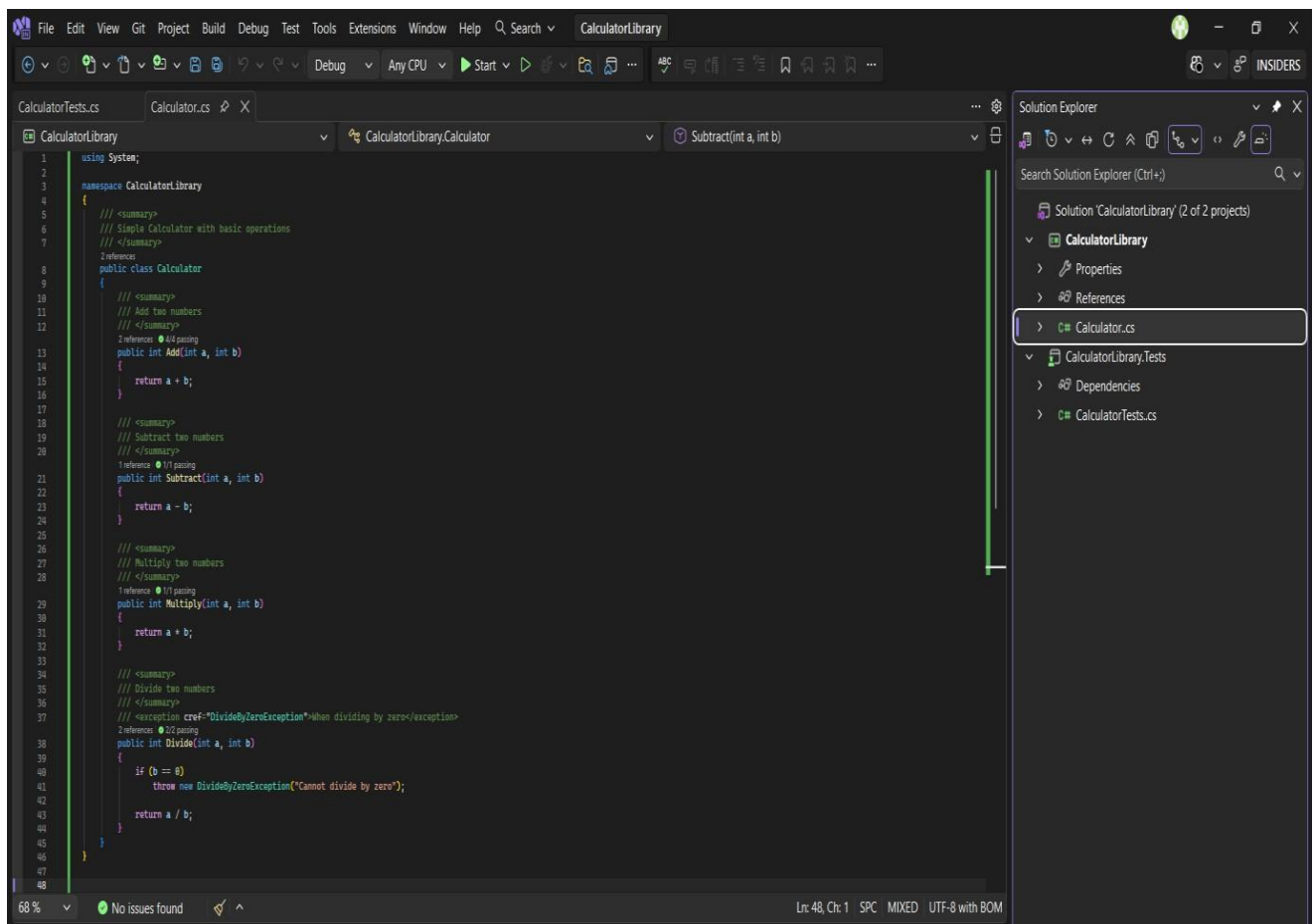


STEP 1: Create Calculator Class Library

1. Click Create a new project
2. Select Class Library (.NET)
3. Click Next
4. Project name: CalculatorLibrary
5. Click Create

