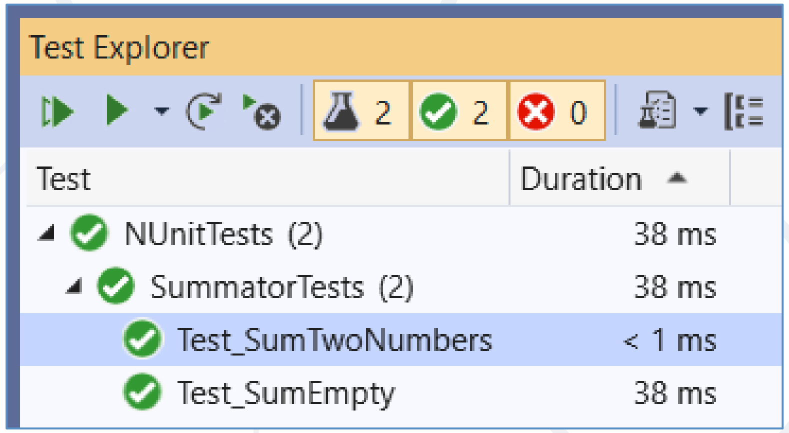
Unit Testing

Unit Testing Concepts. Testing Frameworks.
NUnit. Writing Automated Tests with NUnit



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

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

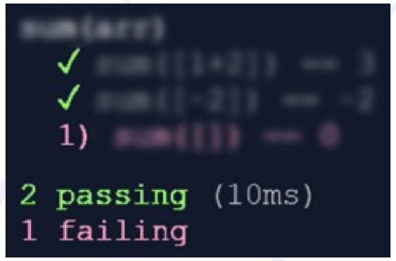
Table of Contents

What is Unit Testing?

Automated Testing of Software Components (Units)

Unit Testing

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



```
[illegible]
  ✓ [illegible]
  ✓ [illegible]
  1) [illegible]

2 passing (10ms)
1 failing
```

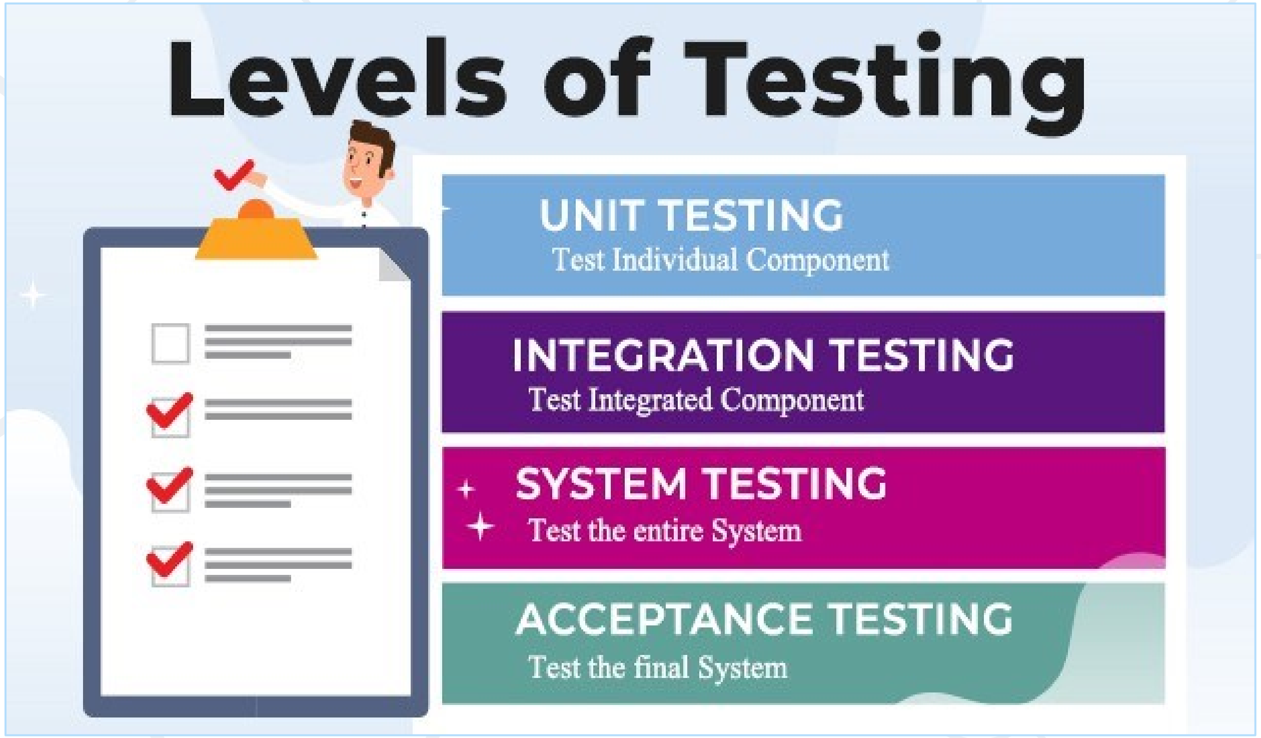
```
int Sum(int[] arr)
{
  int sum = arr[0];
  for (int i=1; i<arr
    .Length; i++)
      sum += arr[i];
  return sum;
}
```

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

```
void Test_SumEmptyArray() {
  if (Sum(new int[]{ }) != 0)
    throw new Exception("sum [] != 0");
}
```

Test Levels

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



Testing Frameworks

Concepts

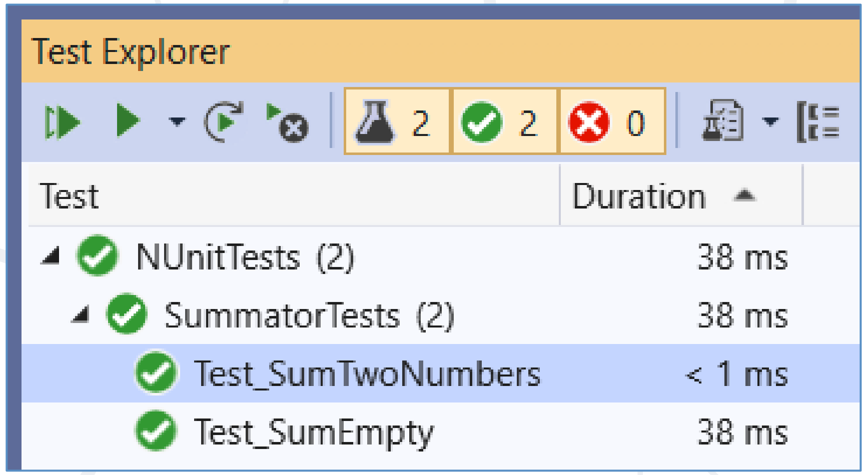
Testing Frameworks

- Testing frameworks provide **foundation for test automation**
  - Consists of **libraries**, code **modules** and **tools** for test automation
  - **Structure the tests** into hierarchical or other form
  - **Implement** test cases, **execute the tests** and **generate reports**
  - **Assert** the execution results and exit conditions
  - Perform initialization at **startup** and cleanup at **shut down**
- **Examples** of testing frameworks:
  - NUnit, xUnit, MSTest (C#), JUnit (Java), Mocha (JS), PyUnit (Python)

