**NUnit Handson Solutions**

**Date:**09-06-2021

**Name:**Asritha Cherukuri

**EmpId:** 916179

**Handson1**

*Given:*

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

*Implementation:*

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using NUnit.Framework;

using CalcLibrary;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestFixture]

public class UnitTest1

{

SimpleCalculator s;

[SetUp]

public void SetUp()

{

s = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

s = null;

}

[Test]

public void TestAdd()

{

double result = s.Addition(10, 10);

double ans = 20;

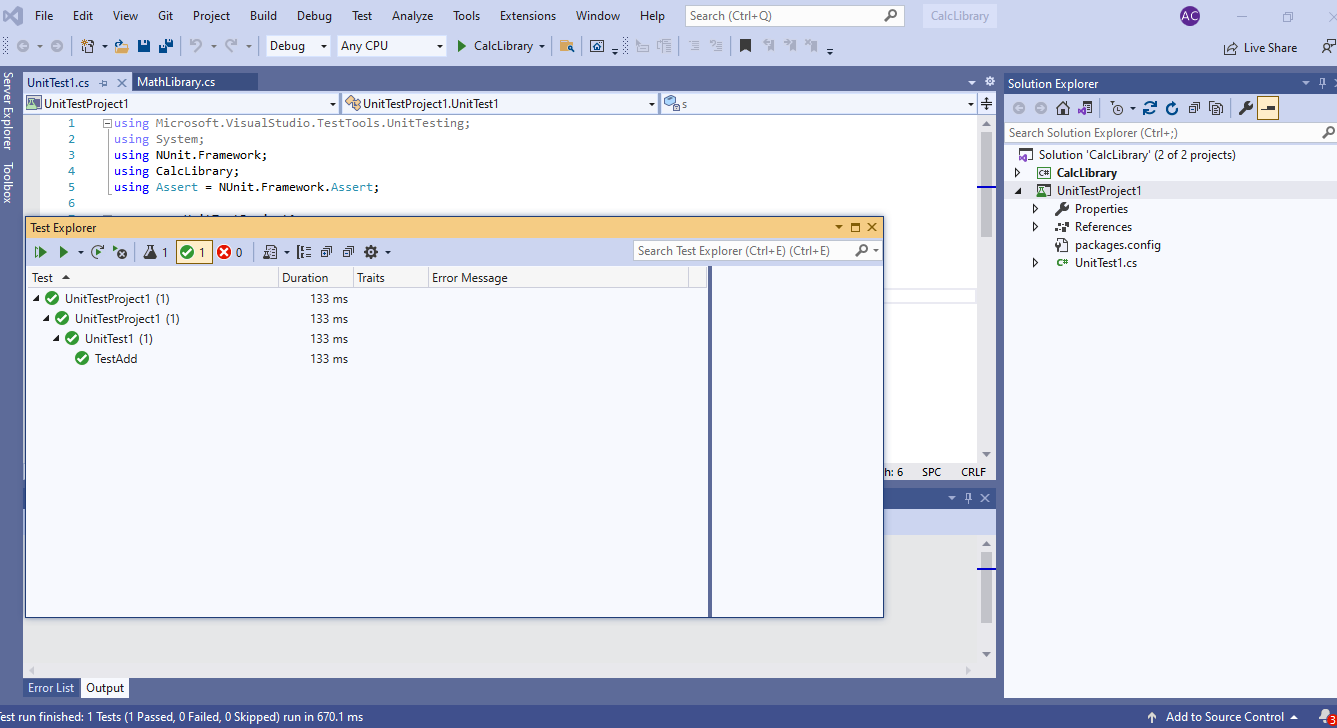
Assert.That(ans, Is.EqualTo(result));

}

}

}

*Results:*

**

**Handson2**

*Given:*

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.

*Implementation:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using CalcLibrary;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestFixture]

class UnitTest2

{

SimpleCalculator s;

[SetUp]

public void SetUp()

{

s = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

s = null;

}

[Test]

public void SubtractionTest1()

{

double result = s.Subtraction(26, 36);

double ans = 10;

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

}

[Test]

public void SubtractionTest2()

{

double result = s.Subtraction(36, 26);

double ans = -10;

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

}

[Test]

public void SubtractionTest3()

{

double result = s.Subtraction(26, 26);

double ans = 0;

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

}

[Test]

public void MultiplicationTest1()

{

double result = s.Multiplication(4, 4);

double ans = 16;

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

}

[Test]

public void MultiplicationTest2()

{

double result = s.Multiplication(0, 4);

double ans = 0;

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

}

[Test]

public void DivisionTest1()

{

double result = s.Division(10, 2);

double ans = 5;

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

}

[Test]

public void DivisionTest2()

{

try

{

var e = Assert.Throws<ArgumentException>(() => s.Division(100, 0));

}

catch {

Assert.Fail("Division by zero");

