Visual C# .Net using framework 4.5

Eng. Mahmoud Ouf

Lecture 01

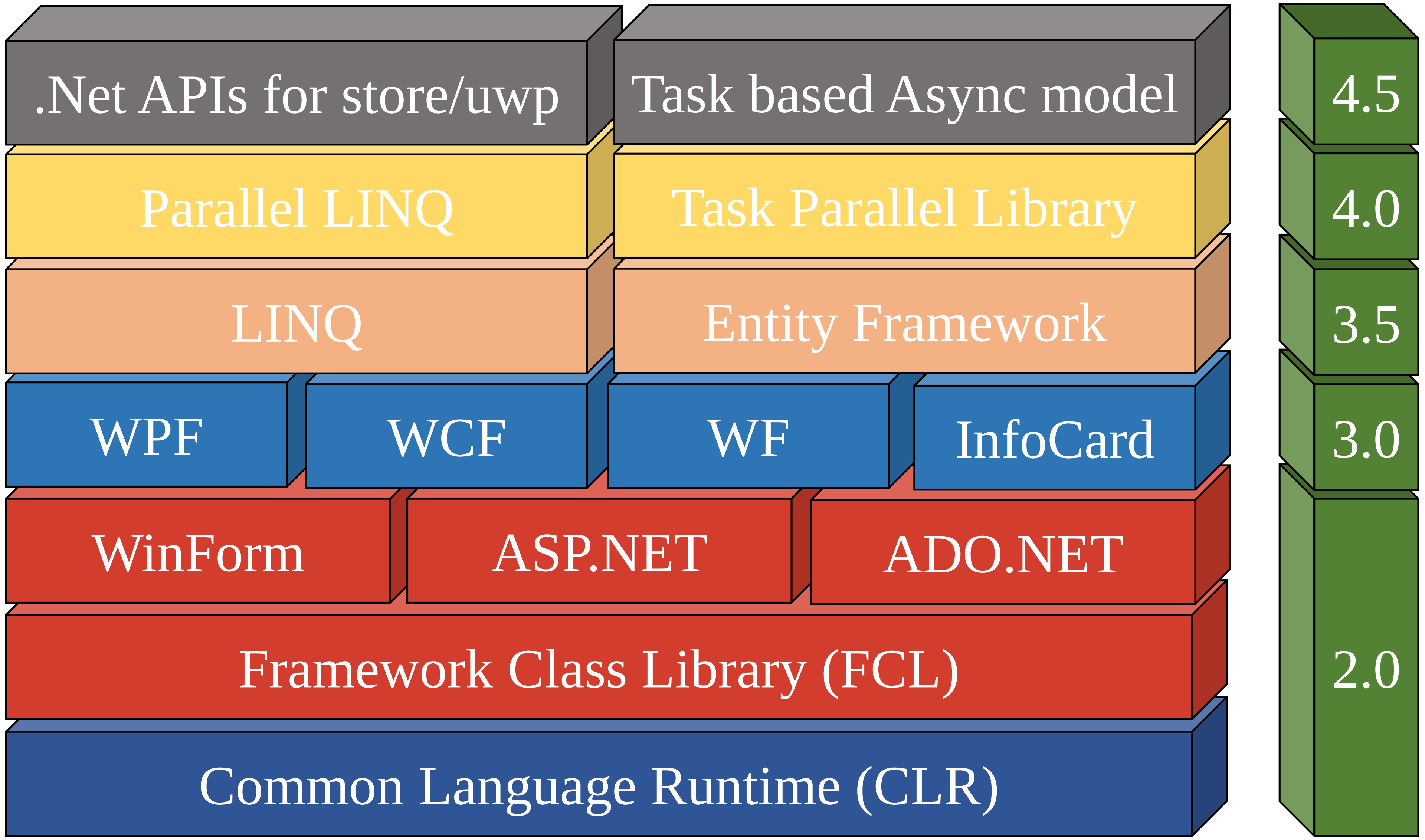
Introduction to .Net Framework

- The development of .Net Framework started the late 1990.
- First version was released early 2002.
- Through the 15 years it passes 8 upgrades
- Some of these upgrades was released with new version of Visual Studio.Net.

