

Industrial Programming

Lecture 2: Introduction to .Net & C#

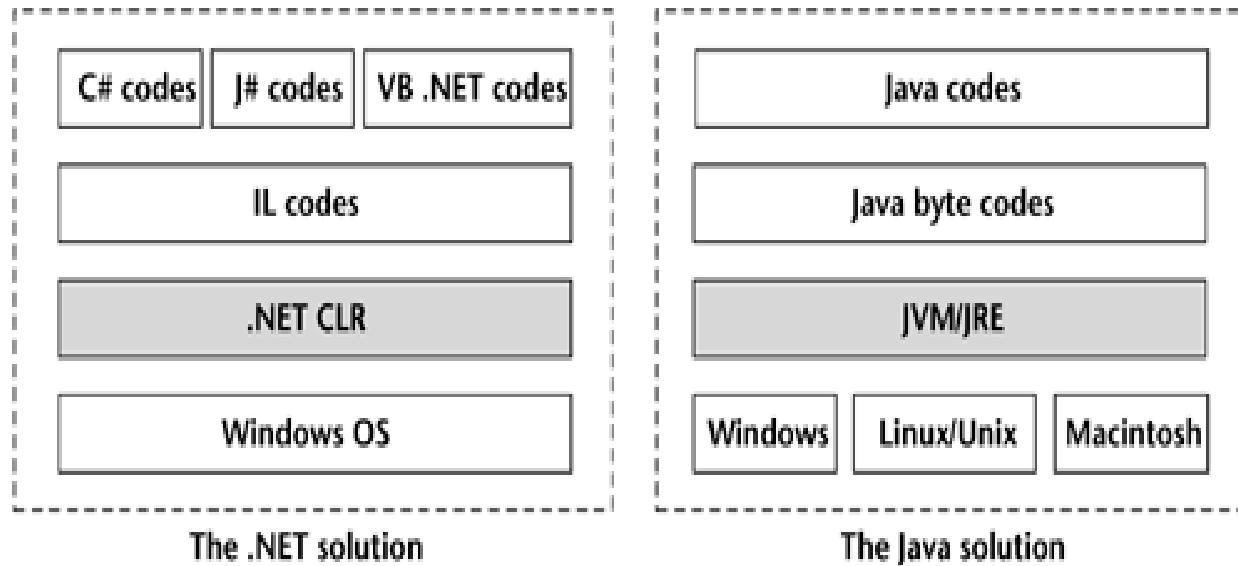
Microsoft .Net

- Microsoft .Net is Microsoft's Internet strategy.
- .Net was originally called NGWS - Next Generation Windows Services.
- .Net is a Internet and Web based infrastructure that will run in any browser.
- .Net and Java/JVM are modern, powerful programming techniques and are equal competitors.

.Net Vs Java/JVM

- Java is a programming language designed to be run on many different platforms, and so uses a **common language** (Java) which has to be compiled and run on different platforms (eg. windows, mac and linux).
- Microsoft, with their offering of .NET, takes on a different approach, by allowing you to program in **any language** (VB.Net, C#, F#) you choose, but has compilers for many different languages that generates a platform specific code (i.e. Microsoft or Windows).

.Net vs Java Framework



Advantages and Disadvantages of .Net

- One advantage of using .Net is that you are not stuck with one language: a multi-language solution is easier to provide than with a Java.
- One disadvantage is that MS has implemented .Net such that it is not nearly as portable as Java. For example, you can't use MS's .Net tools to compile an executable that will run on Linux or Solaris in addition to Windows, while you can do this with Java.

Suitability of .Net

- Java can be used to write programs for **many different operating systems**, and .Net can be used to make any programming language into a Windows program.
- .Net is suitable for professionals who want to develop an application for **windows** easier than using Java for the same.

