

Faculty of Mathematics and Physics  
Charles University in Prague  
12<sup>th</sup> May 2016



C# Made Easy!

# Programming II

Workshop 11 – Functional Testing

# Workshop 11

## Outline

1. Test
2. Functional Testing – What, Why, How
3. Homework



# Test 11

## Test

**Find the test here (no-ads):**

<https://goo.gl/4Hhi1J>

0 vs 0, i vs. l vs. 1

**Permanent link:**

[https://docs.google.com/forms/d/1hml4cekxkb1ep6z\\_y67R5nHloA8pZ4\\_IWR7KSMTclOo/viewform](https://docs.google.com/forms/d/1hml4cekxkb1ep6z_y67R5nHloA8pZ4_IWR7KSMTclOo/viewform)

**Time for the test:**

8 min

# Testing

What?

---

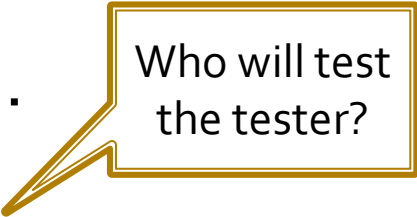
WHAT?

# Testing

## What?

- Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

To put it simply...



Who will test  
the tester?

- Running a code that executes another code and compares results with pre-computed/pre-specified ones.

# Testing

## What?

- Simple example

```
class Calc {  
    int Add(int a1, int a2)  
}
```

```
class CalculatorTest {  
  
    public boolean TestAdd() {  
        Calc c = new Calculator();  
        if (c.Add(1, 1) == 2) return true;  
        return false;  
    }  
  
}
```

# Testing

Why?

---

WHY?

# Testing

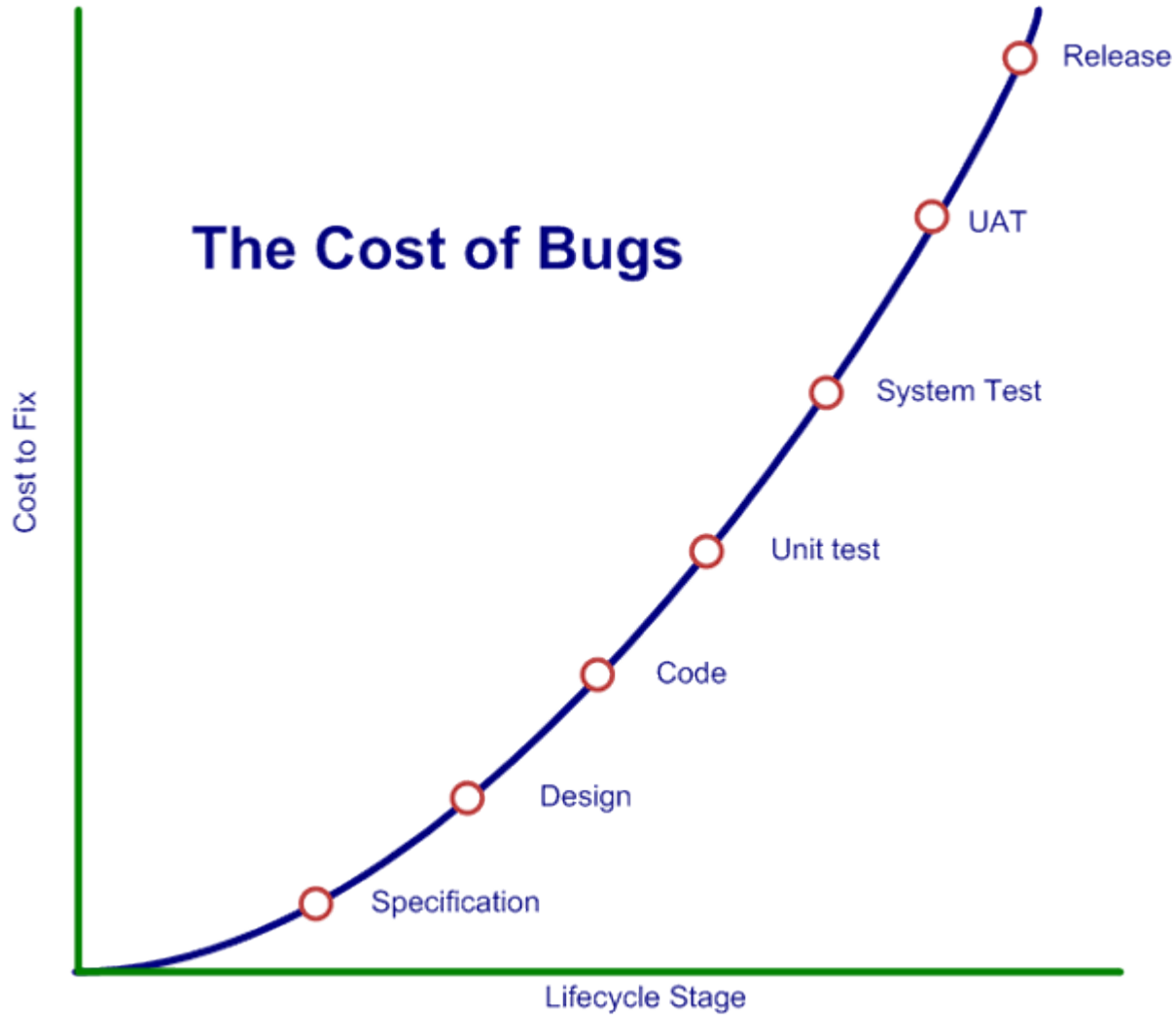
Why?

Because **TIME** (translates as **MONEY**) matters!



