.NET Fundamentals

NYU

School of Continuing & Professional Studies

Division of Programs in Business

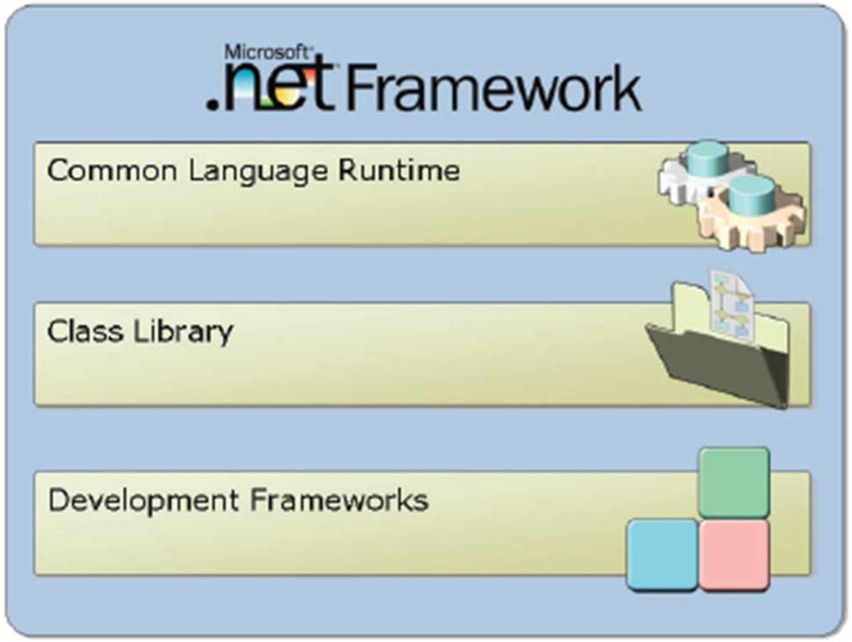
Keith R. Harris (keith9820@hotmail.com)

Agenda

- **Introduction**
  - About me
  - About you
- **Review Syllabus**
- **Session 1**

What is the .NET Framework?

- **Comprehensive development platform**
- **Comprises:**

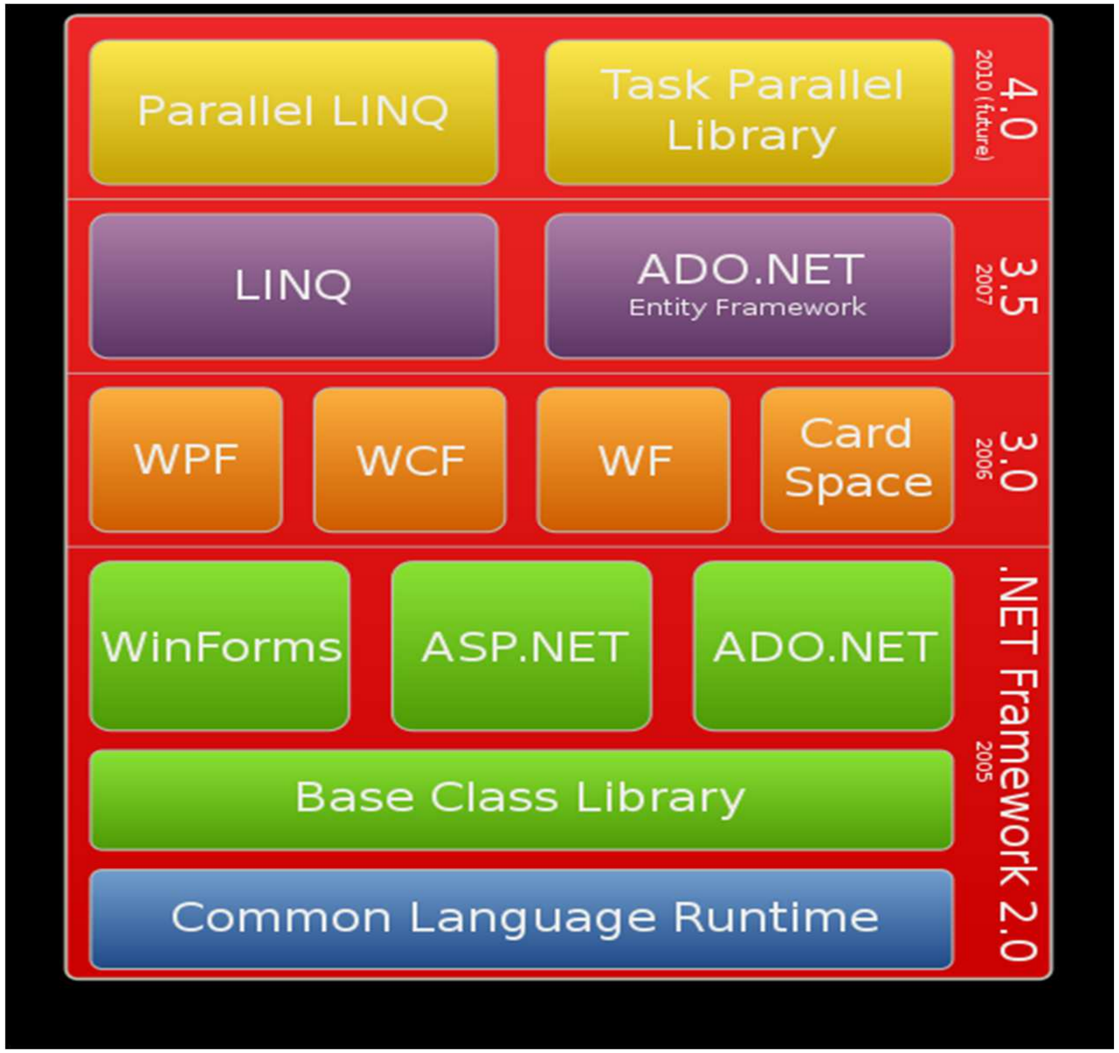


- Manages execution & provides common services such as memory management, transactions, inter-process communications, exception handling, multi-threading, etc…
- Library of reusable classes used to build applications
- Provide the necessary components and infrastructure used to build different types of applications

