ETH Zürich
Chair of Software Engineering

C# Programming in Depth

Prof. Dr. Bertrand Meyer

March 2007 – June 2007

Introducing C# and .NET Framework

Lisa (Ling) Liu

Welcome

- Course web page:
  http://se.inf.ethz.ch/teaching/ss2007/251-0290-00/index.html

- Books:
  - Judith Bishop & Nigel Horspool C# Concisely, Addison Wesley, 2004 ISBN 0 321 15418 5
  - Andrew Troelsen: Pro C# 2005 and the .NET 2.0 Platform, Third Edition, Apress, 2005, ISBN 1-59059-419-3

- Test
  - The exam for C# programming is to deliver the source code and documents for the given project specifition.

Overview

- Why have this course?
- Who should/can take this course?
- What are the components of this course?
- Overview of .NET and C#

Why have this course?

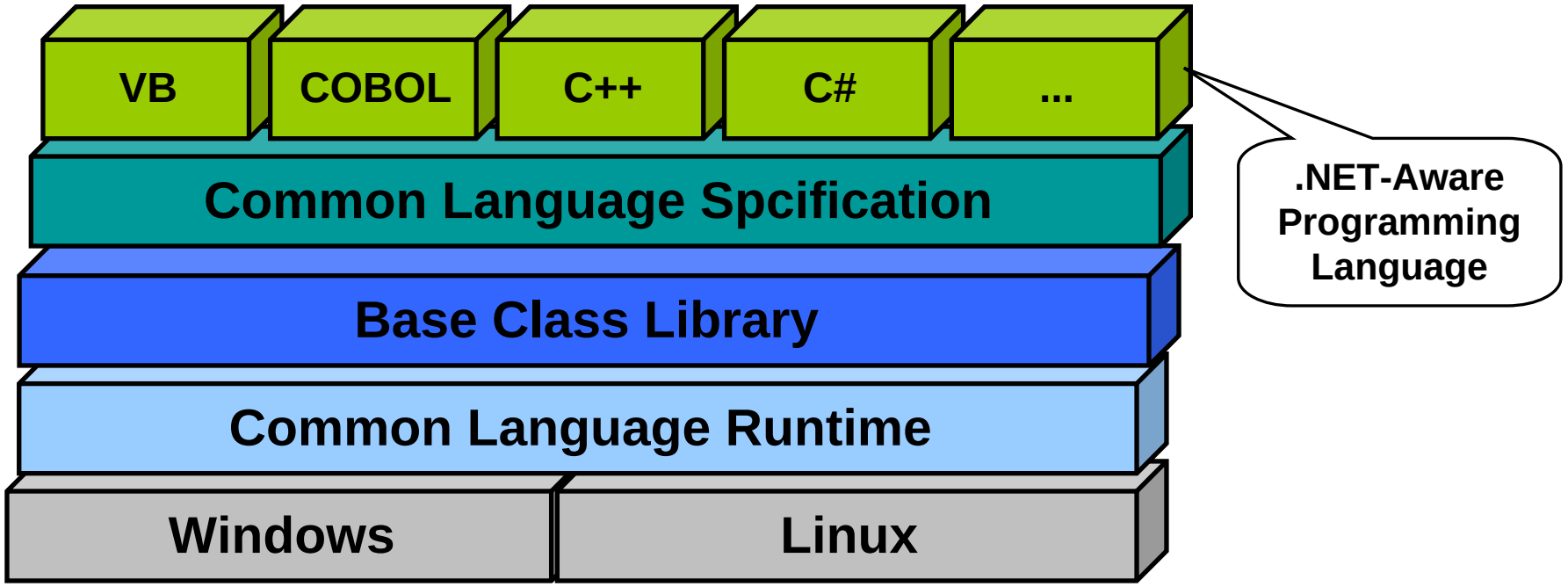
- .NET provides:
  - Full interoperability with existing code
  - Complete and total language integration
  - A common runtime engine shared by all .NET-aware languages

Who should take this course?

- The only requirement is object-oriented programming (OOP) experience

Microsoft .NET framework architecture

.NET can be understood as a new runtime environment and a comprehensive base class library.



