**Hands-On: Stage 3 – .NET Advanced Features** - **NUnit Day 2 - Question 1**

Steps to perform

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.

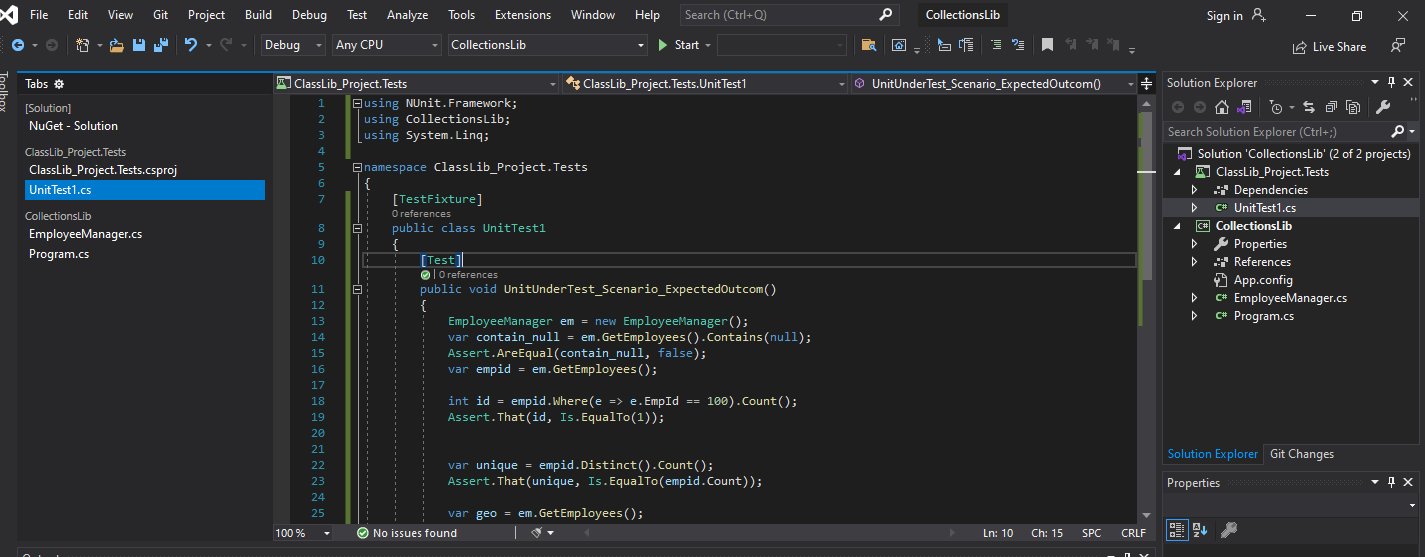
4) Additionally add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

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.

**SOURCE CODE:**

**EmployeeManager.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CollectionsLib

{

public class Employee

{

public int EmpId { get; set; }

public string EmpName { get; set; }

public double Salary { get; set; }

public DateTime DOJ { get; set; }

}

public class EmployeeManager

{

private static readonly List<Employee> employees;

static EmployeeManager()

{

employees = new List<Employee>

{

new Employee { EmpId=100, EmpName="John",DOJ=DateTime.Now.AddYears(-5),Salary=30000},

new Employee { EmpId=101, EmpName="Mary",DOJ=DateTime.Now.AddYears(-2),Salary=10000},

new Employee { EmpId=102, EmpName="Steve",DOJ=DateTime.Now.AddYears(-2),Salary=10000},

new Employee { EmpId=103, EmpName="Allen",DOJ=DateTime.Now.AddYears(-7),Salary=50000},

};

}

public List<Employee> GetEmployees()

{

return employees;

}

public List<Employee> GetEmployeesWhoJoinedInPreviousYears()

{

return employees.FindAll(x=>x.DOJ<DateTime.Now);

}

}

}

**UnitTest1.cs**

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();

