

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

The screenshot shows the Visual Studio IDE interface. The code editor on the left displays the `Calculator.cs` file, which contains the following C# code:

```
1  using System;
2
3  namespace CalculatorLibrary
4  {
5      /// <summary>
6      /// Simple Calculator with basic operations
7      /// </summary>
8      public class Calculator
9      {
10          /// <summary>
11          /// Add two numbers
12          /// </summary>
13          public int Add(int a, int b)
14          {
15              return a + b;
16          }
17
18          /// <summary>
19          /// Subtract two numbers
20          /// </summary>
21          public int Subtract(int a, int b)
22          {
23              return a - b;
24          }
25
26          /// <summary>
27          /// Multiply two numbers
28          /// </summary>
29          public int Multiply(int a, int b)
30          {
31              return a * b;
32          }
33
34          /// <summary>
35          /// Divide two numbers
36          /// </summary>
37          /// <exception cref="DivideByZeroException">When dividing by zero</exception>
38          public int Divide(int a, int b)
39          {
40              if (b == 0)
41                  throw new DivideByZeroException("Cannot divide by zero");
42
43              return a / b;
44          }
45      }
46  }
```

The Solution Explorer window on the right shows the project structure:

- Solution 'CalculatorLibrary' (2 of 2 projects)
 - CalculatorLibrary
 - Properties
 - References
 - Calculator.cs
 - CalculatorLibrary.Tests
 - Dependencies
 - CalculatorTests.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

The screenshot shows a Visual Studio interface with the following details:

- Solution Explorer:** Shows two projects: "CalculatorLibrary" (2 of 2 projects) and "CalculatorLibrary.Tests". Under "CalculatorLibrary", there is a file named "Calculator.cs". Under "CalculatorLibrary.Tests", there is a file named "CalculatorTests.cs".
- Code Editor:** The active tab is "CalculatorTests.cs". The code is as follows:

```
1  using CalculatorLibrary;
2  using NUnit.Framework;
3
4  namespace CalculatorNUnit
5  {
6      [TestFixture]
7      class CalculatorTests
8      {
9          private Calculator _calculator = null;
10
11         [Setup]
12         public void Setup()
13         {
14             _calculator = new Calculator();
15         }
16
17         [Test]
18         public void Add_5And3_Returns8()
19         {
20             int result = _calculator.Add(5, 3);
21             Assert.That(result, Is.EqualTo(8));
22         }
23
24         [Test]
25         public void Subtract_10Minus5_Returns5()
26         {
27             int result = _calculator.Subtract(10, 5);
28             Assert.That(result, Is.EqualTo(5));
29         }
30
31         [Test]
32         public void Multiply_6Times7_Returns42()
33         {
34             int result = _calculator.Multiply(6, 7);
35             Assert.That(result, Is.EqualTo(42));
36         }
37
38         [Test]
39         public void Divide_10By2_Returns5()
40         {
41             int result = _calculator.Divide(10, 2);
42             Assert.That(result, Is.EqualTo(5));
43         }
44
45         [Test]
46         public void Divide_ByZero_ThrowsException()
47     }
```

The screenshot shows the Visual Studio IDE interface. The left pane displays the code for `CalculatorTests.cs`, which contains unit tests for a `Calculator` class. The right pane shows the `Solution Explorer` with two projects: `CalculatorLibrary` and `CalculatorLibrary.Tests`. The `CalculatorLibrary` project contains files for the library itself and its dependencies. The `CalculatorLibrary.Tests` project contains the test classes and their dependencies.

```
17
18    |     int result = _calculator.Add(5, 3);
19    |     Assert.That(result, Is.EqualTo(8));
20    |
21    [Test]
22    |     [References]
23    |     public void Subtract_10Minus5_Returns5()
24    {
25        |         int result = _calculator.Subtract(10, 5);
26        |         Assert.That(result, Is.EqualTo(5));
27    }
28
29    [Test]
30    |     [References]
31    |     public void Multiply_5Times7_Returns42()
32    {
33        |         int result = _calculator.Multiply(6, 7);
34        |         Assert.That(result, Is.EqualTo(42));
35    }
36
37    [Test]
38    |     [References]
39    |     public void Divide_10By2_Returns5()
40    {
41        |         int result = _calculator.Divide(10, 2);
42        |         Assert.That(result, Is.EqualTo(5));
43    }
44
45    [Test]
46    |     [References]
47    |     public void Divide_ByZero_ThrowsException()
48    {
49        |         Assert.Throws<DivideByZeroException>(() => _calculator.Divide(10, 0));
50    }
51
52    // Data-driven test example
53    [TestCase(2, 3, 5)]
54    [TestCase(10, 20, 20)]
55    [TestCase(-5, 5, 0)]
56    |     [References]
57    |     public void Add_DataDrivenTests(int a, int b, int expected)
58    {
59        |         int result = _calculator.Add(a, b);
60        |         Assert.That(result, Is.EqualTo(expected));
61    }
62}
```

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));
}

[TestMethod]
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**