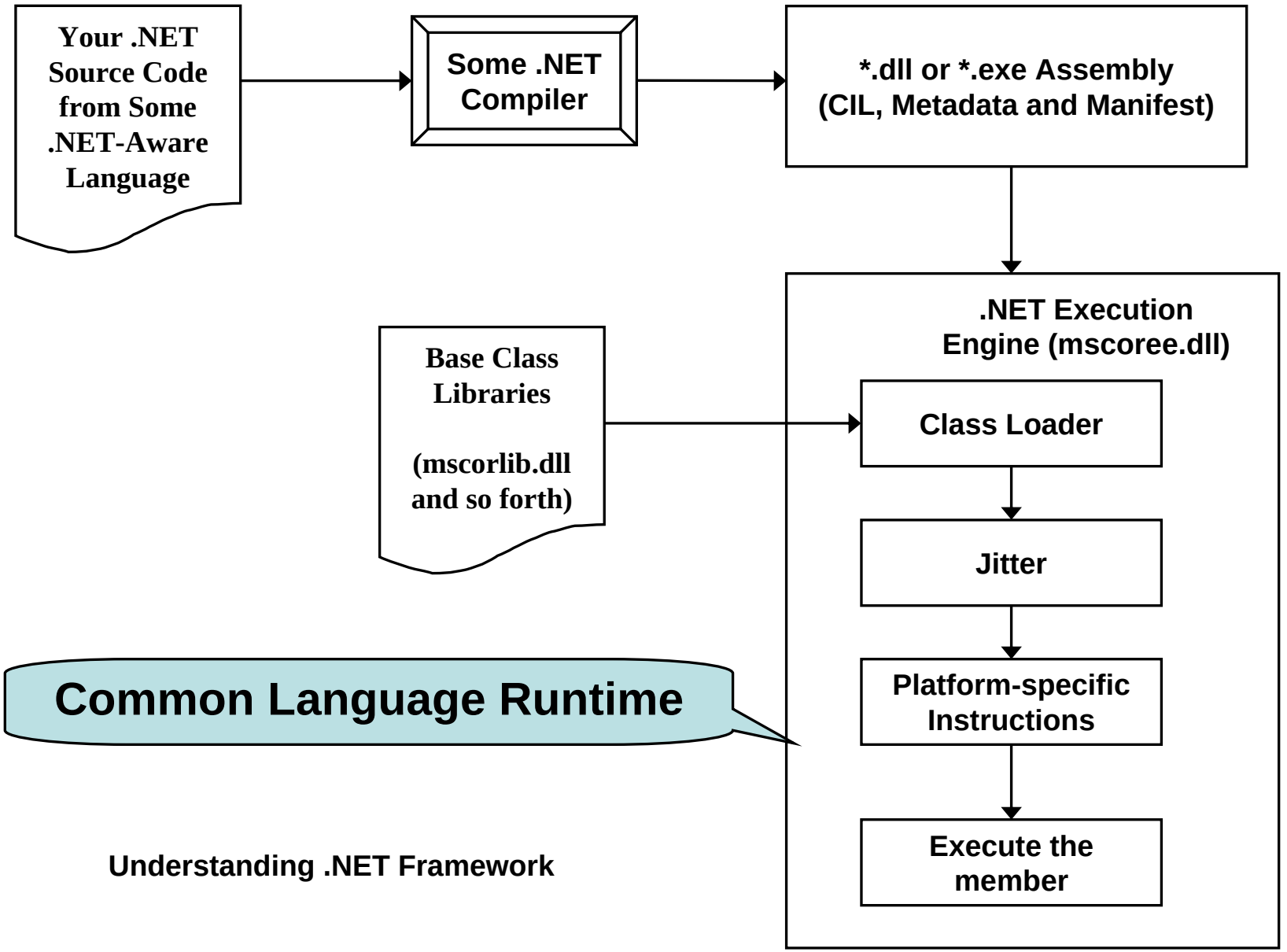
C# features

- No pointers required
- Automatic memory management through garbage collection
- Formal syntactic constructs for enumerations, structures and class properties
- The C++ like ability to overload operators for a custom type, without the complexity
- Using a syntax very similar to C++ templates to build generics
- Full support for interface-based programming techniques
- Full support for aspect-oriented programming techniques via attributes