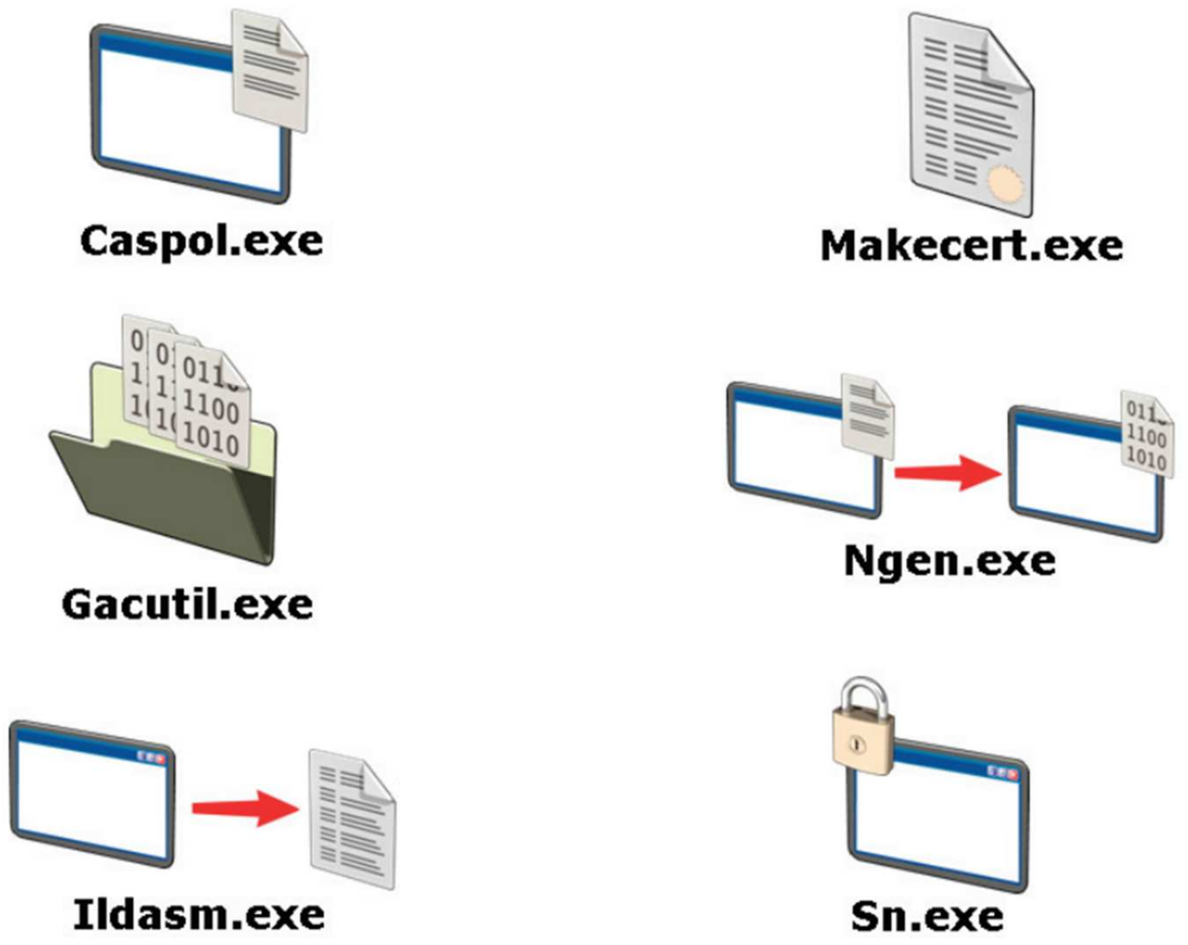
.NET Runtime Versions

Version Name	Release Date
1.0 RTM	01/05/2002
1.0 SP1	03/19/2002
1.0 SP2	07/08/2002
1.0 SP3	08/31/2004
1.1 RTM	04/01/2003
1.1 SP1	08/30/2004
1.1 SP1 (Windows Server 2003 Version)	03/30/2005
2.0 RTM	11/07/2005
3.0 RTM	11/06/2006
3.0 RTM (Vista)	01/30/2007
3.0 SP1	11/19/2007
3.5 RTM	11/19/2007
4.0 RTM	04/12/2010

