

# Unit Testing Methods

Unit Testing Concepts. Testing Frameworks.  
NUnit. Writing Your First Test with NUnit



**SoftUni Team**  
Technical Trainers



**SoftUni**



**Software University**

<https://softuni.bg>

[sli.do](https://sli.do)

**#prgm-for-qa**

1. Unit **Testing** and Testing **Frameworks**
2. **NUnit** Basics: Automated Testing with C# and NUnit
3. The **AAA** Pattern: **A**rrange, **A**ct, **A**ssert
4. Assertions in NUnit: `Assert.That(...)`
5. Code **Coverage**
6. Best Practices

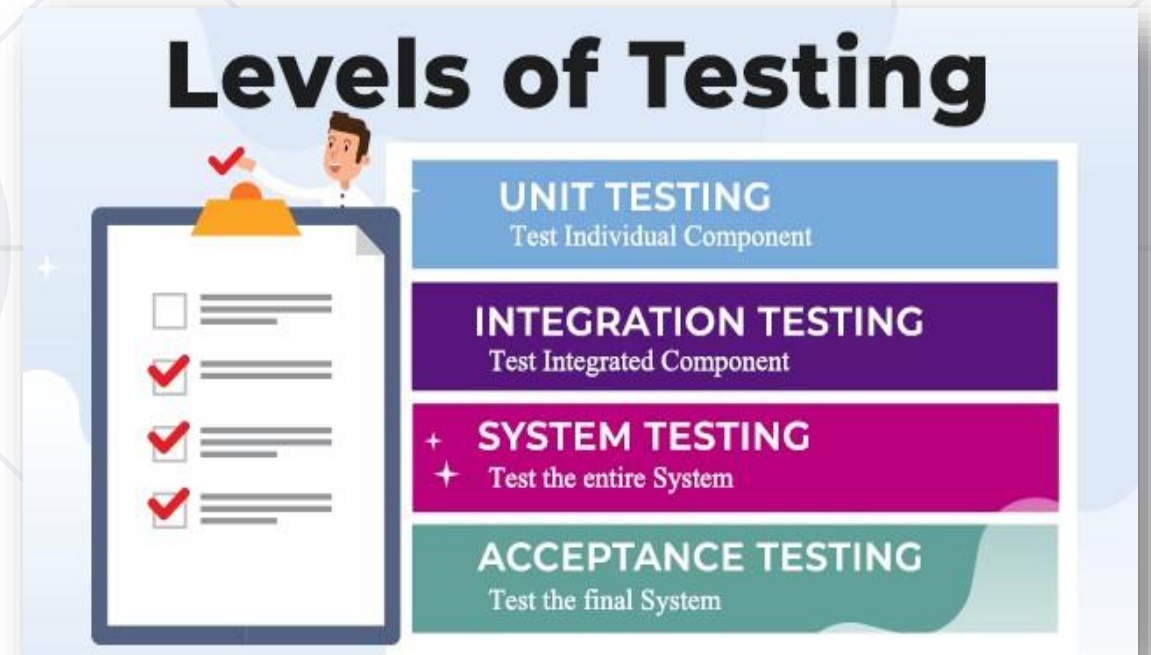




# **What is Unit Testing?**

Automated Testing of Software Components (Units)

- **Unit tests**
  - Test a **single component**
  - NUnit, JUnit, PyUnit, Mocha
- **Integration tests**
  - Test an **interaction** between components, e. g. **API tests**
- **System tests / end-to-end tests**
  - Test the **entire system**
  - Selenium, Appium, Cypress, Playwright



- **Unit test** == a piece of code that **tests specific functionality** in certain software component (unit)

```
int Sum(int[] arr)
{
    int sum = 0;
    foreach (int num in arr)
        sum += num;
    return sum;
}
```

```
void Test_SumTwoNumbers()
{
    if (Sum(new[] {1, 2}) != 3)
        throw new Exception(
            "1+2=3!");
}
```

```
sum(arr)
✓ sum([1+2]) == 3
✓ sum([1-2]) == -1
1) sum([1]) == 0

2 passing (10ms)
1 failing
```



# Testing Frameworks

Concepts

- **Testing frameworks** provide foundation for test automation
  - Structure the tests into hierarchical or other form
  - Create and run **test cases**, then make **reports**
  - Check the **results** and **exit** conditions
  - Perform initialization at **startup** and cleanup at **shut down**
- **Examples of testing frameworks:**
  - **NUnit**, xUnit, MSTest (C#), **Junit**, TestNG (Java), **Mocha**, Jest (JS), **PyTest** (Python)

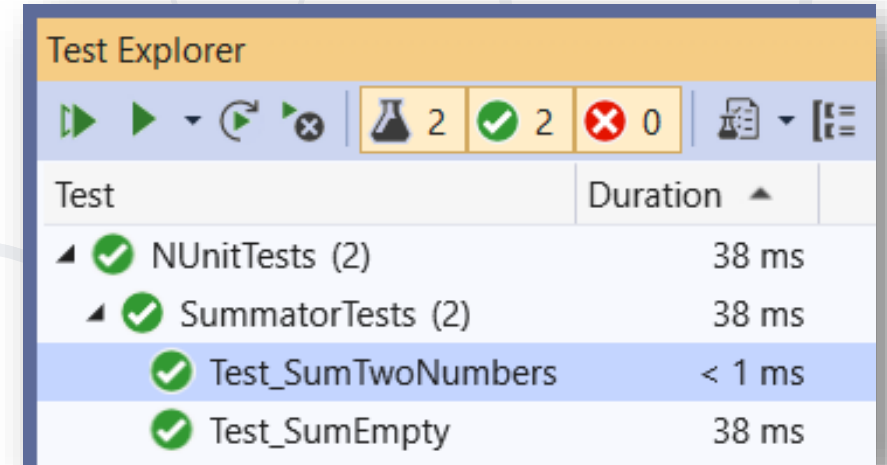




- Testing frameworks simplify automated testing and reporting
  - Example: **NUnit** testing framework for C#

```
using NUnit.Framework;

public class SummatorTests
{
    [Test]
    public void Test_SumTwoNumbers()
    {
        var sum = Sum(new[] { 1, 2 });
        Assert.AreEqual(3, sum);
    }
}
```



The screenshot shows the Test Explorer window with a toolbar at the top containing icons for running tests, a test count summary (2 passed, 2 failed, 0 errored), and a list icon. Below the toolbar is a table with two columns: 'Test' and 'Duration'.