Important point of C#

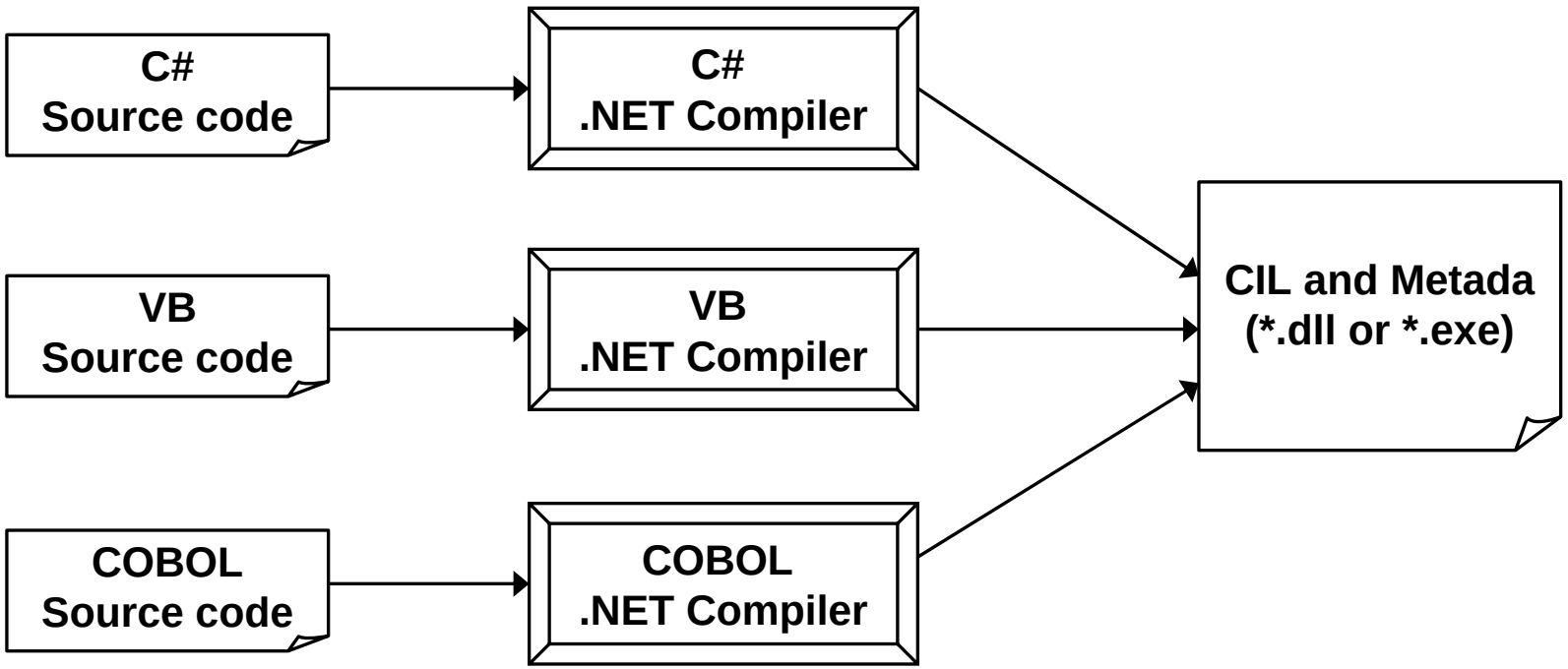
It can only produce code that can execute within the .NET runtime

- Managed code: the term used to describe the code targeting .NET runtime
- Assembly: the binary units that contains the managed code



Understanding .NET Framework

.NET assemblies



Contents of .NET assemblies

- IL/CIL code
- metadata
- manifest

Benefit of CIL

```
//CIL code for C# Calc::add method
.method public hidebysig instance int
    .maxstack 2
    .locals int ([0] int 32 CS$1$0000)
    IL_0000:  ldarg.1
    IL_0001:  ldarg.2
    IL_0002:  add
    IL_0003: stloc.0
    IL_0004: br.s       IL_0006
    IL_0006: ldloc.0
    IL_0007: ret
} //end of method Calc::Add
```

```
//CIL code for VB Calc::add method
.method public instance int 32 Add
    .locals int ([0] int 32 Add)
    IL_0000:  nop
    IL_0001:  ldarg.1
    IL_0002:  ldarg.2
    IL_0003:  add.ovf
    IL_0004: stloc.0
    IL_0005: br.s       IL_0007
    IL_0007: ldloc.0
    IL_0008: ret
} //end of method Calc::Add
```