The .Net Framework

- The .Net Framework is a common environment for building, deploying, and running Web Services and Web Applications.
- Main components:
 - The Common Language Specification (CLS).
 - The Common Type System (CTS).
 - The Common Language Runtime (CLR).
 - The .Net Framework Class Library (FCL).

Common Language Specification (CLS)

- Defines a set of language features, e.g. how methods and constructors are called.
- Allows .Net applications to fully interact with other objects *regardless of the language* in which they were implemented.

Cont. Common Language Specification (CLS)

- In an object-oriented environment such as C# and the .Net platform, *everything is an object*. Once an object is created it needs to be able to communicate or interact with other objects.
- These other objects may have been created in a different language, for example C++, C#, J#, F#.
- The .Net platform overcomes this problem by implementing the Common Type System (CTS). The Common Type System means that every language in .Net uses the same data types (or objects). When you declare a string in C# you are creating an instance of a string class. When you define a string in Visual Basic, you create an instance of the same string class, thus when you pass parameters between languages there is no longer any requirement to convert data types.

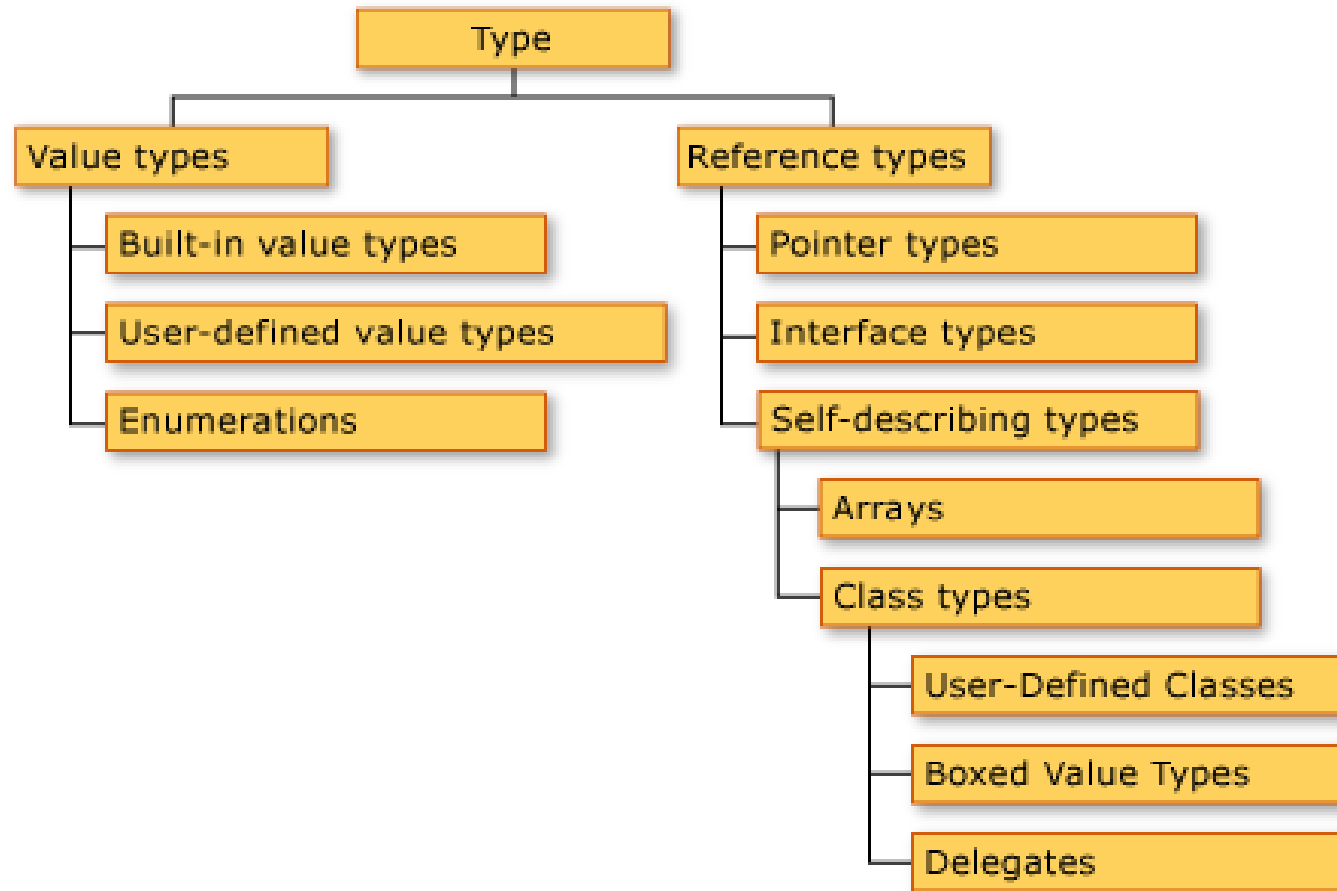
The Common Type System (CTS)

