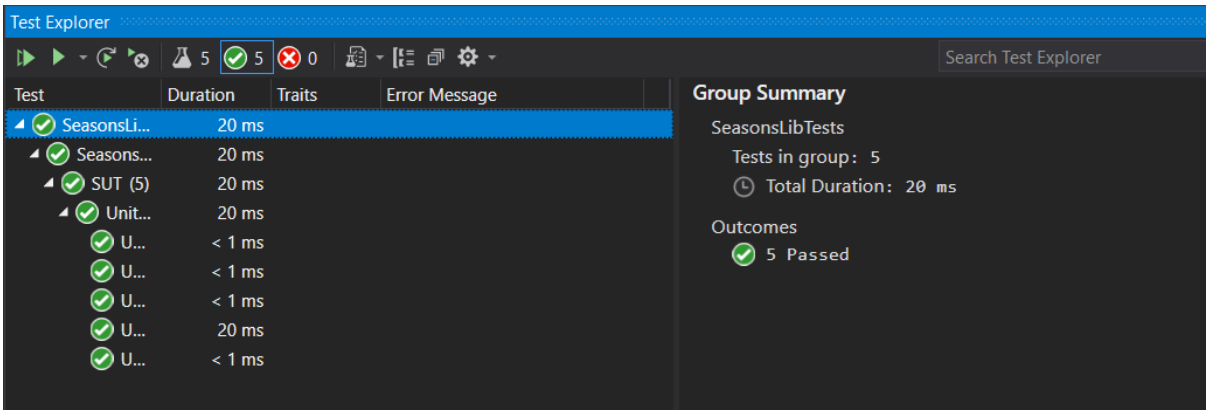
9) Observe the test result.

IMPLEMENTATION:

```
using NUnit.Framework;
using SeasonsLib;
namespace SeasonsLibTests
{
    public class SUT
    {
        SeasonTeller season;
        [SetUp]
        public void Setup()
        {
            season = new SeasonTeller();

        }
        [TearDown]
        public void Teardown()
        {
            season = null;
        }
        [Test]
        [TestCase("February", "Spring")]
        [TestCase("april", "Summer")]
        [TestCase("July", "Monsoon")]
        [TestCase("december", "Winter")]
        [TestCase("ecember", "Invalid Season")]
        public void UnitUnderTest_Scenario_ExpectedOutcome(string a,string expected)
        {
            string result = season.DisplaySeasonBy(a);
            Assert.That(expected, Is.EqualTo(result));
```

```
        }
    }
}
```

OUTPUT:



Hands-On 3:

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the ConverterLib project to the test project.

4) Additionally, add the reference of NUnit, NUnit3TestAdapter and Moq in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

6) Run your tests.

7) Break the test by modifying the source project functionality.

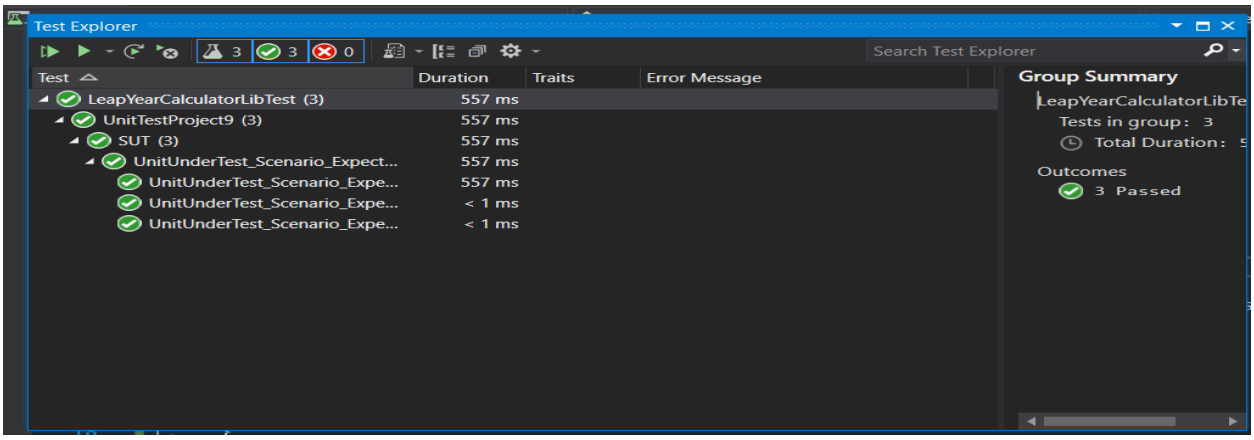
8) Rerun the test.

9) Observe the test result.

IMPLEMENTATION:

```
using NUnit.Framework;
using LeapYearCalculatorLib;

namespace UnitTestProject9

{
    [TestFixture]
    public class SUT
    {
        [Test]
        [TestCase(1752, -1)]
        [TestCase(2000, 1)]
        [TestCase(1998, 0)]
        public void UnitUnderTest_Scenario_ExpectedOutcome(int year, int expectedresult)
        {
            LeapYearCalculator leapYearCalculator = new LeapYearCalculator();
            int result = leapYearCalculator.IsLeapYear(year);
            Assert.That(result, Is.EqualTo(expectedresult));

        }

    }
}
```

OUTPUT:



Hands-On 4:

PANCardNo property reads only 10 characters length value from the user. It is a mandatory property while creating the user.

