**916217 -G pavan kumar**

**Stage3-NUnit Day1 Hands-on1**

· Create a Unit test project(.NET Framework) in the solution provided.

· Add the CalcLibrary project as reference

· Create a class “CalculatorTests” to write all the test cases for the methods in the solution

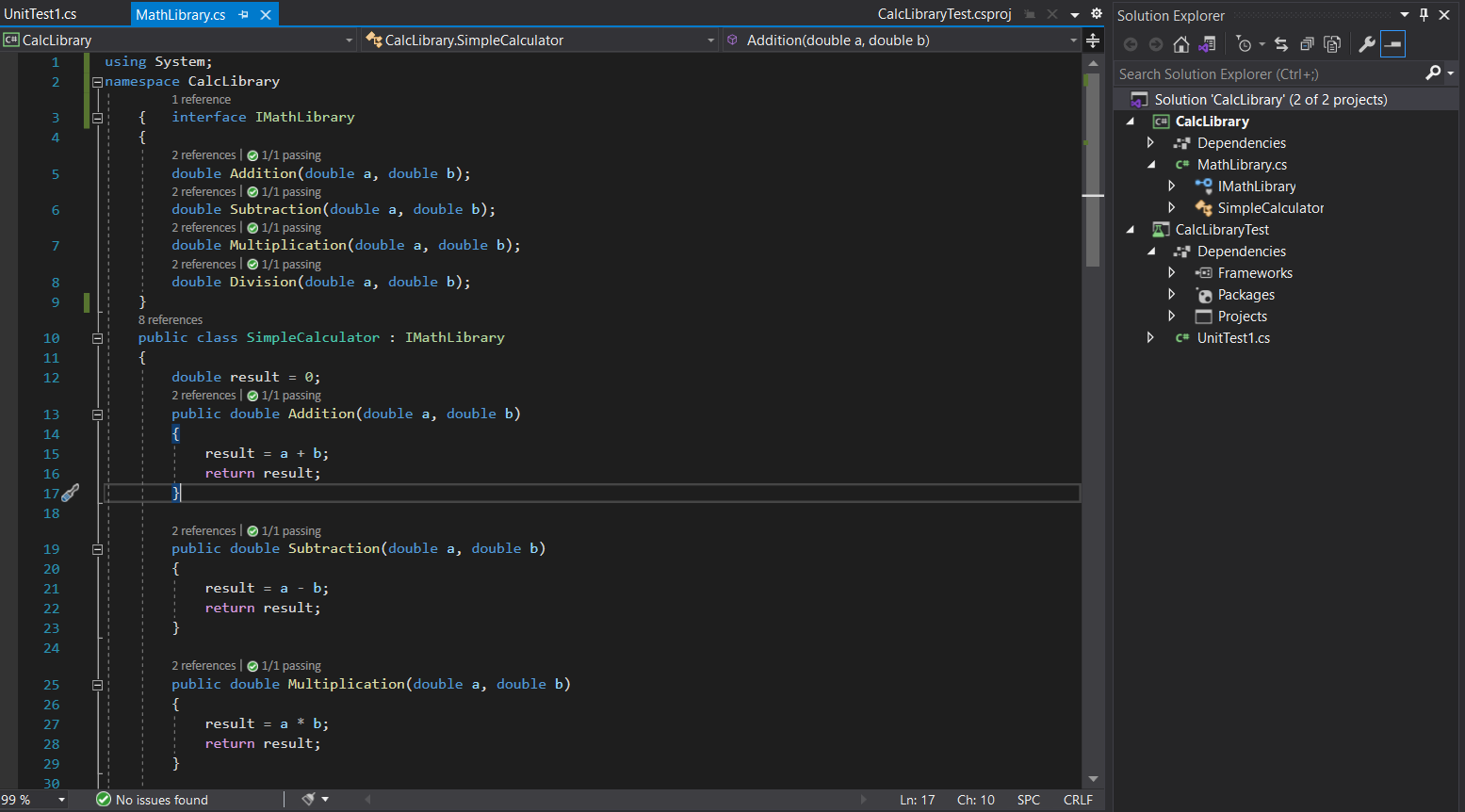
· Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively

· Create a Test method to check the addition functionality

· Use the ‘TestCase’ attribute to send the inputs and the expected result

· Use Assert.That to check the actual and expected result match

**Given Code**



* Add new Nunit3TestProject with name as CalcLibraryTest and install Nunit and

Nunit3TestAdapters from Nuget packages moreover click on the dependence and add reference from solution as CalcLibrary

Implemeantaion

UnitTest.cs

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcLibraryTest

{

[TestFixture]

public class CalculatorTests

{

SimpleCalculator SimpleCalculator;

double Result;

[SetUp]

public void SetUp()

{

SimpleCalculator = new SimpleCalculator();

Result = 0;

}

[TearDown]

public void TearDown()

{

SimpleCalculator = null;

}

[Test]

public void Addition\_result()

{

double expectedResult = 4;

Result = SimpleCalculator.Addition(2, 2);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Subtraction\_result()

{

double expectedResult = 5;

Result = SimpleCalculator.Subtraction(10, 5);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Multiplication\_result()

{

double expectedResult = 16;

Result = SimpleCalculator.Multiplication(8, 2);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Division\_result()

{

double expectedResult = 10;

double Result = SimpleCalculator.Division(20, 2);

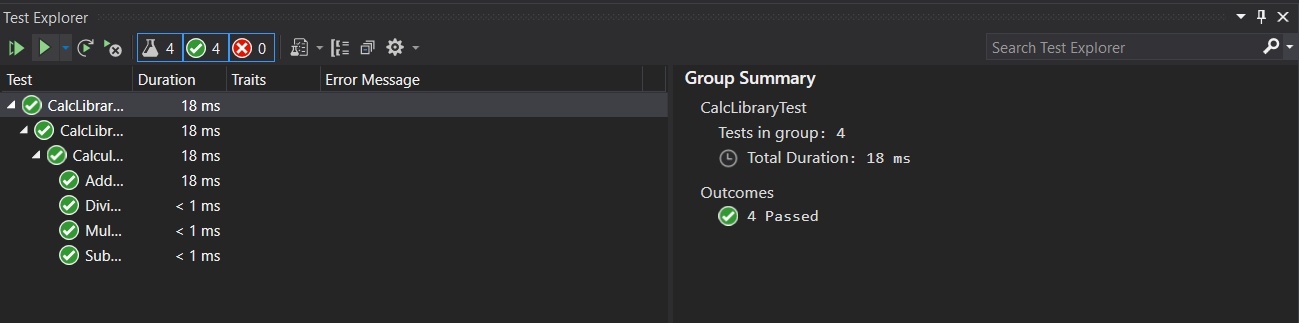
Assert.AreEqual(expectedResult, Result);