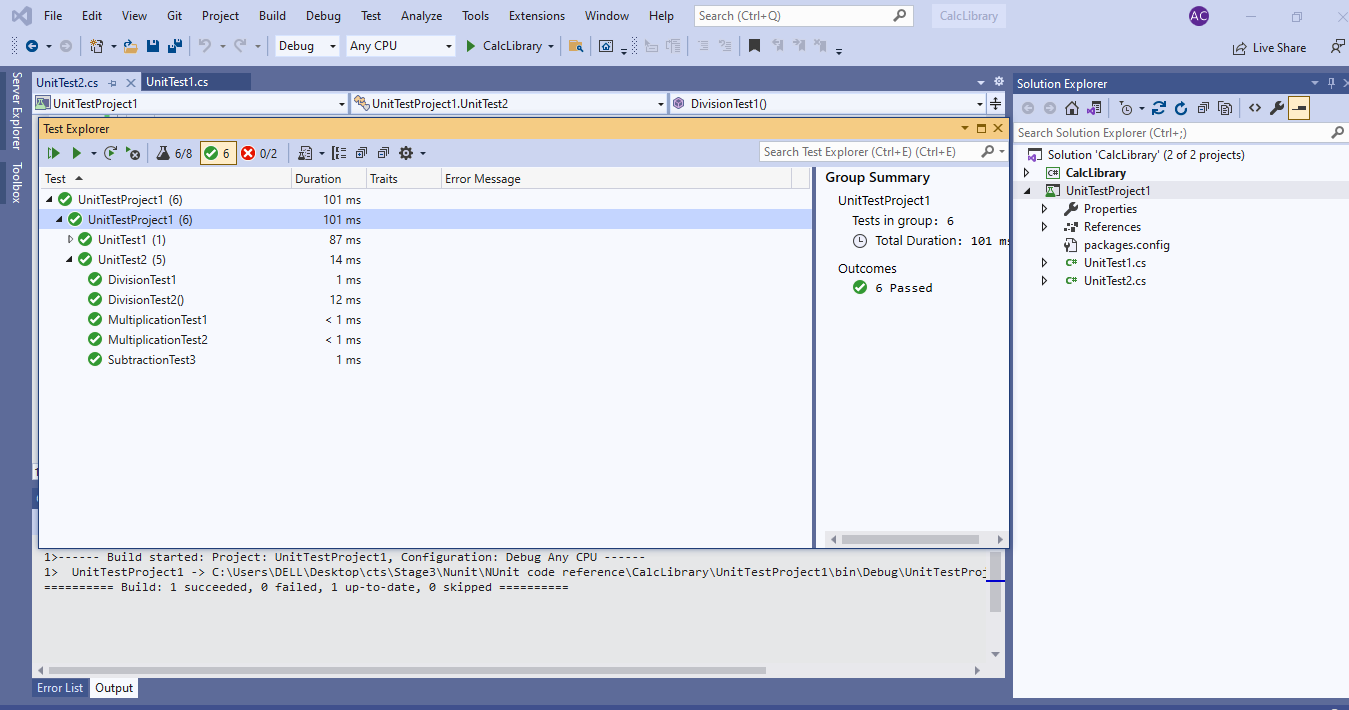
}

}

}

}

*Results:*

**

**Handson3**

*Given:*

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 System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using UtilLib;

using NUnit.Framework;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject1

{

[TestFixture]

public class Tests

{

UrlHostNameParser url;

[SetUp]

public void SetUp()

{

url = new UrlHostNameParser();

}

[TearDown]

public void Dispose()

{

url = null;

}

[Test]

public void Test1()

{

string act = url.ParseHostName("https://Facebook.com");

string exp = "Facebook.com";

Assert.That(act, Is.EqualTo(exp));

}

[Test]

public void Test2()

{

string act = url.ParseHostName("http://twitter.com");

string exp = "twitter.com";

Assert.That(act, Is.EqualTo(exp));

}

[Test]

public void Test3()

{

var ex = Assert.Throws<FormatException>(() => url.ParseHostName("https12://gmail.com"));

