Introduction to .NET Testing with NUnit

Getting Started

Jason Roberts

@robertsjason | dontcodetired.com

Why Write Automated Tests?

Happier development team

Fewer late nights/weekend work

More time to add new features

Happier users

Fewer defects reaching production causing annoyance

Reduced business cost

Defects found earlier in development lifecycle

Reliability

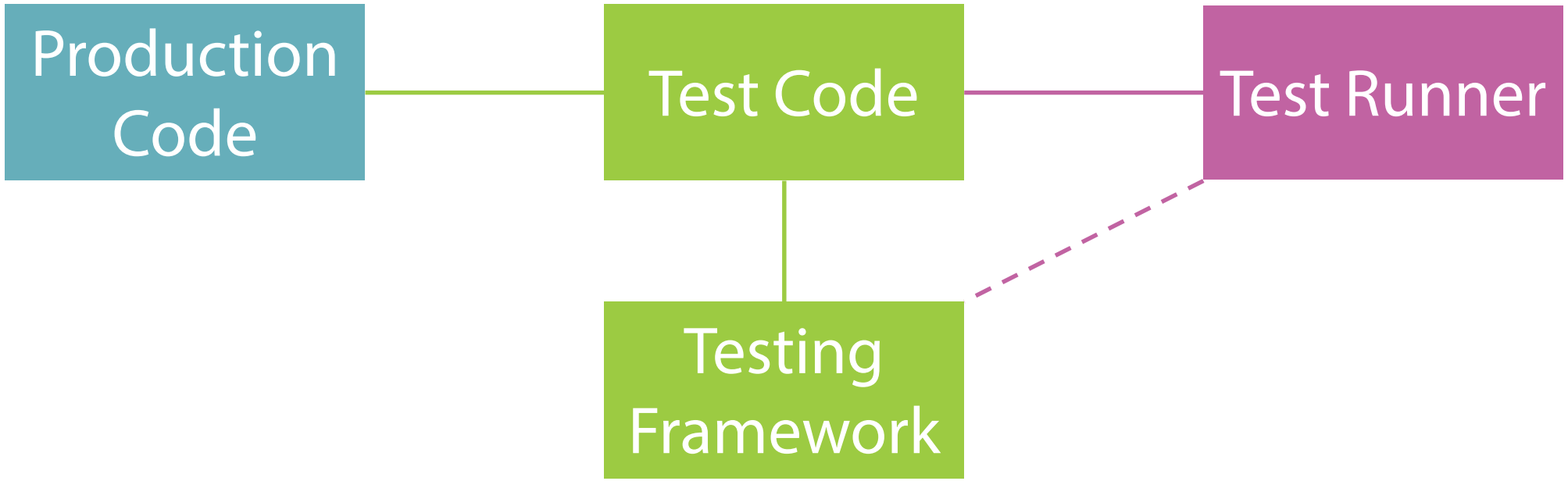
Exactly same test code runs each time

No variance between runs from Human error

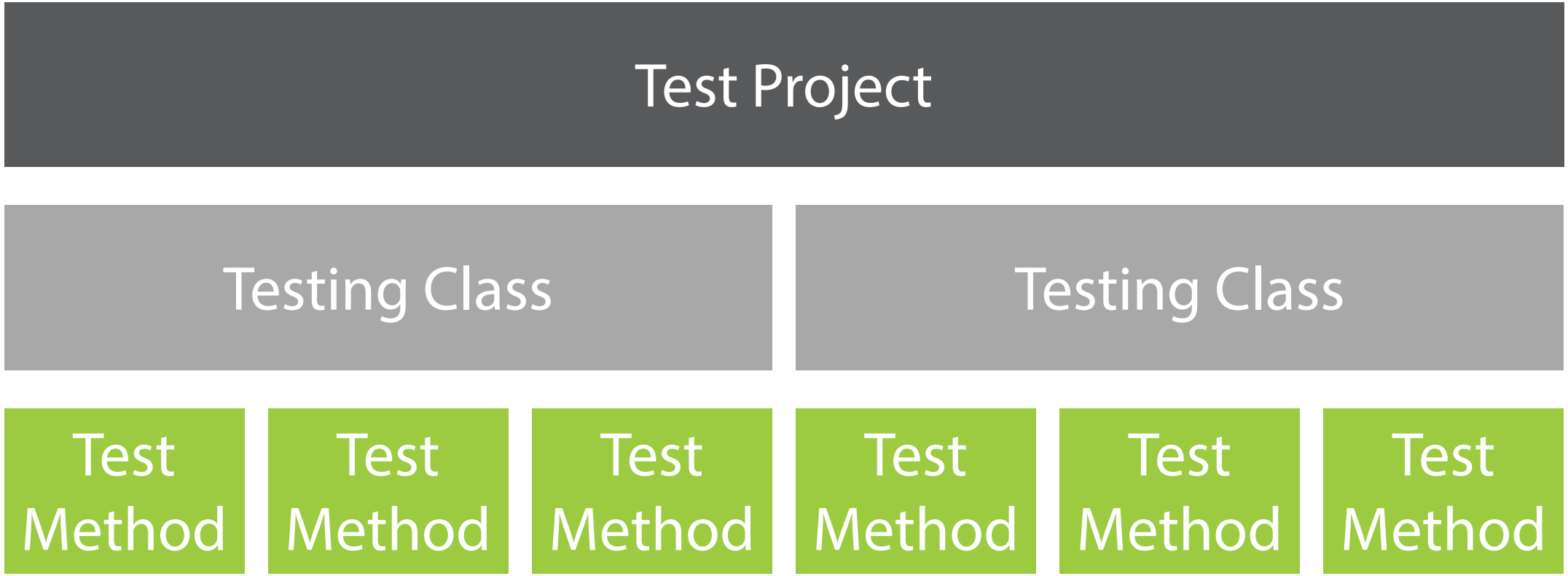
Faster execution

Quicker than a human performing tests manually

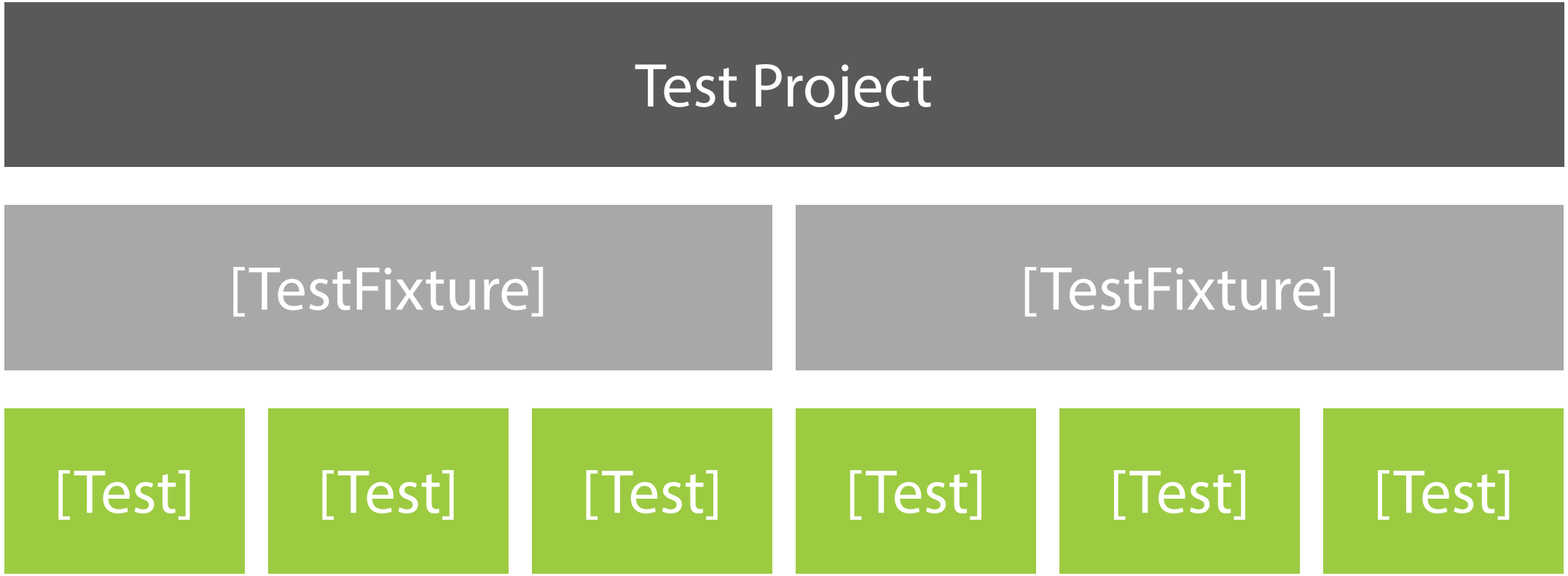
Testing Frameworks and Test Runners



NUnit Test Suite Organizational Structure



Designating Test Code