Testing Framework – Example

- **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 int[] {1, 2});
    Assert.AreEqual(3, sum);
  }
}
```


Test Explorer

2 | 2 | 0

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

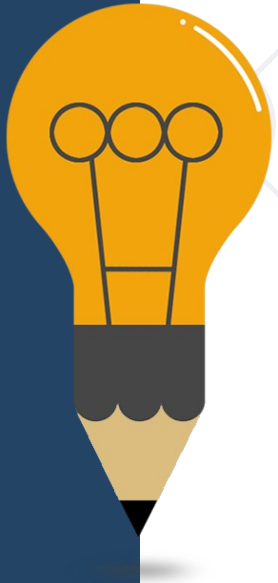
Unit Testing Framework vs. Testing Framework

- **Unit testing framework** == **automated testing framework** == **testing framework** == **test framework**
  - Many names for similar concepts → why?
- Testing frameworks like **JUnit** and **NUnit** were initially designed for **unit testing**, but nothing limits them to wider use
- With additional libraries, NUnit and JUnit are used for:
  - **Integration testing**, **API testing**, Web service testing
  - **End-to-end testing**, **Web UI testing**, **mobile testing**, etc.

Setup and First Test

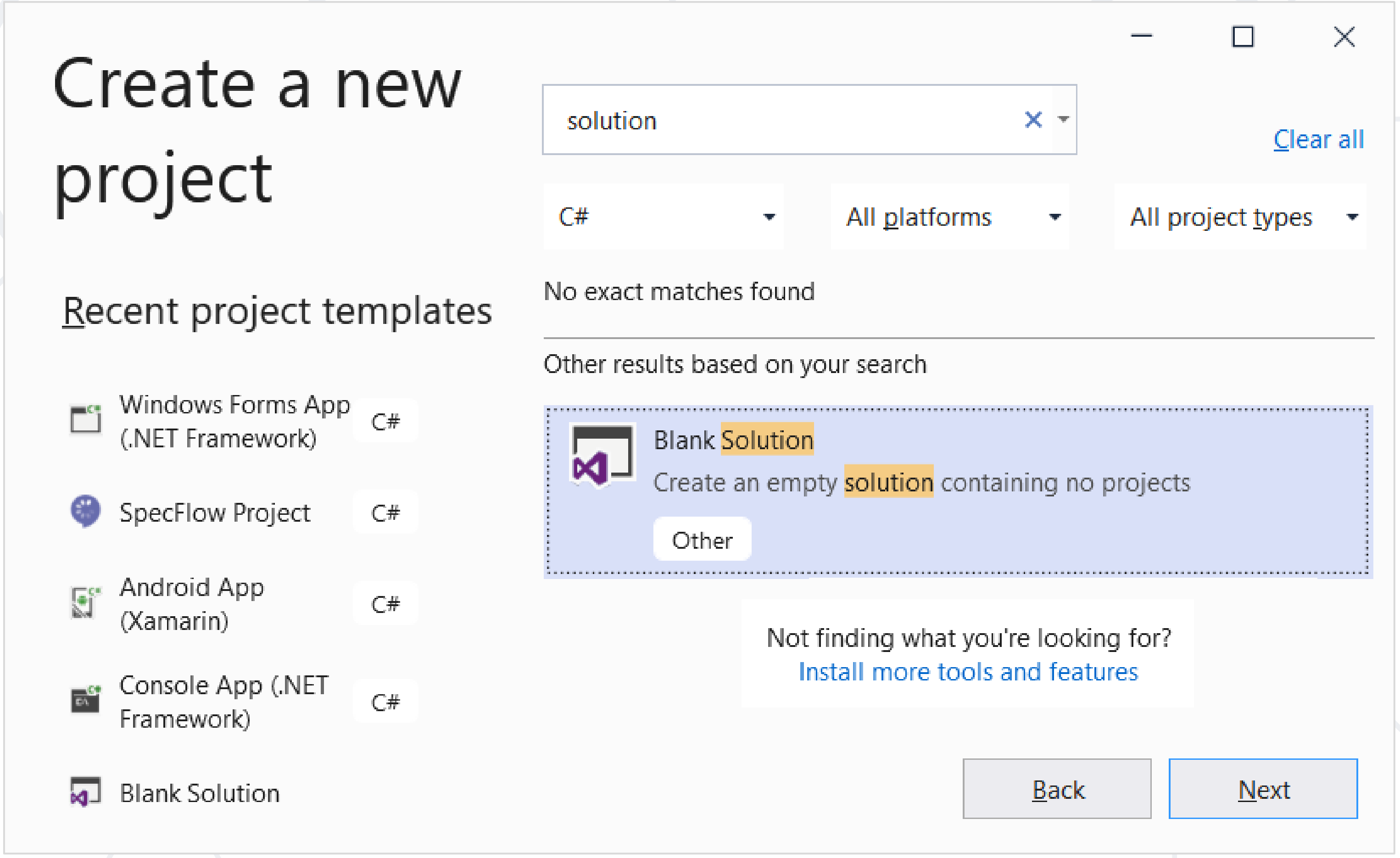
NUnit: Overview

- **NUnit** == popular C# testing framework
  - Supports test suites, test cases, before & after code, startup & cleanup code, timeouts, expected errors, …
  - Like **JUnit** (for Java)
  - Free, open-source
  - Powerful and mature
  - Wide community
  - Built-in support in Visual Studio
  - Official site: 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)



Create a new project

solution

Clear all

C# | All platforms | All project types

Recent project templates

Windows Forms App (.NET Framework)	C#
SpecFlow Project	C#
Android App (Xamarin)	C#
Console App (.NET Framework)	C#
Blank Solution	

No exact matches found

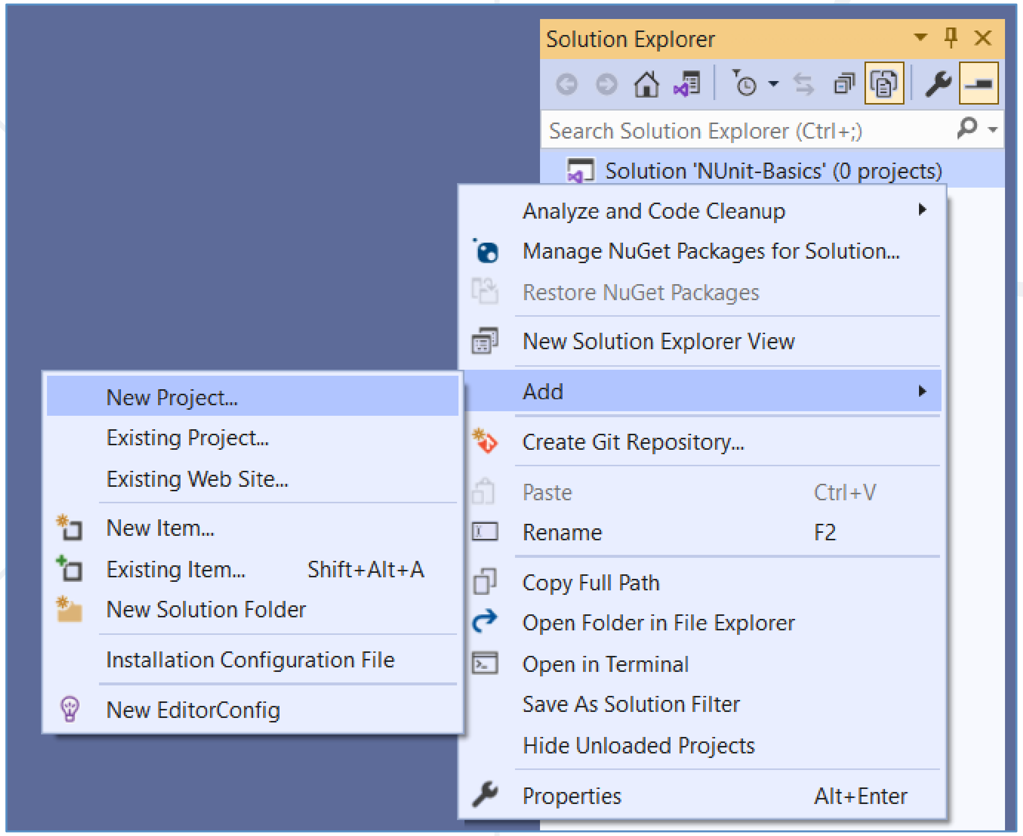
Other results based on your search

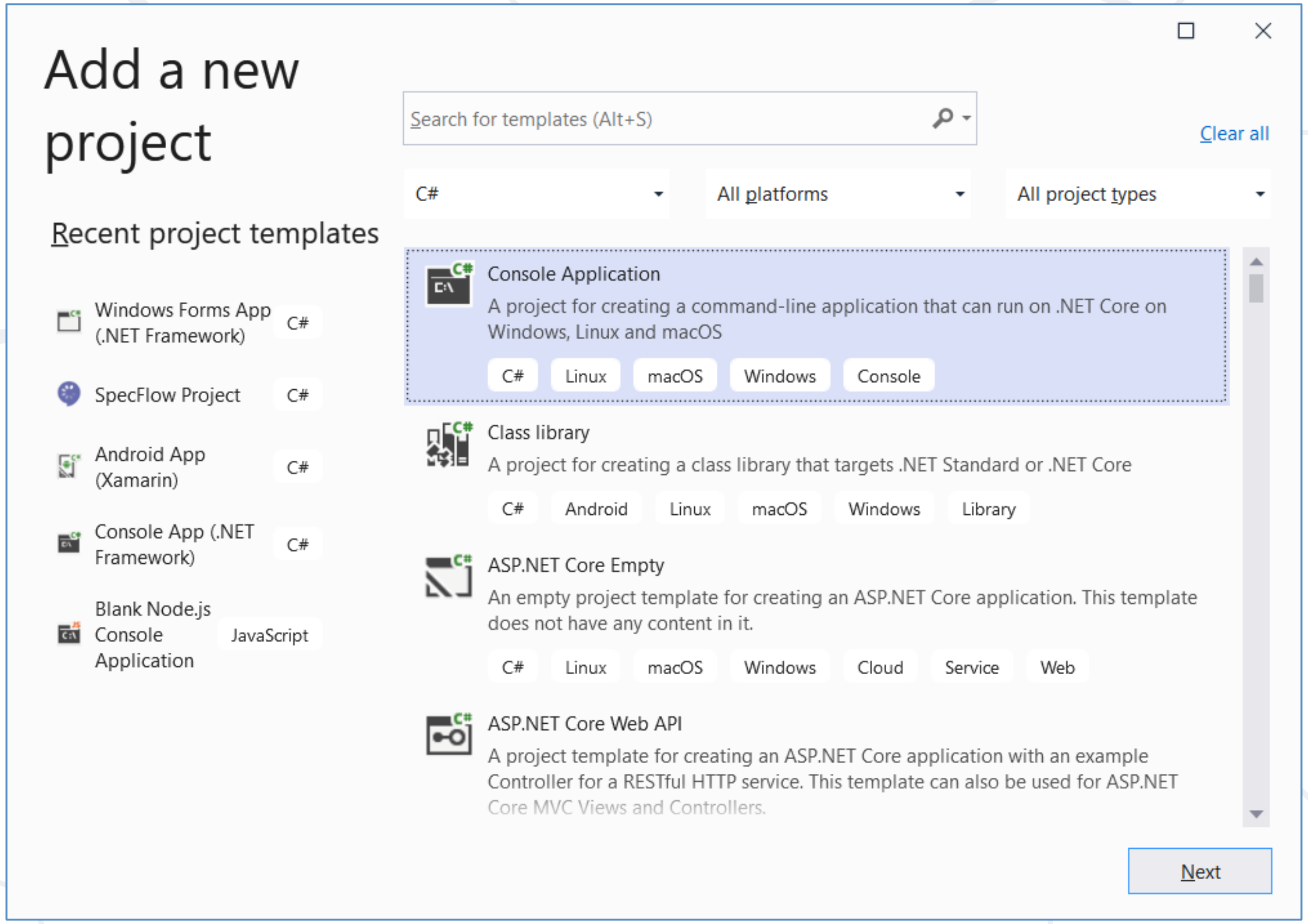
Blank Solution
Create an empty solution containing no projects
Other

Not finding what you're looking for?
Install more tools and features

Creating a Project for Testing (1)

- Create a **console-based app**, to hold the **code for testing**





Add a new project

Search for templates (Alt+S)

Clear all

C# | All platforms | All project types

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

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

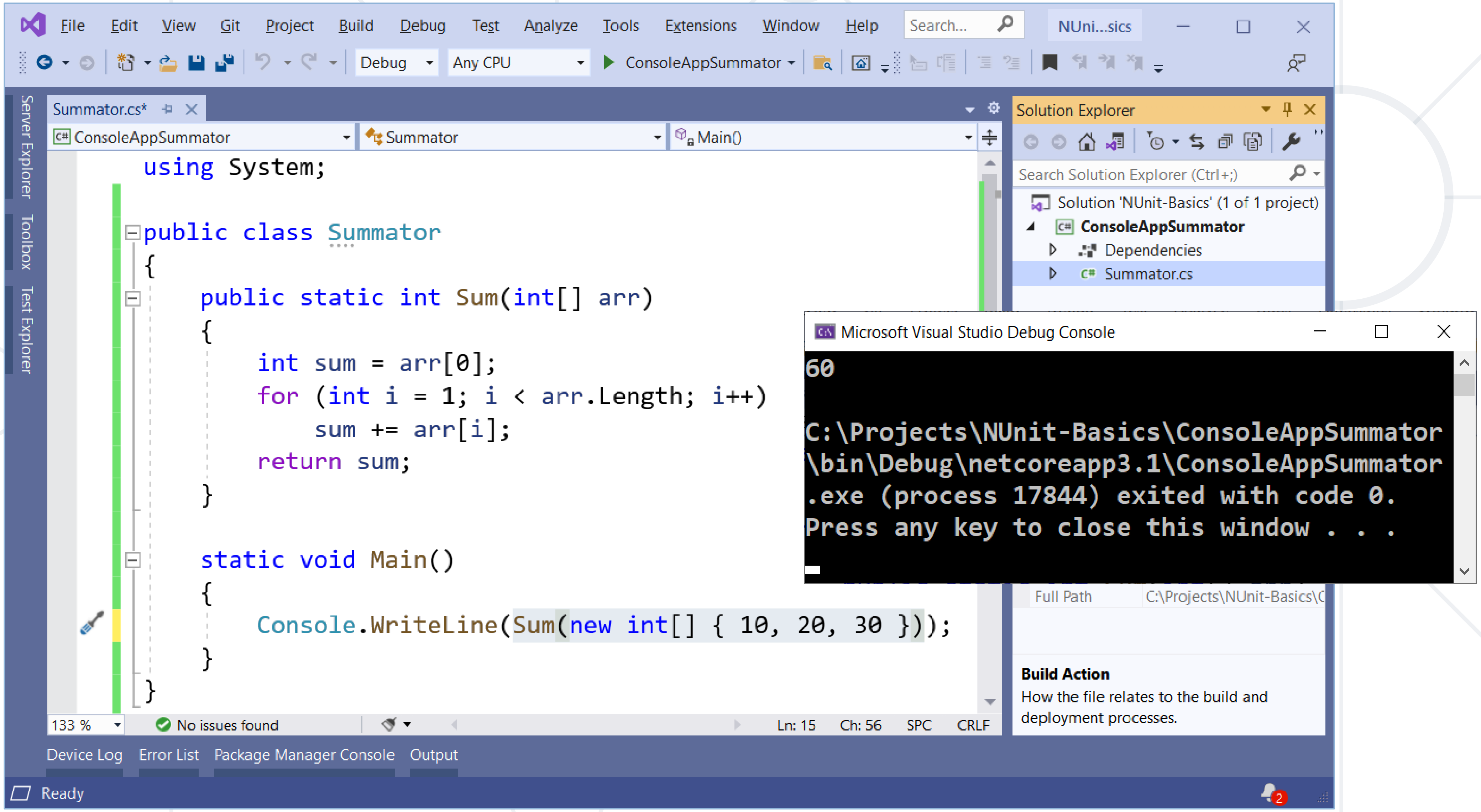
Class library
A project for creating a class library that targets .NET Standard or .NET Core
C# Android Linux macOS Windows Library

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

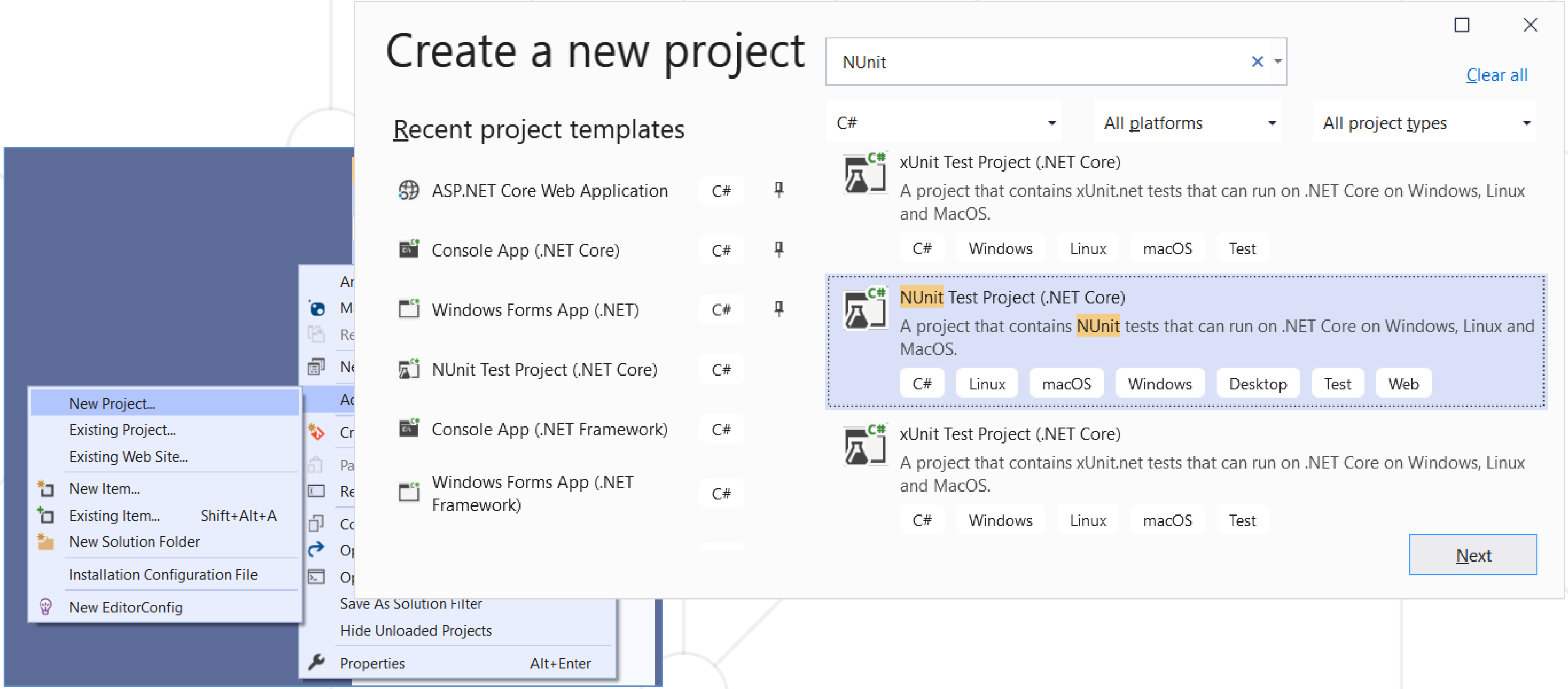
Creating a Project for Testing (2)