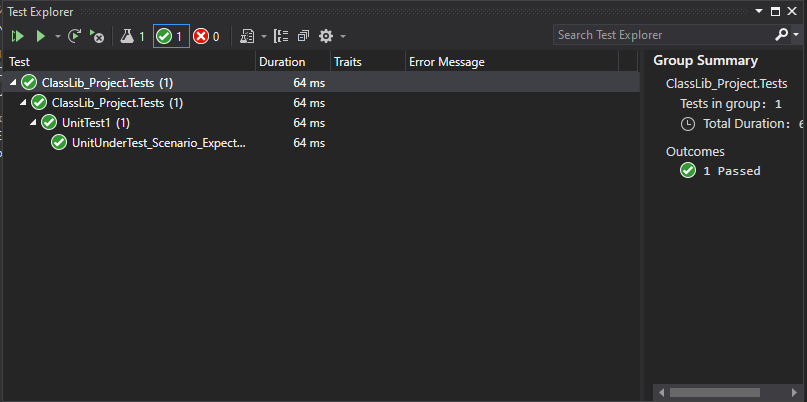
Assert.That(geo, Is.EquivalentTo(gepyo));

}

}

}

**OUTPUT:**



**Hands-On: Stage 3 – .NET Advanced Features** - **NUnit Day 2 - Question 2**

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).

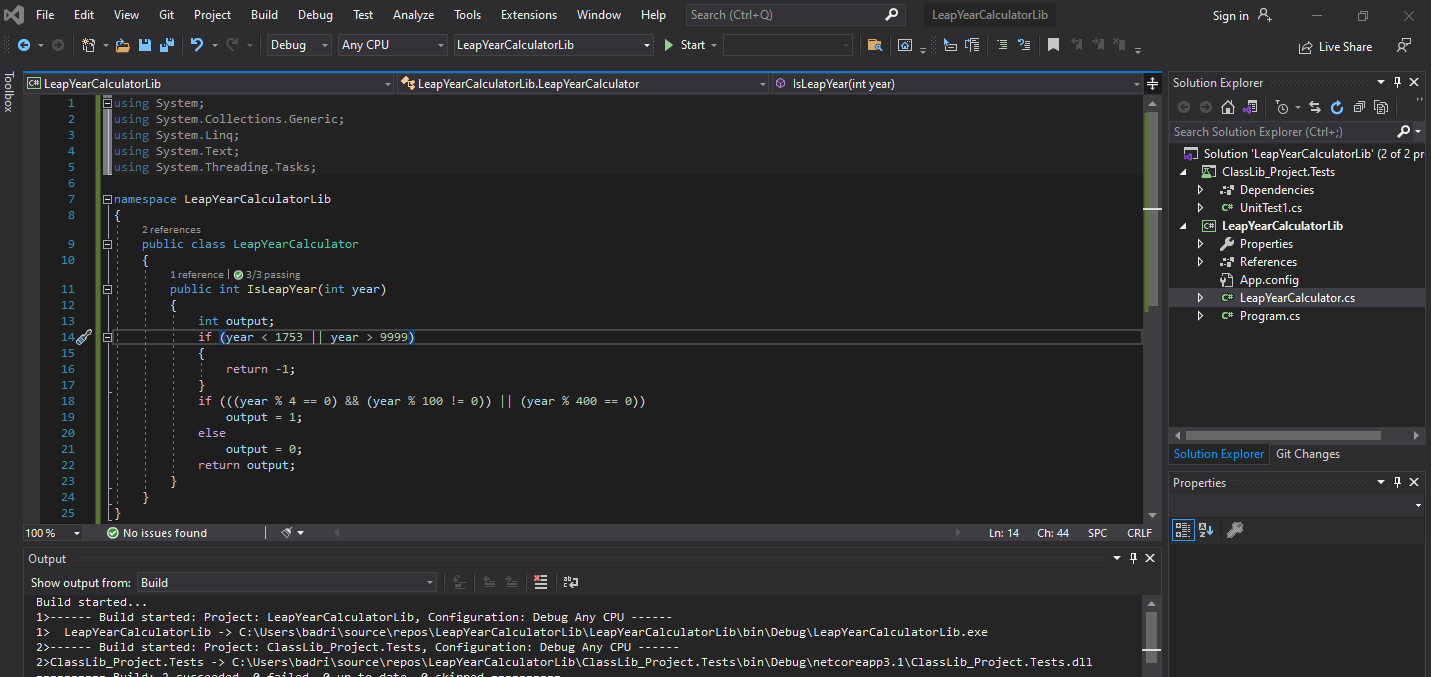
5) Write the suggested test methods.