- Allow you to build applications using your language of choice
- A single code base running on numerous operating systems

JIT/Jitter (just-in-time) Compiler

- .NET runtime environment leverages a JIT compiler for the underlying platform
- Cache the results in memory in a manner suited to the target operating system

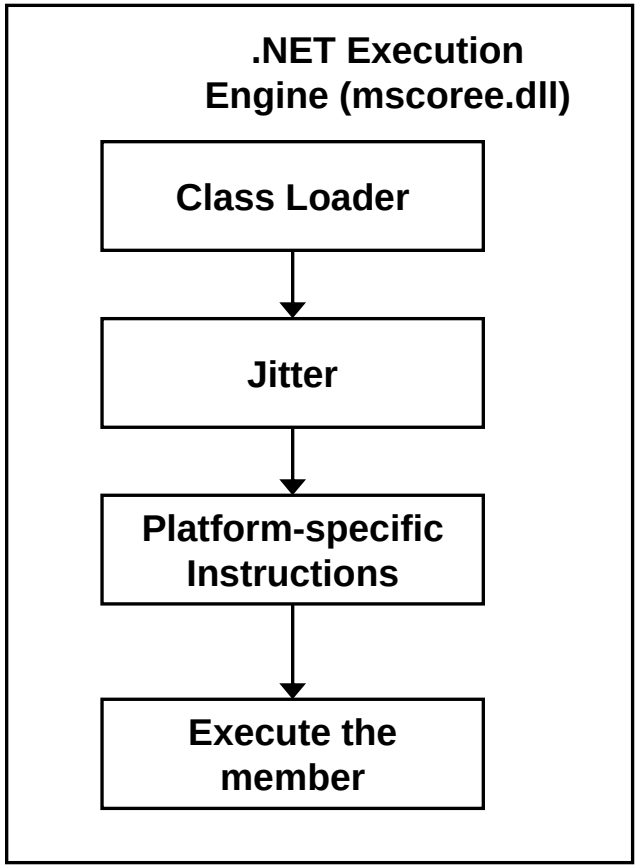
Base class libraries

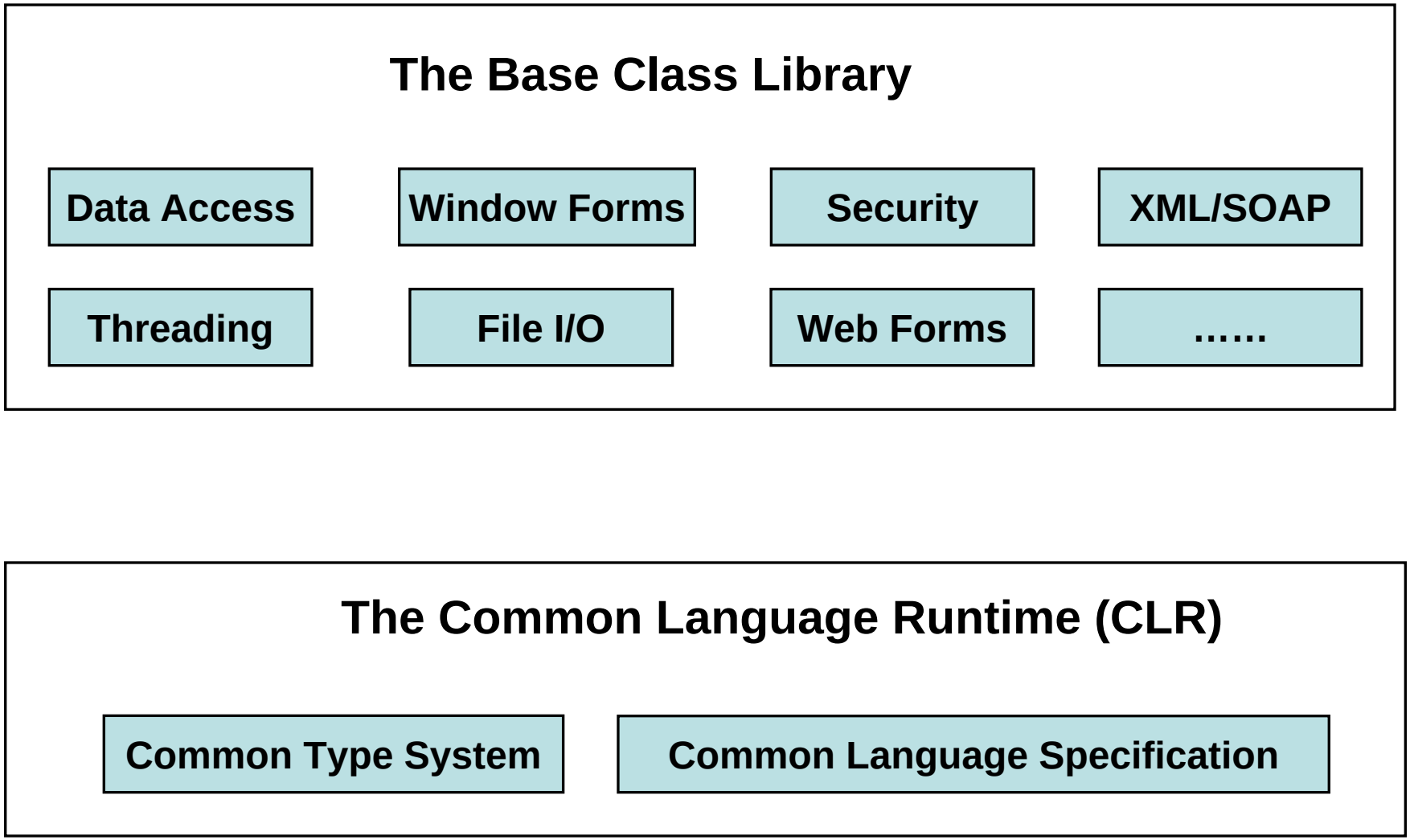
- Encapsulate various primitives
- Provide support for a number of services required by most real-world applications
- Be broken into a number of discrete assemblies

Common Language Runtime (CLR)

Physically represented
by a library named
mscoree.dll
(common object runtime execution engine)

- Locate, load and manage .NET types on your behalf
- Take care of memory management
- Perform security checks




The Base Class Library
Data Access
Window Forms
Security
XML/SOAP
Threading
File I/O
Web Forms
……
The Common Language Runtime (CLR)
Common Type System
Common Language Specification

Common type system (CTS)

- A formal specification that documents how types must be defined in order to be hosted by the CLR
- {class, structure, interface, enumeration, delegate}
- Intrinsic CTS data types (mscorlib.dll)

Common language specification

- A set of rules that compiler builder must conform to, if they intend their products to function seamlessly within the .NET universe.
  - Rule: CLS rules apply only to those parts of a type that are exposed outside the defining assembly
  - Using C# compiler to check your code for CLS compliance:
    [assembly: System.CLSCompliant[true]]

```
public class Calc
{
        //Exposed unsigned data is not CLS compliant
        public int Add (ulong x, ulong y)
        { return x+y; }
 }
```

```
public class Calc
{
        public int Add (int x, int y)
        {
            //As this ulong variable is only used internally
            //we are still CLS compliant

            ulong temp;

            ……
            return x+y;
         }
 }
```

Assembly/Namespace/Type Distinction

- A namespace is a group of related types contained in an assembly.
- A single assemble (such as mscorlib.dll) can contain any number of namespaces.
- Any language targeting the .NET runtime makes use of the same namespace and same types.

```
//Hello world in C#

Using System;

public class MyApp
{
        static void Main()
        {
                Console.WriteLine("Hi from C#");
         }
}
```

```
//Hello world in VB .NET

Imports System

Public Module MyApp
        Sub Main()
                Console.WriteLine("Hi from C#");
         End Sub

End Module
```

```
//Hello world in Manage
Extensions for C++

#include "stdafx.h"
using namespace System;

int main(array<System::String ^>
^args)

{
        Console::WriteLine("Hi from
managed C++");

}
```

Referencing External Assemblies

- Need to tell the C# compiler the name of the assembly containing the actual CIL definition for the referenced type.

Documents

- ECMA-334: The C# Language Specification
- ECMA-335: The Common Language Infrastructure (CLI)