Test	Duration
▲ ✓ NUnitTests (2)	38 ms
▲ ✓ SummatorTests (2)	38 ms
✓ Test_SumTwoNumbers	< 1 ms
✓ Test_SumEmpty	38 ms



# **NUnit: First Steps**

Setup and First Test

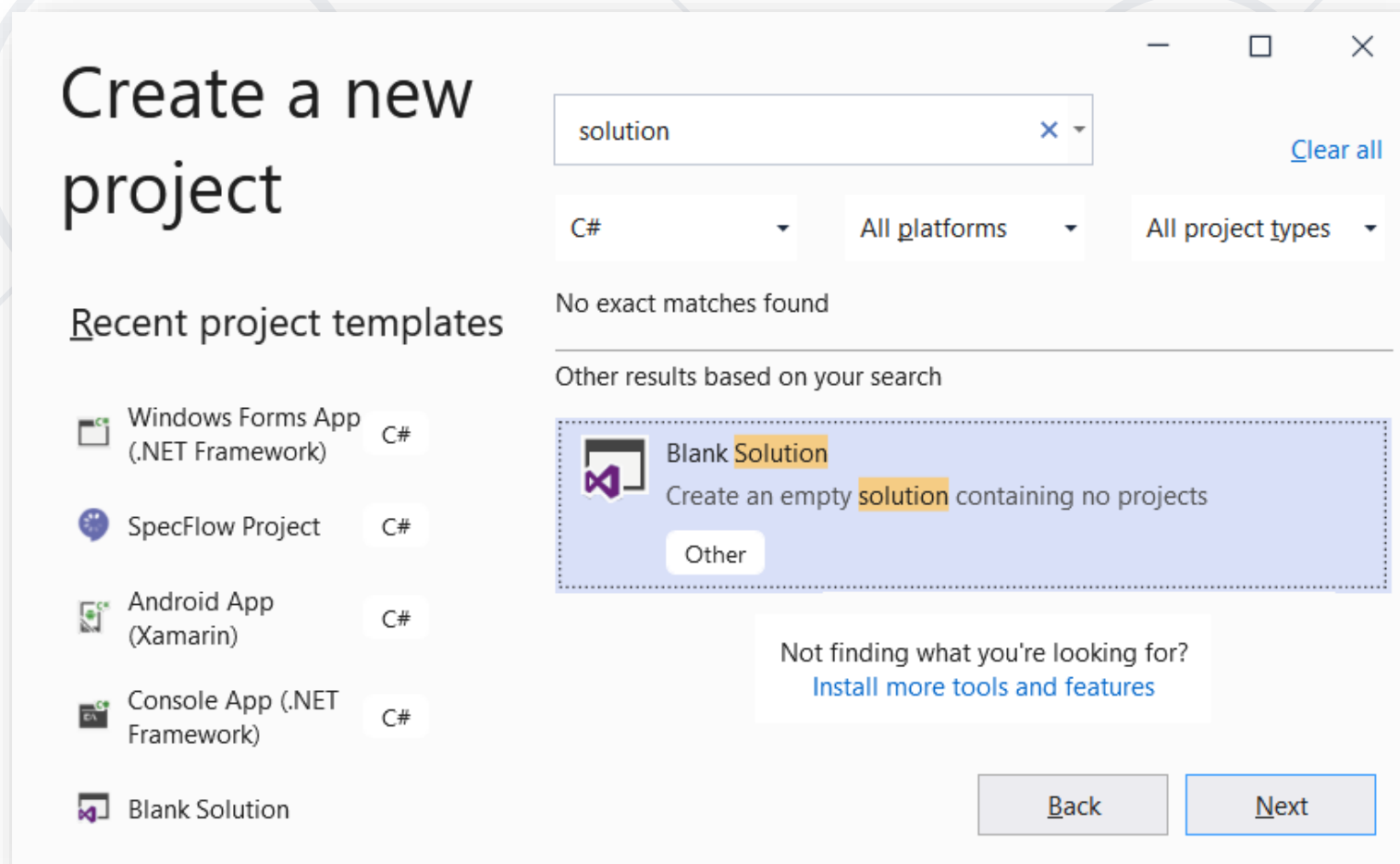
# Setup and First Test

- **NUnit** == popular C# testing framework
  - Supports test **suites**, test **cases**, **before & after** code, **startup & cleanup** code, **timeouts**, expected **errors**, ...
  - Free, open-source
  - Powerful and mature
  - Wide community
  - Built-in support in Visual Studio
  - Official site: [nunit.org](https://nunit.org)