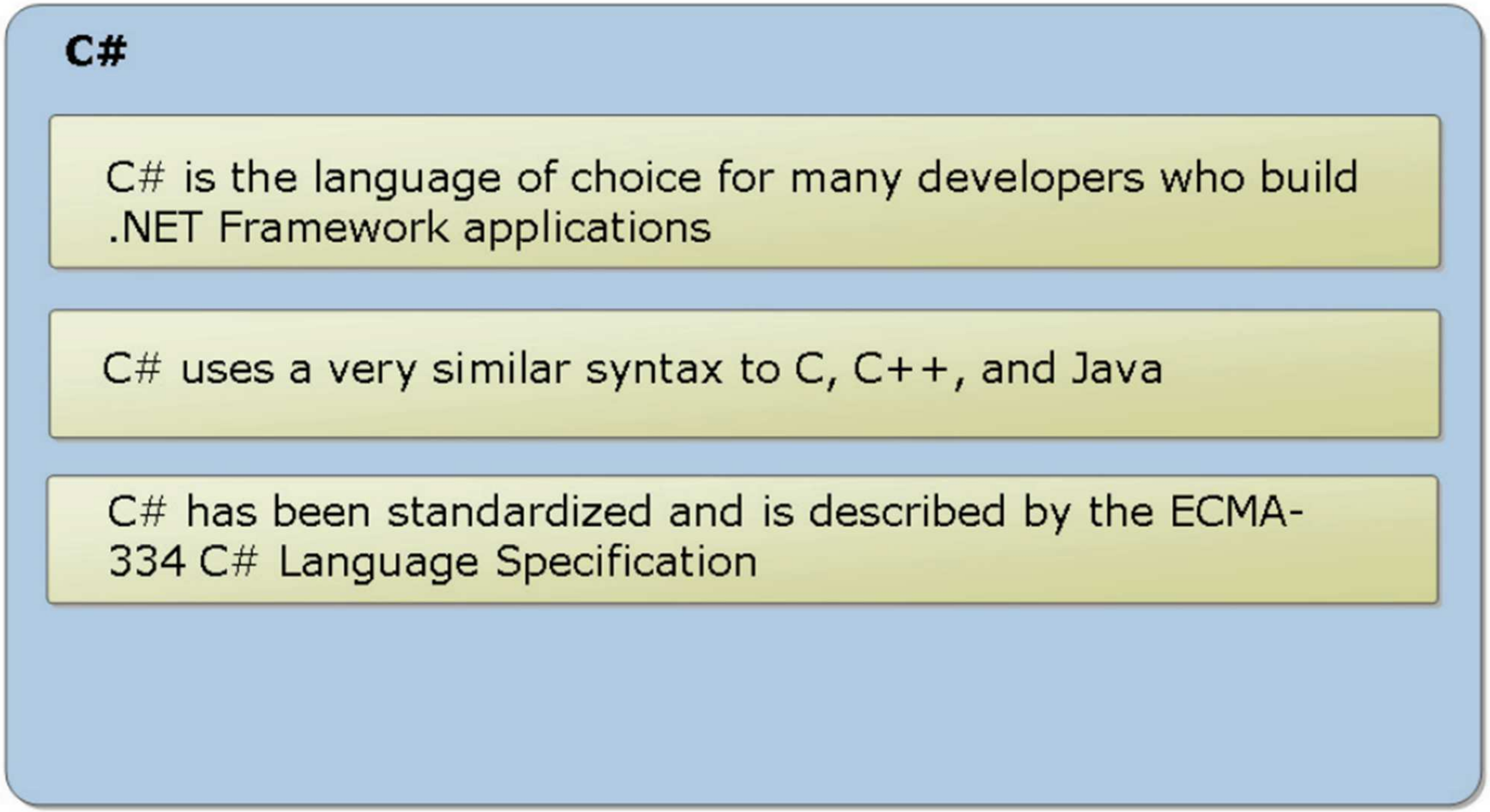
.NET Framework Stack



.NET Framework Tools



Why C#?



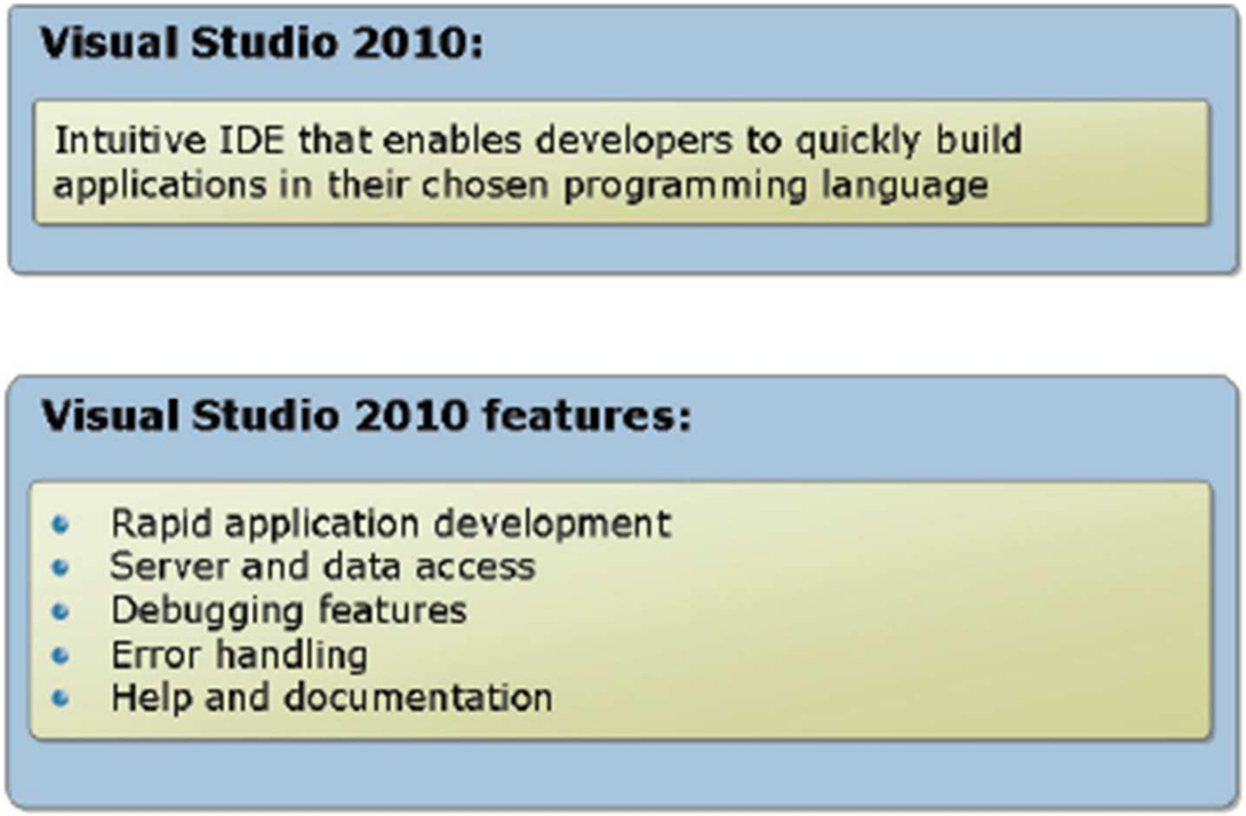
C#

C# is the language of choice for many developers who build .NET Framework applications

C# uses a very similar syntax to C, C++, and Java

C# has been standardized and is described by the ECMA-334 C# Language Specification