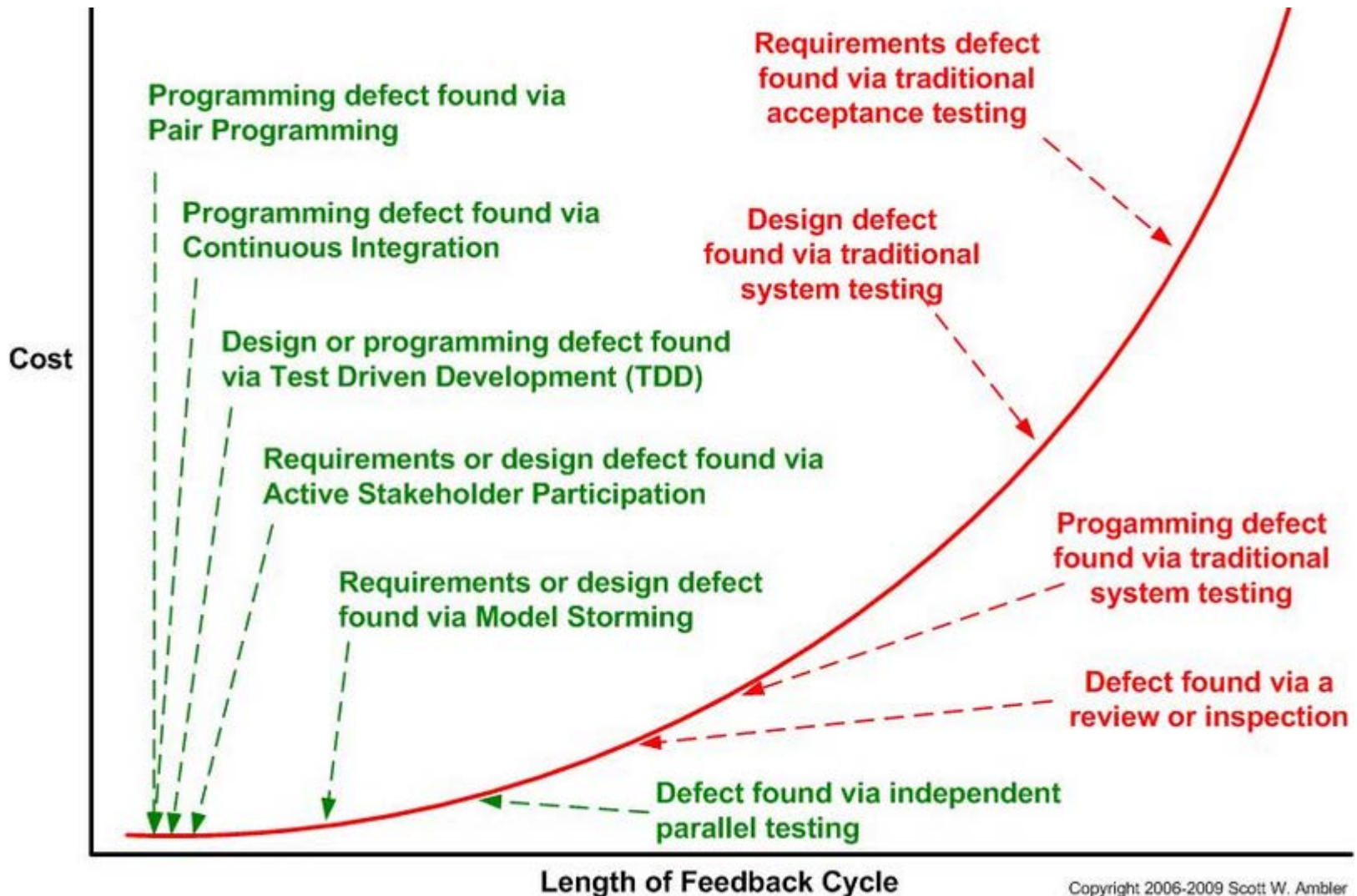
# Testing

Why?



# Testing

## Why?



# Testing

## Why?

- Tests reduce bugs in existing features
- Tests reduce bugs in new features
- Tests (of complex code base) are good documentation
- Tests improve design
- Tests reduce cost of change
- Tests constrain features
- Tests reduce fear of making changes
  - For your colleagues as well as you!

# Testing

Why not to?

WHY NOT TO?

# Testing

## Why not to write tests?

- It takes too much time to write tests
- It takes too much time to execute tests
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

# Testing

## Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

# Testing

## Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually ... and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

# Testing

## Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually ... and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
  - Oh, and you expect to have customers?
- I don't really know how the code should behave so I can't test it!



# Testing

## Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually ... and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
  - Oh, and you expect to have customers?
- I don't really know how the code should behave so I can't test it!
  - You should have not started writing such code from the very beginning!

# Testing

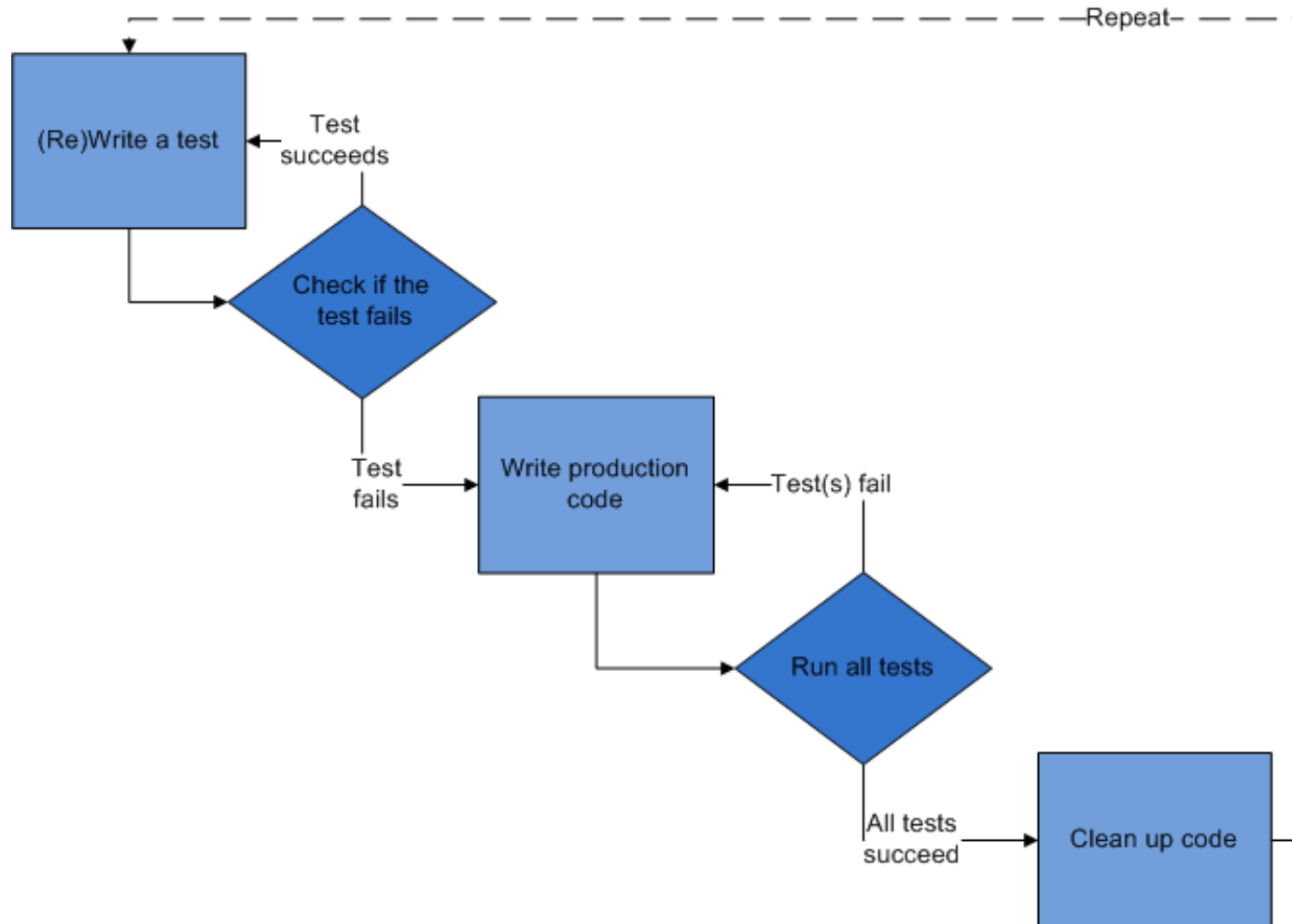
## How?