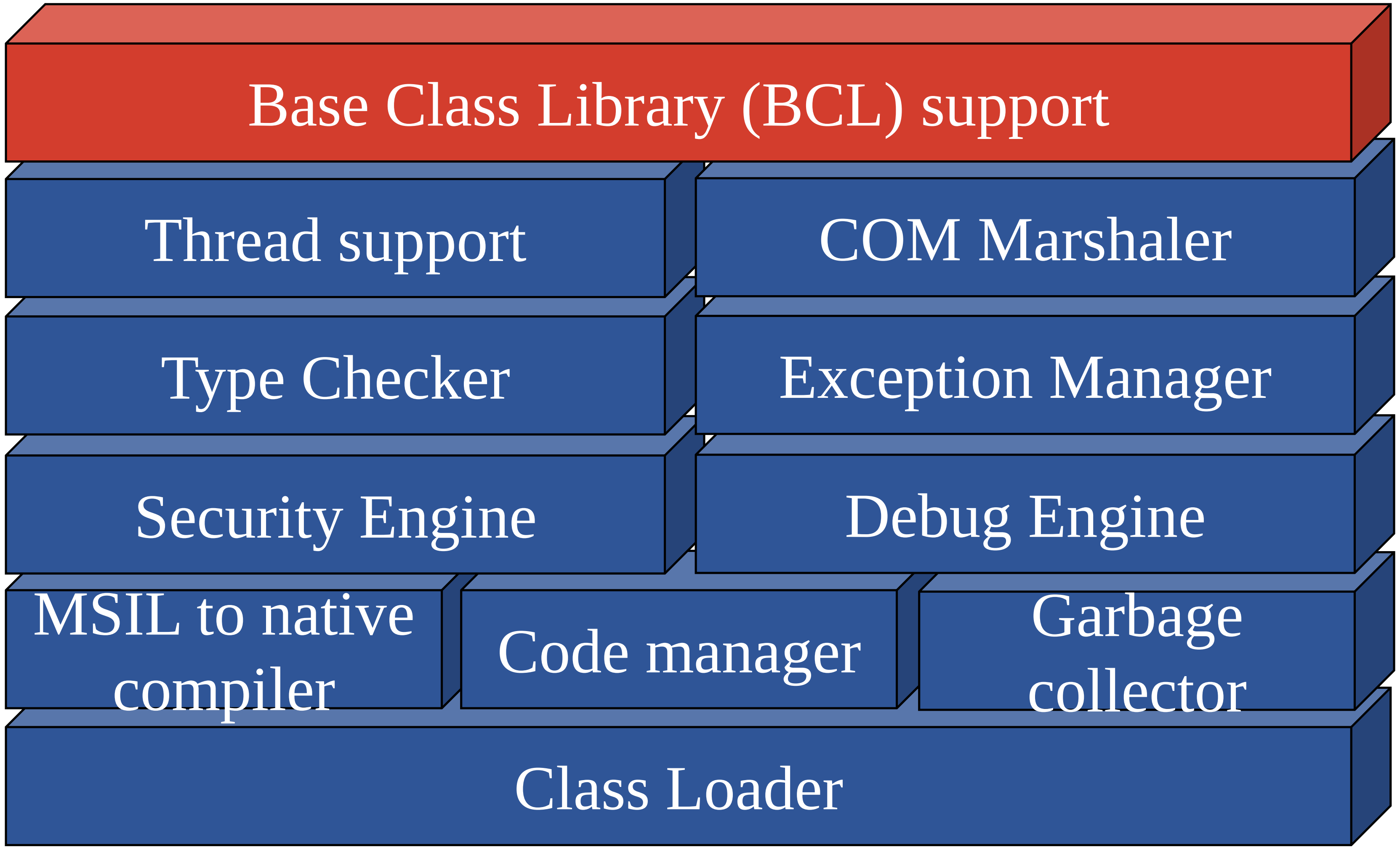
Introduction to .Net Framework

Framework version	Release date	Development tool
1.0	13/02/2002	Visual Studio .Net 2002
1.1	24/04/2003	Visual Studio .Net 2003
2.0	07/11/2005	Visual Studio .Net 2005
3.0	06/11/2006	Visual Studio .Net 2005
3.5	19/11/2007	Visual Studio .Net 2008
4.0	12/04/2010	Visual Studio .Net 2010
4.5	15/08/2012	Visual Studio .Net 2012
4.6	20/07/2015	Visual Studio .Net 2015

