Programming in C#

Dr.M.Paul Arokiadass Jerald

Unit 1- Lecture 1, 2

4-11-2020, 5-11-2020

Syllabus Overview

- **UNIT - I: INTRODUCTION TO C#**

  Introduction to .NET – Features of C# - Data Types – Value Types – Reference Types - Variables and Constants – Declaring – Assigning values – variables of nullable types – Operators – Type Conversions – Implicit and Explicit Type Conversions – Arrays – Single Dimensional and Multidimensional – Control Flow Statements – Selection – Iteration and Jump – Classes and Objects – Access Modifiers – Defining a Class – Variables – Properties and Methods – Creating Objects – Inheritance – Polymorphism- Constructor and Destructors.

Syllabus Overview

- **UNIT - II: WINDOWS FORMS**

  Windows Forms – Form Class – Common Operations on Forms – Creating a Message Box –Handling Events – Mouse Events – Keyboard Events – Common Controls in Windows Forms – Label – TextBox – Button – Combo Box – List Box – Check Box – Radio Button – Group Box – Picture Box – Timer – Open File Dialog – Save File Dialog – Font Dialog – Color Dialog – Print Dialog – Tree View – Menu.

Syllabus Overview

- **UNIT - III: DELEGATES AND EVENTS**

  Delegates – Declaring a Delegate – Defining Delegate Methods – Creating and Invoking Delegate Objects – Multicasting with Delegates – Events – Event Sources – Event Handlers – Events and Delegates.

Syllabus Overview

- **UNIT - V: DATABASE**

  Creating Connection String – Creating a Connection to a Database – Creating a Command Object – Working with Data Adapters – Using Data Reader to work with Databases – Using Dataset.

Syllabus Overview

- **TEXT BOOKS**

1. Vikas Gupta , “Comdex .NET Programming “ , Dream Tech Press, New Delhi, 2011
2. Kogent Solutions, “ C# 2008 Programming Black Book”, Dream Tech Press, New Delhi, Platinum Edition, 2009

Course Requirements

- Requirements for the course
  - Complete 3 assignments
  - Complete 3 Tests
- Intimation of Classes
  - Through WhatsApp.
- Assignments are CIA Test
  - Intimation through Gmail registered in Google Classroom
  - Submission of Assignment through Google Classroom

1. History of C#

- C# was developed by Microsoft within its .NET framework initiative.
- C# programming language is a general-purpose, OOPS based programming language.
- C# development team was lead by "Anders Hejlsberg" in 2002.

History of C#

- Based on Java and C++, but has many additional extensions.
- Java and C# are both being updated to keep up with each other.
- Cross-development with Visual Basic, Visual C++, F#, Python, and many other .NET languages.
  - See: http://en.wikipedia.org/wiki/List_of_CLI_languages

2. Features of C#

- C# is Object-oriented.
- Primarily imperative or procedural.
  - LINQ adds some functional programming language capabilities.
- Structured (as opposed to monolithic).
- Strongly typed.
- ISO and ECMA standardized.

3. C# Version History

Version	.NET Framework	Visual Studio	Important Features
C# 1.0	.NET Framework 1.0/1.1	Visual Studio .NET 2002	• Basic features
C# 2.0	.NET Framework 2.0	Visual Studio 2005	• Generics • Partial types • Anonymous methods • Iterators • Nullable types • Private setters (properties) • Method group conversions • Static classes
C# 3.0	.NET Framework 3.0\3.5	Visual Studio 2008	• Implicitly typed local variables • Object and collection initializers • Auto-Implemented properties • Extension methods • Query expressions
C# 4.0	.NET Framework 4.0	Visual Studio 2010	• Dynamic binding (late binding) • Named and optional arguments • Embedded interop types

C# Version History

Version	.NET Framework	Visual Studio	Important Features
C# 5.0	**.NET Framework 4.5**	Visual Studio 2012/2013	• Async features • Caller information
C# 6.0	**.NET Framework 4.6**	Visual Studio 2013/2015	• Expression Bodied Methods • Auto-property initializer • nameof Expression • Primary constructor • Exception Filter • String Interpolation
C# 7.0	**.NET Core 2.0**	Visual Studio 2017	• Tuples • Pattern Matching • Local functions • Generalized async return types
C# 8.0	**.NET Core 3.0**	Visual Studio 2019	• Readonly members • Default interface methods • Using declarations • Static local functions

4. Advantages of using C#

- Interoperability
  - “Interop” process enables C# programs to do almost anything that a native C++ application can do.
- Ease of Use
  - Syntax allows for users familiar with C, C++, or Java to easily start coding in C# very effortlessly.