STEP 2: Add Calculator Class

1. In Solution Explorer
2. Open Class .cs
3. Rename it to Calculator .cs



```
using System;

namespace CalculatorLibrary
{
    /// <summary>
    /// Simple Calculator with basic operations
    /// </summary>
    public class Calculator
    {
        /// <summary>
        /// Add two numbers
        /// </summary>
        public int Add(int a, int b)
        {
            return a + b;
        }

        /// <summary>
        /// Subtract two numbers
        /// </summary>
        public int Subtract(int a, int b)
        {
            return a - b;
        }

        /// <summary>
        /// Multiply two numbers
```

```
/// </summary>
public int Multiply(int a, int b)
{
    return a * b;
}

/// <summary>
/// Divide two numbers
/// </summary>
/// <exception cref="DivideByZeroException">When dividing by zero</exception>
public int Divide(int a, int b)
{
    if (b == 0)
        throw new DivideByZeroException("Cannot divide by zero");

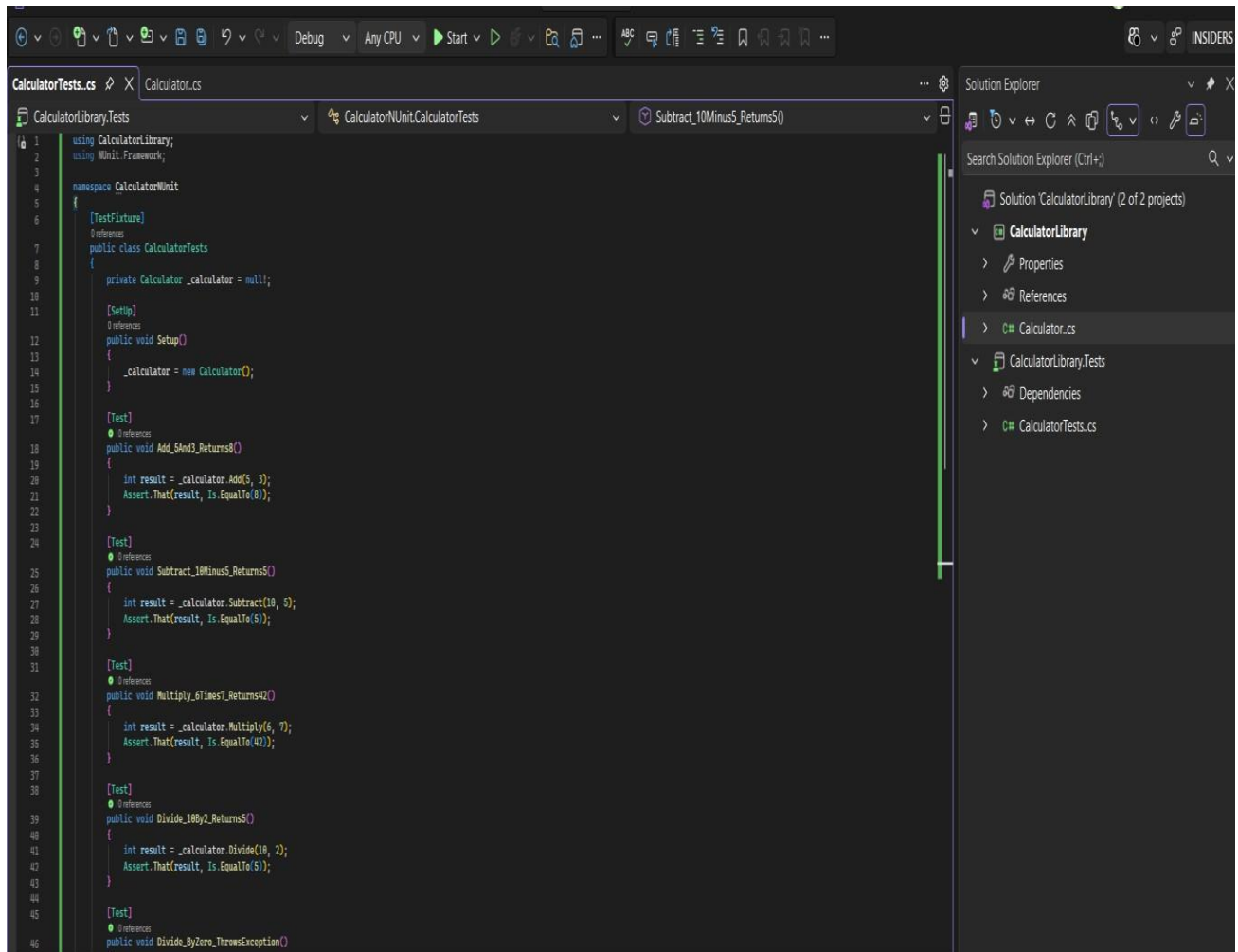
    return a / b;
}
}
```