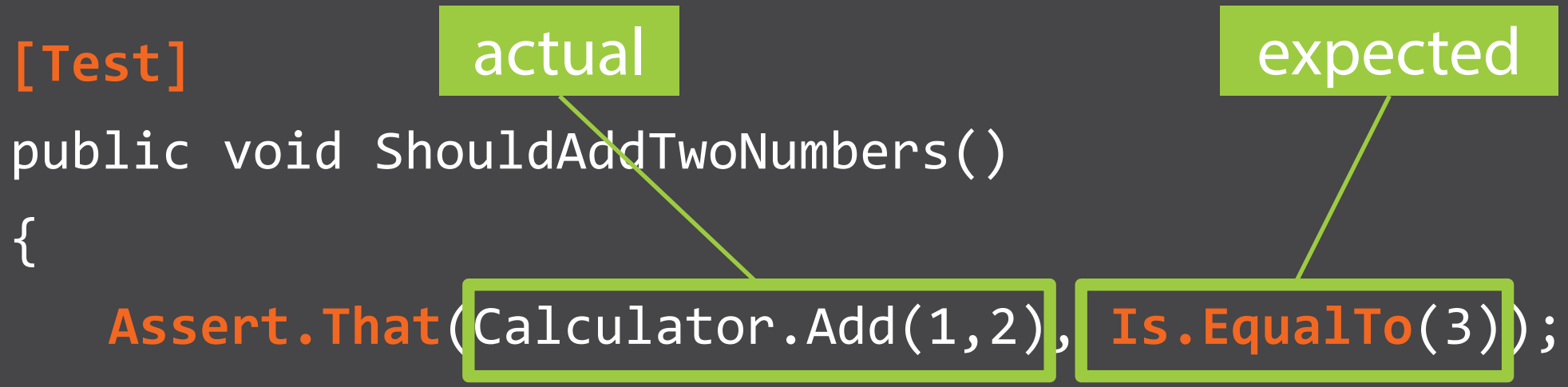


Designating Test Code

```
[TestFixture]
public class CalculatorTests
{
    [Test]
    public void ShouldAddTwoNumbers()
    {
        // Test code omitted
    }
}
```

Asserts tell the test runner whether a test has passed or failed

What are Asserts?



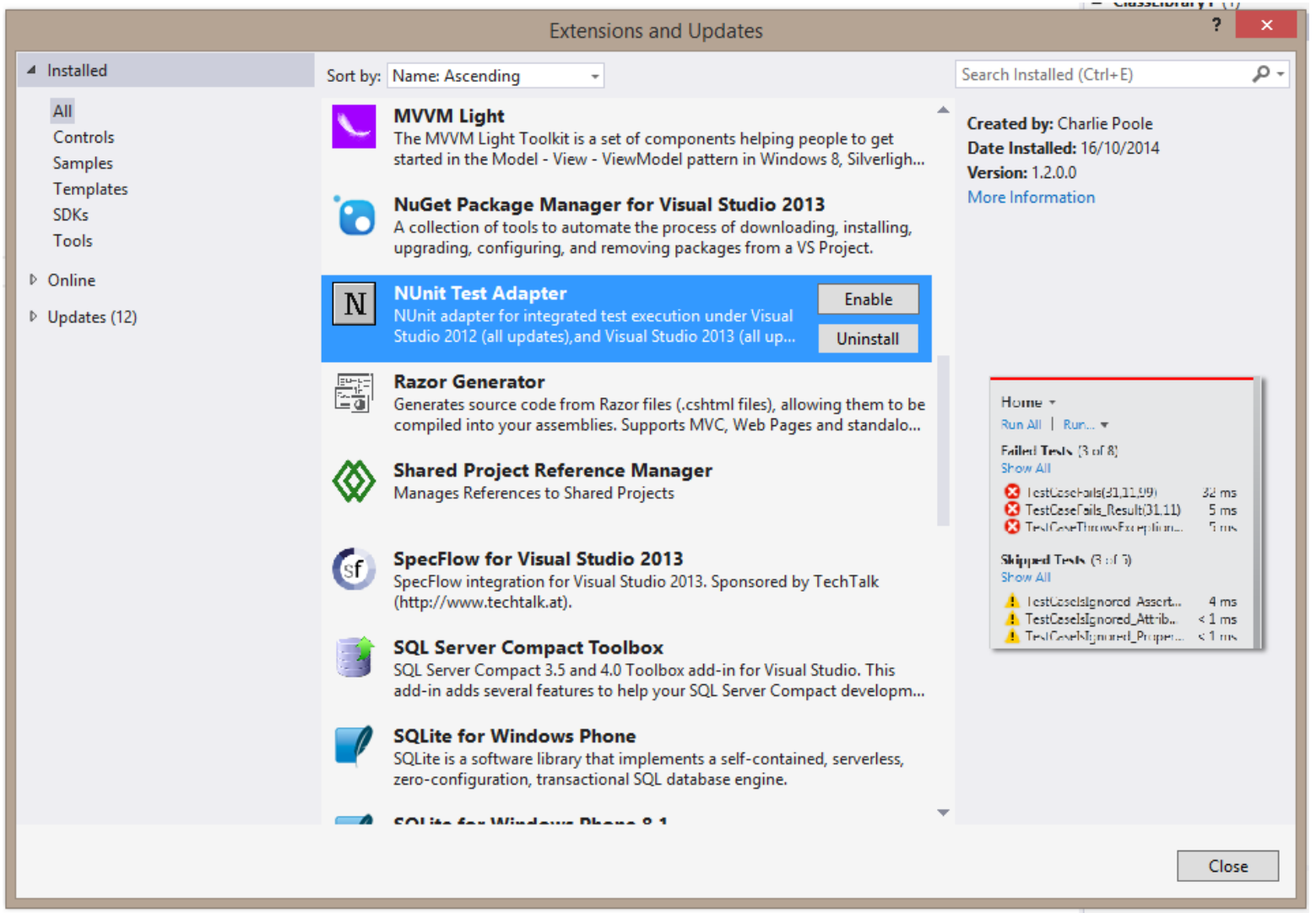
What Makes a Good Test?