```csharp
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 }));
    }
}
```

```
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 . . .
```

Creating an NUnit Project



Create a new project

NUnit

Clear all

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#

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

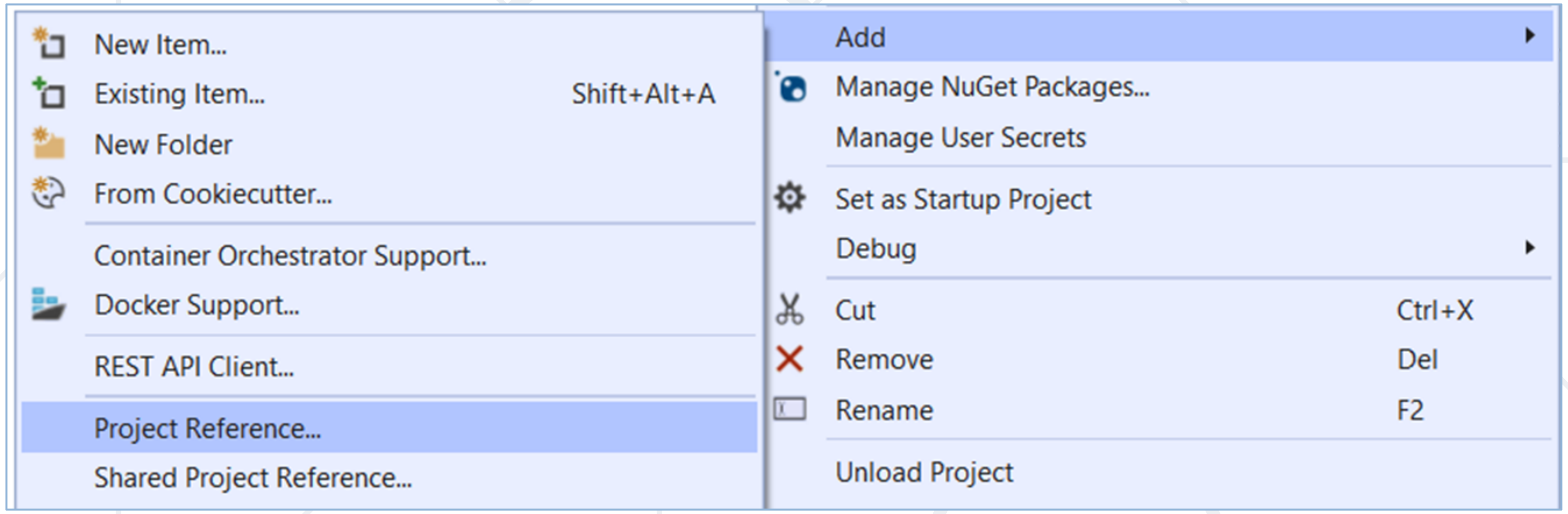
NUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web

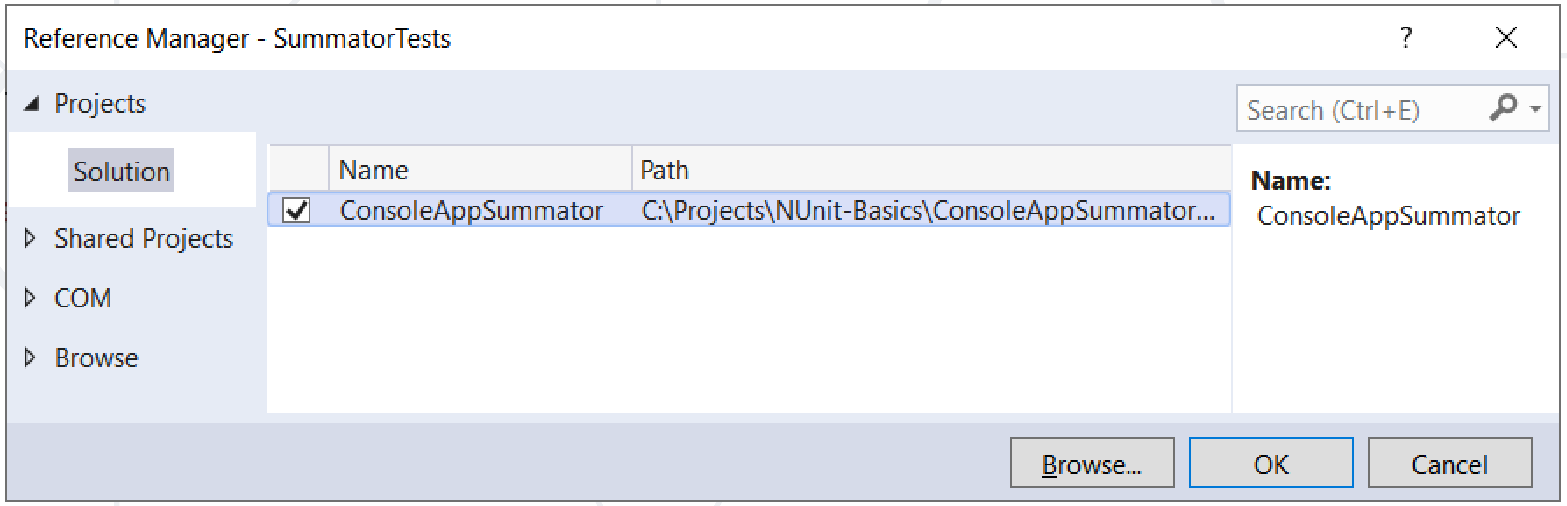
xUnit Test Project (.NET Core)
A project that contains xUnit.net tests that can run on .NET Core on Windows, Linux and MacOS.
C# Windows Linux macOS Test

Next

Adding Project Reference

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



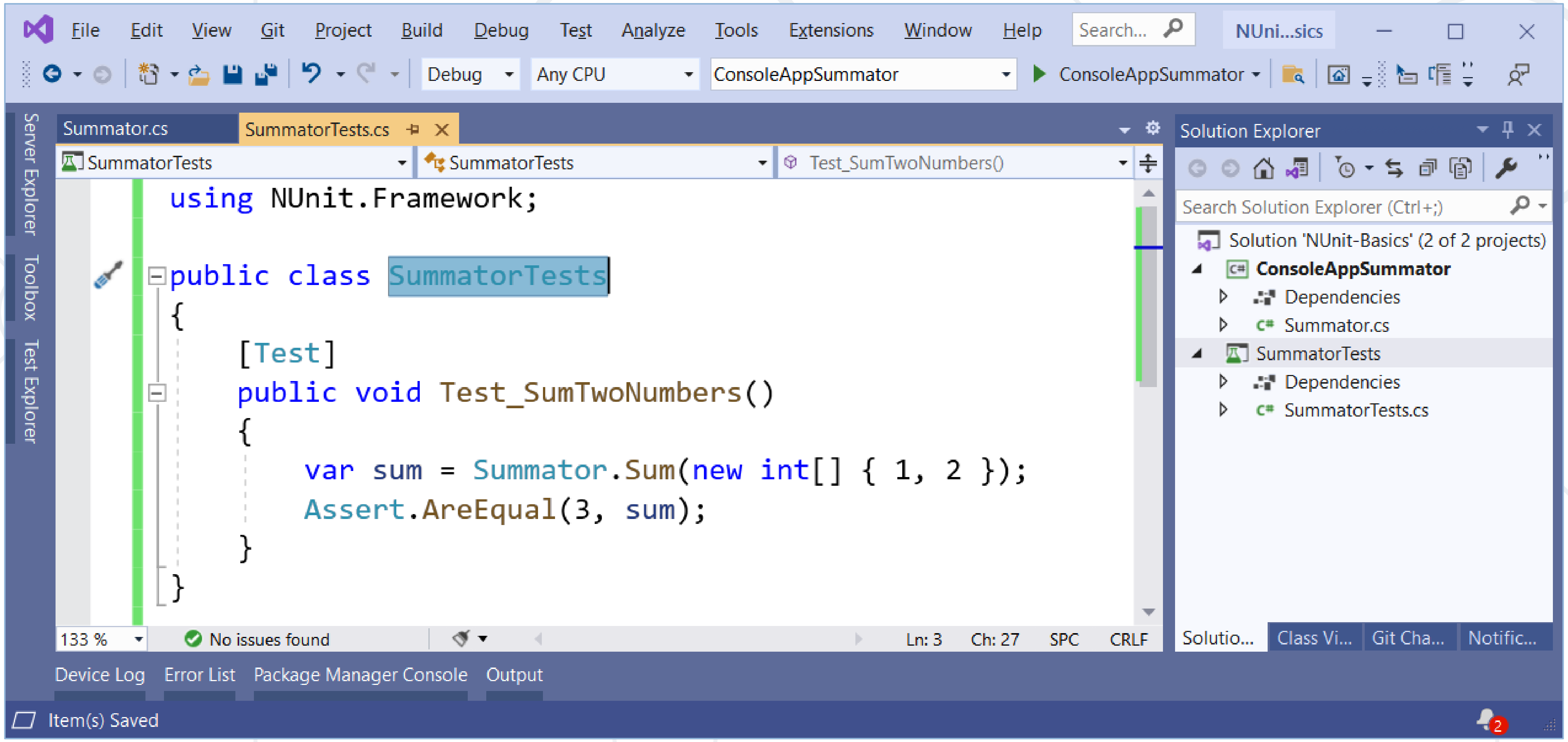


	Name	Path
✓	ConsoleAppSummator	C:\Projects\NUnit-Basics\ConsoleAppSummator...

Name:
ConsoleAppSummator

Writing the First Test

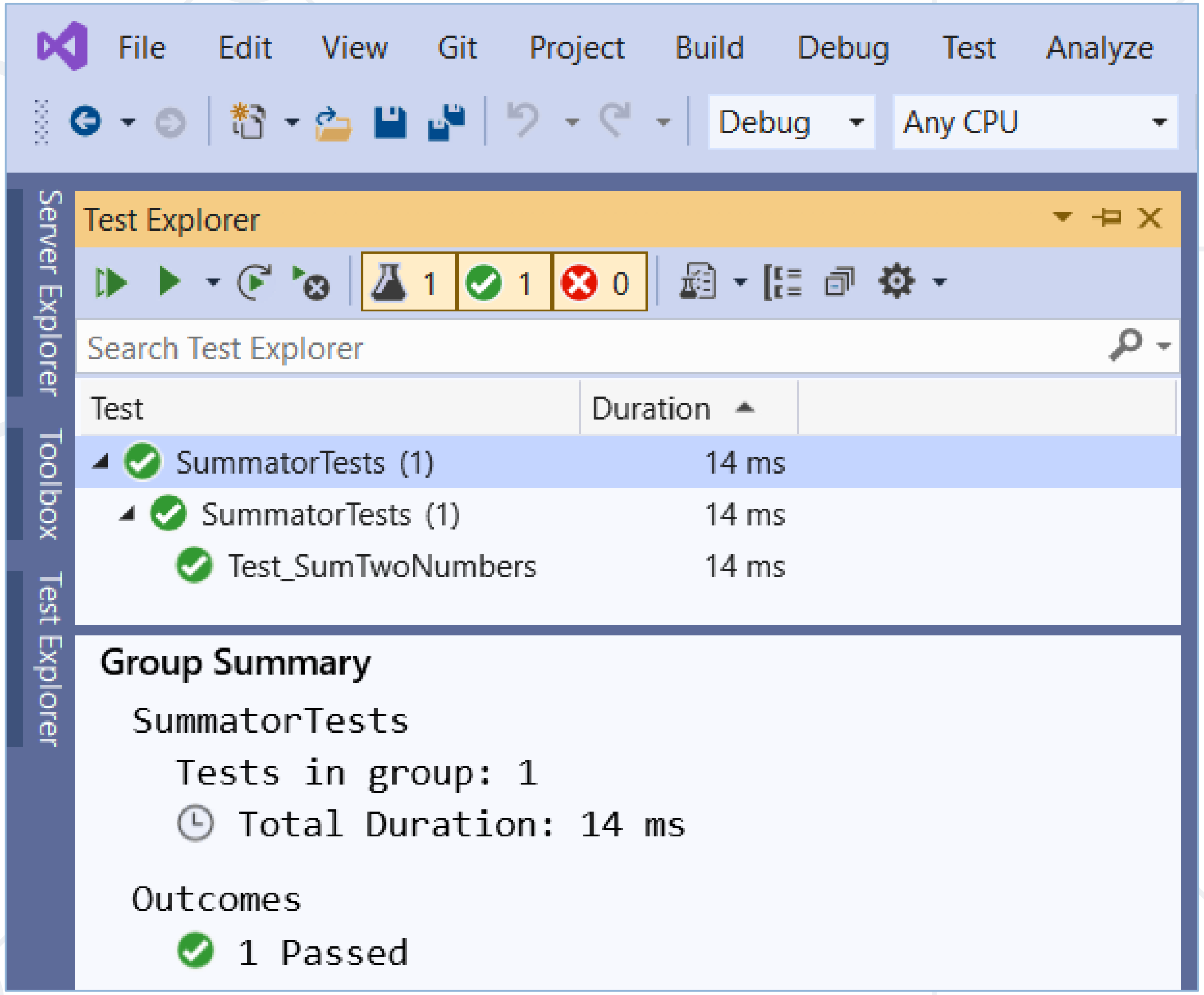
- Writing the first **NUnit test method**:



```
using NUnit.Framework;

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

Running the Tests

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



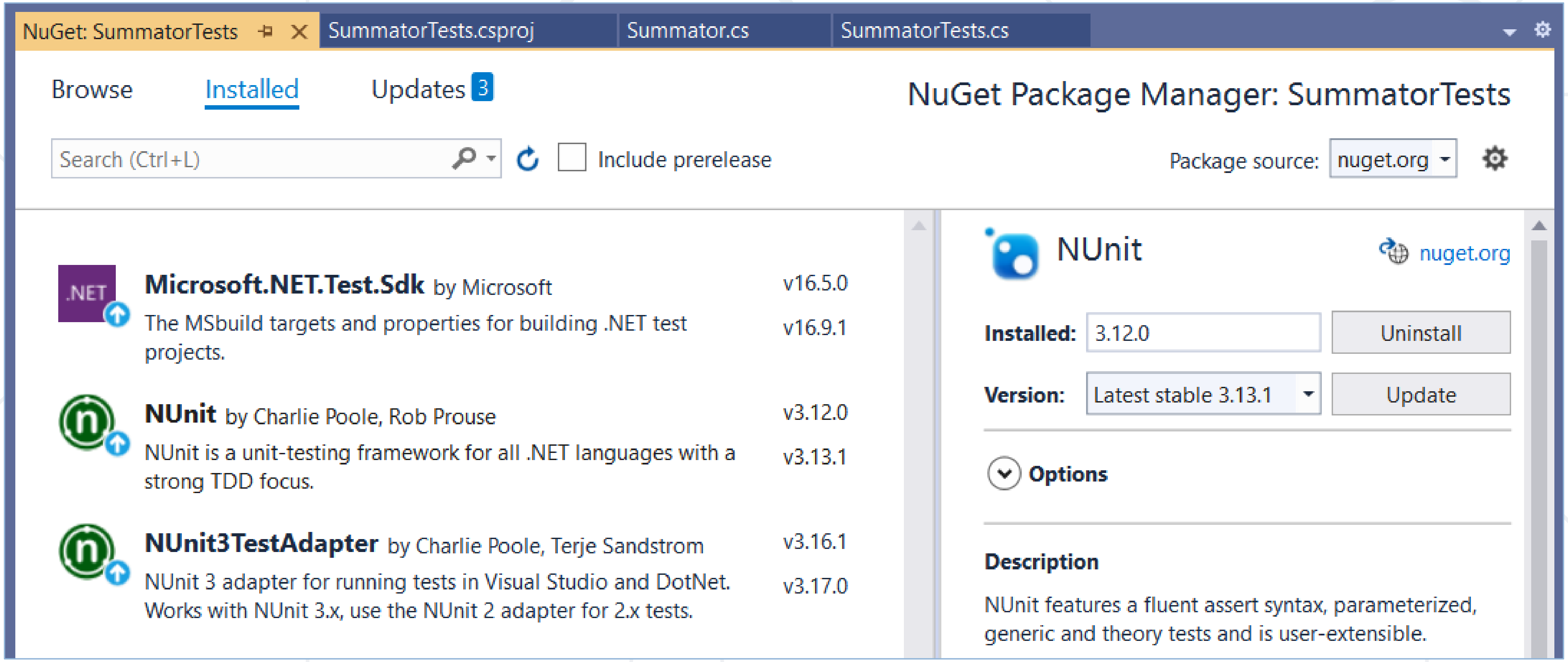
Test	Duration
SummatorTests (1)	14 ms
SummatorTests (1)	14 ms
Test_SumTwoNumbers	14 ms

Group Summary

```
SummatorTests
  Tests in group: 1
  Total Duration: 14 ms

Outcomes
  1 Passed
```

NUnit: NuGet Packages

- **NuGet packages**, required to run NUnit tests in Visual Studio



NuGet: SummatorTests | SummatorTests.csproj | Summator.cs | SummatorTests.cs

Browse | Installed | Updates 3

NuGet Package Manager: SummatorTests

Search (Ctrl+L) Include prerelease Package source: nuget.org

Microsoft.NET.Test.Sdk by Microsoft — v16.5.0 / v16.9.1
The MSbuild targets and properties for building .NET test projects.

NUnit by Charlie Poole, Rob Prouse — v3.12.0 / v3.13.1
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.

NUnit3TestAdapter by Charlie Poole, Terje Sandstrom — v3.16.1 / v3.17.0
NUnit 3 adapter for running tests in Visual Studio and DotNet. Works with NUnit 3.x, use the NUnit 2 adapter for 2.x tests.

NUnit — nuget.org

Installed: 3.12.0 — Uninstall

Version: Latest stable 3.13.1 — Update

Options

Description

NUnit features a fluent assert syntax, parameterized, generic and theory tests and is user-extensible.

Test Classes and Test Methods

- **Test classes** hold **test methods**:

```
using NUnit.Framework;

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

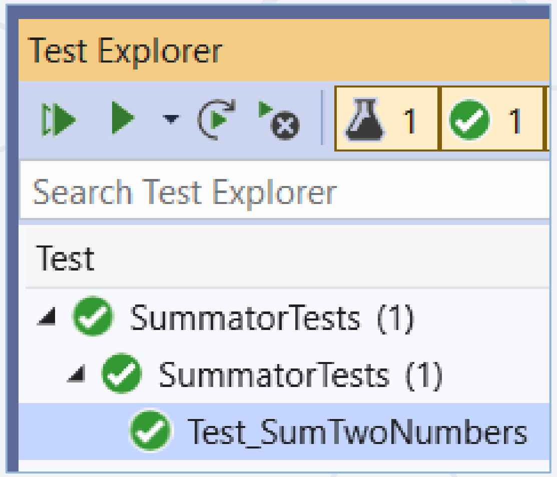
Import NUnit

Optional notation

Test class

Test method

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

AAA

Arrange ⍰ Act ⍰ Assert

The "AAA" Testing Pattern

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

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

  // Act
  var sum = Sum(nums);

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

Checking the Results and Output Conditions

Assertions (1)

- 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);
```

- **Range** assertions

```
double percentage = 99.95;
Assert.That(percentage, Is.InRange(80, 100));
```

Assertions (2)

- **String** assertions

```
Assert.That(string actual,
   Does.Contain(string expected));
```

- Assertions by **regex matching**

```
string date = "7/11/2021";
Assert.That(date, Does.Match(@"^\d{1,2}/\d{1,2}/\d{4}$"));
```

- Assertions for **expected exception**

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

Assertions (3)

- Collection assertions

```
Assert.That(IEnumerable expected,
  Has.Member(object actual));
```

- Collection range assertions

```
var percentages = new int[] { 10, 30, 50, 100 };
Assert.That(percentages, Is.All.InRange(0, 100));
```

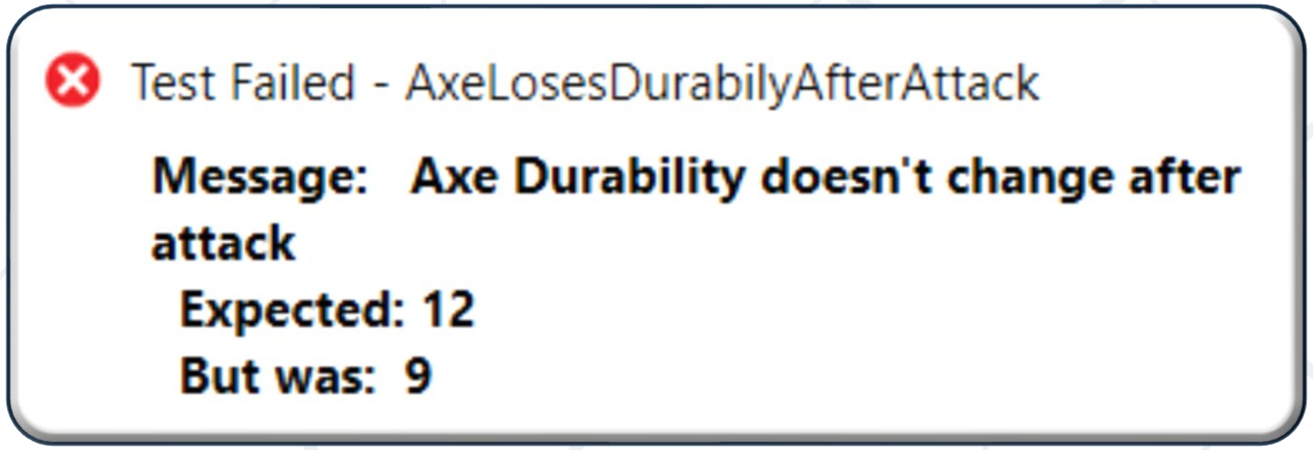
- File / directory assertions

```
Assert.That(string filePath, Does.Exist);
DirectoryAssert.Exists(string path);
```

Assertion Messages

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

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



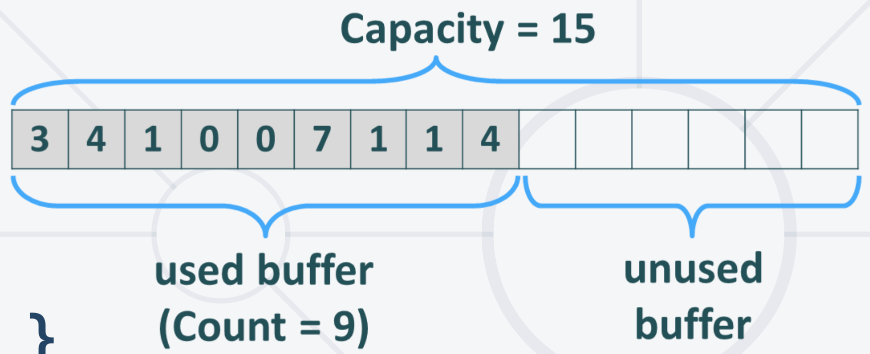
```
Test Failed - AxeLosesDurabilyAfterAttack
  Message:  Axe Durability doesn't change after
  attack
    Expected: 12
    But was:  9
```

Failure messages in the tests help finding the problem

Implementing NUnit Test Cases

Implement Tests for the Collection<T> Class

```
public class Collection<T>
{
  public int Capacity { … }
  public int Count { … }
  public Collection(params T[] items) { … }
  public void Add(T item) { … }
  public void AddRange(params T[] items) { … }
  public T this[int index] { … }
  public void InsertAt(int index, T item) { … }
  public void Exchange(int index1, int index2) { … }
  public T RemoveAt(int index) { … }
  public void Clear() { … }
  public override string ToString() { … }
}
```