6) Run your tests.

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

8) Rerun the test.

9) Observe the test result

**SOURCE CODE:**

**LeapYearCalculator.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LeapYearCalculatorLib

{

public class LeapYearCalculator

{

public int IsLeapYear(int year)

{

int output;

if (year < 1753 || year > 9999)

{

return -1;

}

if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))

output = 1;

else

output = 0;

return output;

}

}

}

**UnitTest1.cs**

using NUnit.Framework;

using LeapYearCalculatorLib;

namespace ClassLib\_Project.Tests

{

[TestFixture]

public class SUT

{

[Test]

[TestCase(1750,-1)]

[TestCase(2010,0)]

[TestCase(2012,1)]

public void UnitUnderTest\_Scenario\_ExpectedOutcome(int year , int expectedoutput)

{

LeapYearCalculator lyc = new LeapYearCalculator();

int result = lyc.IsLeapYear(year);

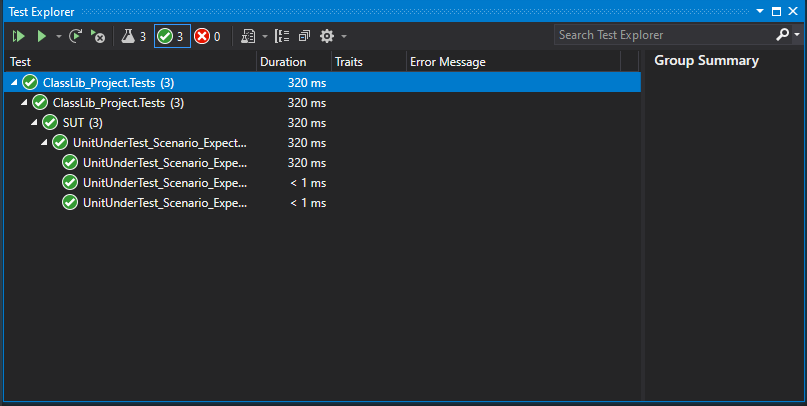
Assert.That(result,Is.EqualTo(expectedoutput));

}

}

}

**OUTPUT:**

****

**Hands-On: Stage 3 – .NET Advanced Features** - **NUnit Day 2 - Question 3**

Steps to perform

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).

