

Module 2.2:

Automatic Testing



TEKNOLOGISK
INSTITUT

Agenda

- ▶ Introducing Automatic Testing
- ▶ Testing in Visual Studio

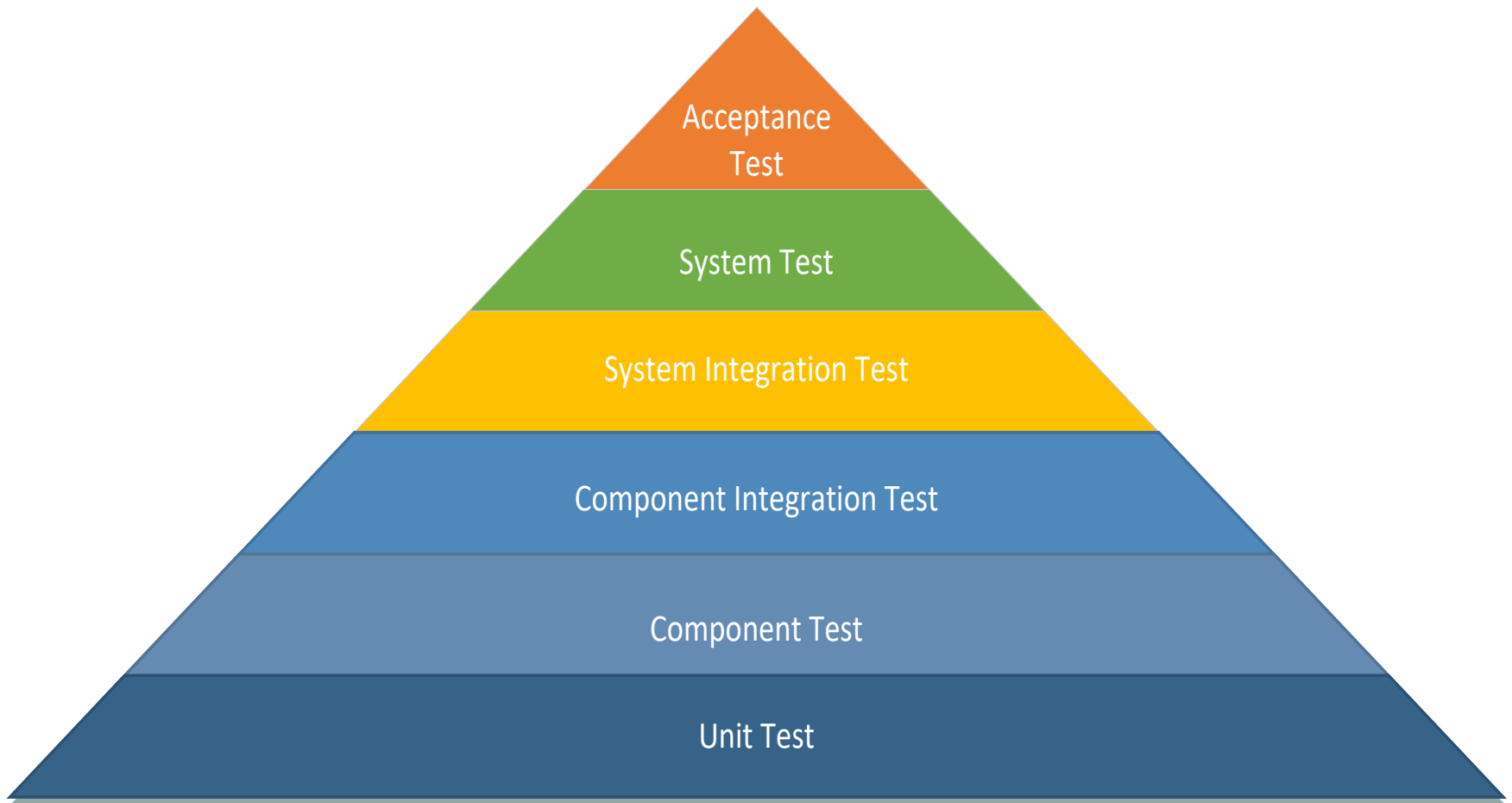
Automatic Testing

- ▶ Automatic “white-box” testing of classes
 - Developers can run tests in Visual Studio
 - Tests can run automatically when code is checked in

- ▶ (Unit) Testing
 - Captures a code-based “specification” of functionality
 - Drives safe refactoring
 - Assists regression testing

- ▶ (Unit) Testing frameworks for C# include
 - MSTest
 - NUnit
 - MbUnit
 - xUnit.net
 - ...