---

# HOW?

# Testing

## How?



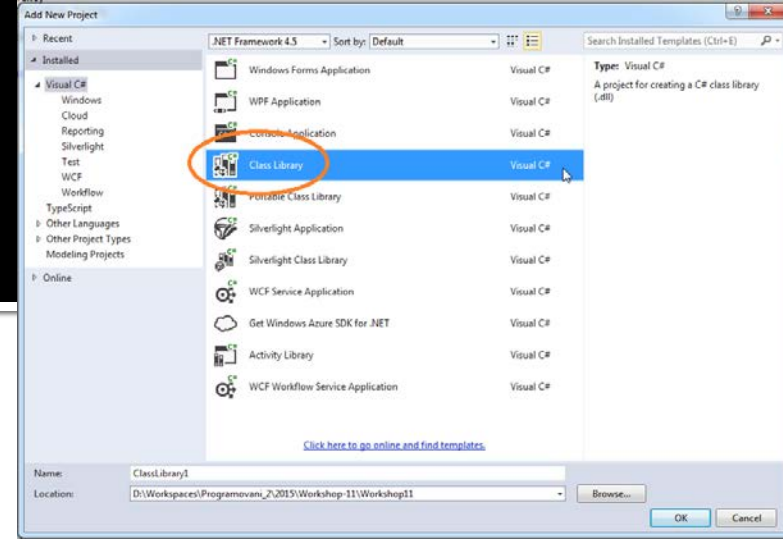
# Testing

How?

Show time...

# Testing

## Creating projects

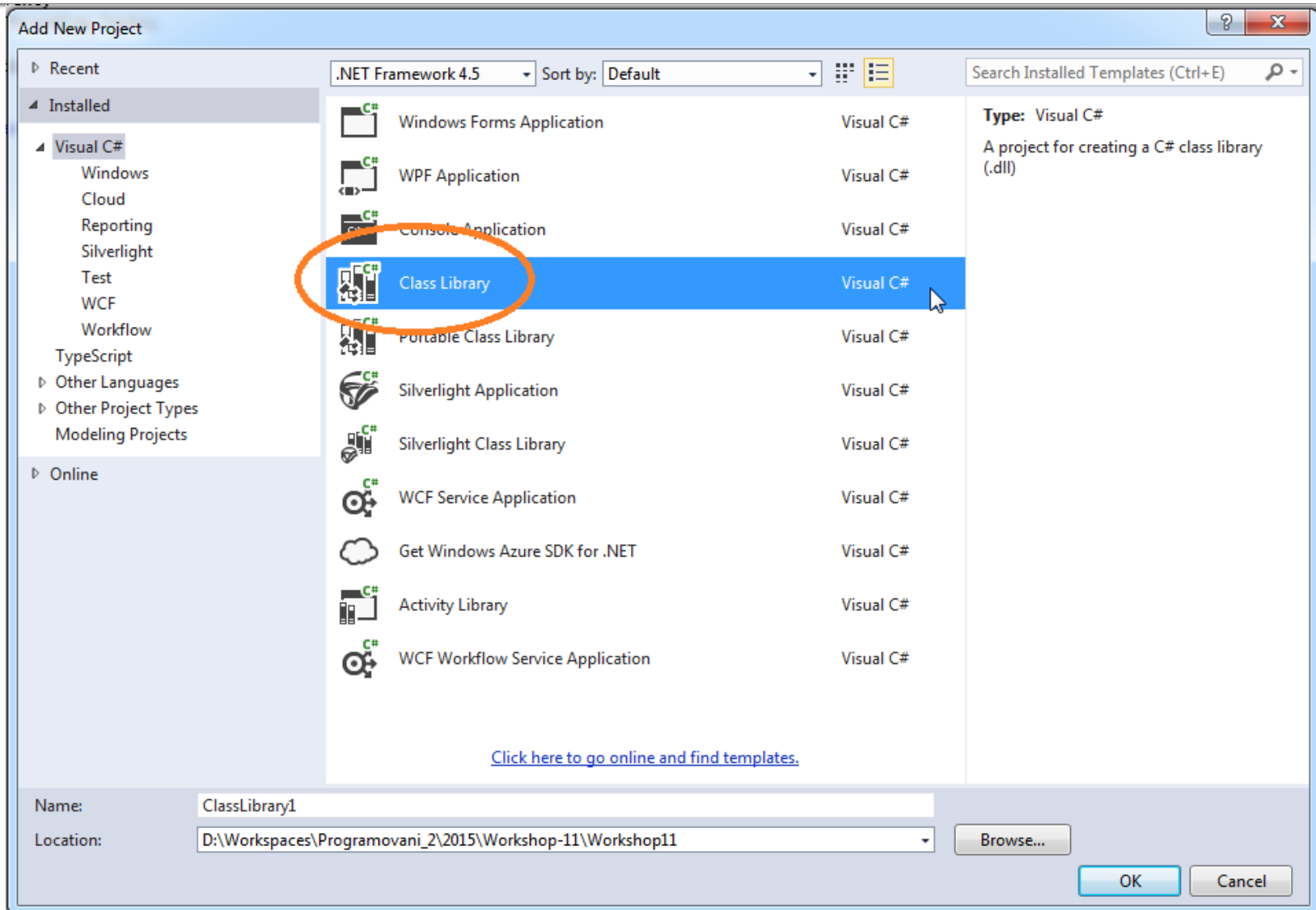


Create two (Class Library) projects

1. First will contain your classes (unit) for testing  
Name it e.g.: MyLibrary
2. Second will contain TESTs that will be executed to test your first project  
Name it e.g.: MyLibrary.Test