Key Features of Visual Studio



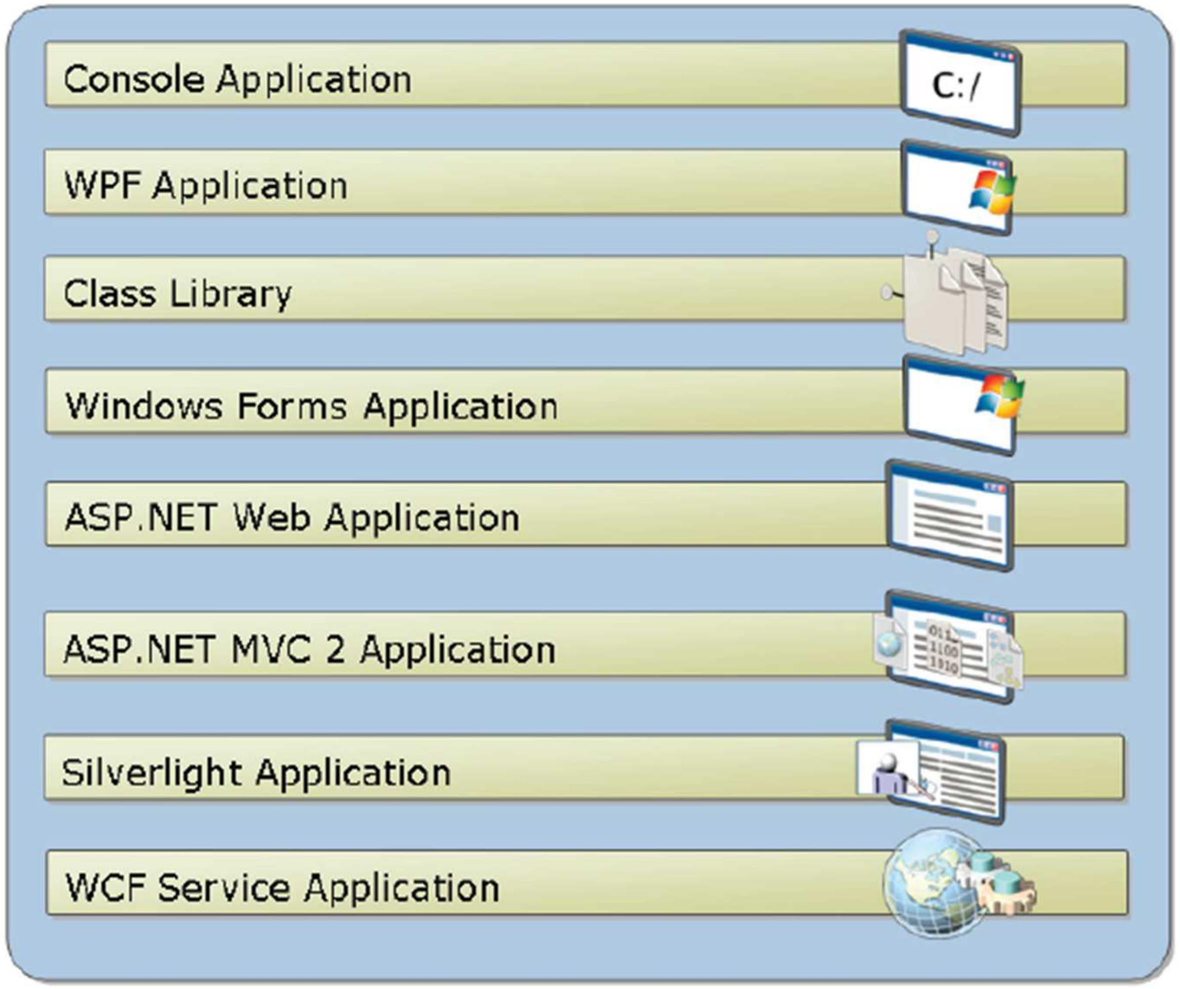
Visual Studio 2010:

Intuitive IDE that enables developers to quickly build applications in their chosen programming language

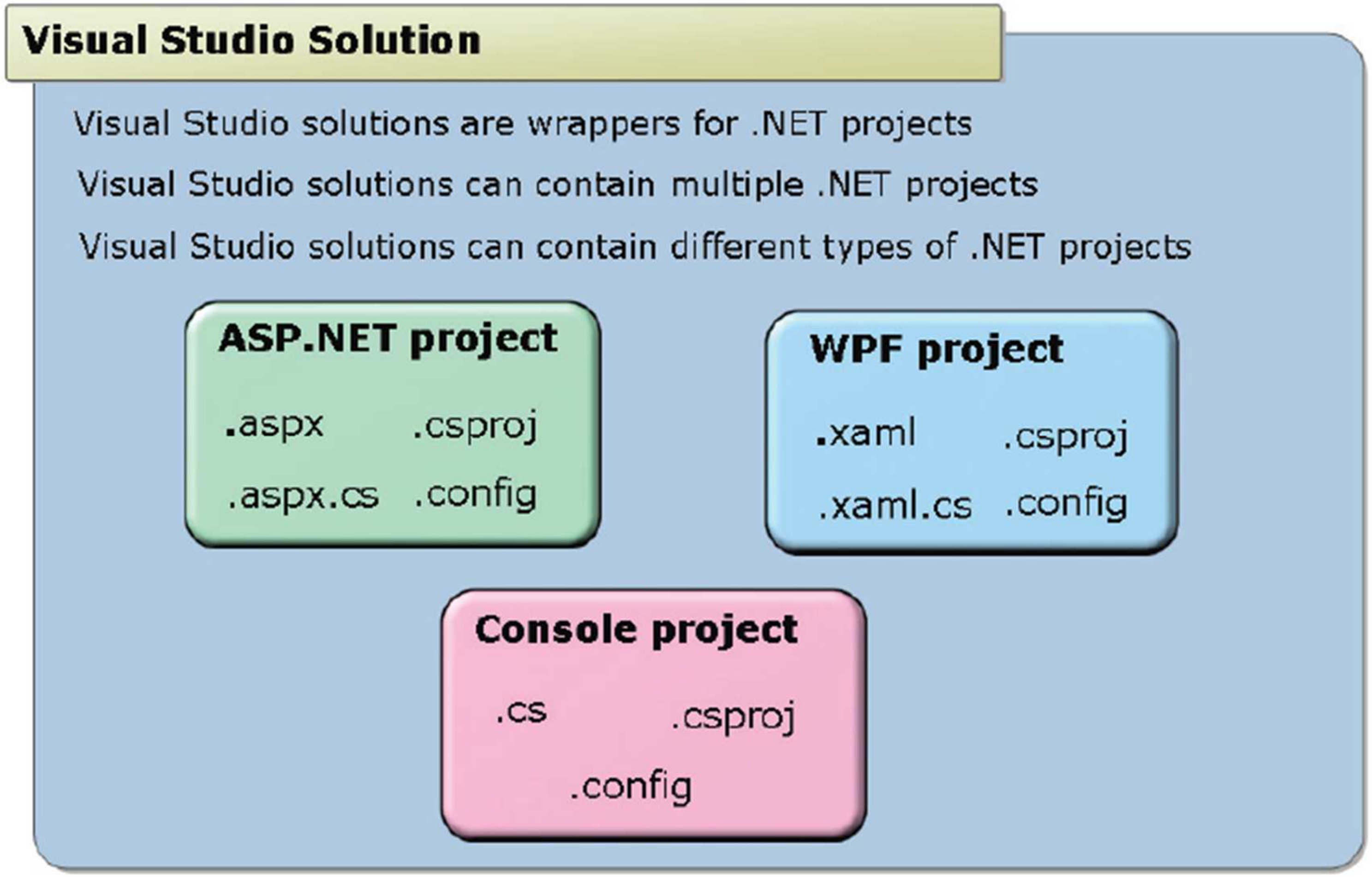
Visual Studio 2010 features:

- Rapid application development
- Server and data access
- Debugging features
- Error handling
- Help and documentation

Template in Visual Studio



Structure of VS Projects and Solutions


Visual Studio Solution
Visual Studio solutions are wrappers for .NET projects
Visual Studio solutions can contain multiple .NET projects
Visual Studio solutions can contain different types of .NET projects
ASP.NET project
.aspx
.csproj
.aspx.cs
.config
WPF project
.xaml
.csproj
.xaml.cs
.config
Console project
.cs
.csproj
.config