Various Kinds of Automatic Tests

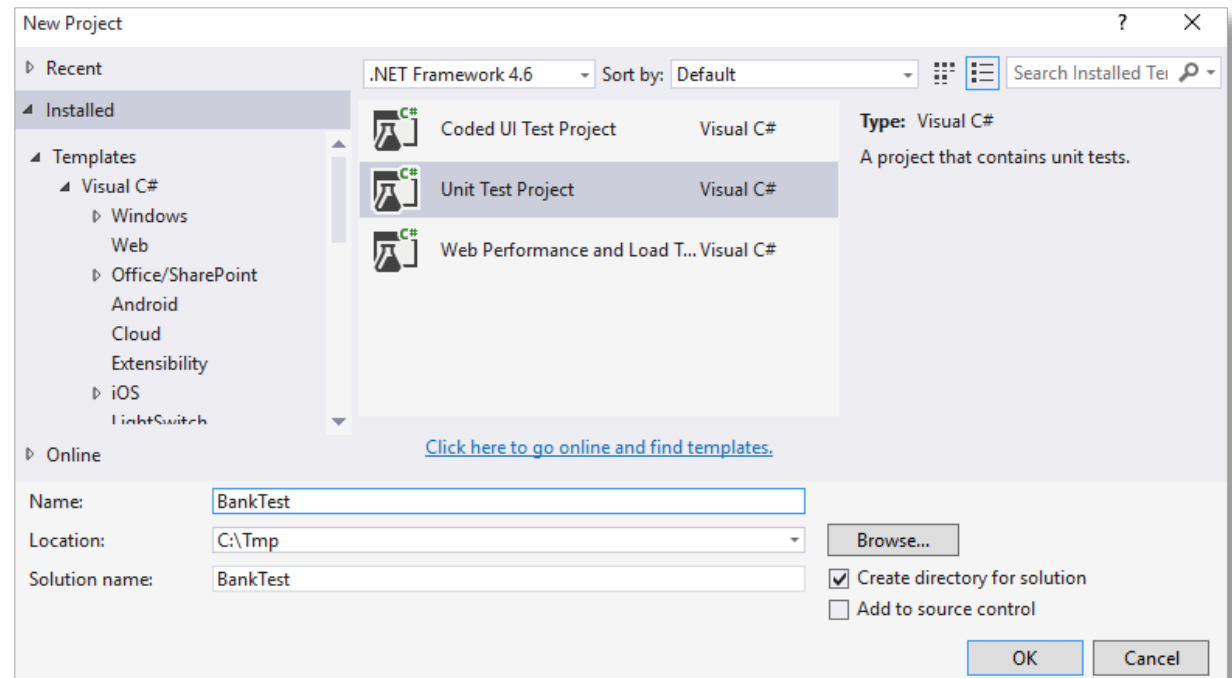


Agenda

- ▶ Introducing Automatic Testing
- ▶ **Testing in Visual Studio**

Unit Testing in Visual Studio

- ▶ Visual Studio includes MSTest (but can install other testrunners!)
 - Unit Test Project



- ▶ Create business logic project(s) "as usual"
- ▶ Create Unit Test Project with a reference to business logic project(s)
- ▶ Author test classes and methods in test project

Test Classes and Test Methods

- ▶ Test methods must be marked with the `[TestMethod]` attribute
 - Cannot have parameters and returns `void`, `Task`, or `Task<T>`

```
[TestClass]
public class BankAccountTest
{
    [TestMethod]
    public void TestDeposit()
    {
        BankAccount account = new BankAccount();
        account.Deposit(87);
        Assert.AreEqual(87, account.Balance);
    }
}
```