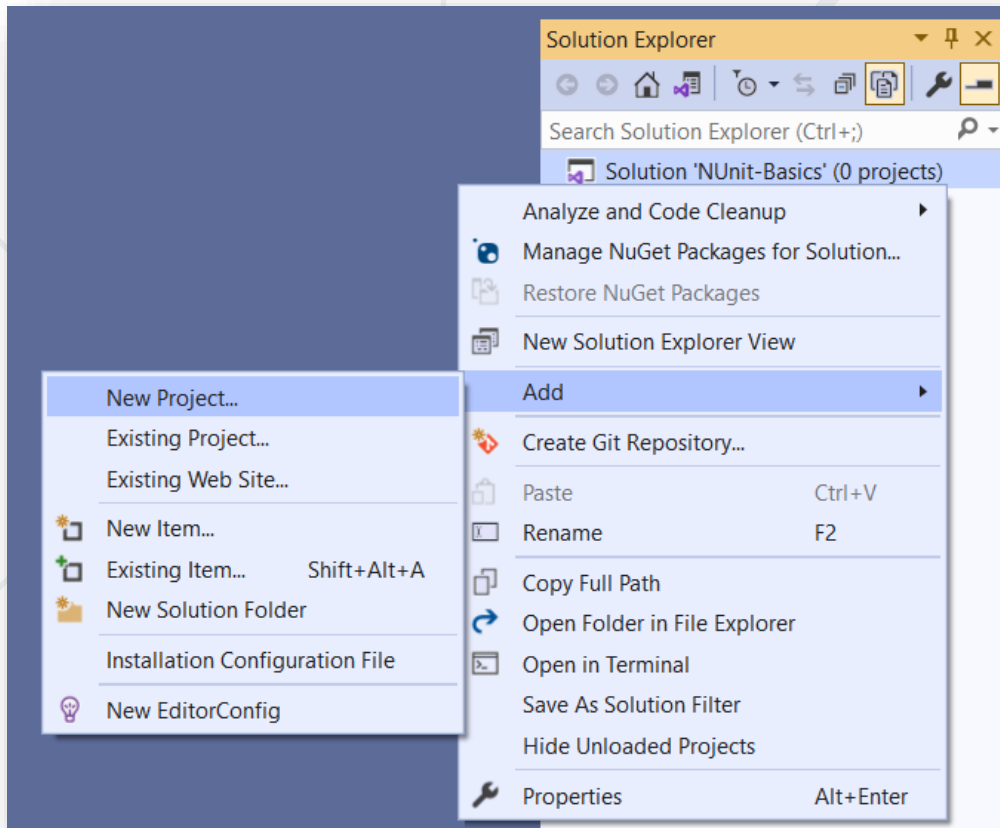
# Creating a Blank Solution

- Create a **blank solution** in Visual Studio
  - It will hold the **project for testing**
  - And the **unit test project (tests)**



# Creating a Project for Testing

- **Add a New Project in your Solution, to hold the code for testing**



## Add a new project

### Recent project templates

- Windows Forms App (.NET Framework) C#
- SpecFlow Project C#
- Android App (Xamarin) C#
- Console App (.NET Framework) C#
- Blank Node.js Console Application JavaScript

Search for templates (Alt+S) [Clear all](#)

C# All platforms All project types

**Console Application**  
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS  
C# Linux macOS Windows Console

**Class library**  
A project for creating a class library that targets .NET Standard or .NET Core  
C# Android Linux macOS Windows Library

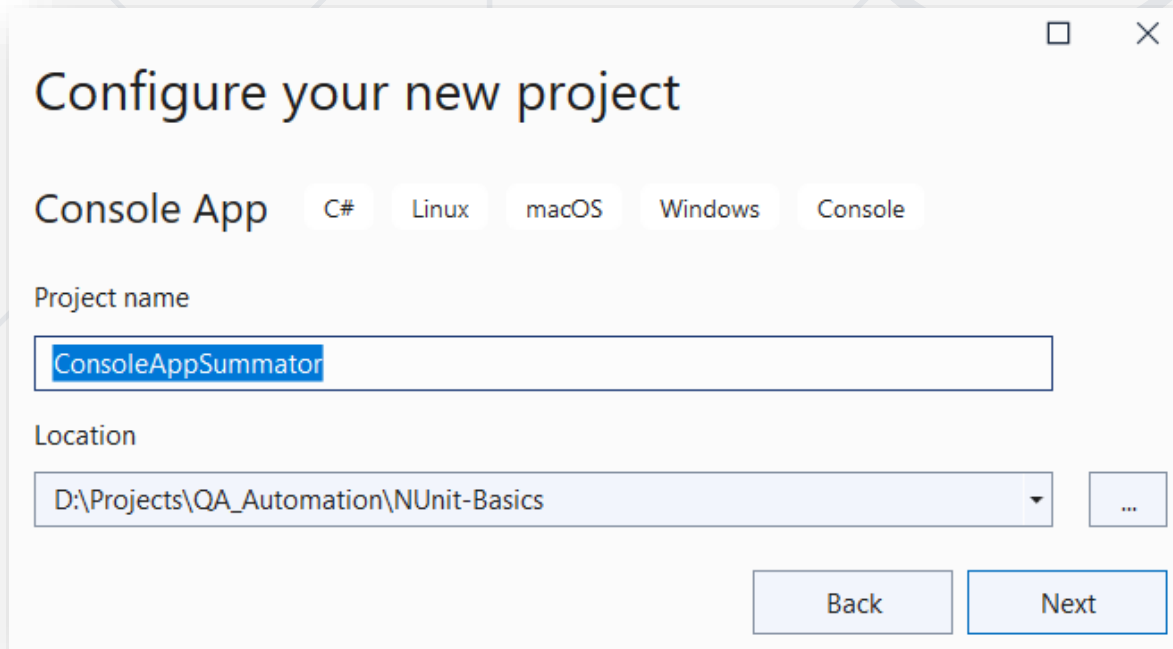
**ASP.NET Core Empty**  
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.  
C# Linux macOS Windows Cloud Service Web

**ASP.NET Core Web API**  
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Next

# Creating a Project for Testing

- Choose a **meaningful name** for your project
- Choose a **place to store it**
- Use **.NET 6.0** (Long Term Support)