Terms

- Types (intrinsic or user-defined)
- Namespace
- Assembly

Types (Intrinsic)

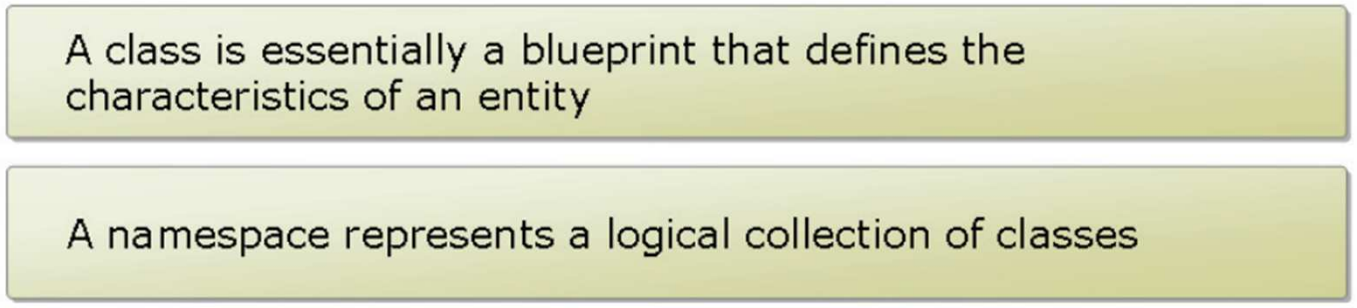
- **Included in .NET:**
  - Int32
  - Int64 (5231)
  - Bool (true / false)
  - String (“Hello”)
  - DateTime (10/4/2011 1:20 PM EST)
  - Decimal (1.234)
  - Single (1.234)
  - …

Types (your own, a.k.a. Class)

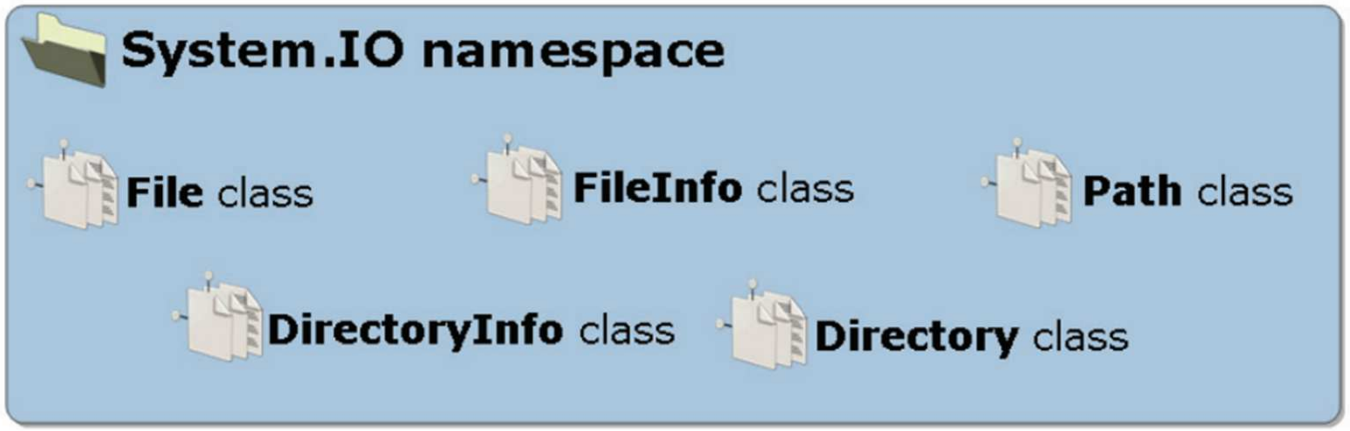
- **User-Defined (Custom) Type**
- **Software model**
- **Contains Members**
  - Fields (private variables)
  - Methods (functions)
    - Properties
    - Constructors
  - Events