Source code: https://github.com/nakov/UnitTestingExample/blob/main/Collections/Collection.cs

Defining the Tests (1)

```
public class CollectionTests
{
  public void Test_Collection_EmptyConstructor() { … }
  public void Test_Collection_ConstructorSingleItem() { … }
  public void Test_Collection_ConstructorMultipleItems() { … }
  public void Test_Collection_Add() { … }
  public void Test_Collection_AddWithGrow() { … }
  public void Test_Collection_AddRange() { … }
  public void Test_Collection_GetByIndex() { … }
  public void Test_Collection_GetByInvalidIndex() { … }
  public void Test_Collection_SetByIndex() { … }
  public void Test_Collection_SetByInvalidIndex() { … }
  …
```

Defining the Tests (2)

```
…
public void Test_Collection_AddRangeWithGrow() { … }
public void Test_Collection_InsertAtStart() { … }
public void Test_Collection_InsertAtEnd() { … }
public void Test_Collection_InsertAtMiddle() { … }
public void Test_Collection_InsertAtWithGrow() { … }
public void Test_Collection_InsertAtInvalidIndex() { … }
public void Test_Collection_ExchangeMiddle() { … }
public void Test_Collection_ExchangeFirstLast() { … }
public void Test_Collection_ExchangeInvalidIndexes() { … }
public void Test_Collection_RemoveAtStart() { … }
public void Test_Collection_RemoveAtEnd() { … }
…
```

Defining the Tests (3)

```
  …
  public void Test_Collection_RemoveAtMiddle() { … }
  public void Test_Collection_RemoveAtInvalidIndex() { … }
  public void Test_Collection_RemoveAll() { … }
  public void Test_Collection_Clear() { … }
  public void Test_Collection_CountAndCapacity() { … }
  public void Test_Collection_ToStringEmpty() { … }
  public void Test_Collection_ToStringSingle() { … }
  public void Test_Collection_ToStringMultiple() { … }
  public void Test_Collection_ToStringNestedCollections() { … }
  public void Test_Collection_1MillionItems() { … }
}
```

Test Cases: Empty Constructor

```
[Test]
public void Test_Collection_EmptyConstructor()
{
  // Arrange
  var nums = new Collection<int>();

  // Assert
  Assert.That(nums.ToString(), Is.EqualTo("[]"));
}
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/3162#0

Test Constructor with Single / Multiple Items

```
[Test]
public void Test_Collection_ConstructorSingleItem()
{
  var nums = new Collection<int>(5);
  Assert.That(nums.ToString(), Is.EqualTo("[5]"));
}
```

```
[Test]
public void Test_Collection_ConstructorMultipleItems()
{
  var nums = new Collection<int>(5, 10, 15);
  Assert.That(nums.ToString(), Is.EqualTo("[5, 10, 15]"));
}
```

Implementing Test Cases: Add

```
public void Test_Collections_Add()
{
  // Arrange
  var nums = new Collection<int>();

  // Act
  nums.Add(5);
  nums.Add(6);

  // Assert
  Assert.That(nums.ToString(), Is.Equal("[5, 6]"));
}
```

Implementing Test Cases: Add Range + Grow

```
[Test]
public void Test_Collection_AddRangeWithGrow()
{
  var nums = new Collection<int>();
  int oldCapacity = nums.Capacity;
  var newNums = Enumerable.Range(1000, 2000).ToArray();

  nums.AddRange(newNums);

  string expectedNums = "[" + string.Join(", ", newNums) + "]";
  Assert.That(nums.ToString(), Is.EqualTo(expectedNums));
  Assert.That(nums.Capacity, Is.GreaterThanOrEqualTo(oldCapacity));
  Assert.That(nums.Capacity, Is.GreaterThanOrEqualTo(nums.Count));
}
```

Test Cases: Get by Index

```
[Test]
public void Test_Collection_GetByIndex()
{
  // Arrange
  var names = new Collection<string>("Peter", "Maria");
  // Act
  var item0 = names[0];
  var item1 = names[1];
  // Assert
  Assert.That(item0, Is.EqualTo("Peter"));
  Assert.That(item1, Is.EqualTo("Maria"));
}
```

Test Cases: Get by Invalid Index

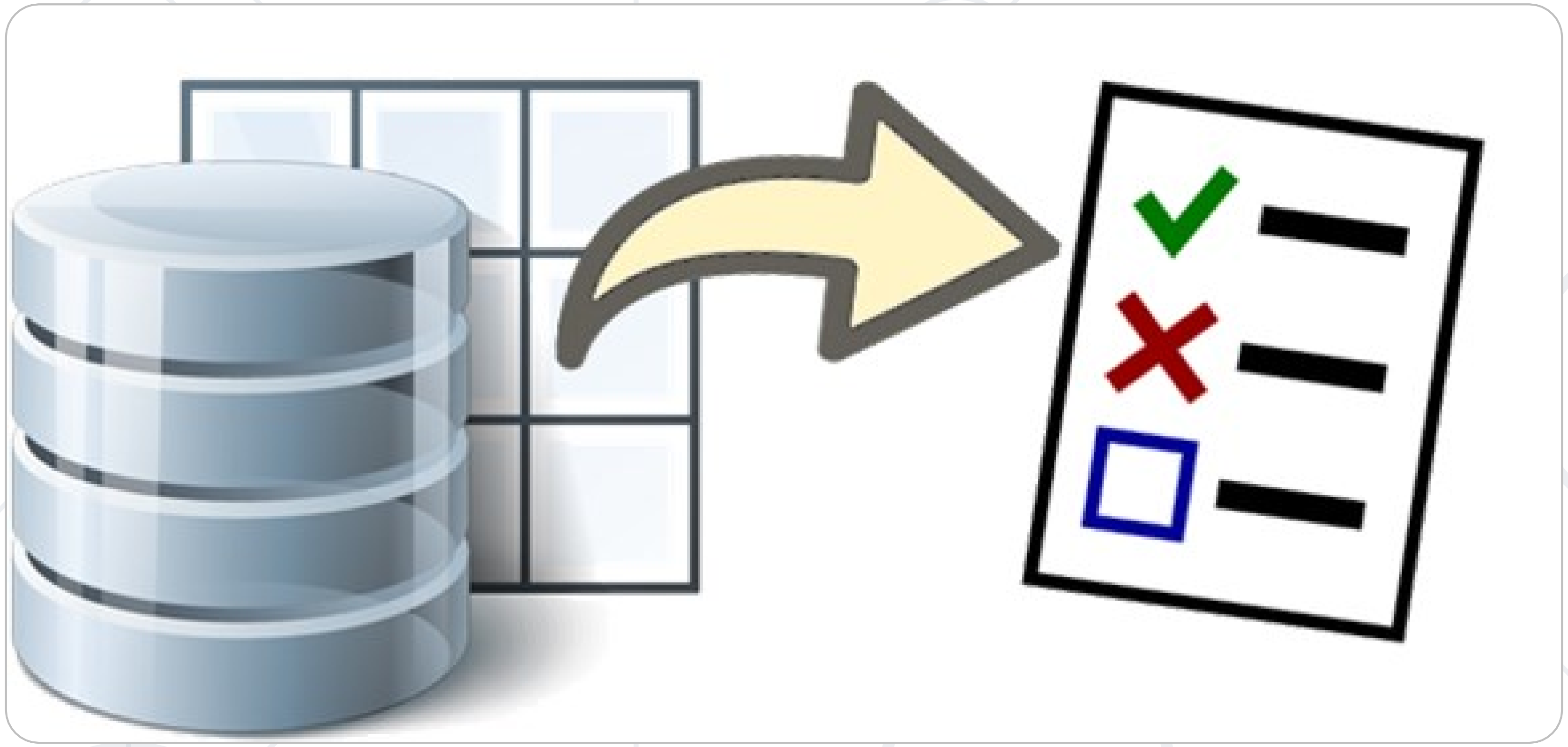
```
[Test]
public void Test_Collection_GetByInvalidIndex()
{
  var names = new Collection<string>("Bob", "Joe");
  Assert.That(() => { var name = names[-1]; },
    Throws.InstanceOf<ArgumentOutOfRangeException>());
  Assert.That(() => { var name = names[2]; },
    Throws.InstanceOf<ArgumentOutOfRangeException>());
  Assert.That(() => { var name = names[500]; },
    Throws.InstanceOf<ArgumentOutOfRangeException>());
  Assert.That(names.ToString(), Is.EqualTo("[Bob, Joe]"));
}
```

Test Cases: ToString() for Nested Collections

```
[Test]
public void Test_Collection_ToStringNestedCollections()
{
  var names = new Collection<string>("Teddy", "Gerry");
  var nums = new Collection<int>(10, 20);
  var dates = new Collection<DateTime>();

  var nested = new Collection<object>(names, nums, dates);
  string nestedToString = nested.ToString();

  Assert.That(nestedToString,
    Is.EqualTo("[[Teddy, Gerry], [10, 20], []]"));
}
```

Performance Test with 1 Million Items

```
[Test]
[Timeout(1000)]
public void Test_Collection_1MillionItems()
{
  const int itemsCount = 1000000;
  var nums = new Collection<int>();
  nums.AddRange(Enumerable.Range(1, itemsCount).ToArray());
  Assert.That(nums.Count == itemsCount);
  Assert.That(nums.Capacity >= nums.Count);
  for (int i = itemsCount - 1; i >= 0; i--)
    nums.RemoveAt(i);
  Assert.That(nums.ToString() == "[]");
  Assert.That(nums.Capacity >= nums.Count);
}
```

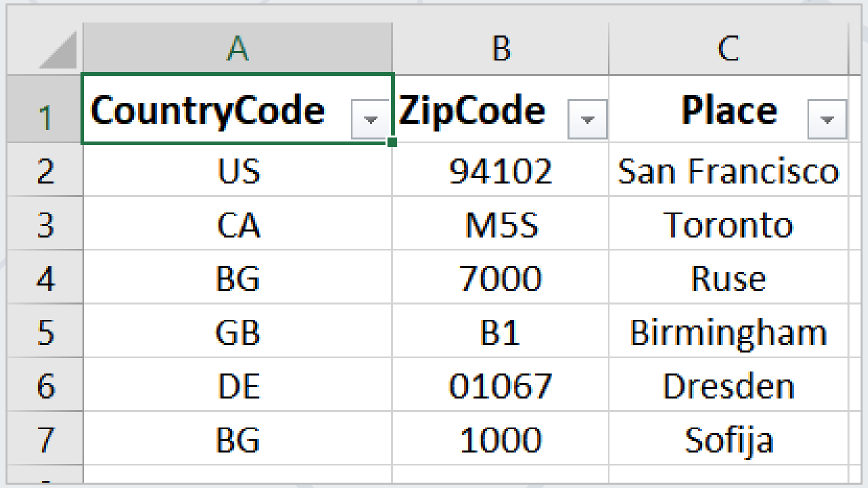


Data-Driven Testing

Data-Driven Testing

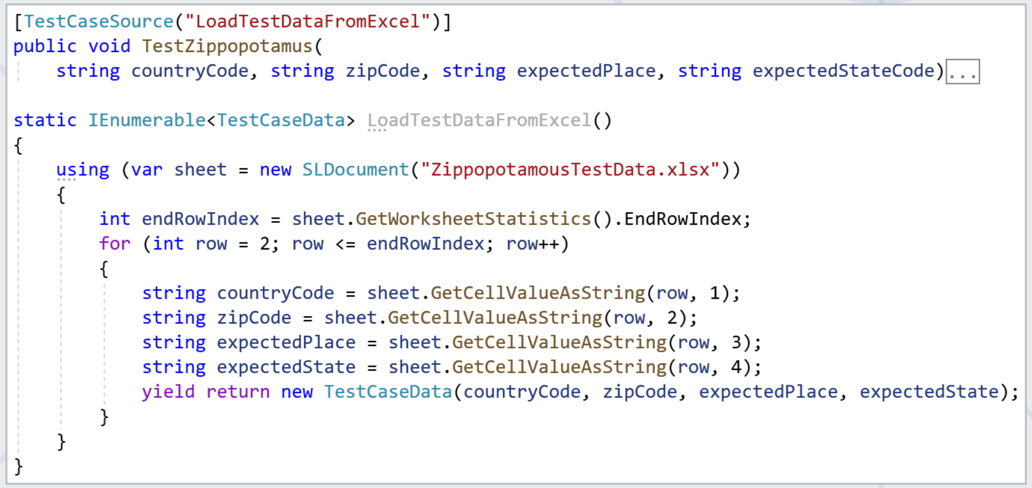
- **Data-driven testing** == running the same test case with multiple data (e. g. datasets in the C# code / Excel spreadsheet)

Data Set



	A	B	C
1	CountryCode	ZipCode	Place
2	US	94102	San Francisco
3	CA	M5S	Toronto
4	BG	7000	Ruse
5	GB	B1	Birmingham
6	DE	01067	Dresden
7	BG	1000	Sofija

Testing Scripts



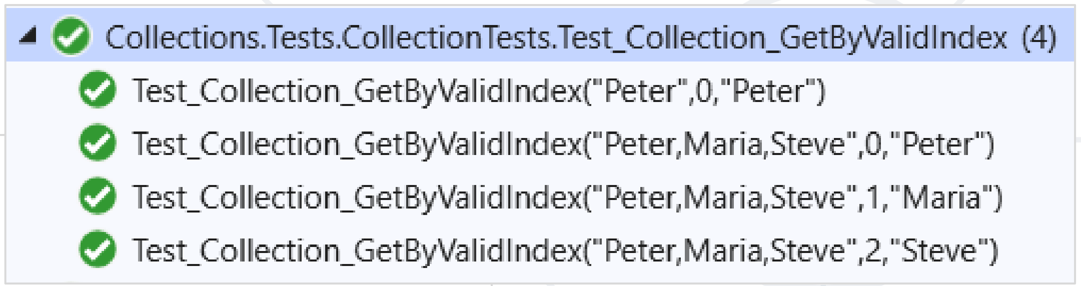
```
[TestCaseSource("LoadTestDataFromExcel")]
public void TestZippopotamus(
    string countryCode, string zipCode, string expectedPlace, string expectedStateCode)...