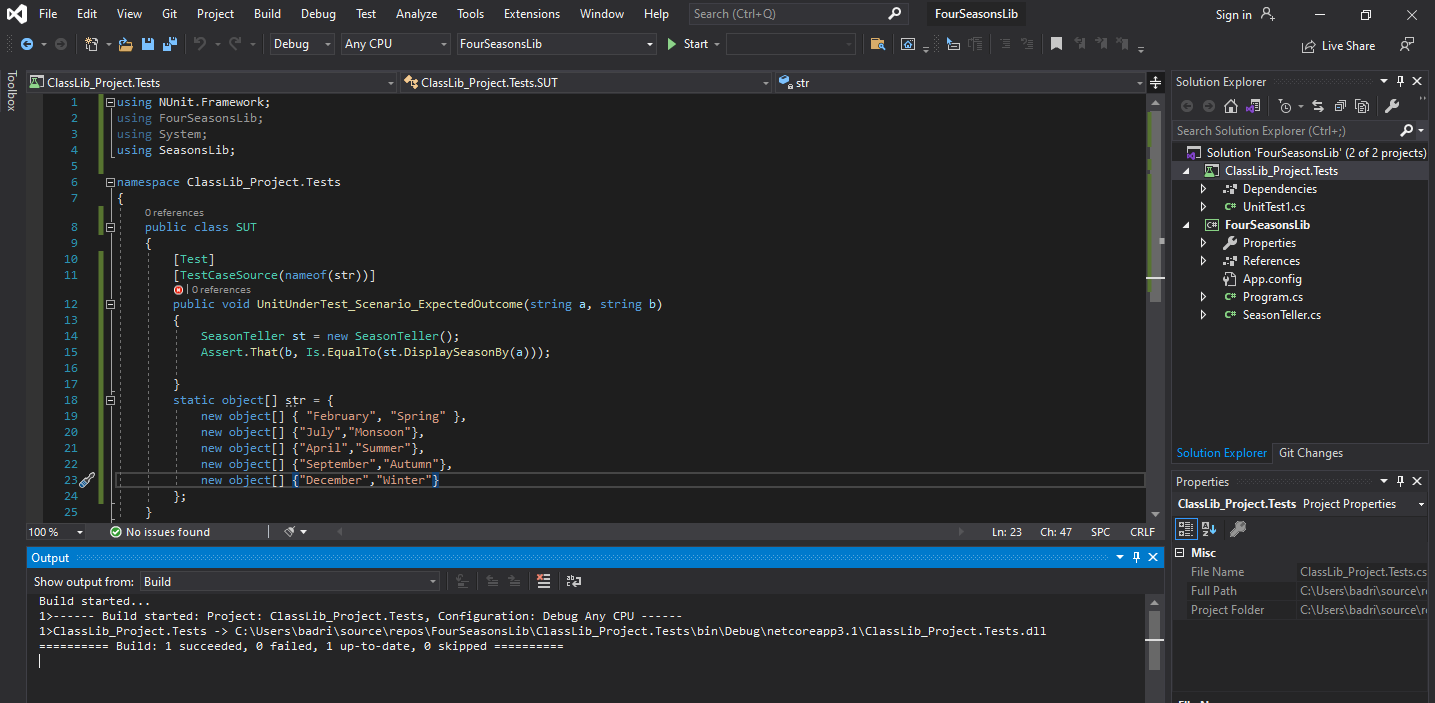
5) Write the suggested test methods.

6) Run your tests.

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

8) Rerun the test.

9) Observe the test result.

**SOURCE CODE:**

**SeasonTeller.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SeasonsLib

{

public class SeasonTeller

{

public string DisplaySeasonBy(string monthName)

{

string seasonName;

if (monthName.Equals("February", StringComparison.OrdinalIgnoreCase) || monthName.Equals("March", StringComparison.OrdinalIgnoreCase))

{

seasonName = "Spring";

}

else if (monthName.Equals("April", StringComparison.OrdinalIgnoreCase) || monthName.Equals("May", StringComparison.OrdinalIgnoreCase) || monthName.Equals("June", StringComparison.OrdinalIgnoreCase))

{

seasonName = "Summer";

}

else if (monthName.Equals("July", StringComparison.OrdinalIgnoreCase) || monthName.Equals("August", StringComparison.OrdinalIgnoreCase) || monthName.Equals("September", StringComparison.OrdinalIgnoreCase))

{

seasonName = "Monsoon";

}

else if (monthName.Equals("October", StringComparison.OrdinalIgnoreCase) || monthName.Equals("November", StringComparison.OrdinalIgnoreCase))

{

seasonName = "Autumn";

}

else if (monthName.Equals("December", StringComparison.OrdinalIgnoreCase) || monthName.Equals("January", StringComparison.OrdinalIgnoreCase))

{

seasonName = "Winter";

}

else

{

return "Invalid Season";