Configure your new project

Console App C# Linux macOS Windows Console

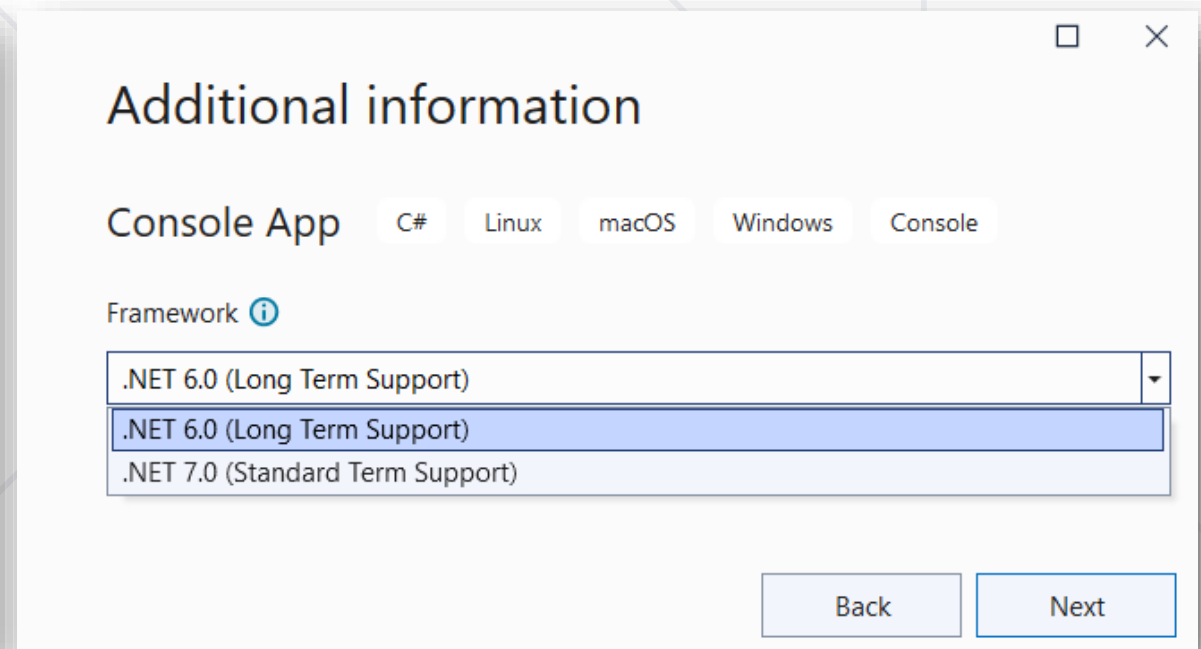
Project name

ConsoleAppSummator

Location

D:\Projects\QA\_Automation\NUnit-Basics

Back Next



Additional information

Console App C# Linux macOS Windows Console

Framework ⓘ

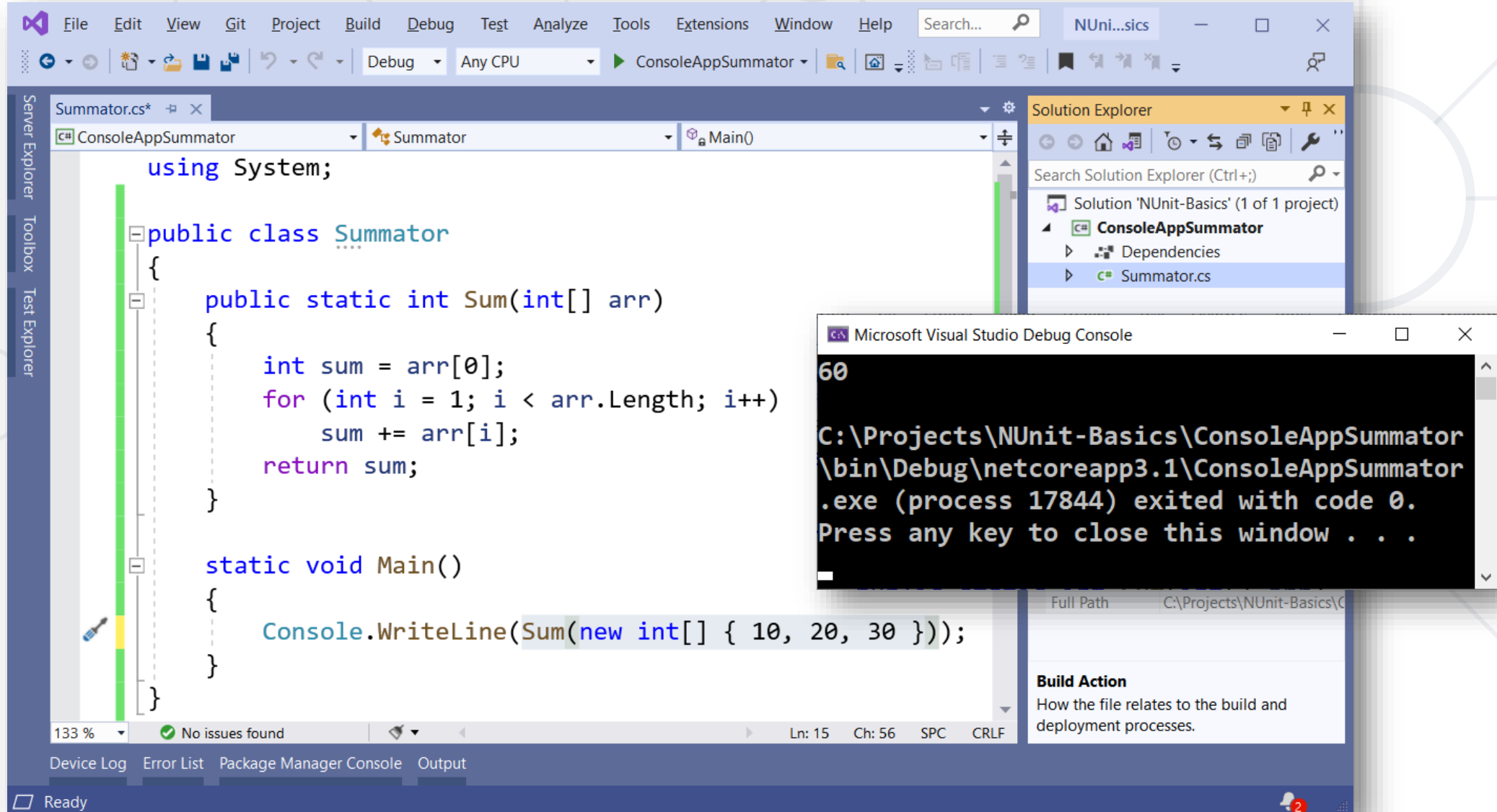
.NET 6.0 (Long Term Support)