- Defines how types are declared, used, and managed in CLR.
- Specifies a set of types that any programming language in CLR should implement.
- Two type categories: Value types and reference types.

Cont. The Common Type System (CTS)

- Languages often define aliases
- E.g. CTS defines `System.Int32` – 4 byte integer
 - `C#` defines `int` as an alias of `System.Int32`

Classification of CTS Types



From MSDN

The Common Language Runtime (CLR)

- A ***virtual machine*** of the .Net framework.
- It is an implementation of Microsoft's Common Language Infrastructure (CLI) standard.
 - Defines an execution environment for program code.
 - It runs a form of bytecode called the Microsoft Intermediate Language (MSIL).

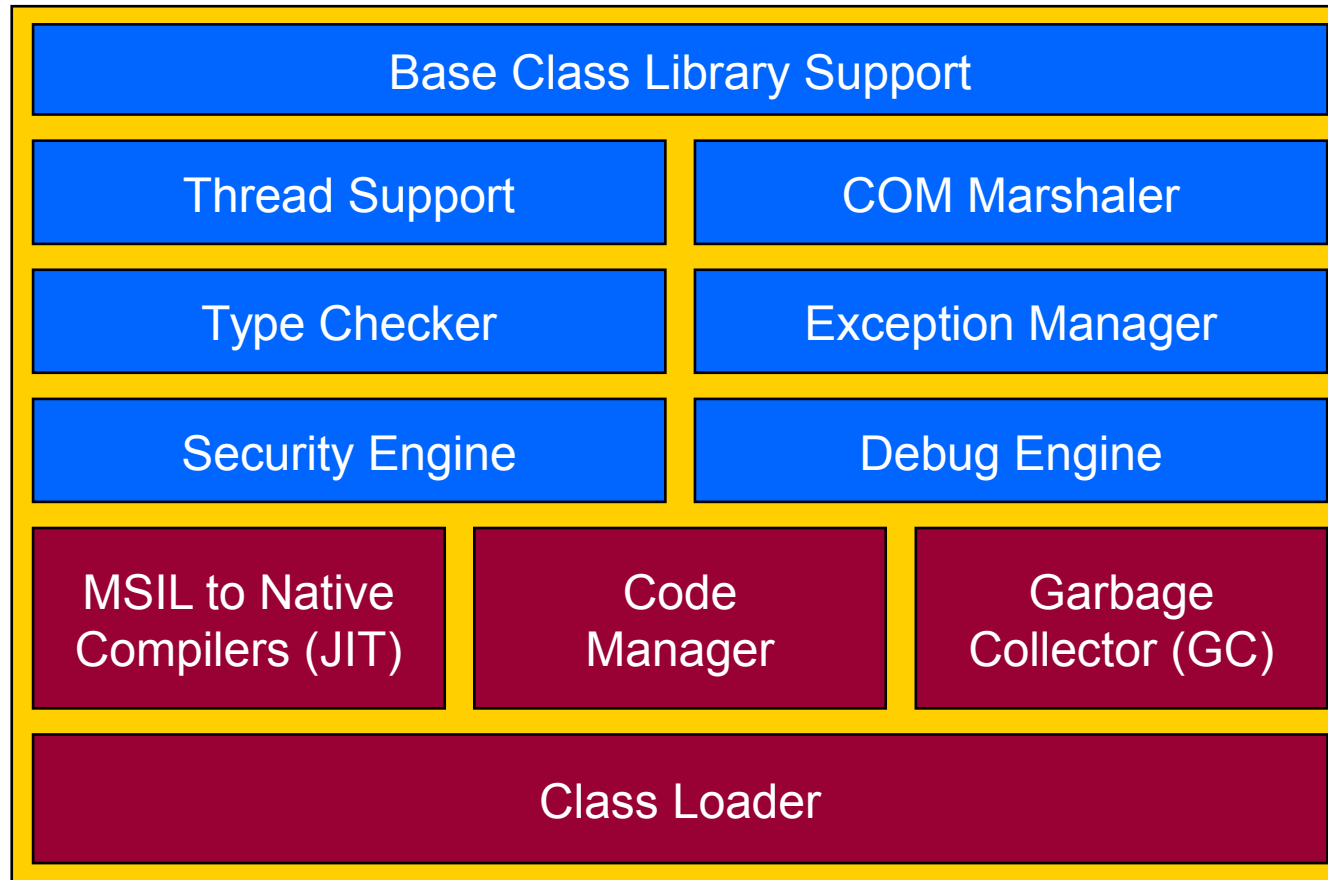
Cont. The Common Language Runtime (CLR)