}

return seasonName;

}

}

}

**UnitTest1.cs**

using NUnit.Framework;

using FourSeasonsLib;

using System;

using SeasonsLib;

namespace ClassLib\_Project.Tests

{

public class SUT

{

[Test]

[TestCaseSource(nameof(str))]

public void UnitUnderTest\_Scenario\_ExpectedOutcome(string a, string b)

{

SeasonTeller st = new SeasonTeller();

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

}

static object[] str = {

new object[] { "February", "Spring" },

new object[] {"July","Monsoon"},

new object[] {"April","Summer"},

new object[] {"September","Autumn"},

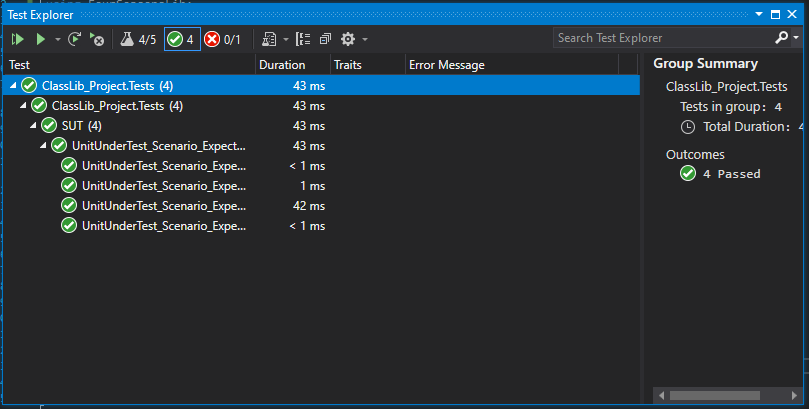
new object[] {"December","Winter"}

};

}

}

**OUTPUT :**



**Hands-On: Stage 3 – .NET Advanced Features** - **NUnit Day 2 - Question 4**

Steps to perform

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.

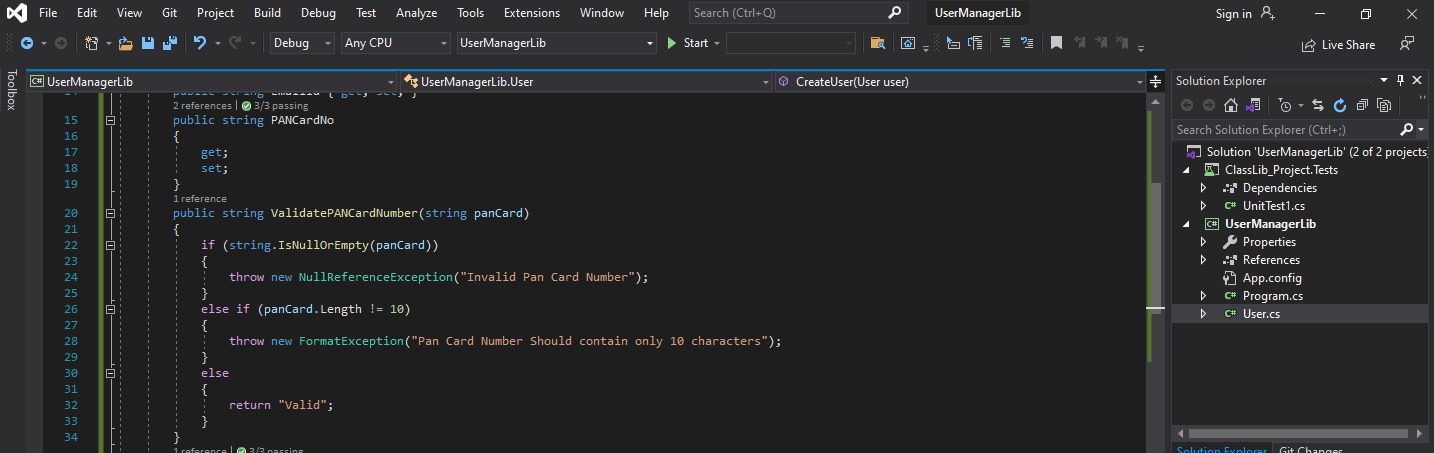
6) Run your tests.