Namespace

- Organizes code
- Group type names, reducing chance of collision


A class is essentially a blueprint that defines the characteristics of an entity

A namespace represents a logical collection of classes



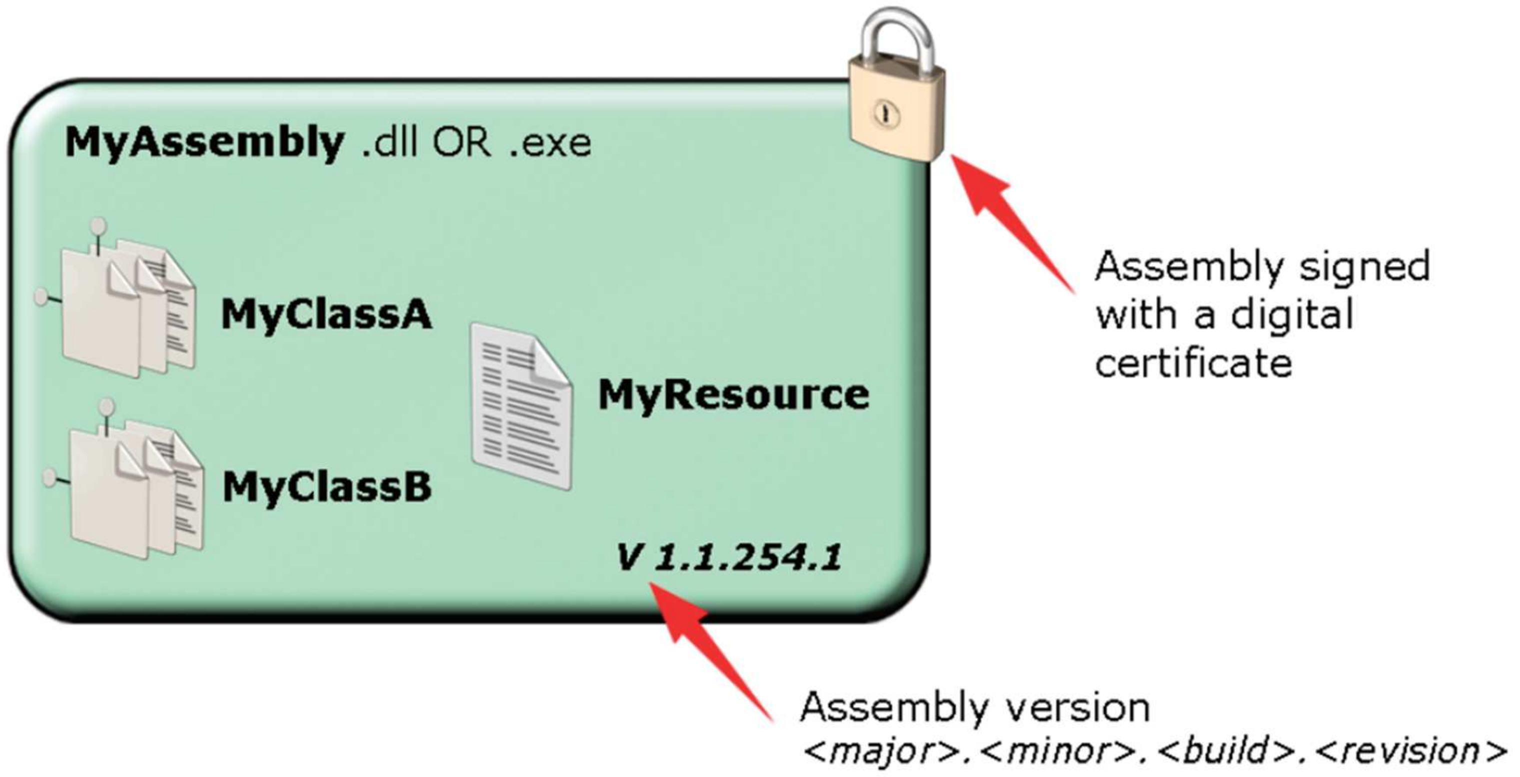
Declaring a namespace

```
namespace Session1.Demo1.NYU
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Assembly

Building blocks of .NET Framework applications

Collection of types and resources that form a logical unit of functionality


MyAssembly .dll OR .exe
MyClassA
MyClassB
MyResource
V 1.1.254.1
Assembly signed with a digital certificate
Assembly version
<major>.<minor>.<build>.<revision>

Types of Projects

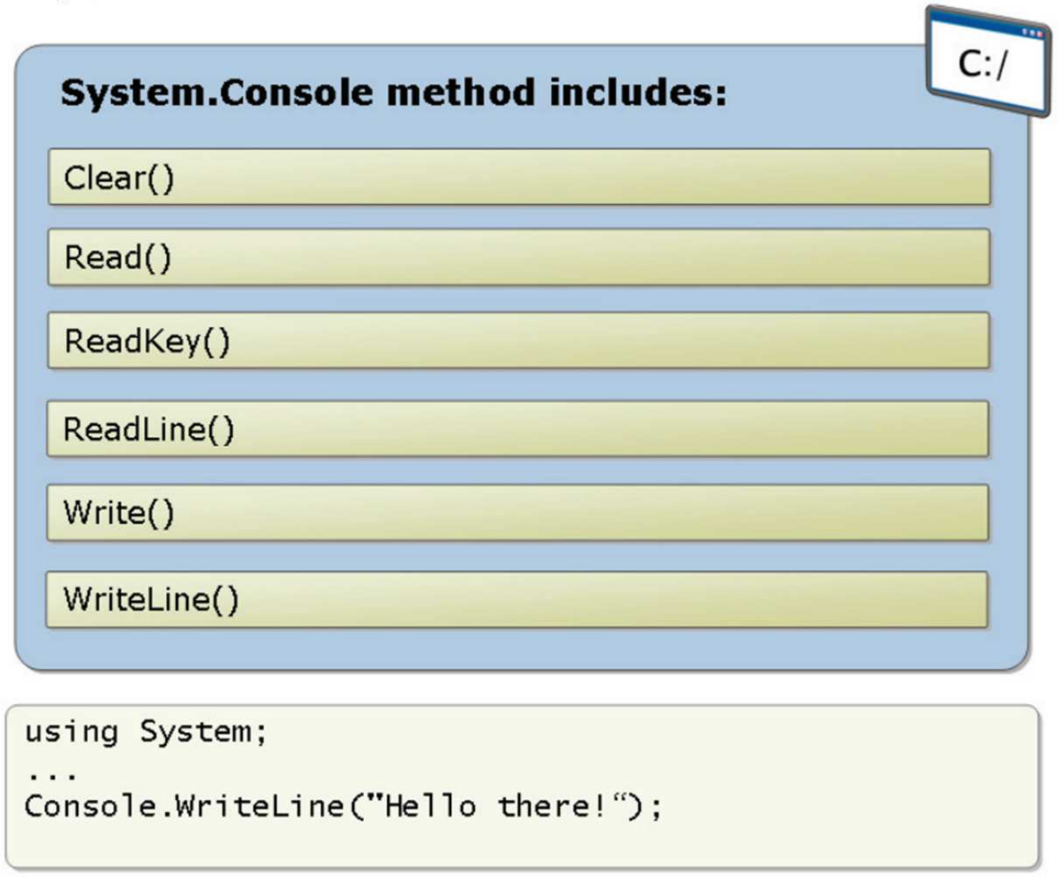
- **Console Application – DOS style program**
- **Windows Application – GUI style program**
- **Windows Service – background process**
- **Web Site / Web Service –**
  - Not standalone applications, run within host executable
  - Require web server (IIS)
- **Library –**
  - Usually contain data structures and business logic
  - not executable, referenced by other projects
  - .dll extension

Console Projects

- “DOS” applications
- No GUI
- Have .exe file extension
- Run from the command line or called from batch files
- Accept input from standard input (keyboard, no mouse)
- Usually write to standard output (screen)

Console Class

- **Represents the standard input and output for console applications**

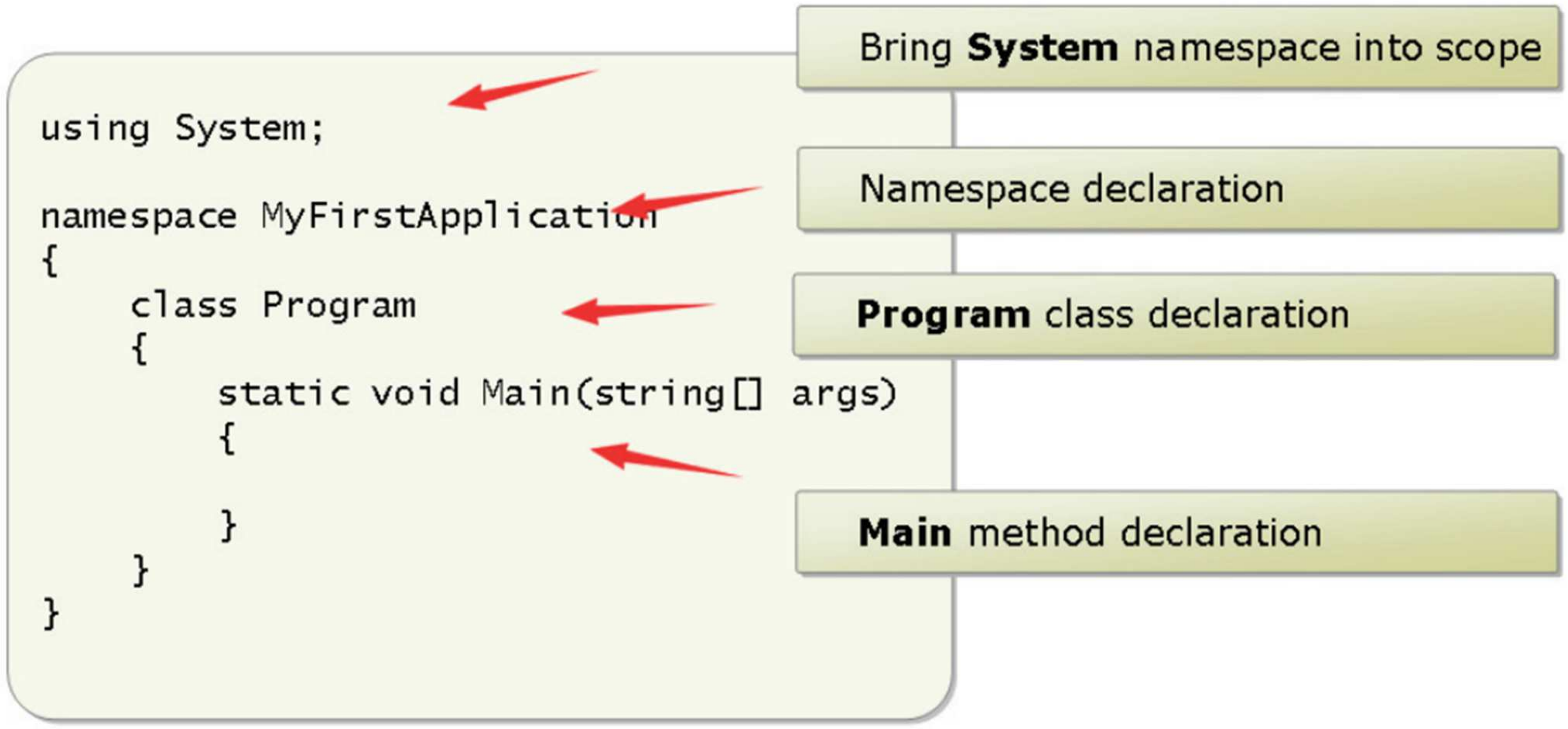


System.Console method includes:

- Clear()
- Read()
- ReadKey()
- ReadLine()
- Write()
- WriteLine()