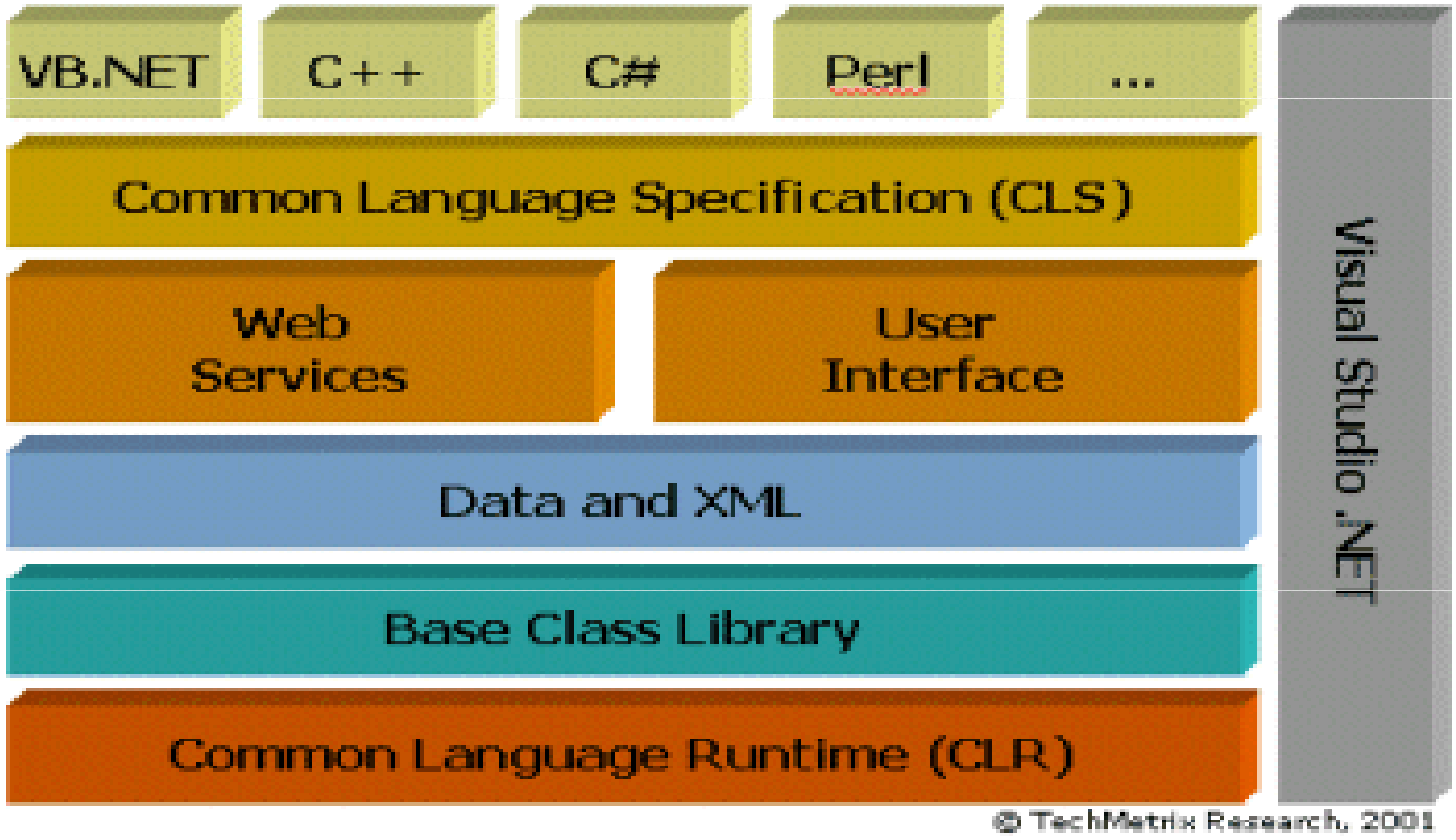
Advantages (contd)

- Reliability
  - Progression of versions gives the user the feeling of reliable mature standard.
- Support of Community
  - It approval from the ISO and ECMA as well as development support from Microsoft give the standard elite standing.