}

}

}

**Output**



**Stage3-NUnit Day1 Hands-on2**

1. Create test case to verify the subtraction feature of the calculator with various input types.

· Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

· Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

· Use Assert.Equal to check the actual and expected results

2. Create a test case to verify the multiplication concepts of calculator

· Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

· Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

· Use Assert.Equal to check the actual and expected results

3. Create a test case to verify the division logic of the calculator

· Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

· Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

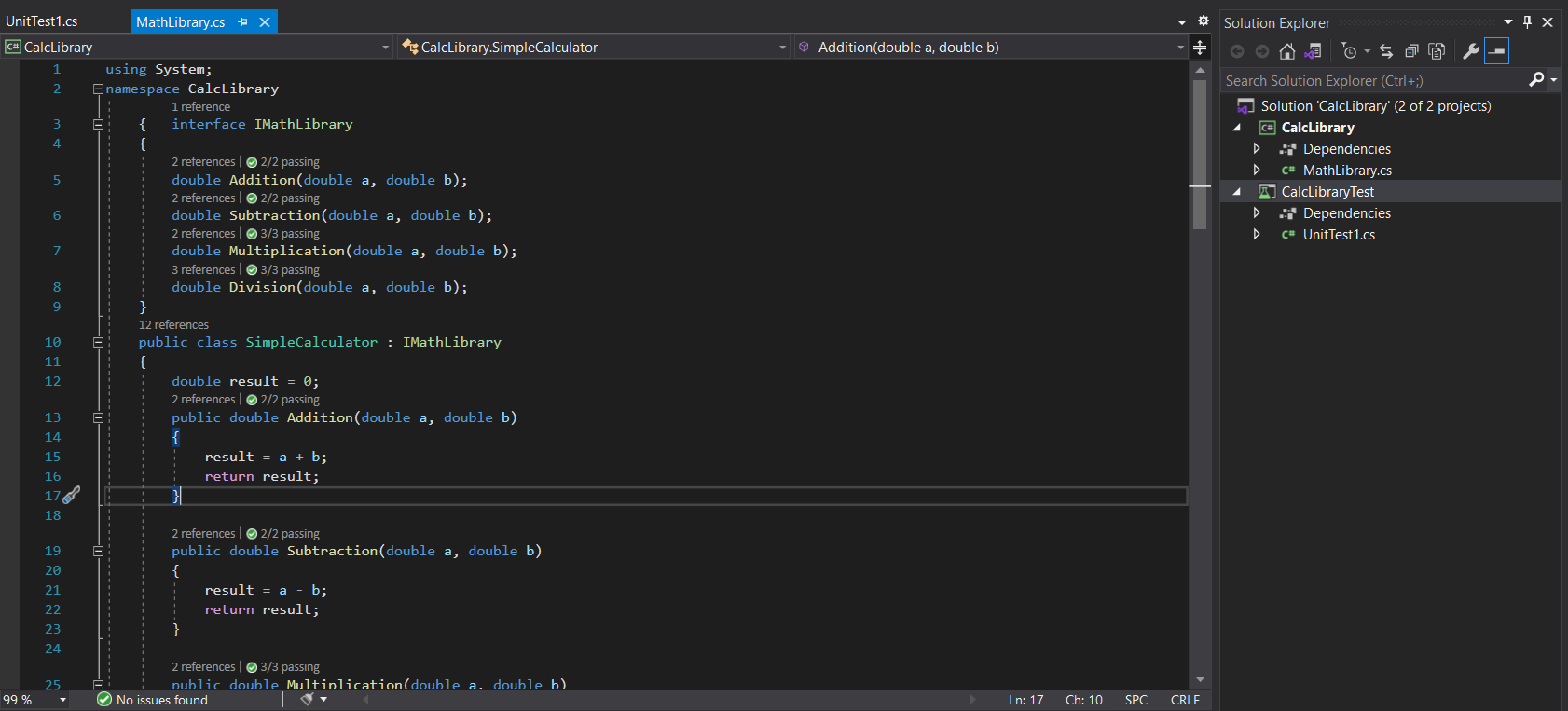
· Use Assert.Equal to check the actual and expected results

· In one of the inputs, provide the divisor value to be 0

· Use Try Catch block to catch the ArgumentException

· Use Assert.Fail to notify the user that the test case has failed. Give the message “Division by zero” in the Assert.Fail, which will be notified to the user. This message will be seen in the test explorer.

**Given code**



**Implementation**

**UnitTest.cs**

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcLibraryTest

{

[TestFixture]

public class CalculatorTests

{

SimpleCalculator SimpleCalculator;

double Result;

[SetUp]

public void SetUp()

{

SimpleCalculator = new SimpleCalculator();

Result = 0;

}

[TearDown]

public void TearDown()

{

SimpleCalculator = null;

}

//double \_result;

[Test]

[TestCase(5, 10, 15)]

[TestCase(5, -5, 0)]

public void Addition\_result(double a, double b, double expectedResult)

{

Result = SimpleCalculator.Addition(a, b);

Assert.AreEqual(expectedResult, Result);

}

[Test]

[TestCase(9, 5, 4)]

[TestCase(5, -9, 14)]

public void Subtraction\_result(double a, double b, double expectedResult)

{

Result = SimpleCalculator.Subtraction(a, b);

Assert.AreEqual(expectedResult, Result);

}

[Test]

[TestCase(9, 5, 45)]

[TestCase(2, 6, 12)]

[TestCase(2, 0, 0)]

public void Multiplication\_result(double a, double b, double expectedResult)

{

Result = SimpleCalculator.Multiplication(a, b);

Assert.AreEqual(expectedResult, Result);

}

[Test]

[TestCase(9, 3, 3)]

[TestCase(6, 6, 1)]

public void Division\_result(double a, double b, double expectedResult)

{

Result = SimpleCalculator.Division(a, b);

Assert.AreEqual(expectedResult, Result);

}

[Test]

[TestCase(10, 0)]

public void Division\_result1(double a, double b)

{

Assert.Throws<ArgumentException>(() => SimpleCalculator.Division(a, b));

}

[TearDown]

public void CleanUp()

{

SimpleCalculator.AllClear();

