

Module 2

"Hello, World"



TEKNOLOGISK
INSTITUT

Agenda

- ▶ **Anatomy of a C# Program**
- ▶ Basic Input and Output in C#
- ▶ Best Practices
- ▶ Lab 2
- ▶ Discussion and Review

"Hello, World" in C#

```
using System;

namespace SimpleCSharpApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine( "Hello, World from C#" );
        }
    }
}
```

Basic Structure

- ▶ A C# application can consist of many files, usually **.cs**-files
- ▶ A C# program consists of classes, structures, and other types
- ▶ The '{' and '}' characters are the foundational block delimiters
- ▶ The ';' character separates statements of the language, if needed
- ▶ A class is a unit of data members and "methods"
- ▶ *Classes will be treated in much more details later*

The `Main()` Method

- ▶ The **`Main()`** method has a special meaning
 - When the program starts, **`Main()`** is executed
 - When **`Main()`** finishes execution, the program terminates
- ▶ Multiple classes can each have a **`Main()`** method
 - Designate a unique **`Main()`** as the entry point
- ▶ Declare **`Main()`** to be **`static void Main`**
- ▶ Note that `C#` is
 - Case-sensitive!
 - Whitespace-insensitive!



Namespaces and using

- ▶ .NET comes equipped with thousands of classes organized into namespaces
 - **System** is the main namespace with core functionality
- ▶ Classes are referred to by their namespace

```
System.Console.WriteLine( "Hello, World from C#" );
```

- ▶ Using statements brings classes into scope

```
using System;
```

```
Console.WriteLine( "Hello, World from C#" );
```

Creating a C# Project in Visual Studio

- ▶ Projects and Solutions in Visual Studio
 - Solution Explorer
 - Solutions
 - Projects
 - Files
- ▶ A brief overview of Visual Studio features and contents
 - Common development environment for
 - Programming languages
 - Project types
 - Data sources
 - ...
- ▶ Compiling a simple C# program
- ▶ Locating errors
- ▶ Running programs with or without the Visual Studio debugger

Agenda

- ▶ Anatomy of a C# Program
- ▶ **Basic Input and Output in C#**
- ▶ Best Practices
- ▶ Lab
- ▶ Discussion and Review

The **System.Console** Class

- ▶ Appropriate for “Console Applications”
- ▶ Write output to the screen by
 - **Console.Write()**
 - **Console.WriteLine()**
- ▶ These methods are overloaded
- ▶ Read from the keyboard via
 - **Console.Read()**
 - **Console.ReadLine()**
- ▶ Console in fact supports colors!

Formatting Console Output

- ▶ Use {0}, {1}, {2} etc. as placeholders for `Console.WriteLine()`

```
Console.WriteLine(  
    "My favorite number is {0}. Not {1}",  
    87, 42  
);
```

```
My favorite number is 87. Not 42
```

- ▶ In Module 4 we will encounter an alternate – and perhaps better – way of formatting such strings

Formatting Numerical Data

- ▶ The placeholder can be further refined by

- 'C' or 'c' Currency
- 'D' or 'd' Decimal numbers
- 'E' or 'e' Exponential notation
- 'F' or 'f' Floating point
- 'N' or 'n' Number
- 'X' or 'x' Hexadecimal

```
Console.WriteLine(  
    "My favorite number is {0:x}",  
    87);
```

```
My favorite number is 57
```

- ▶ Precision of formatting can be specified after the format character
- ▶ Strings can be formatted in a similar fashion using **string.Format()**

```
Console.WriteLine( "Pi is {0:f2}", Math.PI );
```



Agenda

- ▶ Anatomy of a C# Program
- ▶ Basic Input and Output in C#
- ▶ **Best Practices**
- ▶ Lab
- ▶ Discussion and Review

Comments

- ▶ Remember to write your comments when you write your code!
- ▶ Single-line comments

```
// Input the user's name  
Console.WriteLine( "Please input your name: " );  
string name = Console.ReadLine();
```

- ▶ Multi-line comments

```
/* In the section below, we iterate through the list  
   of all the elements. We then compute their values  
   one-by-one before returning the overall value */  
DoStuff();
```

XML Documentation

- ▶ Use `///` to generate XML comments

```
/// <summary>
/// This is an example program for use with
/// the "Grundlæggende C# 6.0" course.
/// </summary>
class Program
{
    /// <summary>
    /// This is the entry point of the application.
    /// </summary>
    /// <param name="args">Command-line arguments
    /// supplied to the application</param>
    static void Main( string[] args )
    {
        Console.WriteLine( ... );
    }
}
```

The Integrated .NET Framework Documentation System

- ▶ Extremely valuable documentation while programming
- ▶ A must to use for any programmer!
- ▶ “HELP” menu item in Visual Studio
- ▶ Press F1 on C# keyword or .NET type

Using the Visual Studio Debugger

```
Program.cs
Wincubate.Module02.Slide16.Program
Main(string[] args)

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Wincubate.Module02.Slide16
8 {
9     class Program
10    {
11        static void Main( string[] args )
12        {
13            string name;
14            Console.WriteLine( "Please enter your name: " );
15            name = Console.ReadLine();
16            Console.WriteLine( "Hello, {0}", name );
17        }
18    }
19 }
20

100 %
```


Customizing Visual Studio

- ▶ Tools -> Options
- ▶ Millions of commands and shortcuts
 - Shortcuts can be (re)defined at will
- ▶ Customizations
- ▶ Code Snippets
- ▶ Extensions and Updates

- ▶ Old, but still very good: "Sara Ford's Tips 'n Tricks"
 - <http://channel9.msdn.com/Blogs/NicFill/Sara-Fords-101-Visual-Studio-Tips-in-55-Minutes-Challenge>

- ▶ Reset everything(!) either via the UI or using
 - **devenv /ResetSettings**



Lab 2: Creating and Debugging C# Programs using Visual Studio



Discussion and Review

- ▶ Anatomy of a C# Program
- ▶ Basic Input and Output in C#
- ▶ Best Practices



WINCUBATE

Jesper Gulmann Henriksen

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : jgh@wincubate.net

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

Hasselvangelen 243

8355 Solbjerg

Denmark