- Developers write programs in C#, F# or VB.Net
 - At compile time, the CLR converts the program into MSIL code.
 - At runtime, the CLR's just-in-time compiler converts the MSIL code (CLI) into native code (suitable for the underlying operating system).

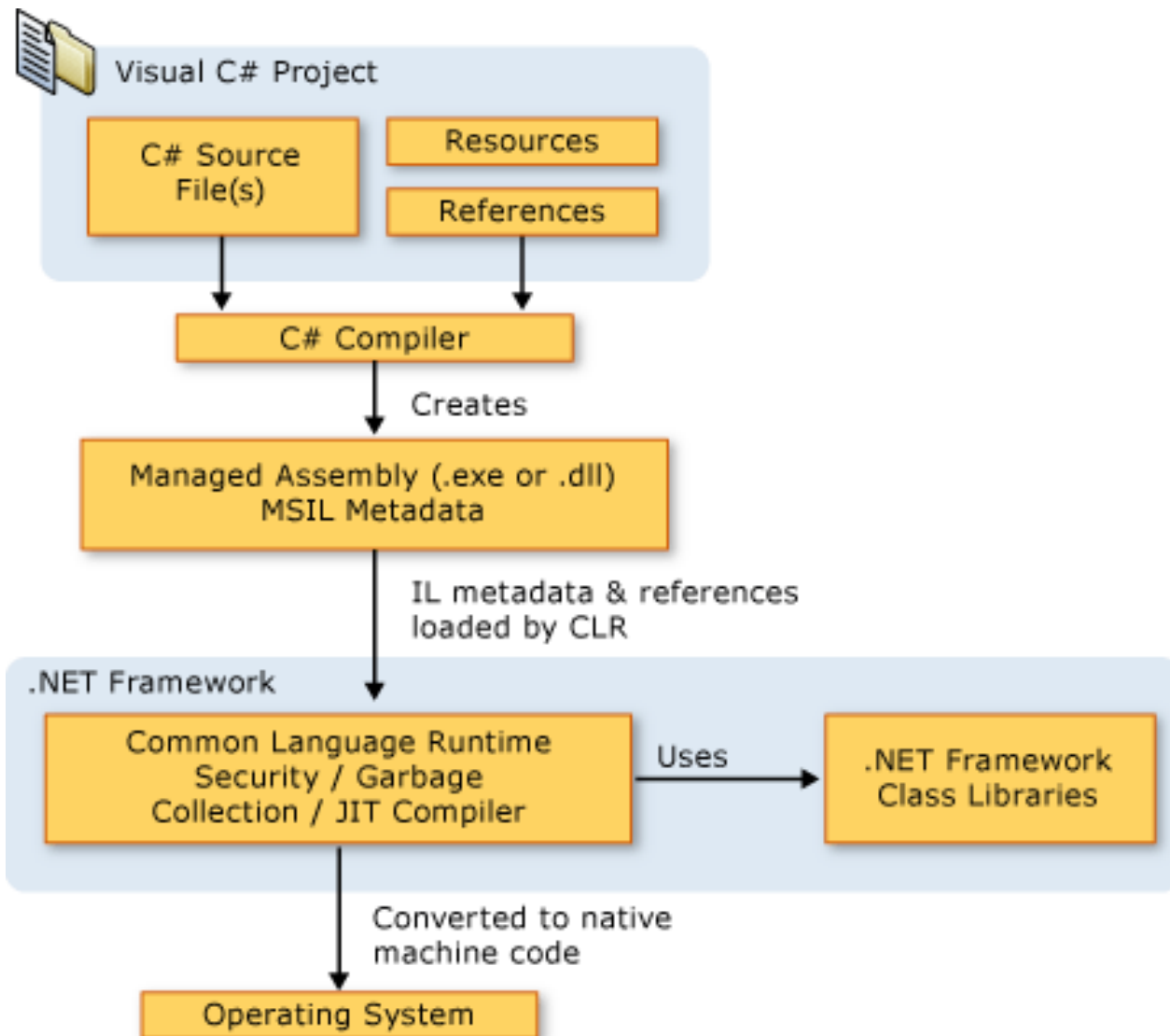
Key Tasks for the CLR

- Class loading
- Assembly integrity check
- Security check
- Memory type safety check
- Just-in-time compilation
- Automatic memory management

CLR Components



Source: MSDN



The .Net Framework Class Library (FCL)