```
using System;
...
Console.WriteLine("Hello there!“);
```

Structure of a C# Program



```
using System;

namespace MyFirstApplication
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Bring **System** namespace into scope
Namespace declaration
Program class declaration
Main method declaration

Compiling Code

Visual Studio

1 In Visual Studio 2010, on the **Build** menu, click **Build Solution**

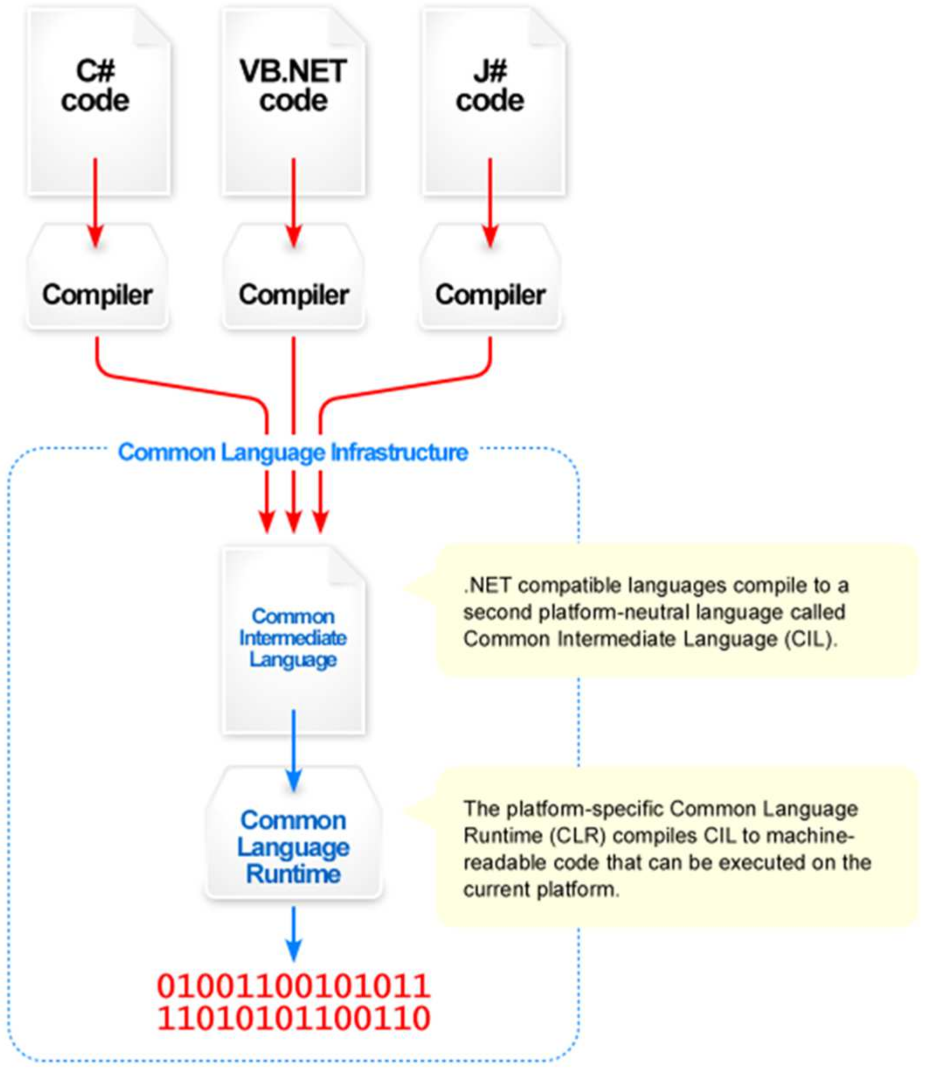
2 On the **Debug** menu, click **Start Debugging**

Command line

C:/

```
csc.exe /t:exe /out:" C:\Users\Student\Documents\Visual Studio
2010\MyProject\myApplication.exe"
"C:\Users\Student\Documents\Visual Studio 2010\MyProject\*.cs"
```

Compiling Code



.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).

The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.