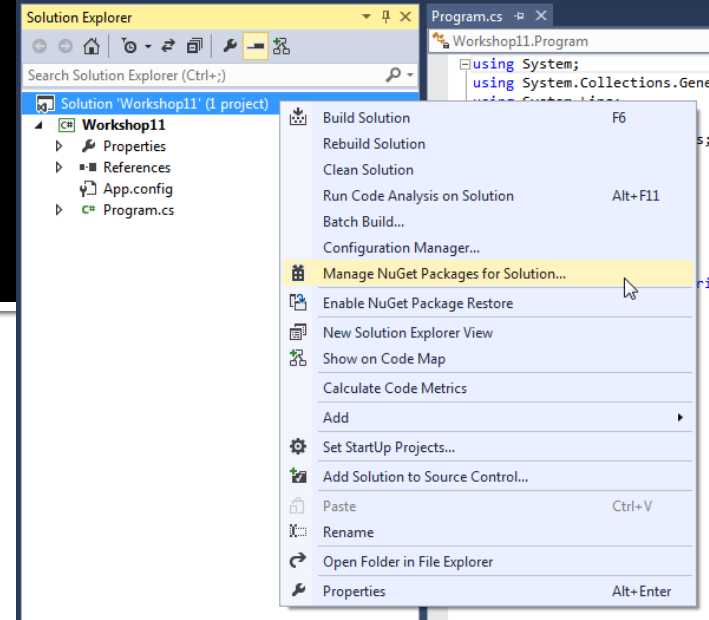
# Testing

## Creating projects



# Testing

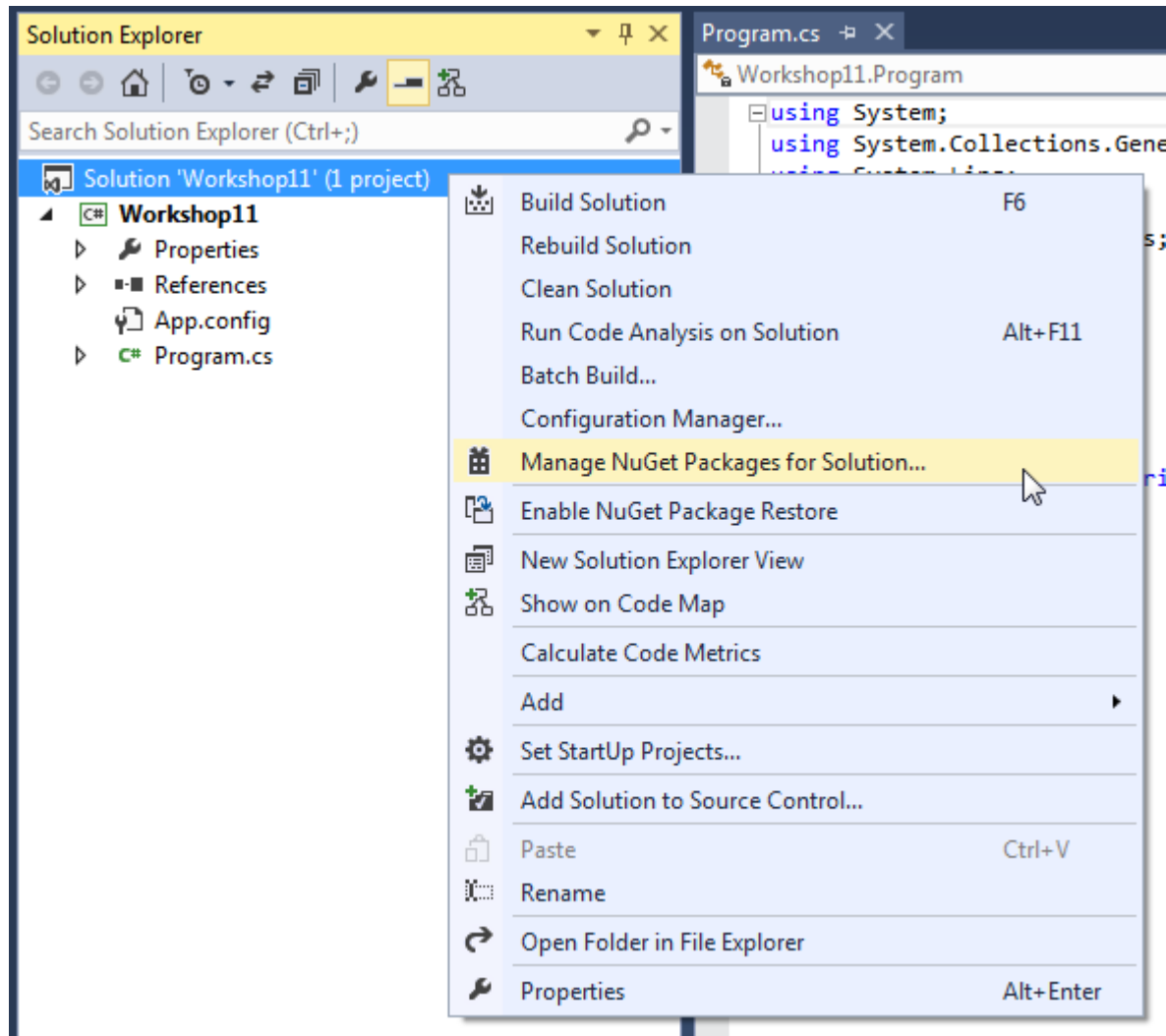
## Creating projects



Import required libraries / packages that you will need to run tests, perform code coverage and generate reports.

# Testing

## Importing required libraries via NuGet





# Testing

## Importing required libraries via NuGet

1. Be sure to manage packages for MyLibraryTest

2. Search for NUnit

NuGet Package Manager: Solution 'MyLibraryTest'

Package source: nuget.org

Filter: All

Include prerelease

nunit



- NUnit**  
NUnit is a unit-testing framework for all .Net languages with a strong TDD focus.
- NUnit.Runners**  
Console runner for version 3.0 of the NUnit unit-testing framework.
- NUnitTestAdapter**  
A package included in the Visual Studio 12/13/15. It provides the adapter to your TFS server.
- SpecFlow.NUnit**  
Combined package to setup SpecFlow with NUnit v2.6+ easily. For running tests with NUnit runners, use SpecFlow.NUnit.Runners package!
- FluentAssertions**  
Fluent methods for asserting the result of TDD/BDD specs for .NET 4.0/4.5 (Desktop and Windows Store), CoreCLR, SL5, WP8, WP8.1 and WPA8.1. Supports the unit test frameworks NUnit, XUnit, XUnit2, MBUnit, G...

5. Do the same for NUnit.Runners

3. Select NUnit

**NUnit**

Action:  Version:

Select which projects to apply changes to:

- 2 project(s)  Show all
- MyLibrary
- MyLibraryTest

**Options**

- Show preview window
- Dependency behavior:
- File conflict action:

[Learn about Options](#)

**Description**

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

This package includes the NUnit 3.0 framework assembly, which is referenced by your tests. You will need to install version 3.0