- Independent & isolated
- Test single behaviour / logical thing
- Clear intent / readable
- Don't test the compiler
- Reliable & repeatable
- Production quality code
- Valuable

Naming Conventions

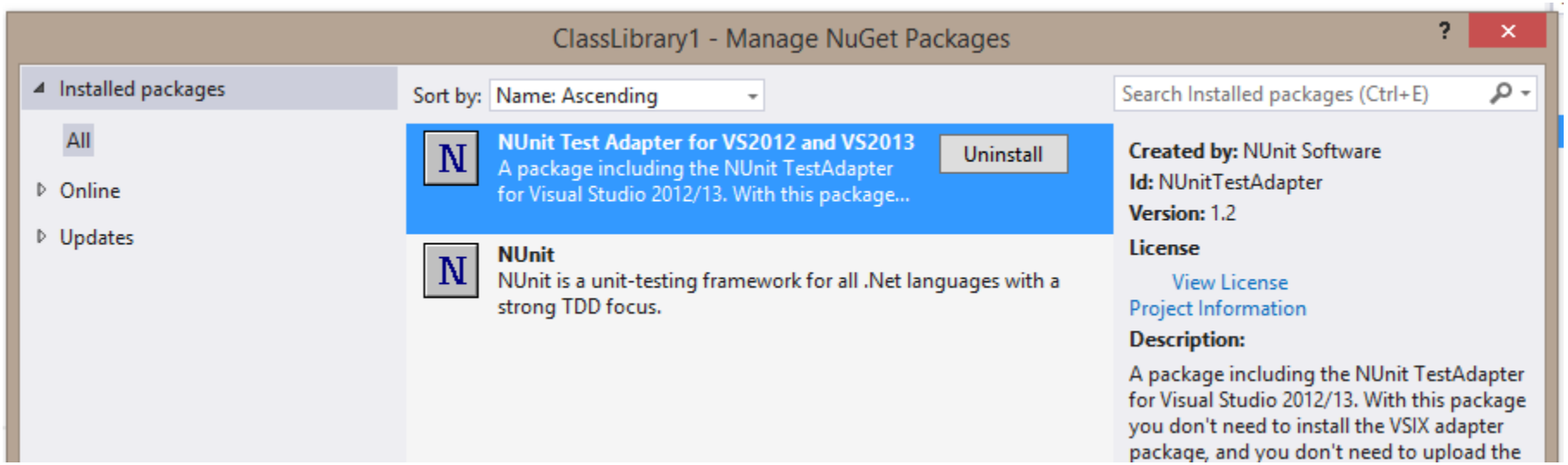
```
[TestFixture]
public class CalculatorTests
{
    [Test]
    public void ShouldAddTwoNumbers()
    {
        var sut = new Calculator();
        // System Under Test
    }
}
```

Enabling NUnit Test Execution in Visual Studio



Tools → Extensions and Updates

Enabling NUnit Test Execution in Visual Studio



Writing an NUnit Test from Scratch

Create new test project

Reference production code project

Install NUnit NuGet package

Write a test class and methods

Refactoring the Test Code

Running Tests in Resharper

Running Tests with NUnit GUI

Download NUnit .zip file

Run NUnit GUI

Browse to test dll

Run tests

Running Tests from the Command Line

Summary

Why write automated tests?

Testing frameworks and runners

[TestFixture] and [Test] attributes

Assert.That(…)

Running tests

Next: Asserts