static IEnumerable<TestCaseData> LoadTestDataFromExcel()
{
    using (var sheet = new SLDocument("ZippopotamousTestData.xlsx"))
    {
        int endRowIndex = sheet.GetWorksheetStatistics().EndRowIndex;
        for (int row = 2; row <= endRowIndex; row++)
        {
            string countryCode = sheet.GetCellValueAsString(row, 1);
            string zipCode = sheet.GetCellValueAsString(row, 2);
            string expectedPlace = sheet.GetCellValueAsString(row, 3);
            string expectedState = sheet.GetCellValueAsString(row, 4);
            yield return new TestCaseData(countryCode, zipCode, expectedPlace, expectedState);
        }
    }
}
```

Data-driven testing framework: code and data stored separately

Data-Driven Testing with NUnit (1)

- The **[TestCase]** attribute in NUnit assigns **multiple datasets** in test method parameters

```
[TestCase("Peter", 0, "Peter")]
[TestCase("Peter,Maria,Steve", 0, "Peter"
[TestCase("Peter,Maria,Steve", 1, "Maria")]
[TestCase("Peter,Maria,Steve", 2, "Steve")]
public void Test_Collection_GetByValidIndex(
    string data, int index, string expectedValue)
{
    var items = new Collection<string>(data.Split(","));
    var item = items[index];
    Assert.That(item, Is.EqualTo(expectedValue));
}
```

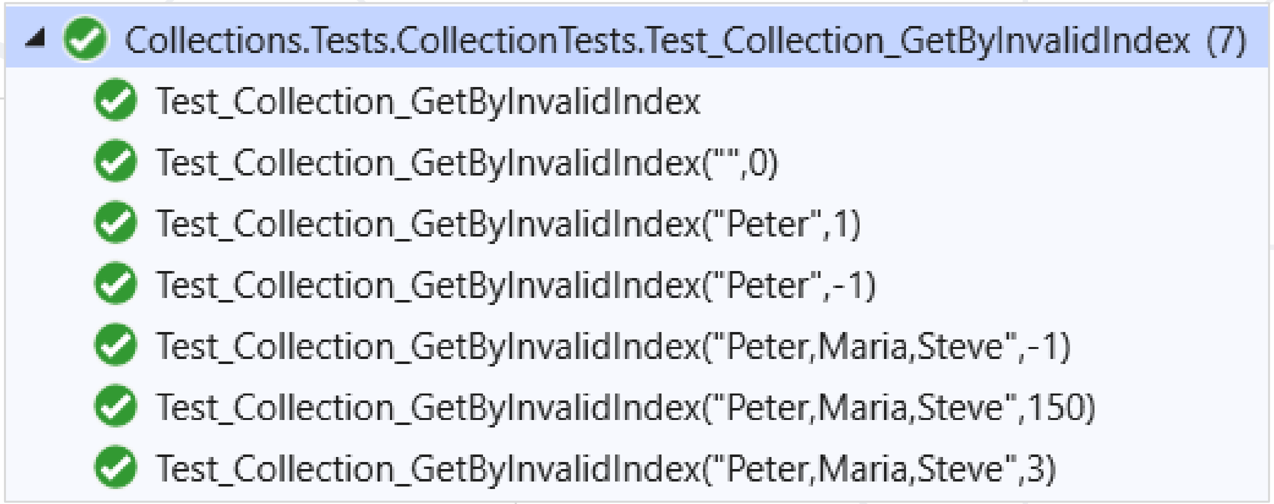


```
Collections.Tests.CollectionTests.Test_Collection_GetByValidIndex (4)
    Test_Collection_GetByValidIndex("Peter",0,"Peter")
    Test_Collection_GetByValidIndex("Peter,Maria,Steve",0,"Peter")
    Test_Collection_GetByValidIndex("Peter,Maria,Steve",1,"Maria")
    Test_Collection_GetByValidIndex("Peter,Maria,Steve",2,"Steve")
```

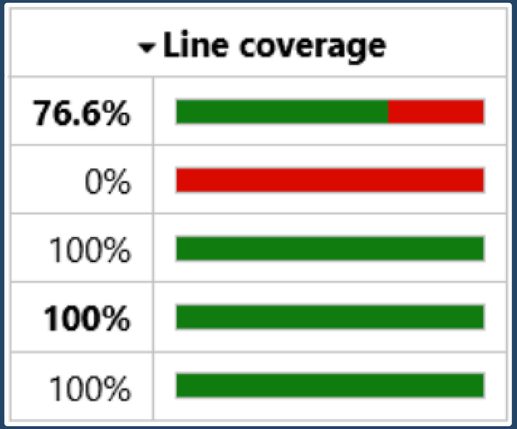
Data-Driven Testing with NUnit (2)

- Another example

```
[TestCase("", 0)]
[TestCase("Peter", -1)]
[TestCase("Peter", 1)]
[TestCase("Peter,Maria,Steve", -1)]
[TestCase("Peter,Maria,Steve", 3)]
[TestCase("Peter,Maria,Steve", 150)]
public void Test_Collection_GetByInvalidIndex(
    string data, int index)
{
    var items = new Collection<string>(data.Split(",",
        StringSplitOptions.RemoveEmptyEntries));
    Assert.That(() => items[index],
        Throws.TypeOf<ArgumentOutOfRangeException>());
}
```



```
◢ ✓ Collections.Tests.CollectionTests.Test_Collection_GetByInvalidIndex (7)
    ✓ Test_Collection_GetByInvalidIndex
    ✓ Test_Collection_GetByInvalidIndex("",0)
    ✓ Test_Collection_GetByInvalidIndex("Peter",1)
    ✓ Test_Collection_GetByInvalidIndex("Peter",-1)
    ✓ Test_Collection_GetByInvalidIndex("Peter,Maria,Steve",-1)
    ✓ Test_Collection_GetByInvalidIndex("Peter,Maria,Steve",150)
    ✓ Test_Collection_GetByInvalidIndex("Peter,Maria,Steve",3)
```

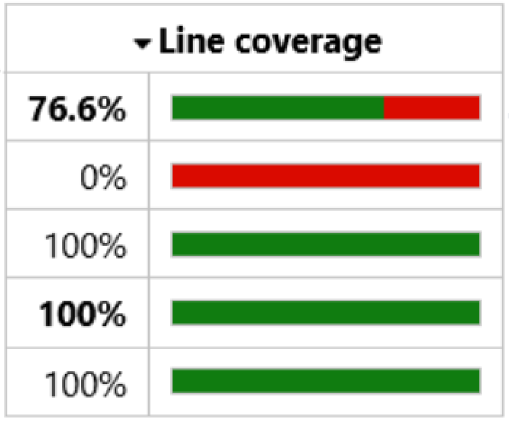


Line coverage	
76.6%	🟩🟥
0%	🟥
100%	🟩
100%	🟩
100%	🟩

Checking the Lines Covered by the Tests

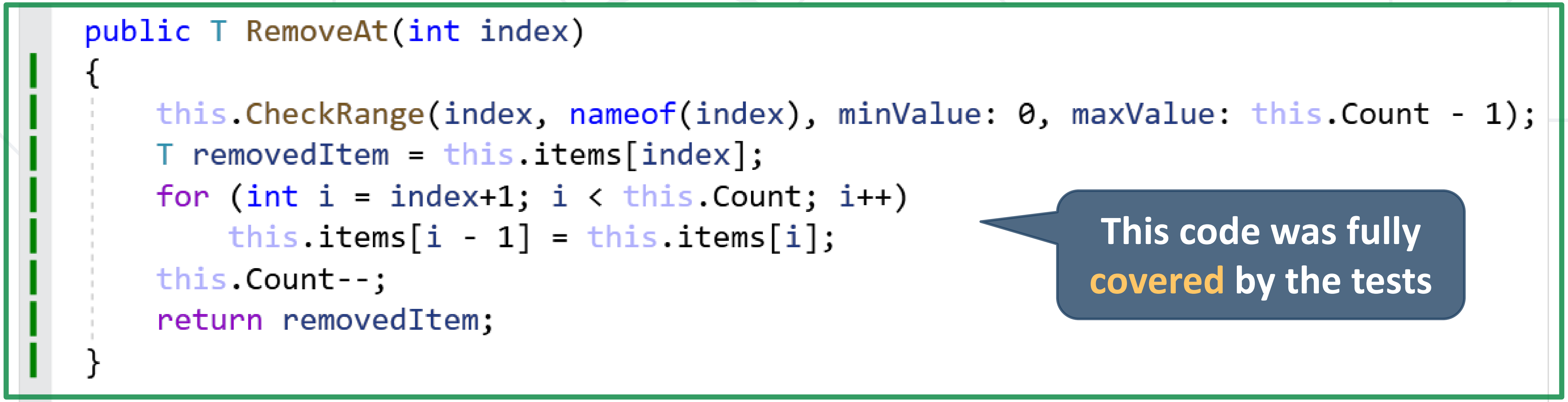
Code Coverage

- **Code coverage** tools measure how many lines of code (LOC) are **covered** by the test execution
  - Lines executed at least once are colored in **green**
  - Lines never executed (untested lines) are **red**
  - Partially executed lines are **orange**
- The **code coverage for the automated tests** is an important **metric** in software engineering
  - Code coverage of **70 - 80%** is a reasonable goal for most projects



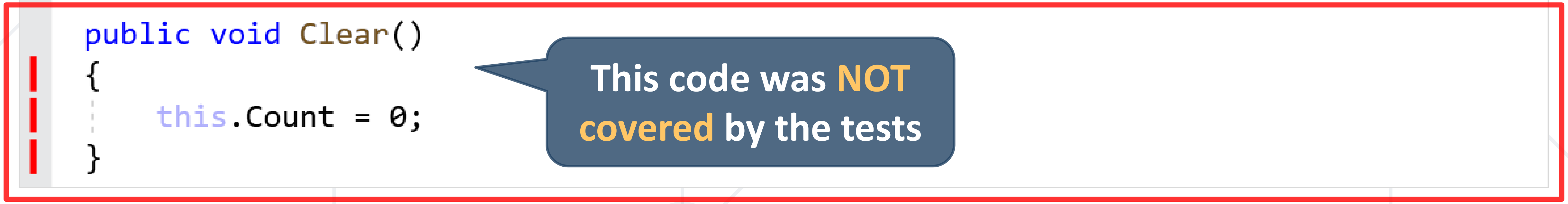
▾ Line coverage	
76.6%	🟩🟩🟥
0%	🟥🟥🟥
100%	🟩🟩🟩
100%	🟩🟩🟩
100%	🟩🟩🟩

Code Coverage: Examples (1)