.NET 6.0 (Long Term Support)

.NET 7.0 (Standard Term Support)

Back Next

# Creating a Project for Testing



Summator.cs\*

```
using System;

public class Summator
{
    public static int Sum(int[] arr)
    {
        int sum = arr[0];
        for (int i = 1; i < arr.Length; i++)
            sum += arr[i];
        return sum;
    }

    static void Main()
    {
        Console.WriteLine(Sum(new int[] { 10, 20, 30 }));
    }
}
```

Solution Explorer

- Solution 'NUnit-Basics' (1 of 1 project)
  - ConsoleAppSummator
    - Dependencies
    - Summator.cs

Microsoft Visual Studio Debug Console

```
60

C:\Projects\NUnit-Basics\ConsoleAppSummator\bin\Debug\netcoreapp3.1\ConsoleAppSummator.exe (process 17844) exited with code 0.
Press any key to close this window . . .
```

Full Path C:\Projects\NUnit-Basics\...

**Build Action**  
How the file relates to the build and deployment processes.

133 % No issues found Ln: 15 Ch: 56 SPC CRLF

Device Log Error List Package Manager Console Output

Ready

# Creating a NUnit Project

## Create a new project

### Recent project templates

- ASP.NET Core Web Application C#
- Console App (.NET Core) C#
- Windows Forms App (.NET) C#
- NUnit Test Project (.NET Core) C#
- Console App (.NET Framework) C#
- Windows Forms App (.NET Framework) C#

NUnit

C#

All platforms

All project types



xUnit Test Project (.NET Core)

A project that contains xUnit.net tests that can run on .NET Core on Windows, Linux and MacOS.

C#

Windows

Linux

macOS

Test



NUnit Test Project (.NET Core)

A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.

C#

Linux

macOS

Windows

Desktop

Test

Web



xUnit Test Project (.NET Core)

A project that contains xUnit.net tests that can run on .NET Core on Windows, Linux and MacOS.

C#

Windows

Linux

macOS

Test

Next

New Project...

Existing Project...

Existing Web Site...

New Item...