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

8) Rerun the test.

9) Observe the test result.

**SOURCE CODE:**

****

**User.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace UserManagerLib

{

public class User

{

public Guid Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string EmailId { get; set; }

public string PANCardNo

{

get;

set;

}

public string ValidatePANCardNumber(string panCard)

{

if (string.IsNullOrEmpty(panCard))

{

throw new NullReferenceException("Invalid Pan Card Number");

}

else if (panCard.Length != 10)

{

throw new FormatException("Pan Card Number Should contain only 10 characters");

}

else

{

return "Valid";

}

}

public void CreateUser(User user)

{

if (ValidatePANCardNumber(user.PANCardNo).Equals("Valid"))

{

}

}

}

}

**UnitTest1.cs**

using NUnit.Framework;

using System;

using UserManagerLib;

namespace ClassLib\_Project.Tests

{

public class SUT

{

[Test]

[TestCase("LQAPS7022A")]

[TestCase("LQWPS7023C")]

[TestCase("AZXSC123AW")]

public void validpancard(string number)

{

User user = new User();

try

{

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

}

catch (NullReferenceException e)

{

Assert.Fail(e.Message);

}

catch (FormatException e)

{

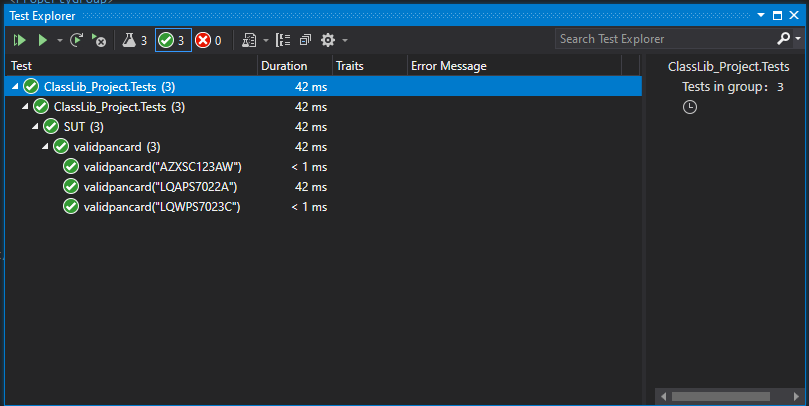
Assert.Fail(e.Message);

}

}

}

}

**OUTPUT :**

**Hands-On: Stage 3 – .NET Advanced Features** - **NUnit Day 2 - Question 5**

Steps to perform

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).

