Unit Testing CalcLibrary with NUnit in Visual Studio Code

# Overview

This document describes the step-by-step process to set up, reference, and unit test a C# calculator library (CalcLibrary) using NUnit in Visual Studio Code. The guide covers project structure, reference management, test writing, troubleshooting, and best practices.

# 1. Project Structure

Exercise-4/  
└── CalcLibrary/  
 ├── CalcLibrary/  
 │ └── CalcLibrary.csproj  
 │ └── SimpleCalculator.cs  
 └── CalcLibrary.Tests/  
 └── CalcLibrary.Tests.csproj  
 └── CalculatorTests.cs

CalcLibrary: Contains the main calculator logic (SimpleCalculator class).

CalcLibrary.Tests: Contains NUnit test cases for the calculator.

# 2. Setting Up the Environment

Install .NET SDK: Ensure the .NET SDK is installed (dotnet --version).

Install Visual Studio Code: Download and install VS Code.

Install Extensions: Add the C# Dev Kit and .NET Test Explorer extensions for enhanced development and testing experience.

# 3. Creating Projects

Create Library Project:

```bash  
dotnet new classlib -n CalcLibrary  
```

Create Test Project:

```bash  
dotnet new nunit -n CalcLibrary.Tests  
```

# 4. Adding a Project Reference

From inside the CalcLibrary.Tests directory, link the test project to the main library:

```bash  
dotnet add reference ../CalcLibrary/CalcLibrary.csproj  
```

This allows the test project to access classes and methods from CalcLibrary.

# 5. Writing the Calculator Class

SimpleCalculator.cs (in CalcLibrary):

```csharp  
using System;

namespace CalcLibrary  
{  
 interface IMathLibrary  
 {  
 double Addition(double a, double b);  
 double Subtraction(double a, double b);  
 double Multiplication(double a, double b);  
 double Division(double a, double b);  
 }

public class SimpleCalculator : IMathLibrary  
 {  
 double result = 0;  
 public double Addition(double a, double b) => result = a + b;  
 public double Subtraction(double a, double b) => result = a - b;  
 public double Multiplication(double a, double b) => result = a \* b;  
 public double Division(double a, double b)  
 {  
 if (b == 0)  
 throw new ArgumentException("Second Parameter Can't be Zero");  
 return result = a / b;  
 }  
 public void AllClear() => result = 0;  
 public double GetResult => result;  
 }  
}

# 6. Writing NUnit Test Cases

CalculatorTests.cs (in CalcLibrary.Tests):

```csharp  
using NUnit.Framework;  
using CalcLibrary;

namespace CalcLibrary.Tests  
{  
 [TestFixture]  
 public class CalculatorTests  
 {  
 private SimpleCalculator \_calculator;

[SetUp]  
 public void SetUp()  
 {  
 \_calculator = new SimpleCalculator();  
 }

[TearDown]  
 public void TearDown()  
 {  
 \_calculator = null;  
 }

[Test]  
 [TestCase(2, 3, 5)]  
 [TestCase(-1, 1, 0)]  
 [TestCase(0, 0, 0)]  
 public void Addition\_WhenCalled\_ReturnsExpectedResult(double a, double b, double expected)  
 {  
 var result = \_calculator.Addition(a, b);  
 Assert.That(result, Is.EqualTo(expected));  
 }  
 }  
}

# 7. Building and Running Tests

Build the Solution:

```bash  
dotnet build  
```

Run Tests:

```bash  
dotnet test  
```

# 8. Troubleshooting

Issue Solution  
Type or namespace not found Ensure correct using directive and that the class is public.  
Project reference error Double-check the relative path in dotnet add reference.  
CS8625: Cannot convert null literal Make the field nullable (SimpleCalculator? \_calculator;) or remove the null assignment in [TearDown].  
Build succeeded with warnings Warnings do not prevent running tests, but can be addressed for code quality.

# 9. Best Practices

Always use the correct class and namespace names in your tests.

Make your classes public if they are to be tested from another project.

Use [TestCase] for parameterized tests.

Clean up resources in [TearDown] if necessary.

Keep your test methods descriptive and focused on one functionality.

# 10. Sample Output

Build succeeded: Indicates successful compilation.

dotnet test: Should show all tests passing or failing with details.

# 11. References

NUnit Documentation

.NET CLI Documentation

Submitted by : Subhodeep Mukherjee