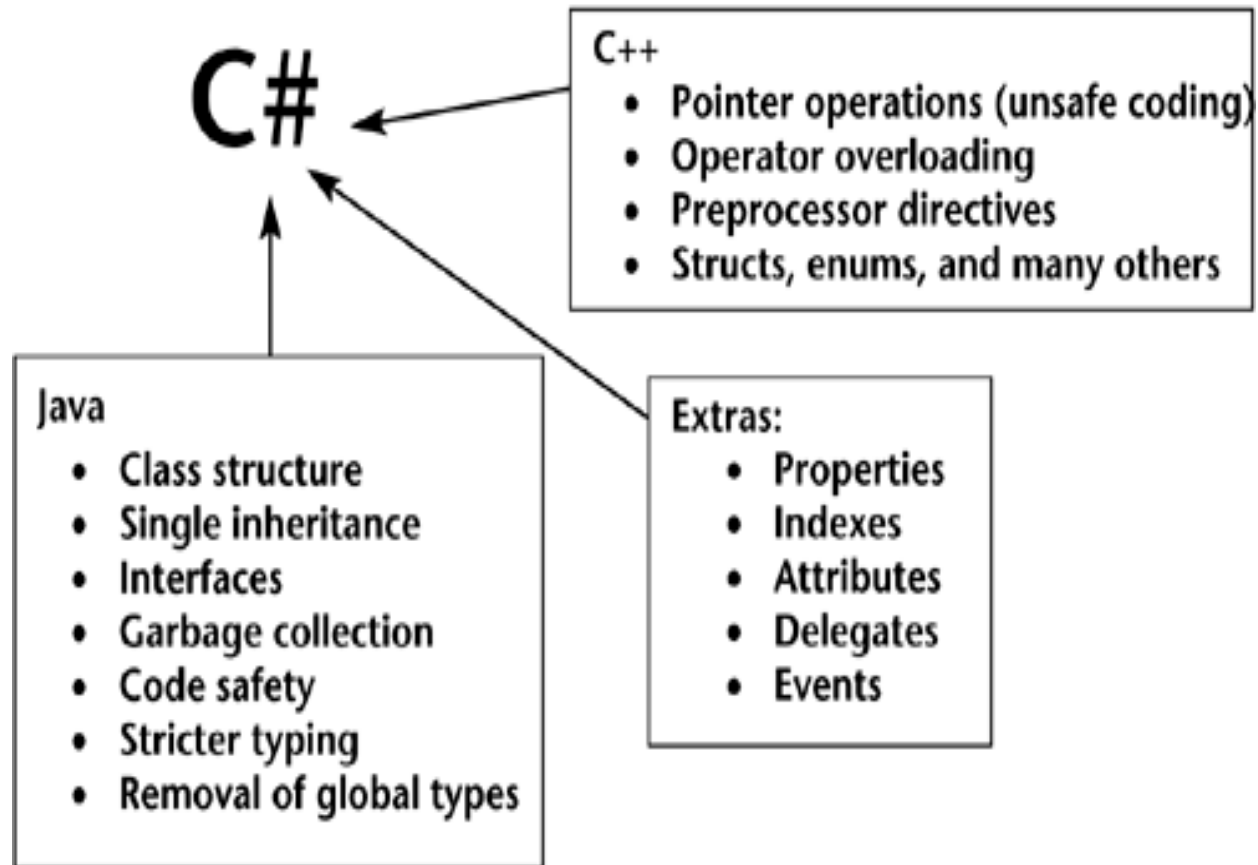
- Contains thousands of reusable classes, interfaces and value types.
- Grouped in namespaces.
- Fundamental functionality is provided by the Base Class libraries (BCL) which is part of FCL.
- Examples of FCL namespaces: System, System.IO, System.Net, System.security

C# Objectives and Features

- Simple, modern, general purpose, object-oriented language;
- Support for ... strong type checking, array bounds checking, check for uninitialised variables, and automatic garbage collection;
- Development of software components suitable for deployment in distributed environments;
- For applications both for hosted and embedded systems

C# Features

- Many features have been picked up from Java
- ... but also from other languages eg C++
- Notable for *systems programming*: support for low-level C-style programming
- C# was developed out of C and C++



Feature Comparison Java vs C#

- Platform portability:
 - Java: high (based on a portable JVM)
 - C#: low (tailored for .Net on Windows)
- Simplicity:
 - Java: simple & easy to learn
 - C#: more difficult to learn, with convenience features
- Intermediate Code:
 - Java: JVM bytecode
 - C#: IL in .Net assemblies

The Hello World C# Program

```
using System;
namespace HelloWorld
{
    class Hello
    {
        static void Main()
        {
            System.Console.WriteLine("Hello World!");
        }
    }
}
}
Ind. Programming
```

Hello World Program Explained

- `using System`:
 - The "using" command allows us to specify which libraries namespaces we want access to.
 - It is similar to Java's `import`.
 - The "System" library contains the namespaces for all common .NET variables such as `System.Int32` (which represents an integer) amongst other things.
 - The *System* namespace also contains core classes including the *Console* class.

Cont. Hello World Program Explained

- `namespace HelloWorld`
 - Defines a new namespace *HelloWorld*
 - Namespaces separate code into individually named containers and avoids the danger of having duplicate variable names within the scope of a block.
- `class Hello`
 - Define a class named *Hello* in the namespace.
 - Just like in Java or C++, C# uses classes to facilitate its object-orientated design methodology. Classes allow us to create individual self-contained units of code, which can be instantiated throughout our application. All C# applications must contain at least **one class**.