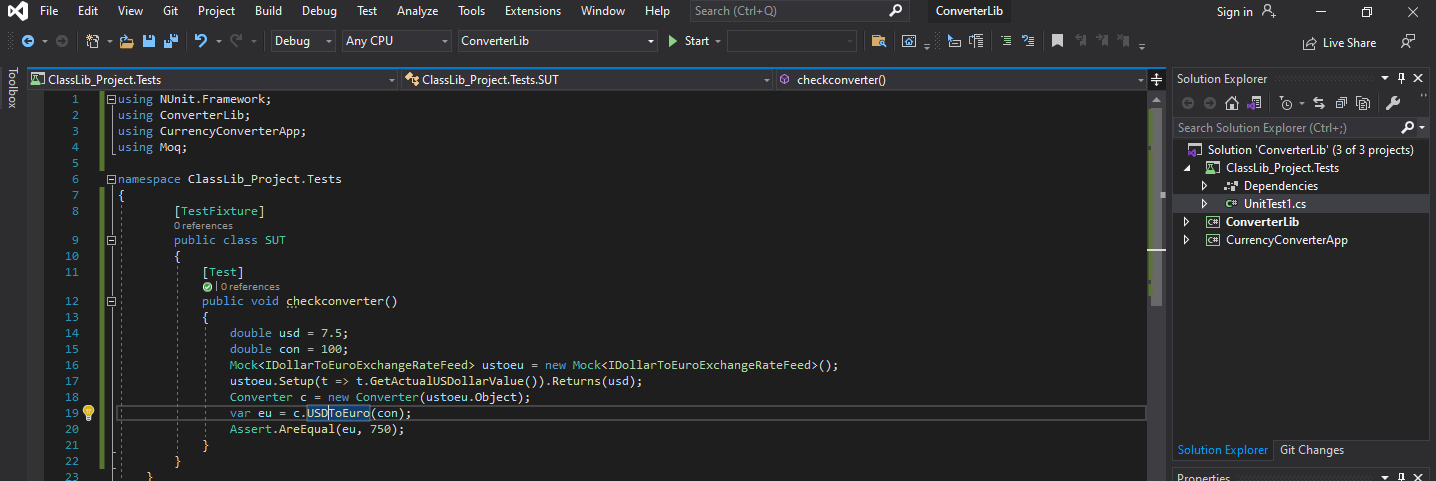
5) Write the suggested test methods using moq.

6) Run your tests.

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

8) Rerun the test.

9) Observe the test result.

**SOURCE CODE:**

**Converter.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CurrencyConverterApp;

namespace ConverterLib

{

public class Converter : IConverter

{

IDollarToEuroExchangeRateFeed \_exchangeRateFeed;

public Converter(IDollarToEuroExchangeRateFeed exchangeRateFeed)

{

\_exchangeRateFeed = exchangeRateFeed;

}

public double CelsiusToKelvin(double celsius)

{

return celsius + 273.15;

}

public double KilogramToPound(double kilogram)

{

return kilogram \* 2.205;

}

public double KilometerToMile(double kilometer)

{

return kilometer / 1.609;

}

public double LiterToGallon(double liter)

{

return liter / 3.785;

}

public double USDToEuro(double dollar)

{

return dollar \* \_exchangeRateFeed.GetActualUSDollarValue();

}

}

}

**IConverter.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConverterLib

{

public interface IConverter

{

double CelsiusToKelvin(double celsius);

double KilogramToPound(double kilogram);

double KilometerToMile(double kilometer);

double LiterToGallon(double liter);

double USDToEuro(double dollar);

}

}

**IDollarToEuroExchangeRateFeed.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CurrencyConverterApp

{

public interface IDollarToEuroExchangeRateFeed

{

double GetActualUSDollarValue();

}

}

**UnitTest1.cs**

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);

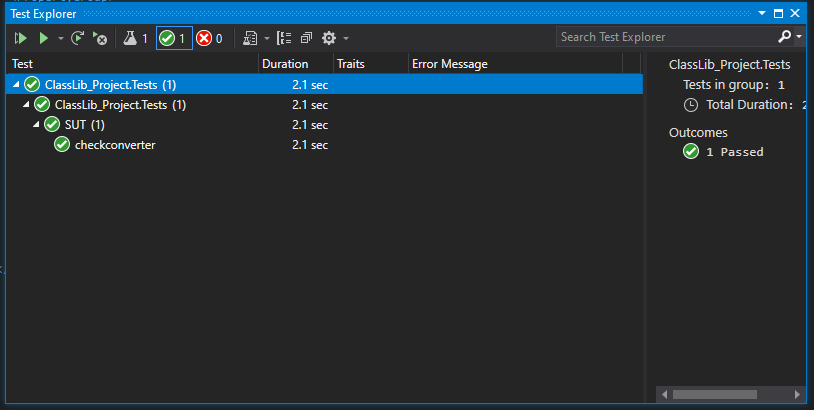
var eu = c.USDToEuro(con);

Assert.AreEqual(eu, 750);

}

}

}

**OUTPUT :**