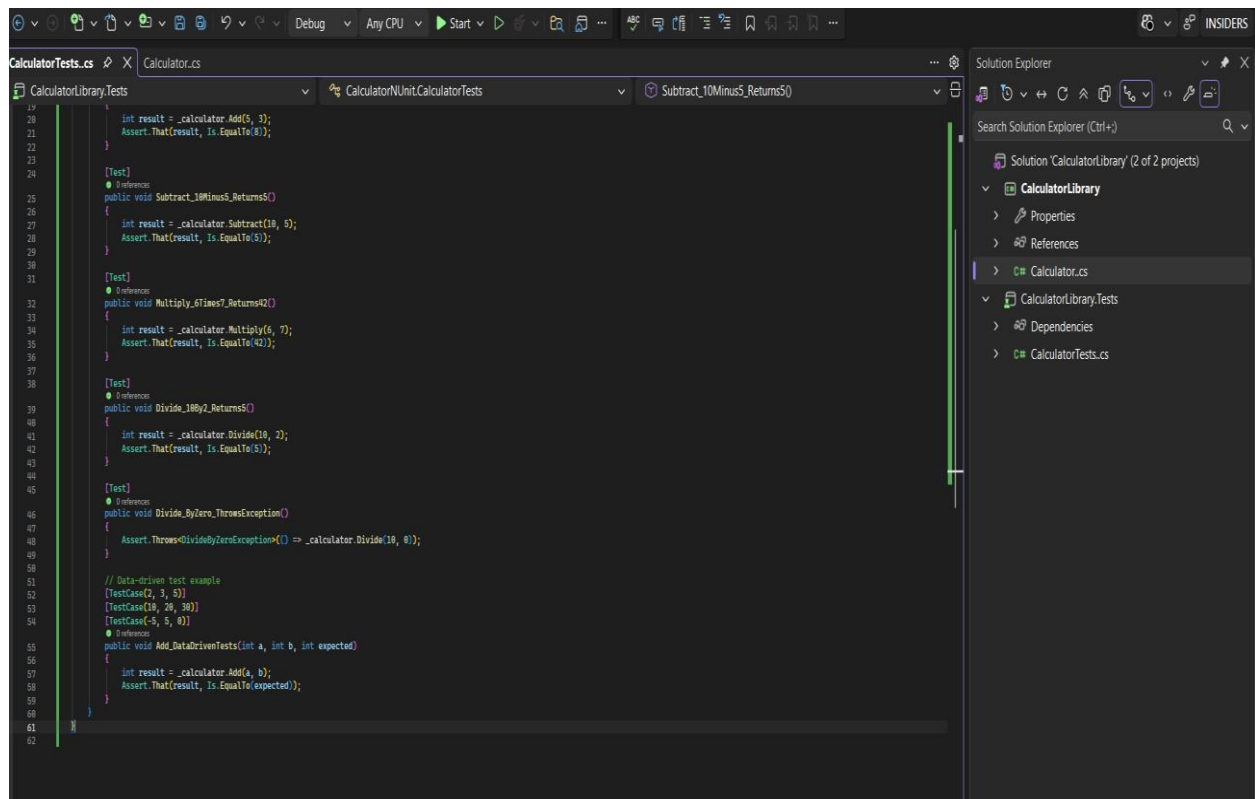
STEP 3: Create NUnit Test Project

1. Right-click Solution
2. Click Add → New Project
3. Select NUnit Test Project (.NET)
4. Click Next
5. Project name: CalculatorLibrary.Tests

STEP 4 : Add Reference to CalculatorLibrary

1. Right-click CalculatorLibrary.Tests
2. Click Add → Project Reference





CalculatorTests.cs Code:

using CalculatorLibrary;

using NUnit.Framework;

namespace CalculatorNUnit

{

[TestFixture]

public class CalculatorTests

{

private Calculator _calculator = null!;

[SetUp]

public void Setup()

{

```
    _calculator = new Calculator();  
}
```

```
[Test]  
  
public void Add_5And3_Returns8()  
{  
    int result = _calculator.Add(5, 3);  
    Assert.That(result, Is.EqualTo(8));  
}
```

```
[Test]  
  
public void Subtract_10Minus5_Returns5()  
{  
    int result = _calculator.Subtract(10, 5);  
    Assert.That(result, Is.EqualTo(5));  
}
```

```
[Test]  
  
public void Multiply_6Times7_Returns42()  
{  
    int result = _calculator.Multiply(6, 7);  
    Assert.That(result, Is.EqualTo(42));  
}
```

```
[Test]  
  
public void Divide_10By2_Returns5()
```

```

    {
        int result = _calculator.Divide(10, 2);
        Assert.That(result, Is.EqualTo(5));
    }

[Test]
public void Divide_ByZero_ThrowsException()
{
    Assert.Throws<DivideByZeroException>(() => _calculator.Divide(10, 0));
}

// Data-driven test example
[TestCase(2, 3, 5)]
[TestCase(10, 20, 30)]
[TestCase(-5, 5, 0)]
public void Add_DataDrivenTests(int a, int b, int expected)
{
    int result = _calculator.Add(a, b);
    Assert.That(result, Is.EqualTo(expected));
}
}
}

```

Step 5 : Run Unit Tests

1. Menu → **Test** → **Test Explorer**
2. Click **Run All Tests**

OUTPUT:

The screenshot displays the Visual Studio interface during a test run. The top toolbar shows the Test Explorer icon, and the status bar indicates 'Test run finished: 8 Tests (8 Passed, 0 Failed, 0 Skipped) run in 371 ms'. The Test Explorer pane on the left lists the test results in a table:

Test	Duration	Traits	Error Message
CalculatorLibrary.Tests (8)	12 ms		
CalculatorUnit (8)	12 ms		
CalculatorTests (8)	12 ms		
Add_5And3_Returns8	6 ms		
Add_DataDrivenTests (3)	< 1 ms		
Divide_10By2_Returns5	< 1 ms		
Divide_ByZero_ThrowsException	6 ms		
Multiply_6Times7_Returns42	< 1 ms		
Subtract_10Minus5_Returns5	< 1 ms		

The right-hand pane shows the 'Run' button and a 'Group Summary' for 'CalculatorLibrary.Tests', indicating 'Tests in group: 8' and a 'Total Duration: 12 ms'. Below this, the 'Outcomes' section shows '8 Passed'. The bottom status bar of the application shows the system clock at 22:34 on 21-01-2026.