Following exceptions may occur while creating the user.

o NullReferenceException- If the input value is empty or null

o FormatException-If the input string does not meet the length criteria.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally, add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

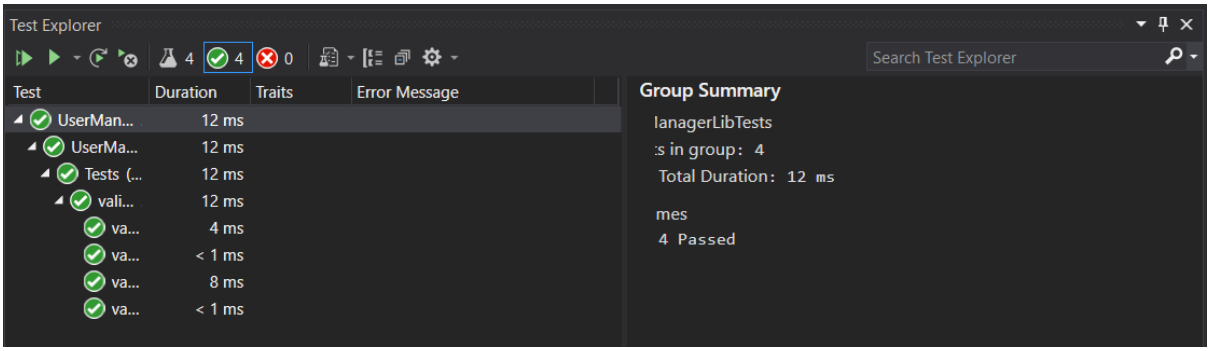
9) Observe the test result.

IMPLEMENTATION:

```
using NUnit.Framework;
using UserManagerLib;
using System;
namespace UserManagerLibTests
{
    public class Tests
    {
        User user;
        [SetUp]
        public void Setup()
        {
            user = new User();
        }
        [Test]
```

```
        [TestCase("CYGPK00189")]
        [TestCase("ABCDEFGHIJ")]
        [TestCase("")]
        [TestCase("HCGYJIL8")]
        public void validpancard(string a)
        {
            try
            {
                user.CreateUser(new User { PANCardNo = a });
            }
            catch (NullReferenceException e)
            {

                Assert.That(e.Message,Is.EqualTo("Invalid Pan Card Number"));
            }
            catch (FormatException e)
            {
                Assert.That(e.Message, Is.EqualTo("Pan Card Number Should contain only 10
characters"));
            }
        }

    }

}
```

OUTPUT:



Hands-On 5:

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the ConverterLib project to the test project.

4) Additionally, add the reference of NUnit, NUnit3TestAdapter and Moq in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

6) Run your tests.

7) Break the test by modifying the source project functionality.

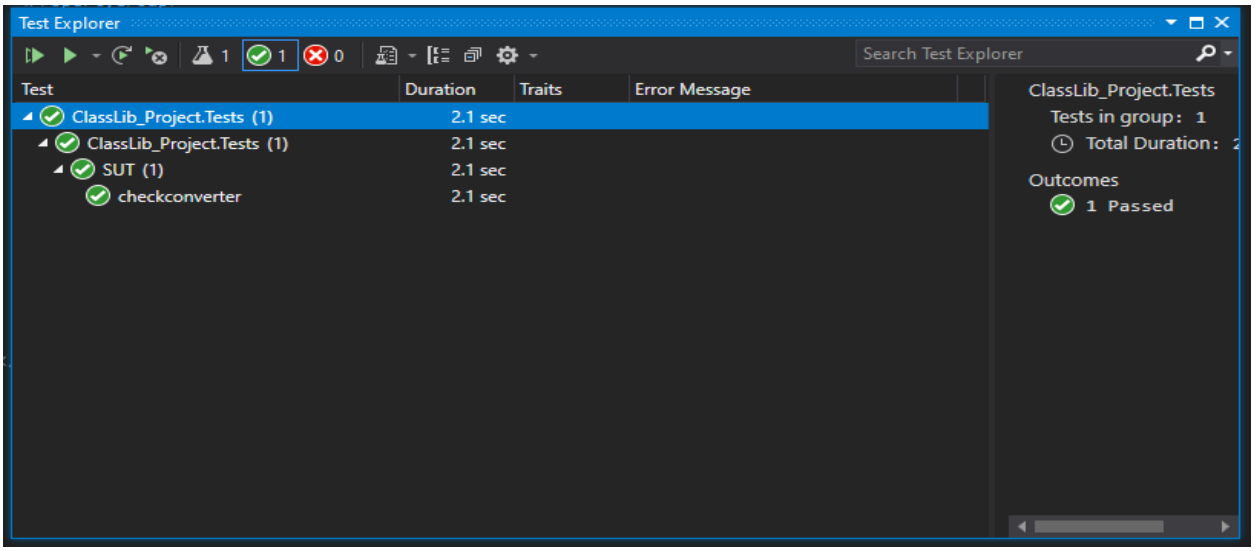
8) Rerun the test.

9) Observe the test result.

IMPLEMENTATION:

```
using NUnit.Framework;
using ConverterLib;
using CurrencyConverterApp;
using Moq;
namespace ClassLib_Project.Tests
{
    [TestFixture]
    public class SUT
    {
        [Test]
        public void checkconverter()
        {
            double usd = 7.5;
            double con = 100;
            Mock<IDollarToEuroExchangeRateFeed> ustoeu = new
Mock<IDollarToEuroExchangeRateFeed>();
            ustoeu.Setup(t => t.GetActualUSDollarValue()).Returns(usd);
            Converter c = new Converter(ustoeu.Object);
            var eu = c.USDToEuro(con);
            Assert.AreEqual(eu, 750);
        }
    }
}
```

OUTPUT:


Test Explorer
1
1
0
Search Test Explorer
Test
Duration
Traits
Error Message
ClassLib_Project.Tests (1)
2.1 sec
ClassLib_Project.Tests (1)
2.1 sec
SUT (1)
2.1 sec
checkconverter
2.1 sec
ClassLib_Project.Tests
Tests in group: 1
Total Duration:
Outcomes
1 Passed