# Date:01/09/2022

# LabSessionNo.:06

# Task 6:Design a test case for testing for Visual C# using NUNIT for arithmetic operations

# Task 6: Testing visual c# using NUnit.

**Aim**: To perform testing visual c# using NUnit.

# Procedure:

**Steps**:

1. Creating Class Library.
   * Click on File and select New project
   * Click on Visual C# and rename it as Program1
   * Click on Finish
2. Add the following code

using System; usingSystem.Collections.Generic; usingSystem.Linq; usingSystem.Text;

namespace Leela64

{

publicclassAddition

{

publicint add(int x, int y)

{

return (x + y);

}

}

publicclassSubtraction

{

publicint sub(int x, int y)

{

return (x - y);

}

}

publicclassDivision

{

publicint div(int x, int y)

{

return (x / y);

}

}

publicclassMultiplication

{

publicintmul(int x, int y)

{return (x \* y);

} } }

1. Add new class Library
   * Right click on Project in Solution explorer.

Click on Add and New Project(Class Library)

* + Type the following code

using System; usingSystem.Collections.Generic; usingSystem.Linq; usingSystem.Text; usingNUnit.Framework;

using Leela64; namespace Test

{

[TestFixture] publicclassClass1

{

[Test] publicvoidprs()

{

Multiplication m1 = newMultiplication(); Addition a1 = newAddition(); Subtraction s1 = newSubtraction(); Division d1 = newDivision(); Assert.AreEqual(20, m1.mul(5, 4));

Assert.AreEqual(1, s1.sub(8, 7));

Assert.AreEqual(7, a1.add(3, 4));

Assert.AreEqual(1, d1.div(5, 5));

}

[Test] publicvoid prs1()

{

Multiplication m1 = newMultiplication(); Addition a1 = newAddition(); Subtraction s1 = newSubtraction(); Division d1 = newDivision(); Assert.AreEqual(10, m1.mul(2, 4));

Assert.AreEqual(1, s1.sub(8, 7));

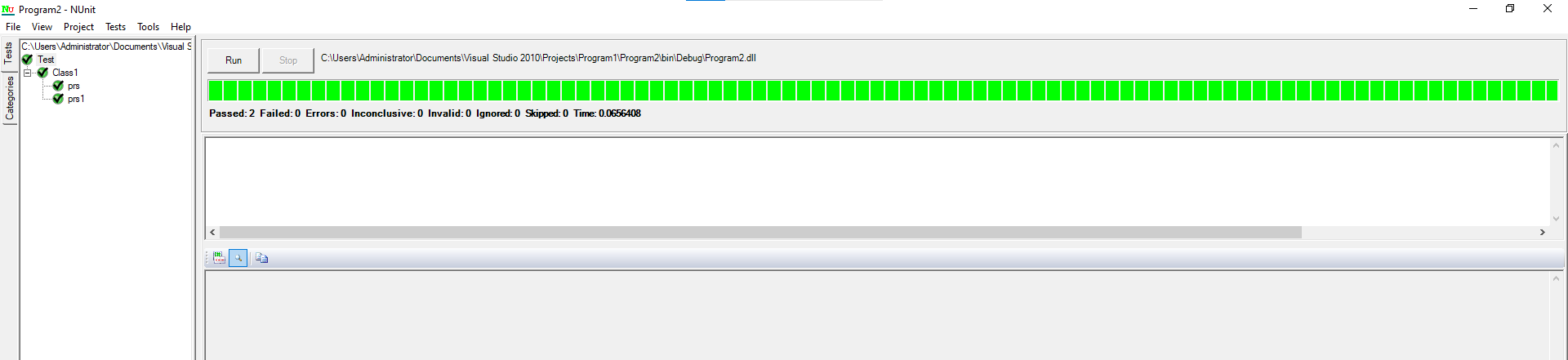
Assert.AreEqual(7, a1.add(3, 4));