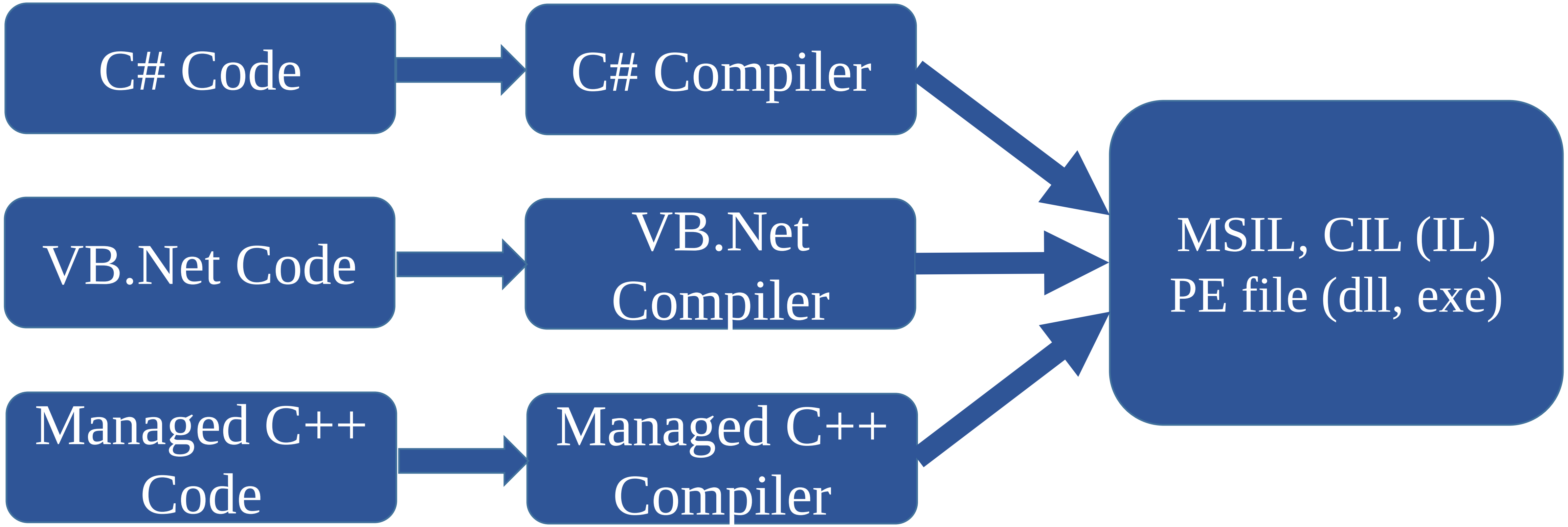
Overview of .Net Framework



Common Language Runtime (CLR)



From source code to executable



.Net Framework 1.1

1. Built in support for mobile ASP.Net controls
2. Enables Code Access Security in ASP.Net application
3. Built-in support for ODBC and Oracle Database
4. .Net Compact Framework
5. Support Internet Protocol version 6 (IPv6)

.Net Framework 2.0

1. Full 64-bit support
2. Numerous API changes
3. Microsoft SQL Server integration
4. Additional and improved ASP.Net web controls
5. New personalization features for ASP.Net
6. Partial classes
7. Nullable types
8. Anonymous methods

.Net Framework 3.0

1. Windows Presentation Foundation (WPF)
2. Windows Communication Foundation (WCF)
3. Windows Workflow Foundation (WF)
4. Windows CardSpace

.Net Framework 4.0

1. Parallel Extension to improve support of parallel programming
2. New Visual basic and C# features
3. Include new types
4. Introduced Common Language Runtime (CLR) 4.0

.Net Framework 4.5

1. .Net for Metro Style apps
2. Managed Extensibility Framework (MEF)
3. Core Features
4. ASP .Net
5. Networking

.Net Framework 4.6

1. Just In Time (jit) compiler for 64 bit system
2. WPF and Windows Forms both have been updated for high DPI scenarios
3. Support for TLS 1.1 and TLS 1.2 has been added to WCF ASP .Net
4. The cryptographic API in .NET Framework 4.6 uses the latest version of Windows CNG cryptography API.

Structure of C# program

- In C#, an application is a collection of one or more classes.
- The classes for a C# application can be written in more than one file and multiple classes can be put in one file. One class could be written in multiple files (partial class).

```
class HelloWorld
{
        public static void Main()
        {
                System.Console.WriteLine(“Hello, World”);
        }
}
```

Structure of C# program (cont.)

- The entry point to a C# application is the Main() method, which must be: contained in a class, begin with capital M, and public static.
- public: modifier tells us that the method is accessible by everyone.
- static: means that the method can be called without creating an instance of the class.
- The .Net Framework is made up of many namespaces, the most important of which is called **System**; it contains the classes that most application uses for interacting with the operating System.
- We can refer to objects in namespace by:

1. prefixing them explicitly with the identifier of namespace
   System.Console.WriteLine(“Hello, World”);
2. specifing a namespace by placing a using directive at the beginning of the application before the first class is defined
   using System

Basic Input / Output operation

- **Output:**

Use the following 2 methods for output:

Console.Write()

Console.WriteLine()

The difference is that WriteLine() append a new line/carriage return to the end of the output.

To print a constant value:

```
Console.WriteLine(99);
Console.WriteLine("Hello, World");
```

To print a variable value:

```
int x = 99;
Console.WriteLine(x);
```

To print a combination from variable and constant value:

```
Console.WriteLine("The Sum of {0} and {1} is {2}", x, x, x+x);
```

Basic Input / Output operation

- **Input:**

Use the following 2 methods for iutput:
Console.Read(), which read single character and return its ASCII
Console.ReadLine(), which read a string

```
string input = Console.ReadLine();
```

To read an integer, use the Parse:

```
string s = Console.ReadLine();
int n = int.Parse(s);
```