Hello World (CIL)

```
.method public static void Main() cil managed
{
   .entrypoint
   .maxstack 1
   ldstr "Hello, world!"
   call void
   [mscorlib]System.Console::WriteLine(string) ret
}
```

How the CLR Executes Code

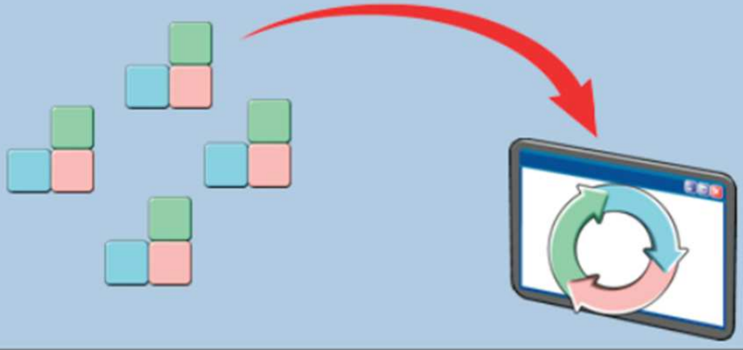
Assemblies contain MSIL code, which is not actually executable

The CLR loads the MSIL code from an assembly and converts it into the machine code that the computer requires

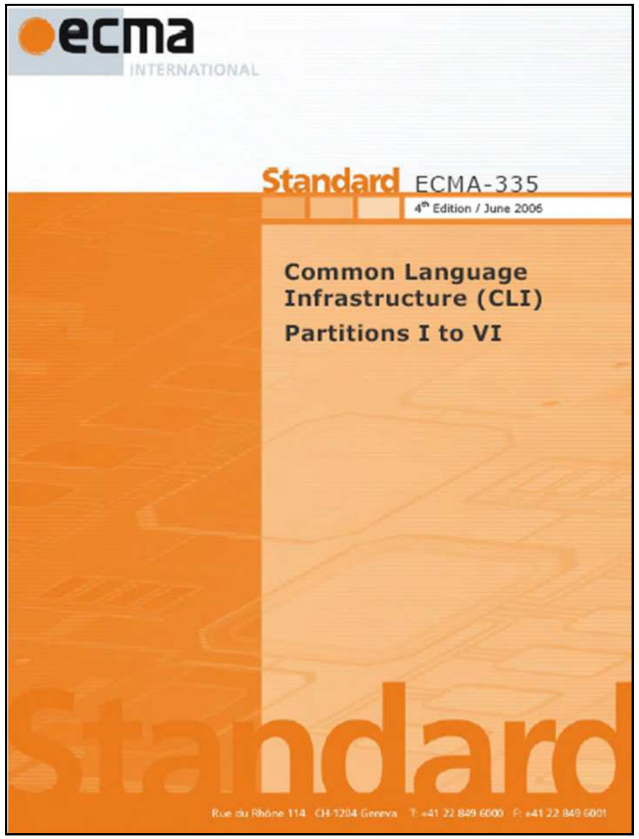
1 Loads assemblies that the application references

2 Verifies and compiles assemblies into machine code

3 Runs the executable assembly



The CLR Standard


ecma INTERNATIONAL

Standard ECMA-335

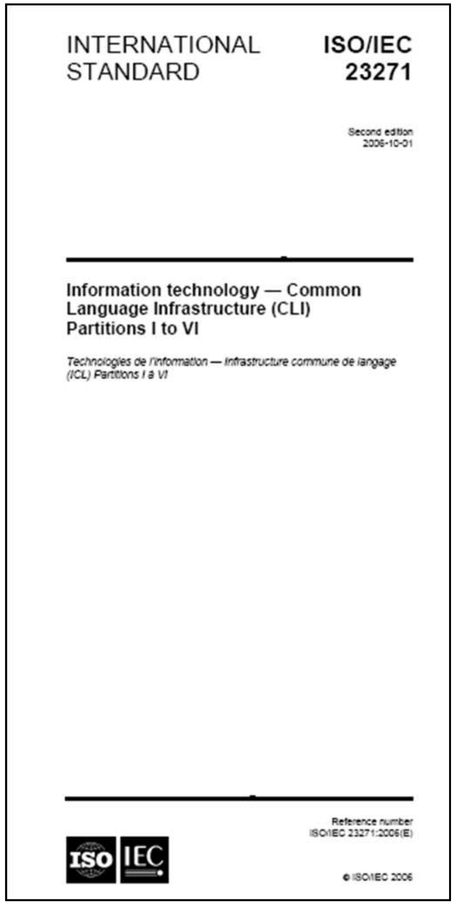
4th Edition / June 2006

Common Language Infrastructure (CLI)
Partitions I to VI

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

12/2001


INTERNATIONAL STANDARD

ISO/IEC 23271

Second edition 2006-10-01

Information technology — Common Language Infrastructure (CLI) Partitions I to VI

Technologies de l'information — Infrastructure commune de langage (ICL) Partitions I à VI

Reference number ISO/IEC 23271:2006(E)

ISO IEC

© ISO/IEC 2006

04/2003

CLI
Common Language Infrastructure

- **Open specification developed by MS**
- **Describes the executable code and runtime environment that form the core and the .NET framework**
- **Also describes:**
  - Common Type System (CTS)
  - Metadata
  - Common Language Specification (CLS)
  - Virtual Execution System (VES)

CLI Implementaions

- **Microsoft:**
  - Shared Source CLI (formerly Rotor)
  - .NET Framework
  - .NET Compact Framework
- **Others:**
  - Mono development platform (open source project)
  - Portable .NET (dotGNU project)

CLR
Common Language Runtime

- **Microsoft’s implementation of CLI standard**
- **Virtual machine component of .NET**
- **Compiles bytecodes to machine instructions**
- **Provides run-time services such as:**
  - Memory management
  - Thread management
  - Exception handling
  - Garbage collection
  - Security

Metadata

- Information about compiled types
- Similar to COM type library
- Enables applications to discover interfaces, classes, types, methods and fields in assembly
- Read using reflection

CIL
Common Intermediate Language

- **IL formerly MSIL**
- **Bytecodes**
- **Machine independent**
- **Executed by a VES**

CTS
Common Type System

- **.NET languages must abide**
- **Allows interoperability among languages**
- **Concerns:**
  - types

CLS
Common Language Specification

- **Enables interoperability between languages**
- **Concerns:**
  - Inheritance
  - Polymorphism
  - Exception handling

FCL
Framework Class Library

- Available to all languages
- Extends the BCL (System namespace)

JIT
Just In Time compilation

- Aka Dynamic Translation
- Technique for improving the runtime performance of a program
- Converts IL