Existing Item... Shift+Alt+A

New Solution Folder

Installation Configuration File

New EditorConfig

Ar

M

Re

Ne

Ac

Cr

Pa

Re

Co

Op

Op

Save As Solution Filter

Hide Unloaded Projects

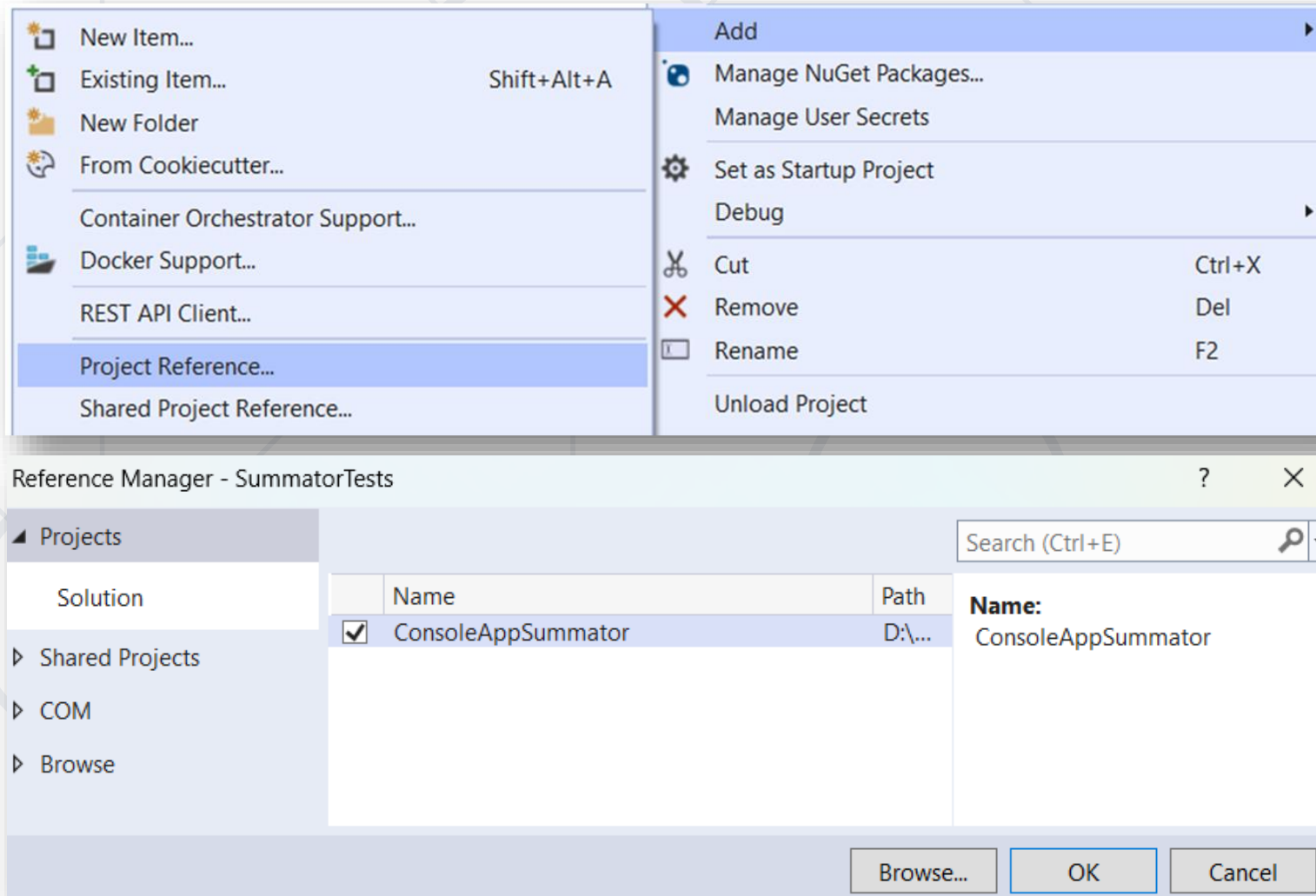
Properties

Alt+Enter



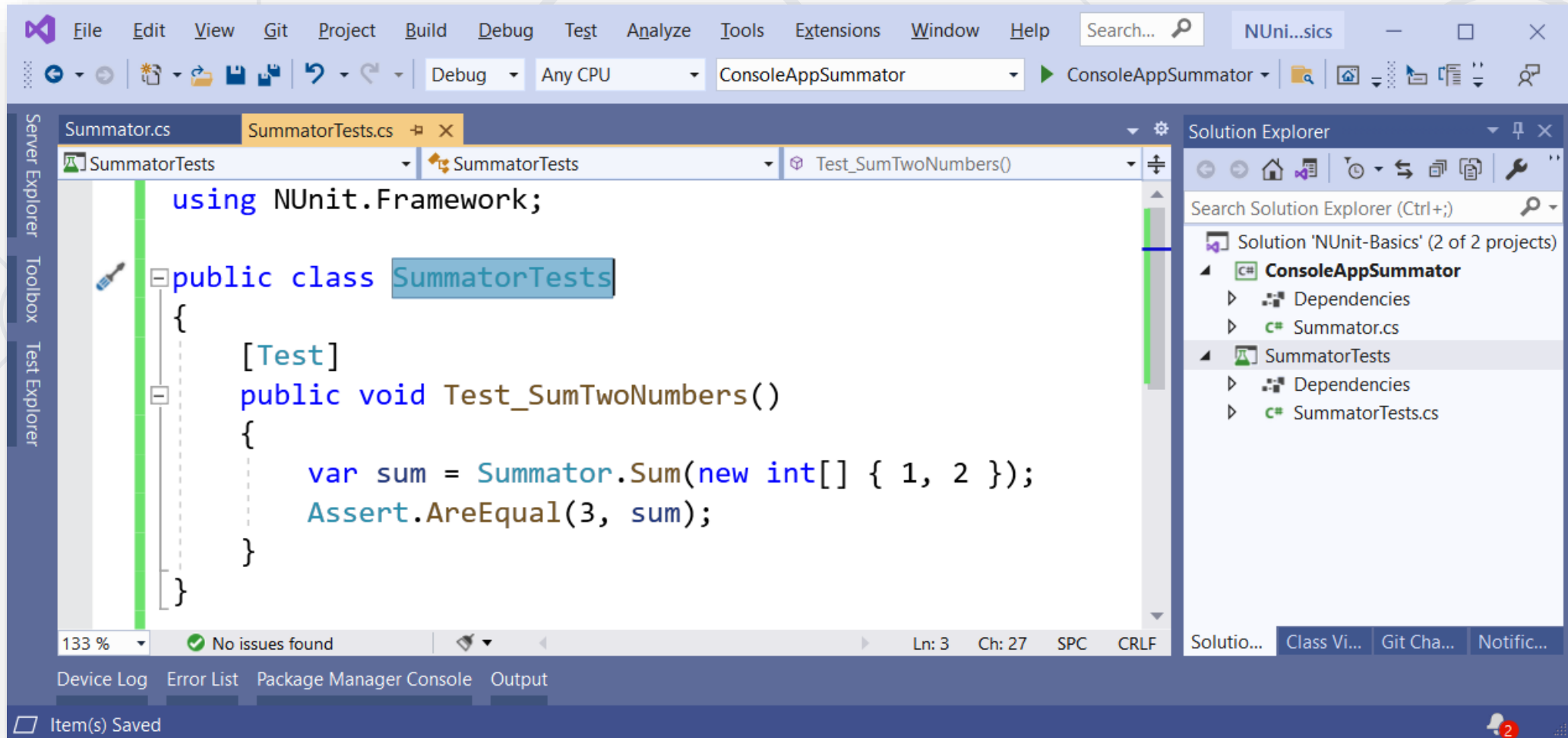
# Adding Project Reference

- Add **Project Reference** to target the project for testing:



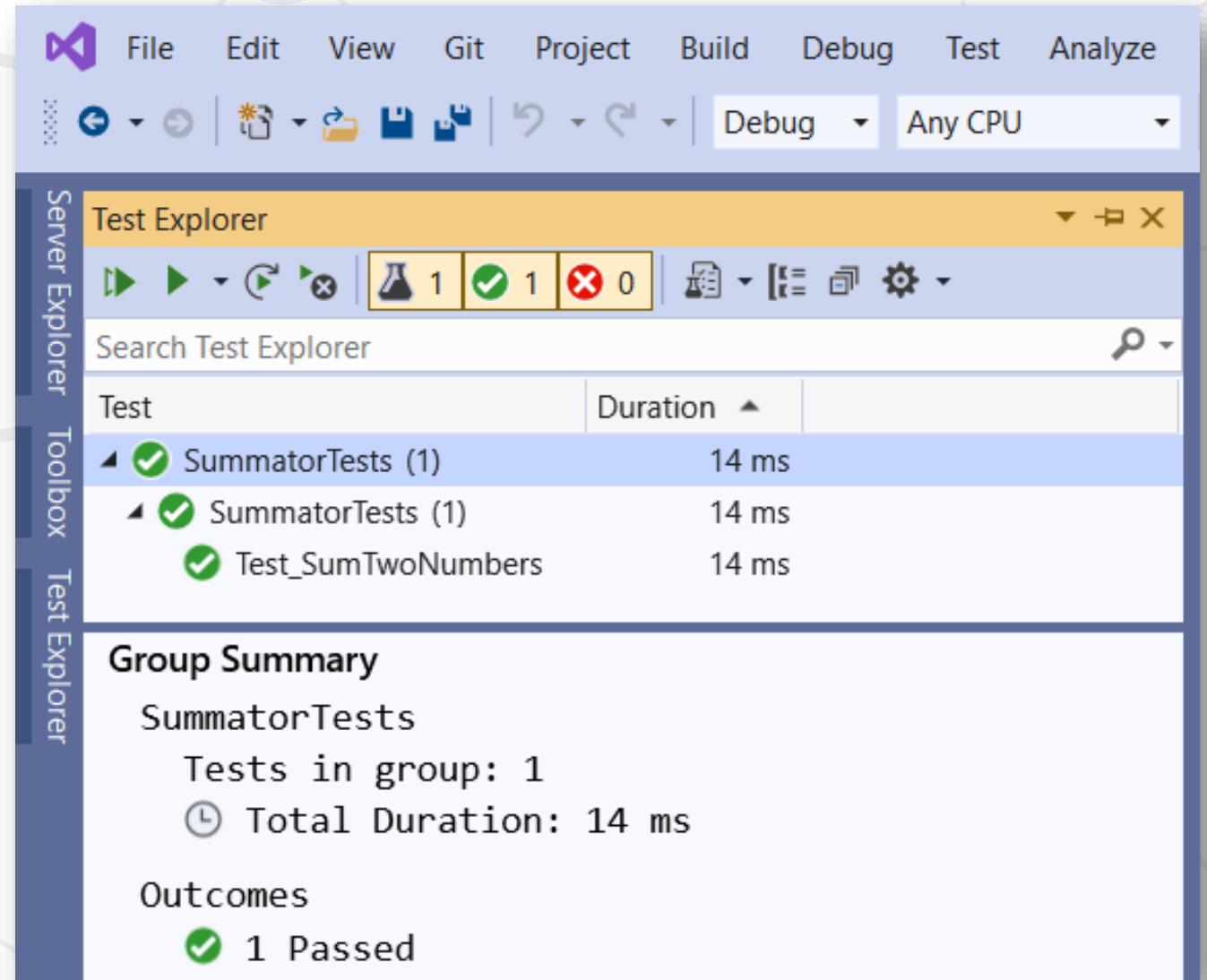
# Writing your First Test

- Writing a NUnit test method:


















































# Test Explorer

- The **[Test Explorer]** tool in Visual Studio
  - Open with **[Ctrl + E] + T**
  - Visualizes the **hierarchy** of tests
  - **Executes** tests
  - **Reports** results




































# Failing Tests Produce Errors

Test Explorer



12 10 2



Search

Test run finished: 1 Tests (0 Passed, 1 Failed, 0 Skipped) run in 249 ms

Test	Duration	Error Message
Unit-Tests (12)	133 ms	
<Empty Namespace> (12)	133 ms	
GradesCalculatorTests (8)	6 ms	
SubArrayTests (1)	74 ms	
SummatorTests (1)	53 ms	
Test_Sum	53 ms	Expected: 8 But was: 7
TriangleTests (2)	< 1 ms	

Test Detail Summary

Test\_Sum

Source: [SummatorTests.cs](#) line 4

Duration: 53 ms

Message:

Expected: 8

But was: 7

Stack Trace:

[SummatorTests.Test\\_Sum\(\)](#) line 8

20



# **NUnit: Basics**

Test Classes and Test Methods

# Test Classes and Test Methods

- Test classes hold **test methods**:

```
using NUnit.Framework;
```

Import NUnit

```
public class SummatorTests  
{
```

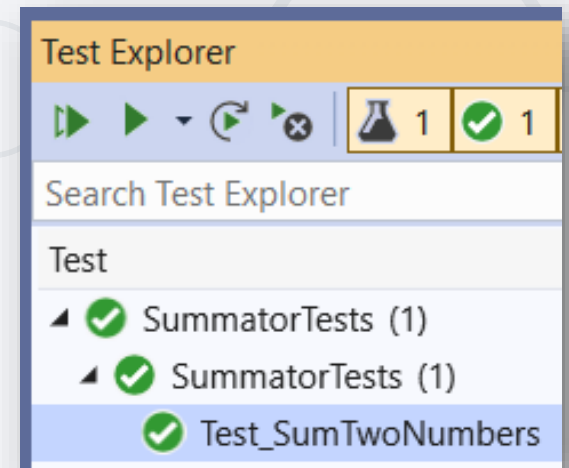
Test class

```
[Test]
```

Test method

```
public void Test_SumTwoNumbers() {  
    var sum = Sum(new int[] { 1, 2 });  
    Assert.AreEqual(3, sum);  
}  
}
```

Assertion



# Initialization and Cleanup Methods

```
private Summator summator;
```

```
[SetUp] // or [OneTimeSetUp]
```

```
public void TestInitialize()
```

```
{
```

```
    this.summator = new Summator();
```

```
}
```

```
[TearDown] // or [OneTimeTearDown]
```

```
public void TestCleanup()
```

```
{
```

```
    // ...
```

```
}
```

Executes **before**  
each test

Executes **after**  
each test

A background network diagram consisting of a central dark blue circle with the text 'AAA' inside. Surrounding this central circle are several smaller, light gray circles connected by thin gray lines, forming a web-like structure. The circles are of varying sizes and are distributed across the frame, with some lines extending towards the edges.