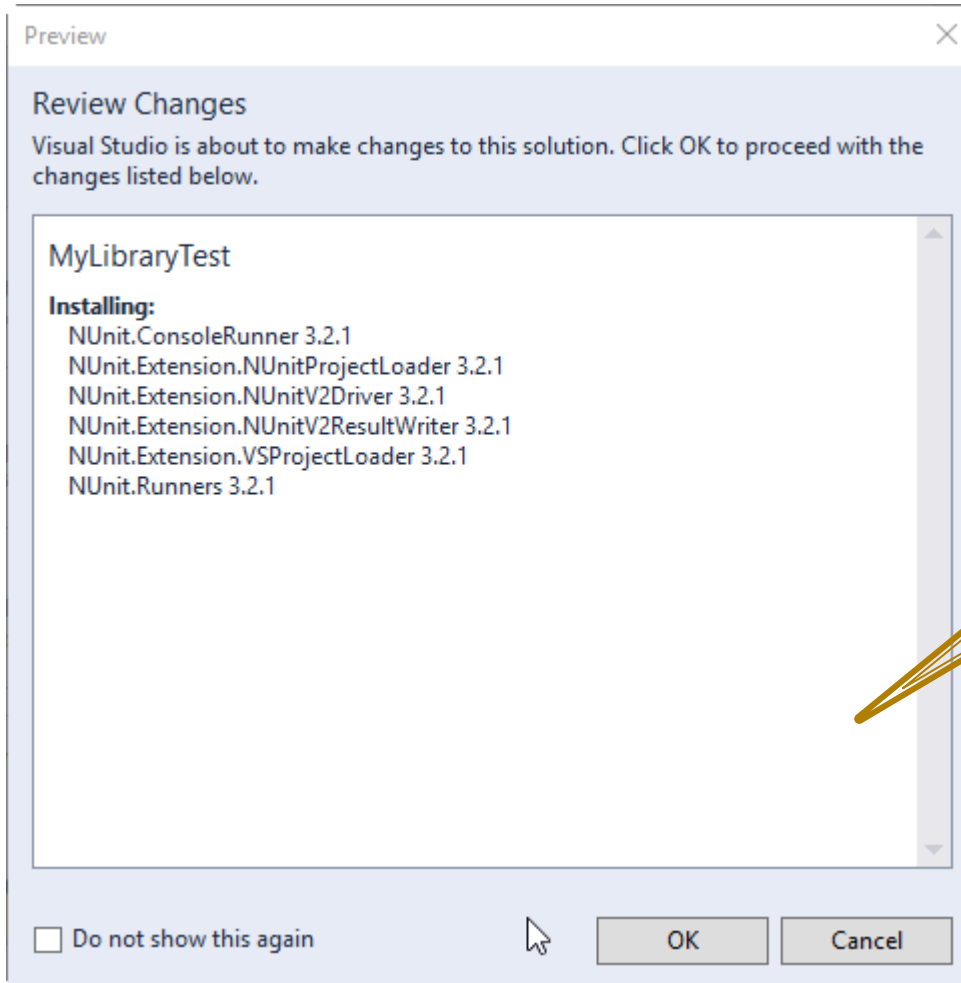
4. Install it!

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

# Testing

## Importing required libraries via NuGet



1. Confirm you want to install it...

# Testing

## Importing required libraries via NuGet

1. Be sure to manage packages for MyLibraryTest

2. Search for OpenCover

3. Select OpenCover

4. Install it!


NuGet Package Manager: HeapLibraryTest


Package source:


Filter:


Include prerelease




 **OpenCover**  
An open source code coverage tool (branch and sequence point) for all .NET Frameworks 2 and above. Also capable of handling 32 and 64 bit processes.

 **OpenCoverToCoberturaConverter**  
Converts OpenCover reports to Cobertura reports.


 **NunitRunnerTask**  
simple msbuild task to run nunit/xunit tests with and without coverage

 **coveralls.net**  
Coveralls.io uploader for .NET Code Coverage.

 **psake.net**  
Convention based automation for .NET leveraging psake

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

 **OpenCover**

Action:

Version:

Options

Show preview window

Dependency behavior:

File conflict action:

[Learn about Options](#)

Description

An open source code coverage tool (branch and sequence point) for all .NET Frameworks 2 and above (including Silverlight). Also capable of handling 32 and 64 bit processes. Use ReportGenerator 1.9 for best viewing results (also available via Nuget).

Author(s): sawilde

License: <https://github.com/opencover/opencover/blob/master/License.md>

Downloads: 307,523

Project URL: <https://github.com/opencover/opencover>

Report Abuse: <https://www.nuget.org/packages/OpenCover/4.6.519/ReportAbuse>

# Testing

## Importing required libraries via NuGet

1. Be sure to manage packages for MyLibraryTest

2. Search for ReportGenerator


NuGet Package Manager: HeapLibraryTest


Package source:


Filter:


Include prerelease




 **ReportGenerator**  
ReportGenerator converts XML reports generated by OpenCover, PartCover, Visual Studio or NCover into a readable reports in various formats. The reports do not only show the coverage quota, but also include the...

 **nBuildKit.Tools.VsCoverageToReportGenerator**  
A tool that converts VS code coverage files to a file type that can be...

 **HtmlWarningsReportGenerator**  
Comand line tool generating html report from xml. It render warni...  
code. It can be easy integrated with continous integration servers and reports can be published as artifacts.

 **Dapper.DynamicReportGenerator**  
A dynamic generator for Dapper

 **ReportUnit**  
ReportUnit is a report generator for the test-runner family. It uses stock reports from NUnit, MSTest, xUnit, TestNG and Gallio and converts them HTML reports with dashboards.

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

3. Select ReportGenerator

 **ReportGenerator**

Action:

Version:

Options

Show preview window

Dependency behavior:

File conflict action:

[Learn about Options](#)

Description

ReportGenerator converts XML reports generated by OpenCover, PartCover, Visual Studio or NCover into a readable reports in various formats. The reports do not only show the coverage quota, but also include the source code and visualize which line has been covered.

Author(s): Daniel Palme

License: <https://raw.githubusercontent.com/danielpalme/ReportGenerator/master/LICENSE.txt>