```
public T RemoveAt(int index)
{
    this.CheckRange(index, nameof(index), minValue: 0, maxValue: this.Count - 1);
    T removedItem = this.items[index];
    for (int i = index+1; i < this.Count; i++)
        this.items[i - 1] = this.items[i];
    this.Count--;
    return removedItem;
}
```

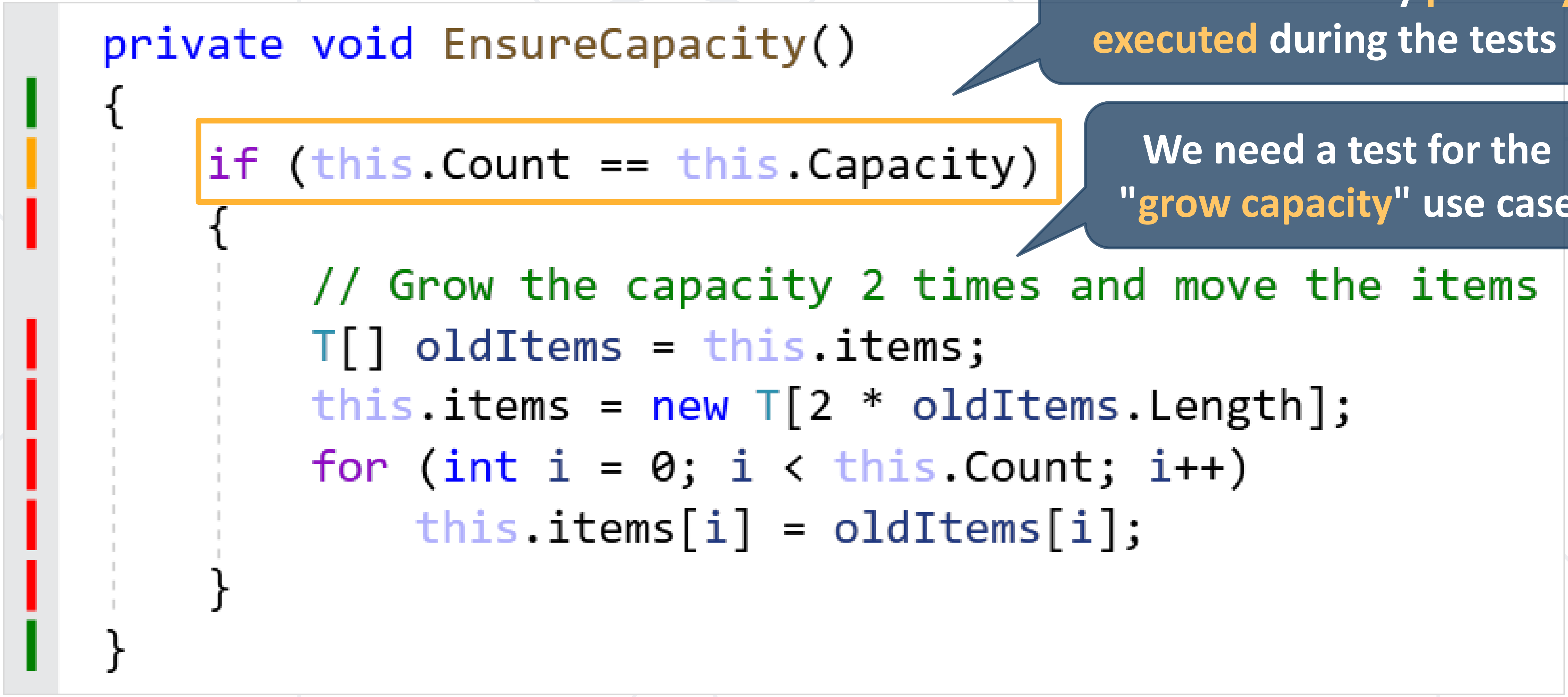
This code was fully **covered** by the tests



```
public void Clear()
{
    this.Count = 0;
}
```

This code was **NOT covered** by the tests

Code Coverage: Examples (2)



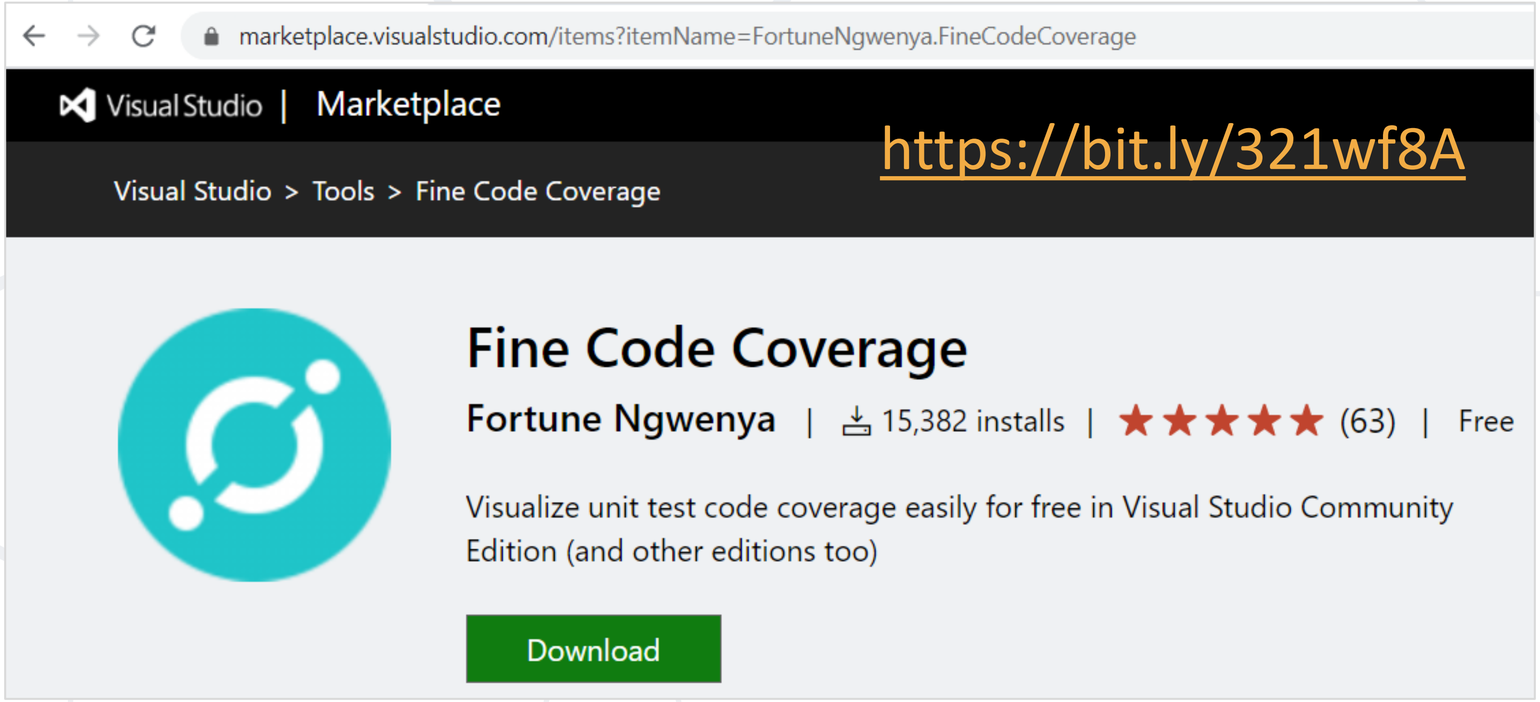
```
private void EnsureCapacity()
{
    if (this.Count == this.Capacity)
    {
        // Grow the capacity 2 times and move the items
        T[] oldItems = this.items;
        this.items = new T[2 * oldItems.Length];
        for (int i = 0; i < this.Count; i++)
            this.items[i] = oldItems[i];
    }
}
```

This code was only **partially executed** during the tests

We need a test for the "**grow capacity**" use case

Code Coverage Tools for C#

- **Visual Studio** supports **code coverage** for C# only in the **Enterprise edition** (paid product)
- Alternative: the **Fine Code Coverage** free extension for VS



marketplace.visualstudio.com/items?itemName=FortuneNgwenya.FineCodeCoverage

Visual Studio | Marketplace

Visual Studio > Tools > Fine Code Coverage

https://bit.ly/321wf8A

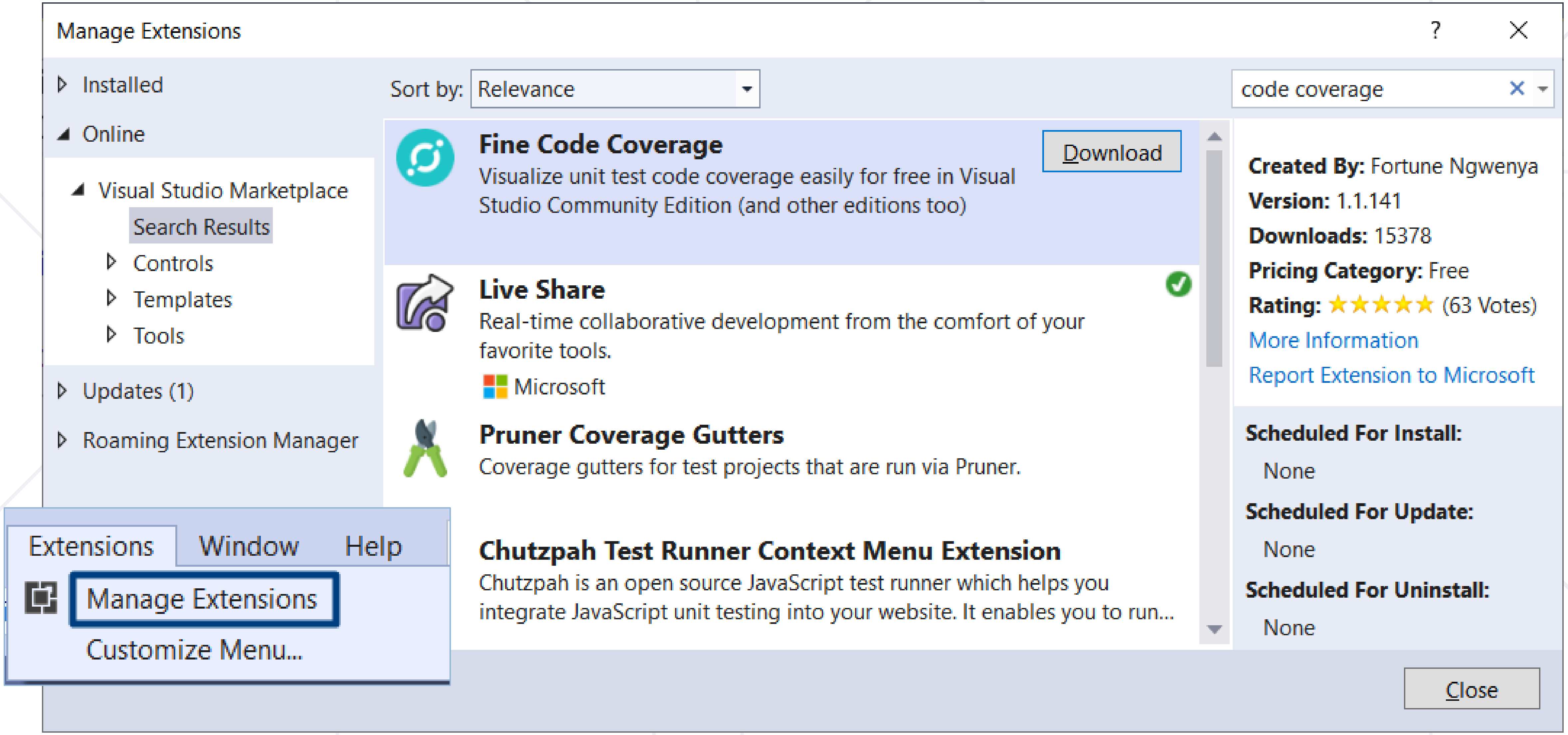
Fine Code Coverage

Fortune Ngwenya | 15,382 installs | ★★★★★ (63) | Free

Visualize unit test code coverage easily for free in Visual Studio Community Edition (and other editions too)

Download

Installing "Fine Code Coverage" in VS



Fine Code Coverage
Visualize unit test code coverage easily for free in Visual Studio Community Edition (and other editions too)

Live Share
Real-time collaborative development from the comfort of your favorite tools.
Microsoft

Pruner Coverage Gutters
Coverage gutters for test projects that are run via Pruner.

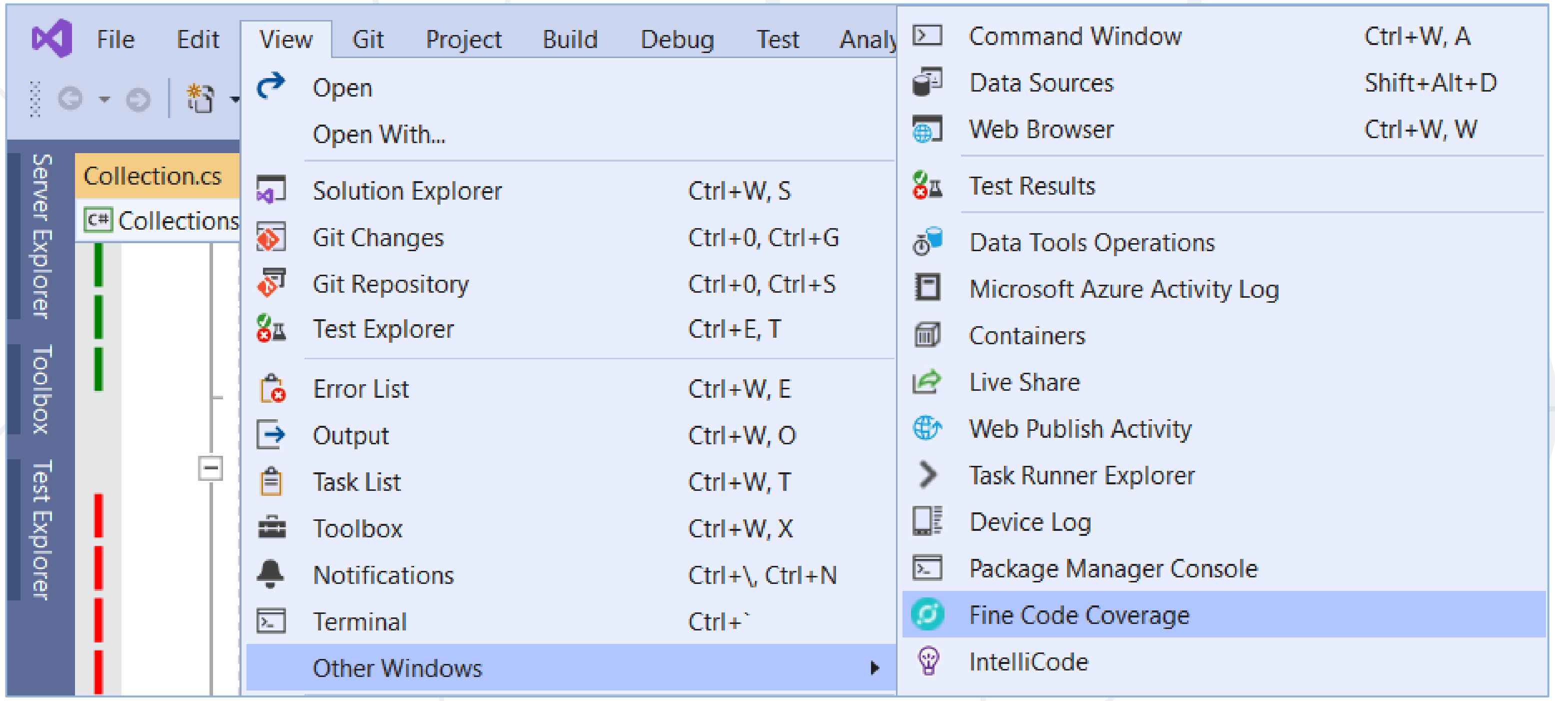
Chutzpah Test Runner Context Menu Extension
Chutzpah is an open source JavaScript test runner which helps you integrate JavaScript unit testing into your website. It enables you to run...

Created By: Fortune Ngwenya
Version: 1.1.141
Downloads: 15378
Pricing Category: Free
Rating: ★★★★★ (63 Votes)
More Information
Report Extension to Microsoft

Scheduled For Install:
None
Scheduled For Update:
None
Scheduled For Uninstall:
None

Activating "Fine Code Coverage" in VS

- Run the unit tests to view **the covered lines** in **green** / **red** / **orange** (be patient, the coloring comes after a while)



- View your code coverage **report** in the [Fine Code Coverage] window

Naming, Repeatable, No Dependencies

Naming the Test Methods

- **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() {}
```