Assert.AreEqual(1, d1.div(5, 5));

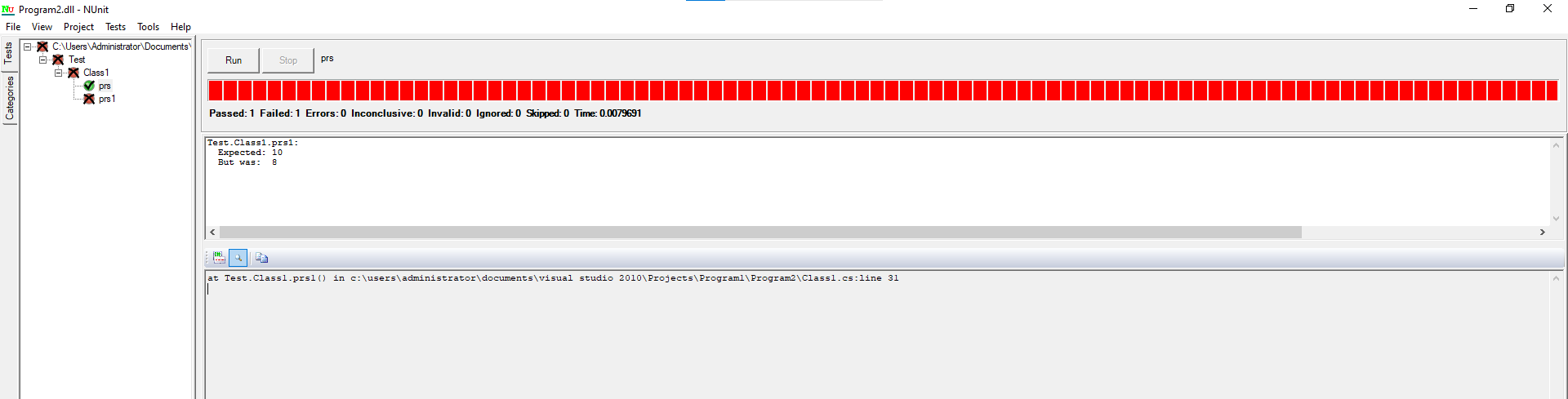
}}}

1. Add references
   * Right Click on Test and click on Add references
   * Add the Existing Project and also nunit.framework.dll
   * Click on OK
2. Build Solution
   * Click on Build
   * Now click on Build Solution
3. Testing using NUnit
   * Open NUnit and click on File
   * Click on Open project and select the Visual Project
   * Select the test code project.
   * Click on bin and then on debug and then on Test.nunit.dll
   * Click on Run

# Output: Pass Case



**Fail Case**



# Test suite design:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Name**: NUnit Testing | | | | | | |
| **Test case id**: ID\_6 **Test Designed by:SK.Suhana**  **Test Priority**: low **Test Designed Date**:01/09/22  **Module Name**: Nunit Testing **Test Executedby**:**Sk.Suhana**  **Test Title**: Blackbox testing  **Test Executed Date**: 01/09/22  **Description:** Test case for problem using NUnit Testing | | | | | | |
| **Pre-Condition**: User should give two input numbers and one expected output | | | | | | |
| **Stag e** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **status (Pass**  **/Fail)** | **Remarks** |
| 1 | Addition | 6,4 | 10 | 10 | Pass | Nil |
| 2 | Subtractio n | 7,3 | 4 | 4 | Pass | Nil |
| 3 | Multiplicat ion | 2,4 | 8 | 8 | Pass | Nil |
| 4 | Division | 8,4 | 2 | 3 | Fail | Nil |
| **Post condition**: Expected result should match with value returned by function | | | | | | |

**Result:** Performing testing visual c# using NUnit has been implemented successfully.

**Evaluator’s Observation**

**Marks Secured: out of**

**Full Name of the Evaluator: Student’s Signature**

**Signature of the Evaluator: Date of Evaluation:**