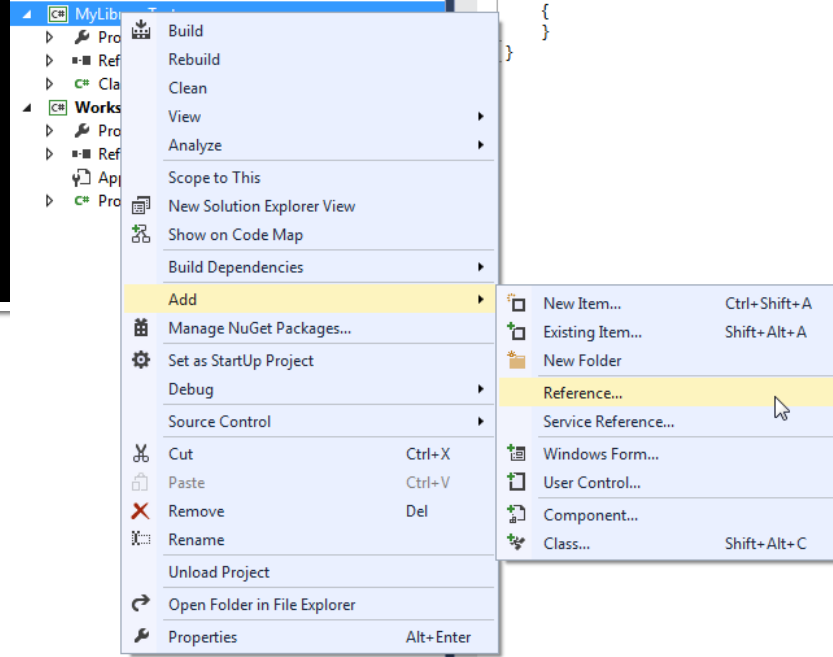
Downloads: 339,984

Project URL: <https://github.com/danielpalme/ReportGenerator>

4. Install it!

# Testing

## Add Reference

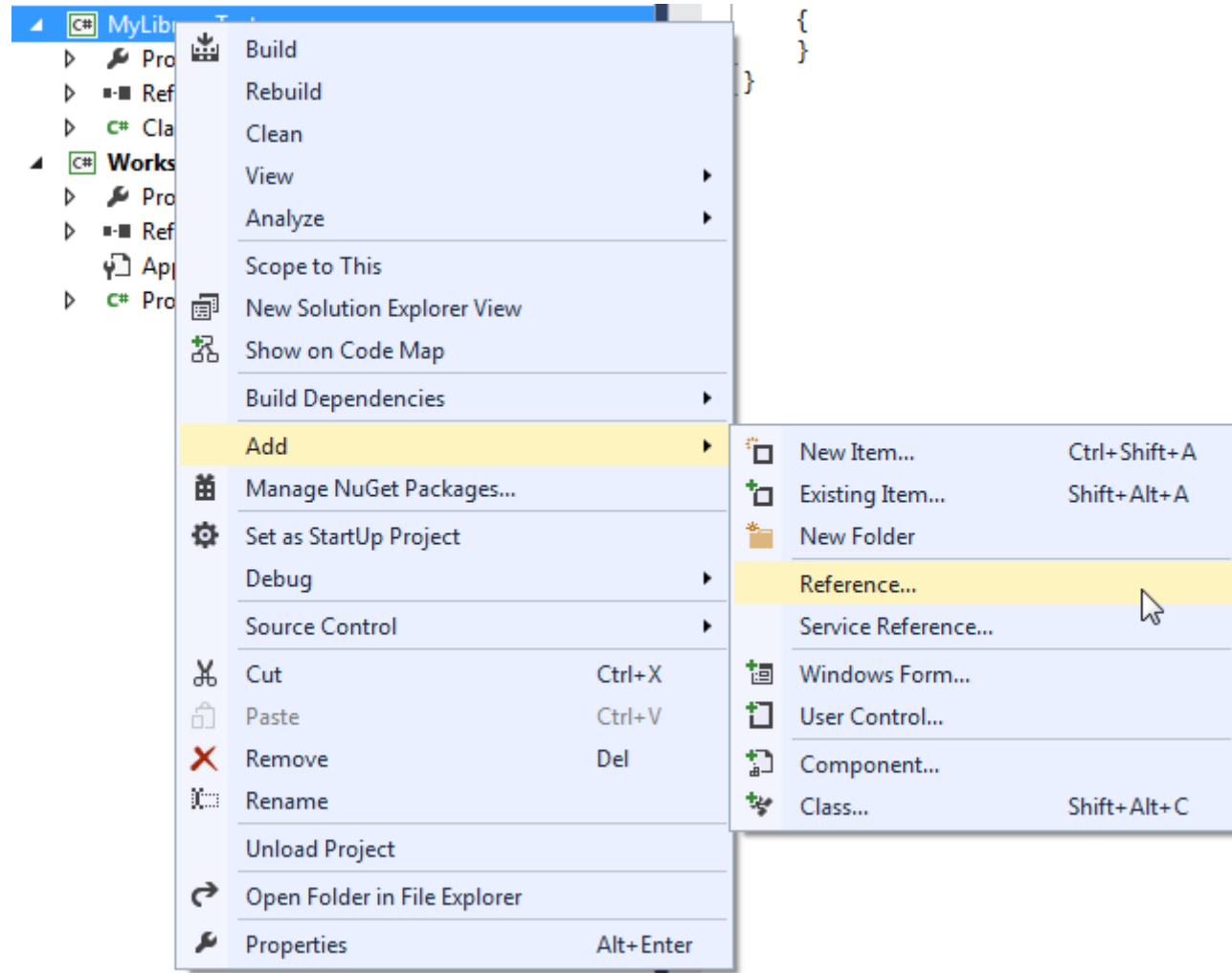


In order to be able to test code that exists in MyLibrary, we have to tell Visual Studio that project MyLibraryTest references MyLibrary in order to be able to use namespaces from MyLibrary within MyLibraryTest.