# AAA

## The "AAA" Pattern

Arrange → Act → Assert



# The "AAA" Testing Pattern

- Automated tests usually follow the "**AAA**" pattern
  - **Arrange**: prepare the input data
  - **Act**: invoke the action for testing
  - **Assert**: check the output received

```
[Test]
public void Test_SumNumbers()
{
    // Arrange
    int[] nums = new[] {3, 5};

    // Act
    int sum = Sum(nums);

    // Assert
    Assert.AreEqual(8, sum);
}
```



# Assertions

Checking the Results and Output Conditions

- **Constraint** Model (**Assert.That**)

- The **logic** necessary to carry out each assertion is expressed as a **constraint** passed as the second parameter

```
Assert.That(expected, Is.EqualTo(actual));
```

- **Classic** Model (**Assert.AreEqual**)

- **Assert.AreEqual**, **Assert.True**, **Assert.False**, ...
- Uses a separate method to express each individual assertion:

```
Assert.AreEqual(expected, actual);
```

- Assert that **condition** is true

```
Assert.That(bool condition);
```

- **Comparison** (equal, greater than, less than or equal, ...)

```
Assert.That(actual, Is.EqualTo(expected));
```

```
Assert.AreEqual(expected, actual);
```

- Assertions for **expected exception**

```
Assert.That(() => { code },  
    Throws.InstanceOf<ArgumentOutOfRangeException>());
```

- Assertions can **show messages** to help with **diagnostics**

```
Assert.That(axe.DurabilityPoints, Is.EqualTo(12),  
    "Axe Durability doesn't change after attack");
```

✖ Test Failed - AxeLosesDurabilityAfterAttack  
**Message: Axe Durability doesn't change after attack**  
**Expected: 12**  
**But was: 9**

Failure messages in the tests help finding the problem



# **Unit Testing Best Practices**

Naming, Repeatable, No Dependencies

- **Test names** should answer the question "*what's inside?*"
  - Should use **business domain terminology**
  - Should be **descriptive** and **readable**

```
IncrementNumber() {}  
Test1() {}  
TestTransfer() {}
```



```
Test_DepositAddsMoneyToBalance() {}  
Test_DepositNegativeShouldNotAddMoney() {}  
Test_TransferSubtractsFromSourceAddsToDestAccount() {}
```



- Test cases must be **repeatable**
  - Tests should **behave the same** if you run them many times
  - The expected results must be **consistent** and easily verified
- Test cases should **have no dependencies**
  - The order of test execution should never be important
  - Input data and entrance conditions should be set in the test
  - Test cases may depend on the test initialization only: **[SetUp]**
  - Tests should **cleanup** properly any resources used




# Automation Tests: Good Practices

- **Single scenario** per test case, not multiple

```
[Test]
0 references
public void Test_Calculator()
{
    // Arrange:

    // Act:
    int resultOne = Calculator.Add(a: 1, b: 2);
    int resultTwo = Calculator.Subtract(a: 3, b: 2);


    // Assert:
    Assert.That(resultOne, Is.EqualTo(3));
    Assert.That(resultTwo, Is.EqualTo(1));
}
```



```
[Test]
0 references
public void Test_Calculator_Add()
{
    // Arrange:

    // Act:
    int result = Calculator.Add(a: 1, b: 2);

    // Assert:
    Assert.That(result, Is.EqualTo(3));
}
```



```
[Test]
0 references
public void Test_Calculator_Subtract()
{
    // Arrange:

    // Act:
    int result = Calculator.Subtract(a: 3, b: 2);

    // Assert:
    Assert.That(result, Is.EqualTo(1));
}
```

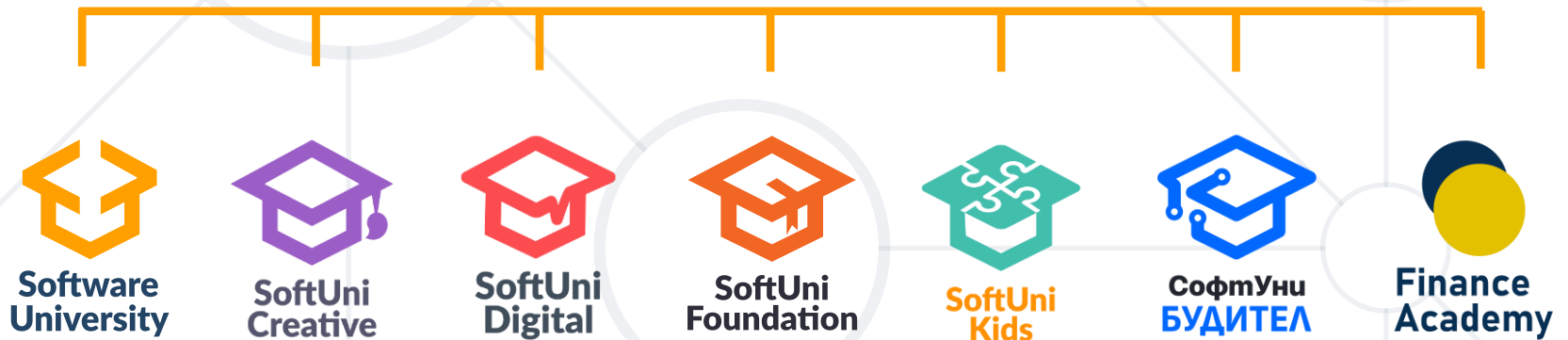
- **Unit testing** - automated testing of single component (unit)
- **NUnit** – automated testing framework for C#
  - **[Test], Assert, [SetUp], [TearDown]**
- The **AAA pattern**: Arrange, Act, Assert
- **Assertion** – checking results / exit conditions
- **Best practices** – naming and conventions



# Questions?



SoftUni



# SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