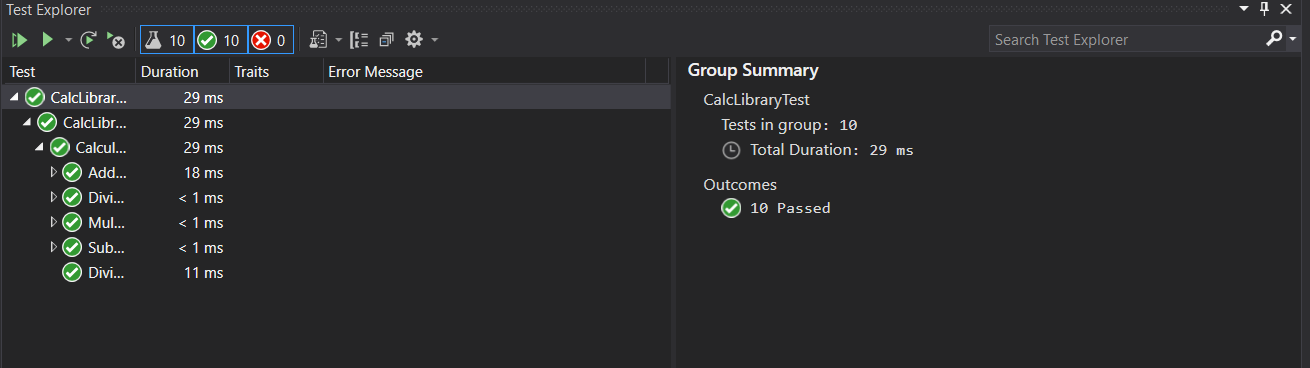
double result = SimpleCalculator.GetResult;

}

}

}

**Output**



**Stage3-NUnit -Day1 Hands-on3**

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.

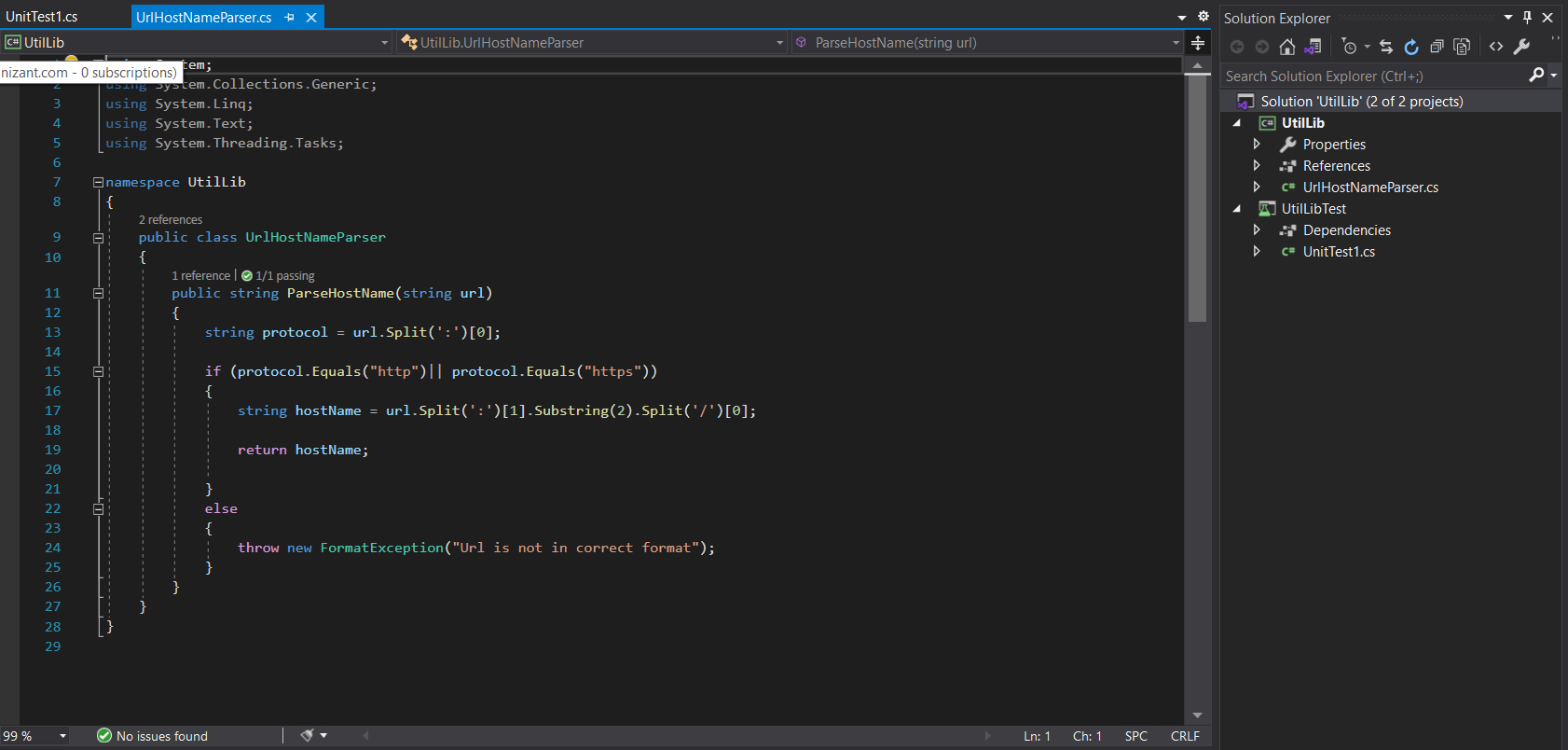
6) Run your tests.