# Testing

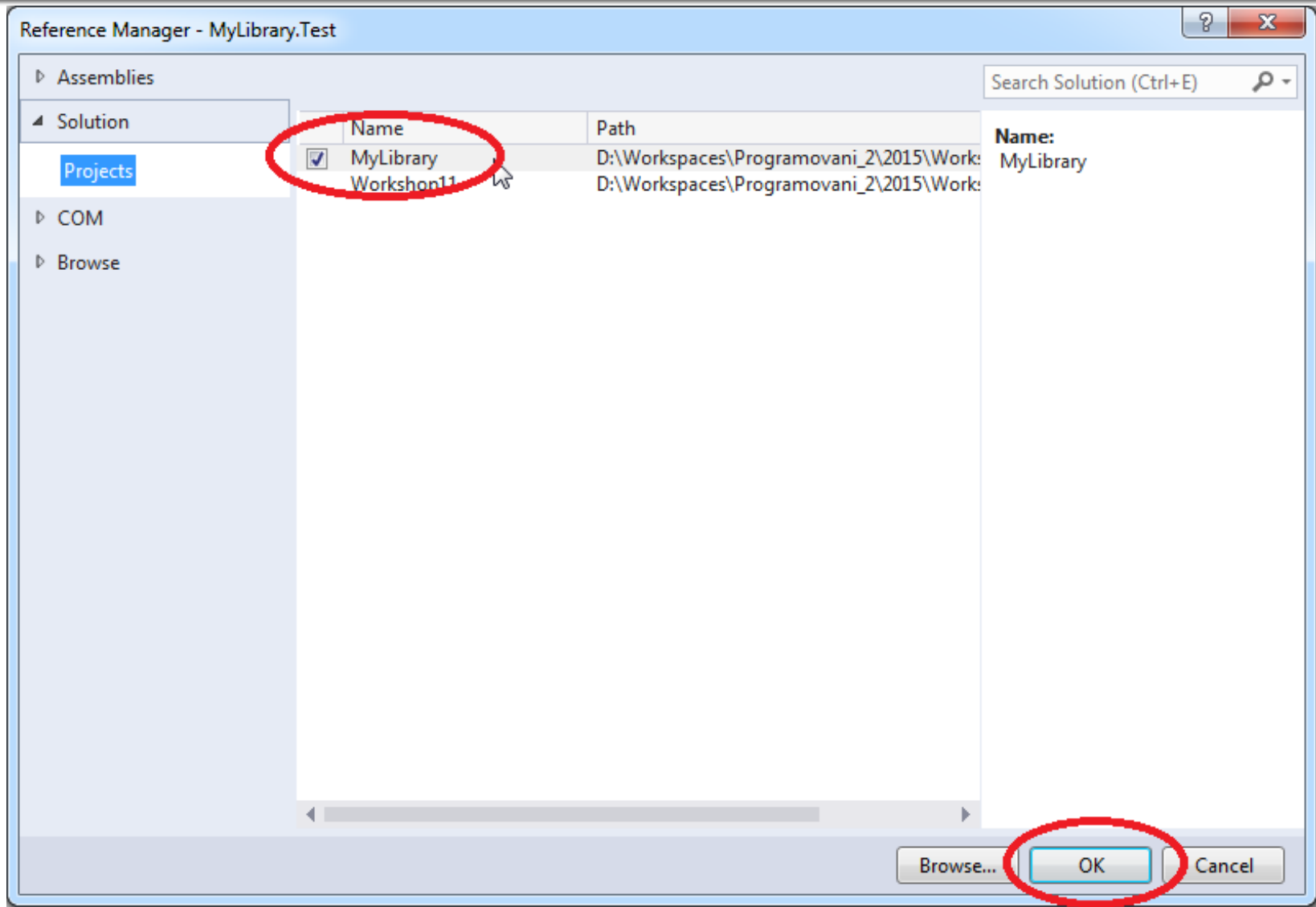
## Add Reference to MyLibrary

Right-click  
MyLibraryTest  
and navigate  
to Add->  
Reference.



# Testing

## Add Reference to MyLibrary



# Testing

## Add simple Calc class to MyLibrary

```
namespace MyLibrary
{
    public class Calc
    {
        public int Add(int a, int b)
        {
            return a + b;
        }

        public int Mul(int a, int b)
        {
            return a * b;
        }
    }
}
```



# Testing

## Add simple CalcTest class to MyLibraryTest

```
namespace MyLibraryTest
{
    [TestFixture]
    public class CalcTest
    {

        [Test]
        public void TestAdd(int a, int b)
        {
            int paramA = 1;
            int paramB = 1;

            int expectedResult = 2;

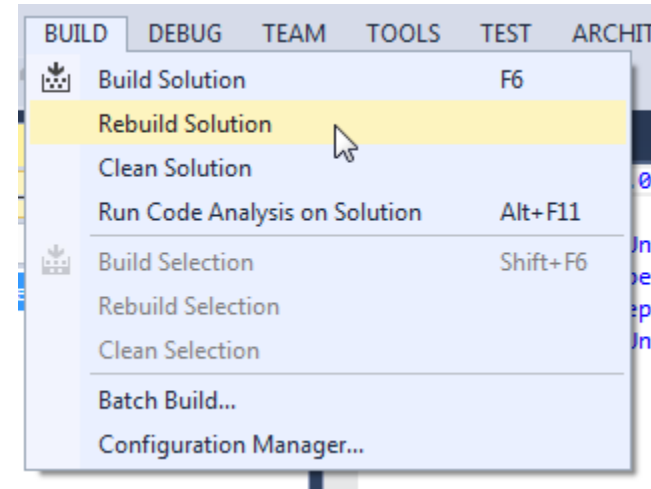
            int givenResult = FirstClass.Add(paramA, paramB);

            Assert.AreEqual(expectedResult, givenResult);
        }
    }
}
```

# Testing

## Build the solution

When “building” the solution Visual Studio will create .dll files of your libraries. We will need both to perform tests and generate reports about code coverage.



# Testing

## Running the tests

Navigate to the folder of your solution.  
There you will find subdirectory  
'packages'.

The content should look like the picture  
to the right (even though versions  
might be different).

↑ Name	Ext	Size
↑ [..]		
📁 [NUnit.3.2.1]		
📁 [NUnit.ConsoleRunner.3.2.1]		
📁 [NUnit.Extension.NUnitProjectLoader.3.2.1]		
📁 [NUnit.Extension.NUnitV2Driver.3.2.1]		
📁 [NUnit.Extension.NUnitV2ResultWriter.3.2.1]		
📁 [NUnit.Extension.VSProjectLoader.3.2.1]		
📁 [NUnit.Runners.3.2.1]		
📁 [OpenCover.4.6.519]		
📁 [ReportGenerator.2.4.5.0]		

# Testing

## Running the tests

---

Now you will need to create 3 batch files that will

1. Run tests
2. Perform code coverage
3. Generate HTML report

Put those batch files into the folder of your solution.

# Testing

## Running the tests

Batch File 1: test.bat

2 lines

```
del TestResult.xml  
.\packages\NUnit.ConsoleRunner.3.2.1\tools\nunit3-console.exe .\MyLibraryTest\bin\Debug\MyLibraryTest.dll
```

```
del TestResult.xml  
.\packages\NUnit.ConsoleRunner.3.2.1\tools\nunit3-  
console.exe  
.\MyLibrary.Test\bin\Debug\MyLibrary.Test.dll
```

You might need to adjust texts in red to match your configuration

Explanation: here we're running NUnit that executes code within your "Test" project producing "TestResult.xml" file with the report.

# Testing

## Running the tests

### Batch File 2: test-cover.bat

2 lines

```
del results.xml  
.\packages\OpenCover.4.6.519\tools\OpenCover.Console.exe -target:test.bat -register:user -filter:+[MyLibrary]*
```

```
del results.xml  
.\packages\OpenCover.4.6.519\OpenCover.Console.exe -  
target:test.bat -register:user -filter:+[MyLibrary]*
```

You might need to adjust texts in red to match your configuration

Explanation: here we're running tests again but now under observation of OpenCover that will generate Code Coverage report for namespace "MyLibrary" (that's why you might need to change that...).

# Testing

## Running the tests

### Batch File 3: test-cover-report.bat

3 lines

```
call test-cover.bat  
  
.\packages\ReportGenerator.2.4.5.0\tools\ReportGenerator.exe -reports:results.xml -targetdir:coverage  
  
start firefox "file://%CD%/coverage/index.htm"
```

```
call test-cover.bat  
.\packages\ReportGenerator.2.4.5.0\tools\ReportGenerator.  
exe -reports:results.xml -targetdir:coverage  
start firefox "file://%CD%/coverage/index.htm"
```

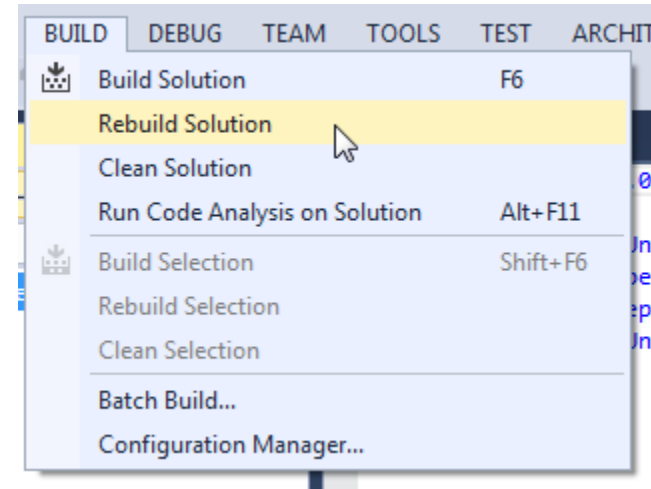
You might need to adjust text in red to match your configuration.