- ▶ Test classes must be marked with the `[TestClass]` attribute

Using the **Assert** Class and Attributes

- ▶ The `Microsoft.VisualStudio.TestTools.UnitTesting` namespaces includes e.g.
 - **Assert.**
 - `AreEqual()`
 - `AreNotEqual()`
 - `Fail()` ...
 - `[ExpectedException]` attribute

```
[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
public void TestWithdraw()
{
    BankAccount account = new BankAccount();
    account.Withdraw(87);
}
```

- `[TestInitialize]` + `[TestCleanup]` attributes

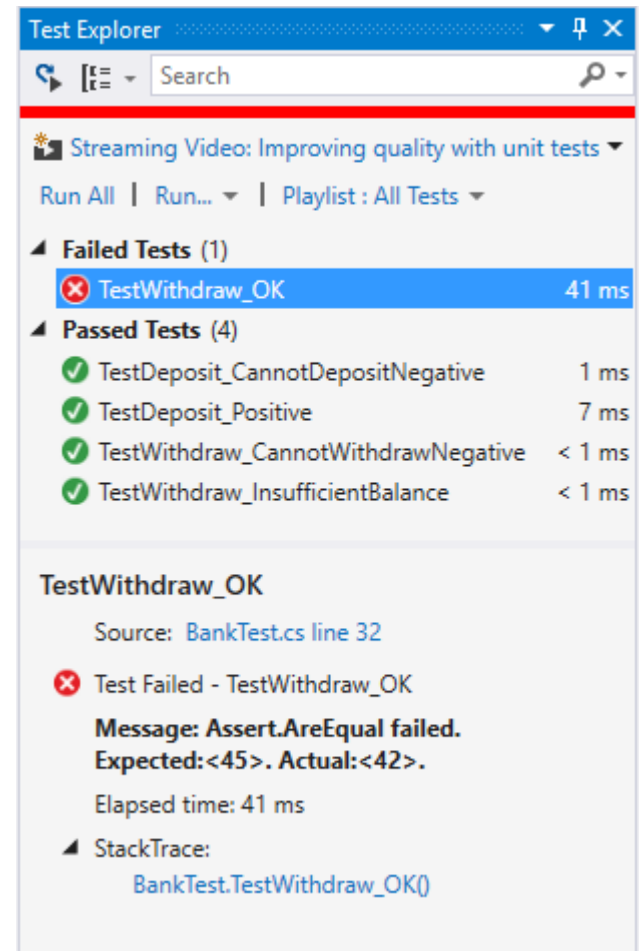
Async Test Methods and Tasks

- ▶ Test methods using await should always return **Task** or **Task<T>**
 - Never **void!!**

```
[TestClass]
public class BankAccountTest
{
    [TestMethod]
    public async Task TestDepositAsync()
    {
        BankAccount account = new BankAccount();
        await account.DepositAsync(87);
        Assert.AreEqual(87, account.Balance);
    }
}
```

Running the Tests

- ▶ Test Explorer
 - Test -> Windows -> Test Explorer
- ▶ Status annotations in source code in editor



Code Coverage Analyzer

- ▶ Some versions of Visual Studio have additional testing tools
 - Test > Analyze Code Coverage > All Tests
 - ...

Code Coverage Results				
jespe_DESKTOP-IO4GN8M 2015-10-05 22_52				
Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
jespe_DESKTOP-IO4GN8M 2015-...	3	7,50 %	37	92,50 %
bank.dll	0	0,00 %	18	100,00 %
{ } Bank	0	0,00 %	18	100,00 %
BankAccount	0	0,00 %	18	100,00 %
Deposit(double)	0	0,00 %	6	100,00 %
Withdraw(double)	0	0,00 %	10	100,00 %
get_Balance()	0	0,00 %	1	100,00 %
set_Balance(double)	0	0,00 %	1	100,00 %
banktest.dll	3	13,64 %	19	86,36 %

Test-Driven Development (TDD)

- ▶ Test-Driven Development (TTD)
 - Write unit tests before class itself
 - Class is complete when all units tests pass
 - Additional features and/or bug fixes incur yet more unit tests etc.
- ▶ Visual Studio has “TDD-friendly” IntelliSense mode
 - **CTRL-ALT-Space** toggles between modes



WINCUBATE

Jesper Gulmann Henriksen

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : jgh@wincubate.net

WWW : <http://www.wincubate.net>

Ringgårdsvej 4A

8270 Højbjerg

Denmark