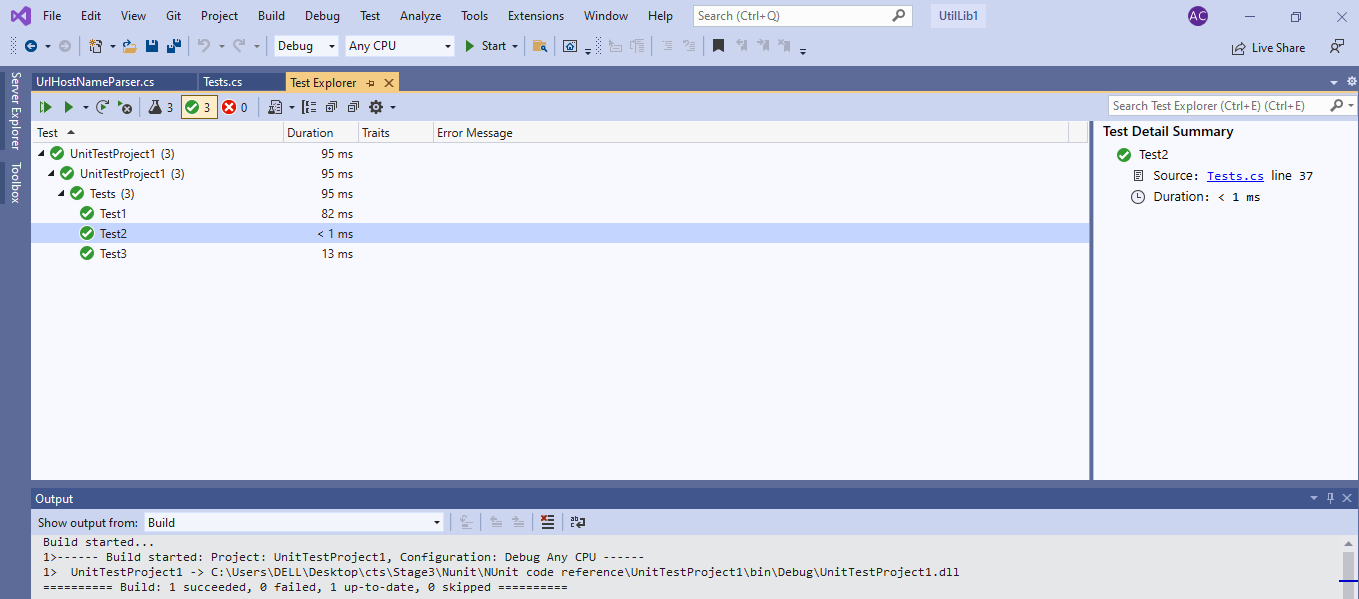
Assert.That(ex.Message, Is.EqualTo("Url is not in correct format"));

}

}

}

*Results:*

**

**Handson4**

*Given:*

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 System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using AccountsManagerLib;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProj1

{

[TestFixture]

public class Tests

{

AccountsManager am;

[SetUp]

public void SetUp()

{

am = new AccountsManager();

}

[TearDown]

public void Dispose()

{

am = null;

}

[Test]

public void LoginTest1()

{

string exp = "Invalid user id/password";

string act = am.ValidateUser("user\_22", "secret@user12s");

Assert.That(act, Is.EqualTo(exp));

}

[Test]

public void LoginTest2()

{

string act = am.ValidateUser("user\_11", "secret@user11");

string exp = string.Format("Welcome user\_11!!!");

Assert.That(act, Is.EqualTo(exp));

}

[Test]

public void LoginTest3()

{

string act = am.ValidateUser("user\_12", "secret@user12");

string exp = string.Format("Invalid user id/password");

Assert.That(act, Is.EqualTo(exp));

}

[Test]

public void LoginTest4()

{

string exp = "Invalid user id/password";

string act = am.ValidateUser("user\_23", "secret@user12s");

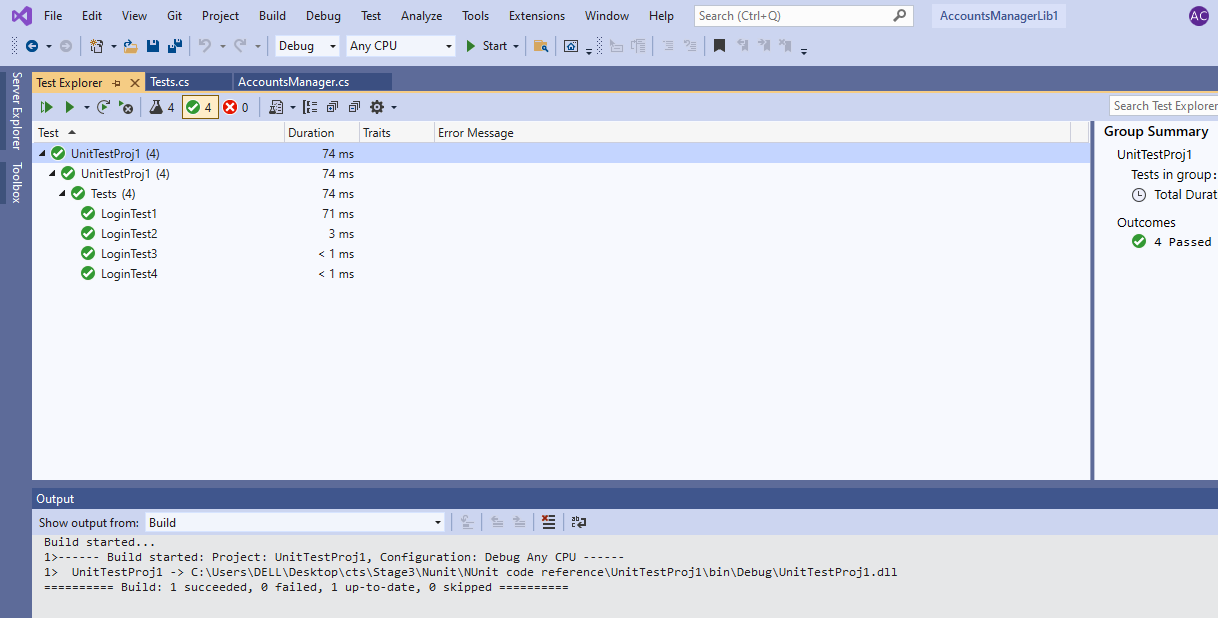
Assert.That(act, Is.EqualTo(exp));

}

}

}

*Results:*

**