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

8) Rerun the test.

9) Observe the test result.

**Given Code**



**Implementation**

using NUnit.Framework;

using System;

using UtilLib;

namespace UtilLibTest

{

[TestFixture]

public class SUT

{

UrlHostNameParser url;

string result;

[SetUp]

public void Setup()

{

url = new UrlHostNameParser();

result = "";

}

[TearDown]

public void TearDown()

{

url = null;

}

[Test]

public void UnitUnderTest\_Scenario\_ExpectedOutcome()

{

result = url.ParseHostName("https://www.javatpoint.com/");

Assert.That(result, Is.EqualTo("www.javatpoint.com"));

}

[Test]

public void UnitUnderTest\_Scenario\_ExpectedOutcome1()

{

result= url.ParseHostName("http://www.Youtube.com");

Assert.That(result, Is.EqualTo("www.Youtube.com"));

}

[Test]

public void UnitUnderTest\_Scenario\_ExpectedOutcome2()

{

var ex = Assert.Throws<FormatException>(() => url.ParseHostName("https101://cognizant.com"));

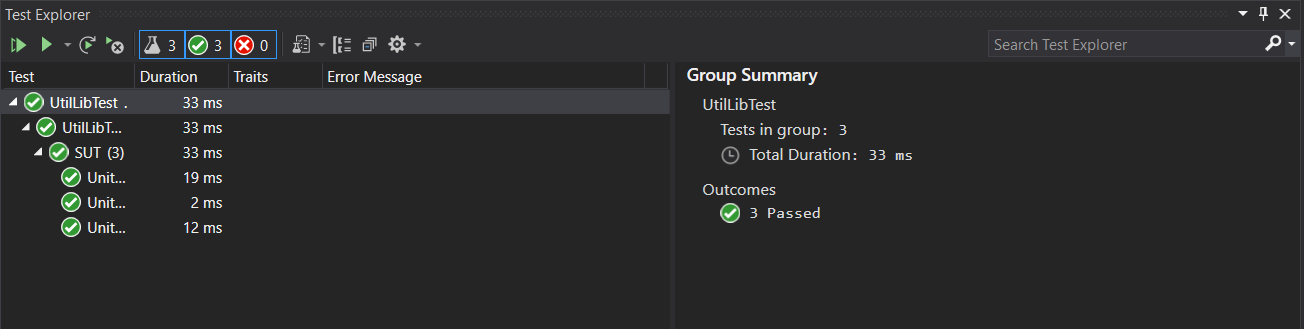
Assert.That(ex.Message, Is.EqualTo("Url is not in correct format"));

}

}

}

**Output**



**Stage3-NUnit -Day1 Hands-on4**

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.