Cont. Hello World Program Explained

- `static void Main()`
 - All C# applications must have a `Main()` function (Note: capital 'M'), which is the entry point for execution of the application.
 - In contrast to Java, `Main()` can be declared without arguments.
 - The `Main()` function is declared as `public` by default, since it is called from the outside.
 - `Main` can return either an integer variable type or nothing, with `void`. `Main` can take an array of strings as (command-line) arguments, and can return an `int` value as return code eg:

```
public static int Main(string []args)
{

}
```

Cont. Hello World Program Explained

- `System.Console.WriteLine("Hello World!");`
 - A call to the `WriteLine()` function of the `Console` class.
 - It writes “Hello World” to the console.
 - A message “Hello World” appears on the screen on compiling and executing this code.

Compiling and Executing the code

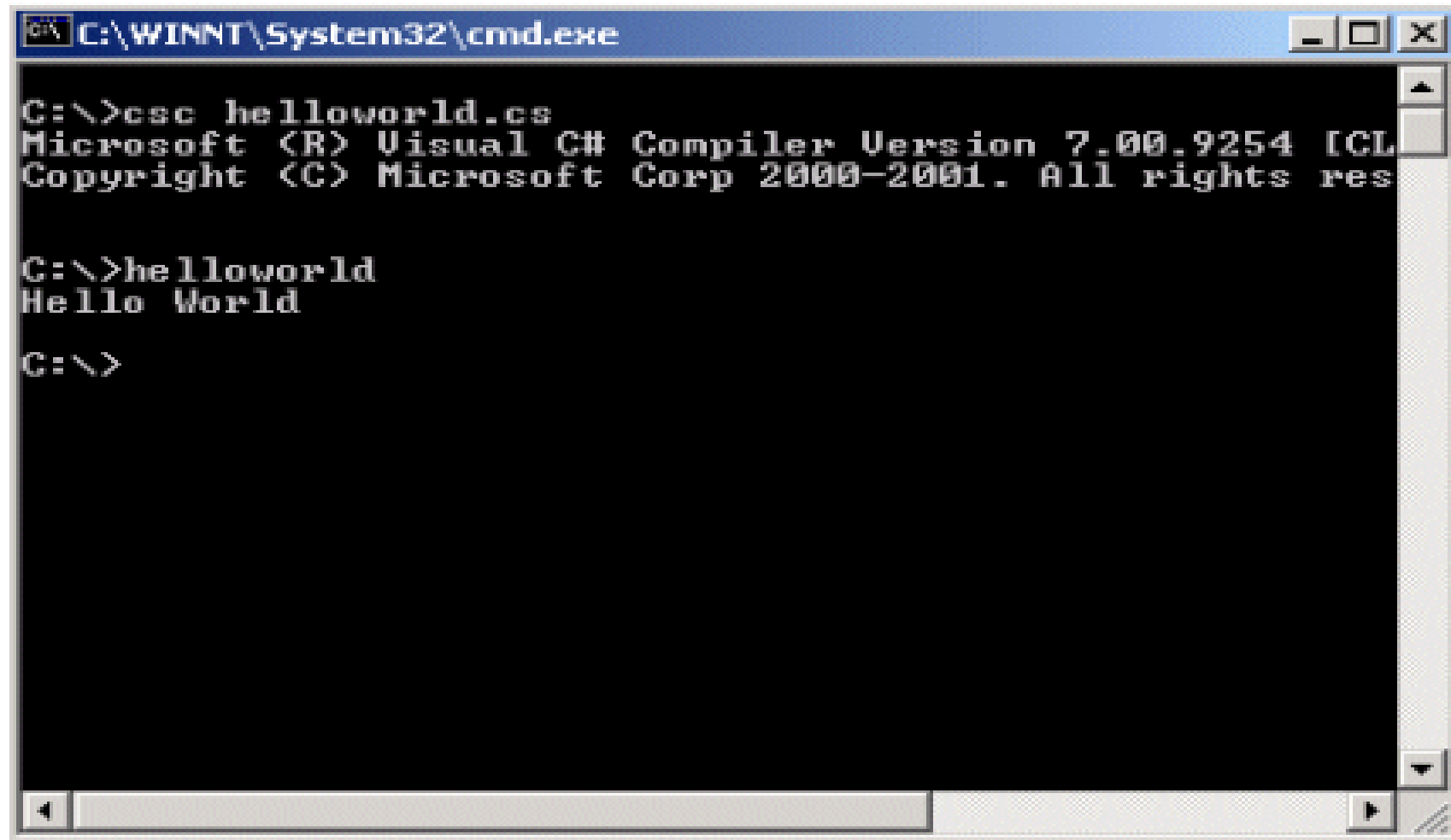
- Type the code in the notepad and save the file into c:\ or any directory you wish, and then minimise notepad.
- Now that we've created our source code file, we have to compile it using .NET's built-in C# compiler, "csc.exe".
- Open a new command prompt window and change the path to the directory where you saved the file "helloworld.cs".
- Enter the following command to compile our C# source code file:

`csc helloworld.cs`

To execute our application, simply type:

`helloworld`
- You should get the result now.

Your screen should look like this



```
C:\WINNT\System32\cmd.exe

C:\>csc helloworld.cs
Microsoft (R) Visual C# Compiler Version 7.00.9254 [CL
Copyright (C) Microsoft Corp 2000-2001. All rights res

C:\>helloworld
Hello World

C:\>
```

Using Visual Studio

- Under projects, choose a new ConsoleApplication
- Enter the program in the main window that comes up
- To compile, *build project*
- To run, choose *debug menu*

Another Example

```
class TestClass {
    public static void Main () {
        MyClass m = new MyClass();
        System.Console.WriteLine("99*2 = {0}",
            m.Double(99));
    }
}

class MyClass {
    public int Double(int val) {
        return val*2;
    }
}
```

Differences to Java

- The filename can be different to the class name
- A file can contain several public classes and interfaces
- Each of the classes can have its own Main function
- To denote the starting point of the execution, the main class has to be specified on the command line, eg.

```
csc Hello.cs /main:Class1
```

- To compile C# code into a library, rather than a standalone executable, call csc like this

```
csc /target:library Hello.cs
```


Download Instructions

- As student, you can download Visual Studio for free, through the Microsoft Dreamspark programme.
- Goto to this HWU software page:
<http://www.hw.ac.uk/is/it-essentials/software.htm>
- From there follow the 'Microsoft Dreamspark' link to
<https://www.dreamspark.com/>
- You need to request access through the central help-desk
IThelp@hw.ac.uk

Linux Setup for C#

- Download from <http://www.go-mono.com/>
- Install a recent version of mono eg.

`mono-2.4.2.2-1mdv2009.1`

- Compile like this

`gmcs hello.cs`

- Run like this

`mono hello.exe`

- For an IDE use

`monodevelop-2.0-1mdv2009.1`

Useful Links

.Net Framework to run C# applications - Download Instructions:

<http://msdn.microsoft.com/en-us/netframework/default.aspx>

C# Basics and Hello World Program

- <http://www.devarticles.com/c/a/C-Sharp/Csharp-Introduction>
- <http://www.devarticles.com/c/a/C-Sharp/The-Basics-of-Csharp-A-HelloWorld-Application/1/>
- <http://www.devarticles.com/c/a/C-Sharp/The-Basics-of-Csharp-A-HelloWorld-Application/2/>

More C# Tutorials

- <http://www.csharp-station.com/Tutorials.aspx>

[http://msdn.microsoft.com/en-us/library/aa288436\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/aa288436(VS.71).aspx)