✓

Automated Tests: Good Practices (1)

- 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

Automated Tests: Good Practices (2)

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

```
[Test]
public void Te
{
    var names
    var remove
    Assert.Tha
    var remove
    Assert.Tha
    var removedL
    Assert.That(
    Assert.That(
}
```

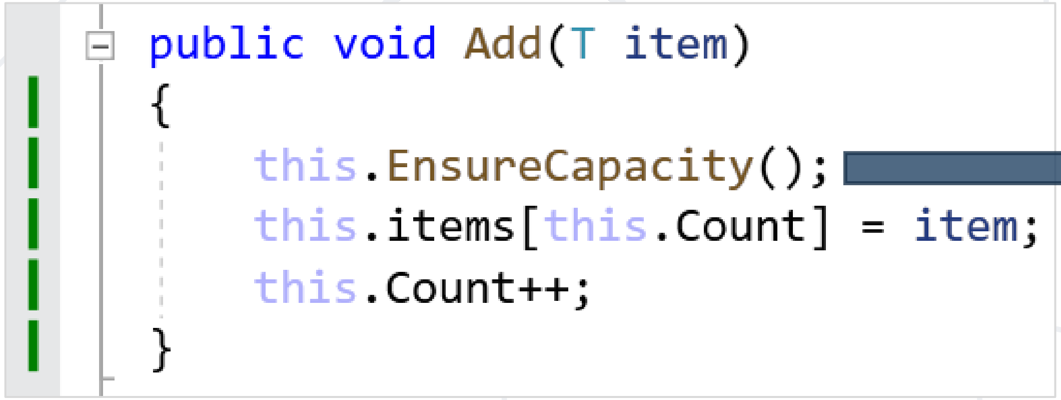
```
[Test]
public void Test_Collections_RemoveAtStart()
{
}
```

```
[Test]
public void Test_Collections_RemoveAtEnd()
{
}
```

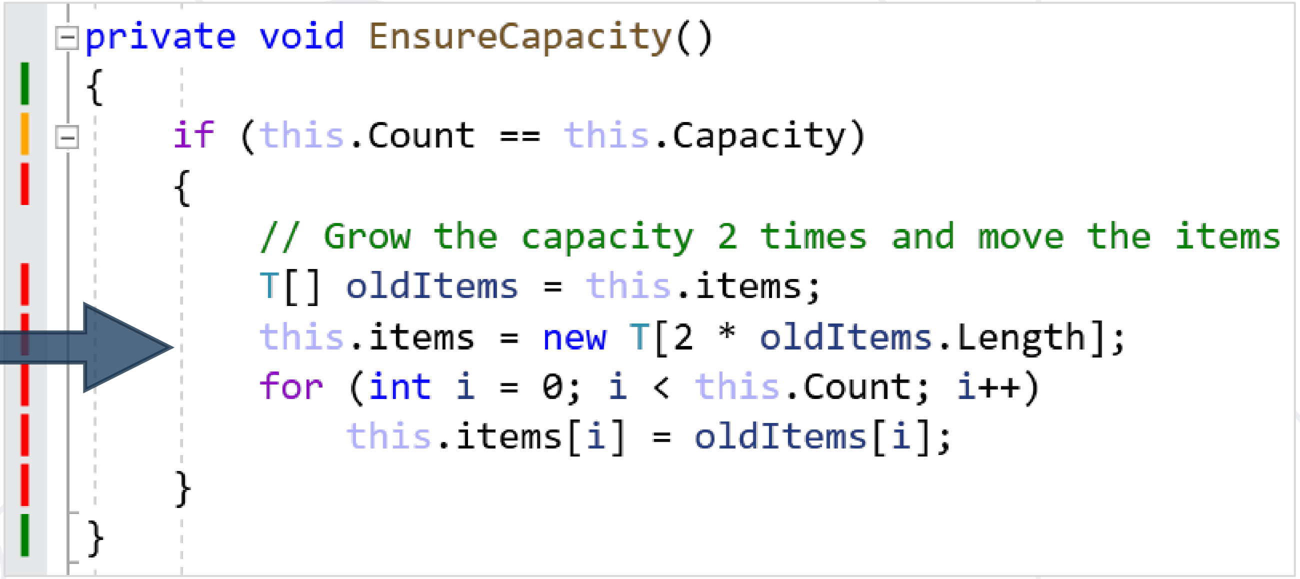
```
[Test]
public void Test_Collections_RemoveAtMiddle()
{
    var names = new Collection<string>("Peter", "Maria", "Steve", "Mia");
    var removed = names.RemoveAt(1);
    Assert.That(removed, Is.EqualTo("Maria"));
    Assert.That(names.ToString(), Is.EqualTo("[Peter, Steve, Mia]"));
}
```

Testing Private Methods

- **Private methods** should be tested indirectly
  - By testing the **public methods** with certain inputs and entrance conditions, that will invoke the target private methods
  - Check the **code coverage** to ensure all code is tested!
- Example:



```
public void Add(T item)
{
    this.EnsureCapacity();
    this.items[this.Count] = item;
    this.Count++;
}
```

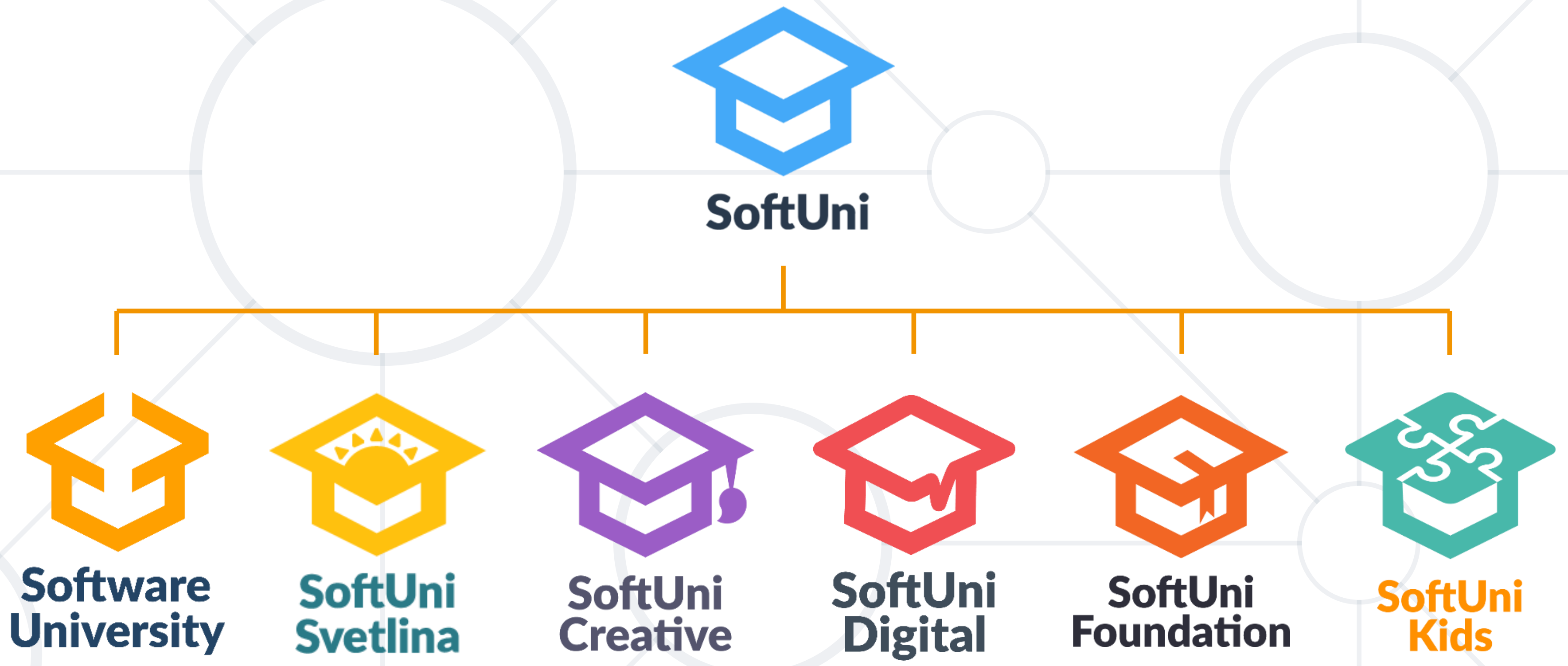


```
private void EnsureCapacity()
{
    if (this.Count == this.Capacity)
    {
        // Grow the capacity 2 times and move the items
        T[] oldItems = this.items;
        this.items = new T[2 * oldItems.Length];
        for (int i = 0; i < this.Count; i++)
            this.items[i] = oldItems[i];
    }
}
```

Summary

- **Unit testing** == automated testing of single component (unit)
- **Testing framework** == foundation for writing tests
- **NUnit** == automated testing framework for C#
- The **AAA pattern**: Arrange, Act, Assert
- **Assertion** == checking results / exit conditions
- **Code coverage** == tracks which LOC are executed

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

License