Explanation: Here we run ReportGenerator on the report generated by the OpenCover producing HTML pages visualizing the report.

# Testing

## Be sure to rebuild the solution

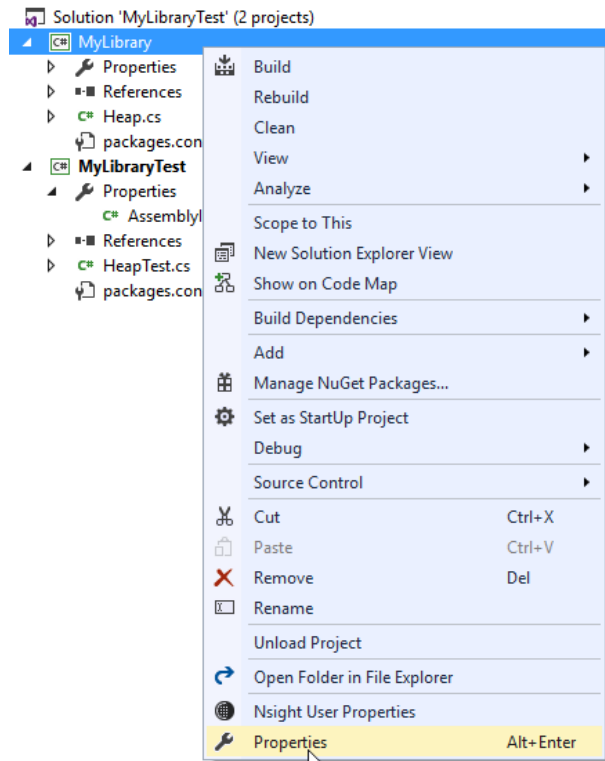
Don't forget to rebuild your solution every time you do any changes to any of your projects!





# Pitfall

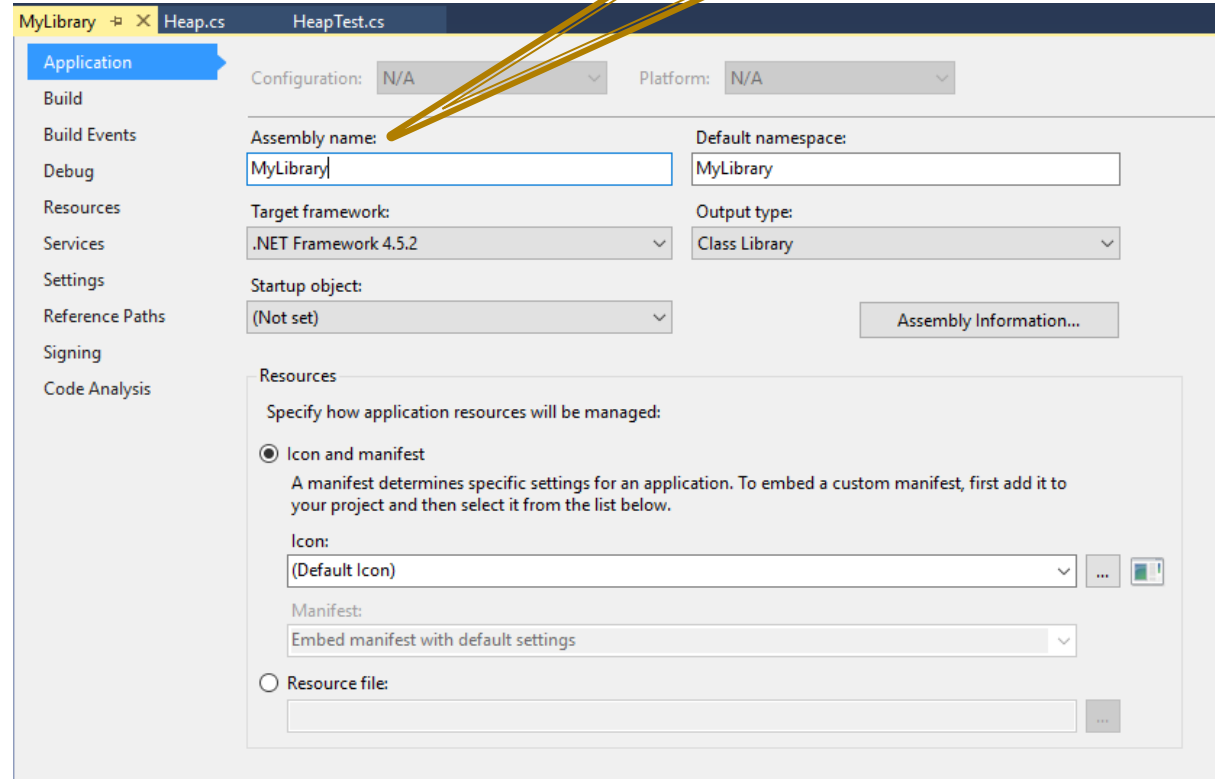
## OpenCover not generating the report?



2. Right-click your project and select Properties

1. Check the assembly name of your library

3. Check the Assembly name, must match test-cover.bat filter



# Functional Testing

## Task – Test the Heap

- Download the template: <http://alturl.com/zrhxq>
  - <http://artemis.ms.mff.cuni.cz/gemrot/lectures/prg2/2016/Workshop11-Homework.zip>
- Code Heap tests to provide complete code coverage!

### Assemblies

Collapse all | Expand all

Grouping:  Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Branch coverage
HeapLibrary	48	60	108	340	44.4%	32.5%
HeapLibrary.Heap`1	43	60	103	170	41.7%	32.5%
HeapLibrary.HeapItem`1	5	0	5	170	100%	

# Assignment 11

## Send me an email

- Email: [jakub.gemrot@gmail.com](mailto:jakub.gemrot@gmail.com)
- Subject: **Programming II – 2016 – Assignment 11**
- Zip up the whole solution and send it
- You WILL NOT find the assignment in CoDex!
- Deadline:
  - 18.5.2016 23:59
- Points: 10 + 3 (meeting the deadline)

# Questions?

I sense a soul in search of answers...

- In case of doubts about the assignment or some other problems don't hesitate to contact me!
  - Jakub Gemrot
    - [gemrot@gamedev.cuni.cz](mailto:gemrot@gamedev.cuni.cz)