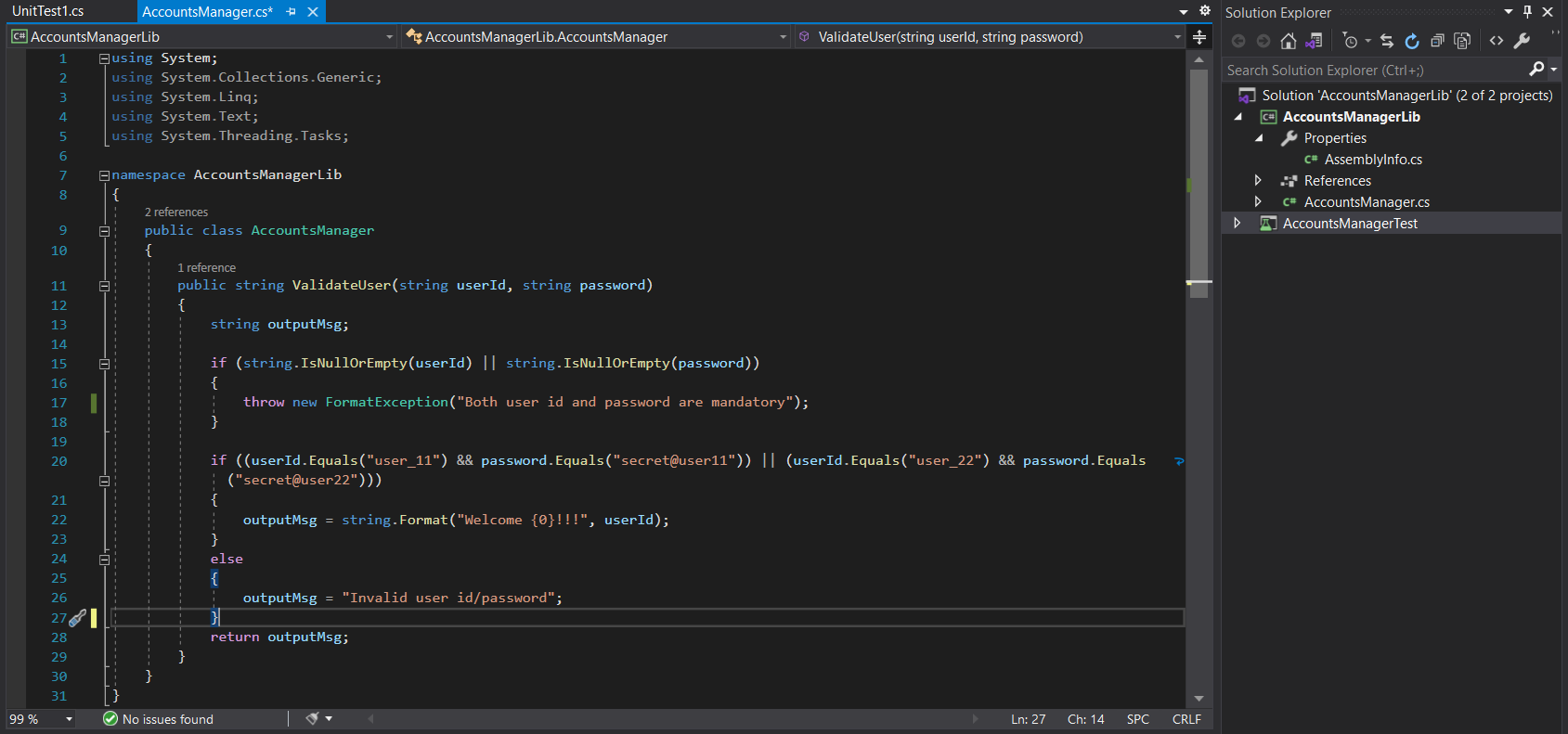
6) Run your tests.

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

8) Rerun the test.

9) Observe the test result.

**Given code**



**Implementation**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Linq;

using System.Text;

using NUnit.Framework;

using AccountsManagerLib;

namespace AccountsManagerTest

{

[TestFixture]

public class AccountsTest

{

[TestFixture]

public class SUT

{

[Test]

[TestCase("user\_11", "secret@user11", "Welcome user\_11!!!")]

[TestCase("user\_22", "secret@user22", "Welcome user\_22!!!")]

[TestCase("user\_11", "secret@user22", "Invalid user id/password")]

public void UnitUnderTest\_Scenario\_ExpectedOutcome1(string id, string password, string message)

{

AccountsManager account = new AccountsManager();

string result = account.ValidateUser(id, password);

NUnit.Framework.Assert.That(result, Is.EqualTo(message));

}

}

}

}

**Output**