Disadvantages

- Microsoft uses C# in its Base Class Library (BCL) which is the foundation of its proprietary .NET framework.
  - Proprietary features may deter other idependent implementations of the full framework.
- Monetary concerns.

5. Microsoft's .NET Technologies



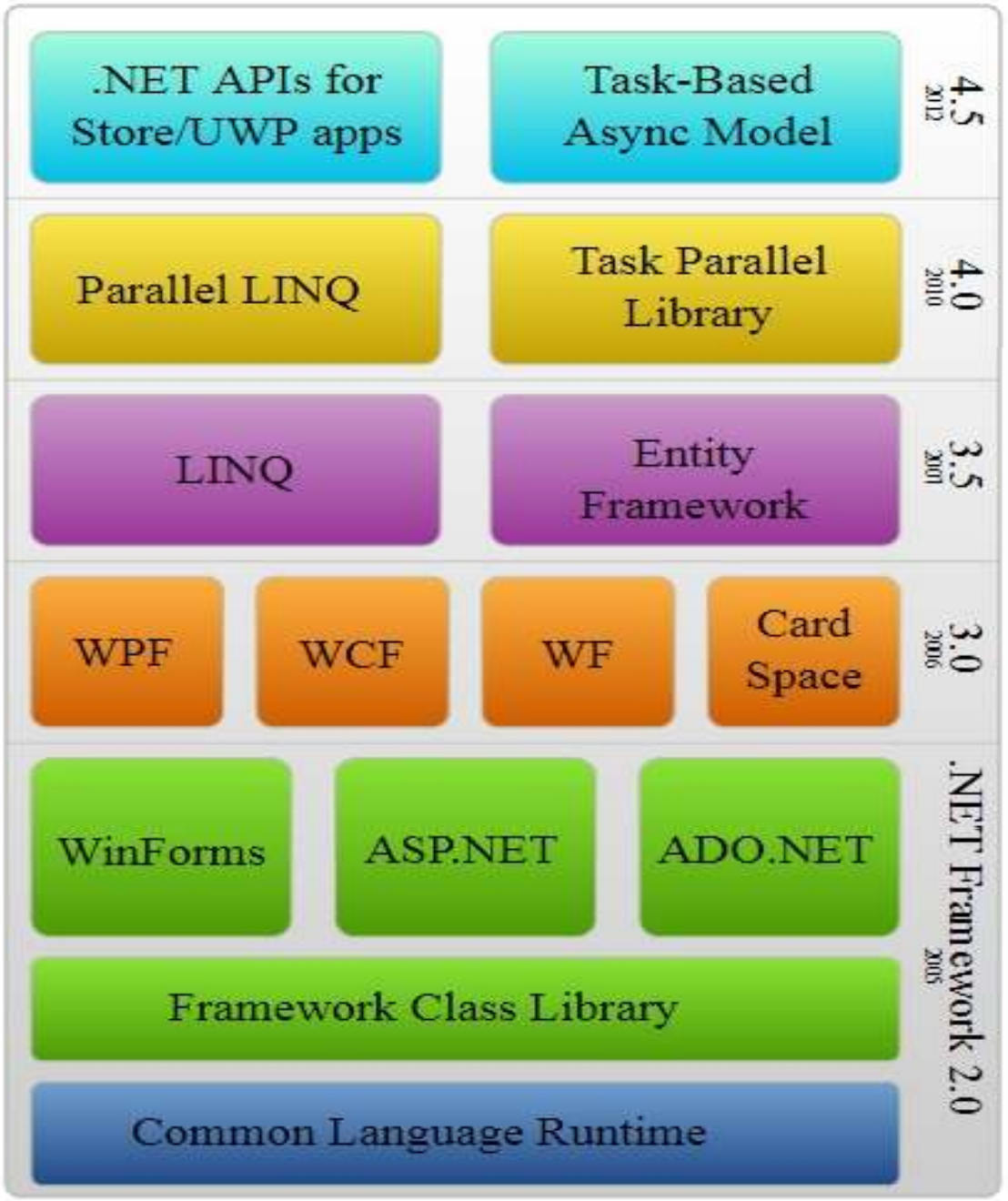
The Class Libraries

- The common classes that are used in many programs
  - System.Console.WriteLine
  - XML, Networking, Filesystem, Crypto, containers
  - Can inherit from many of these classes
- Many languages run on .NET framework
  - C#, C++, J#, Visual Basic
  - even have Python (see IronPython)

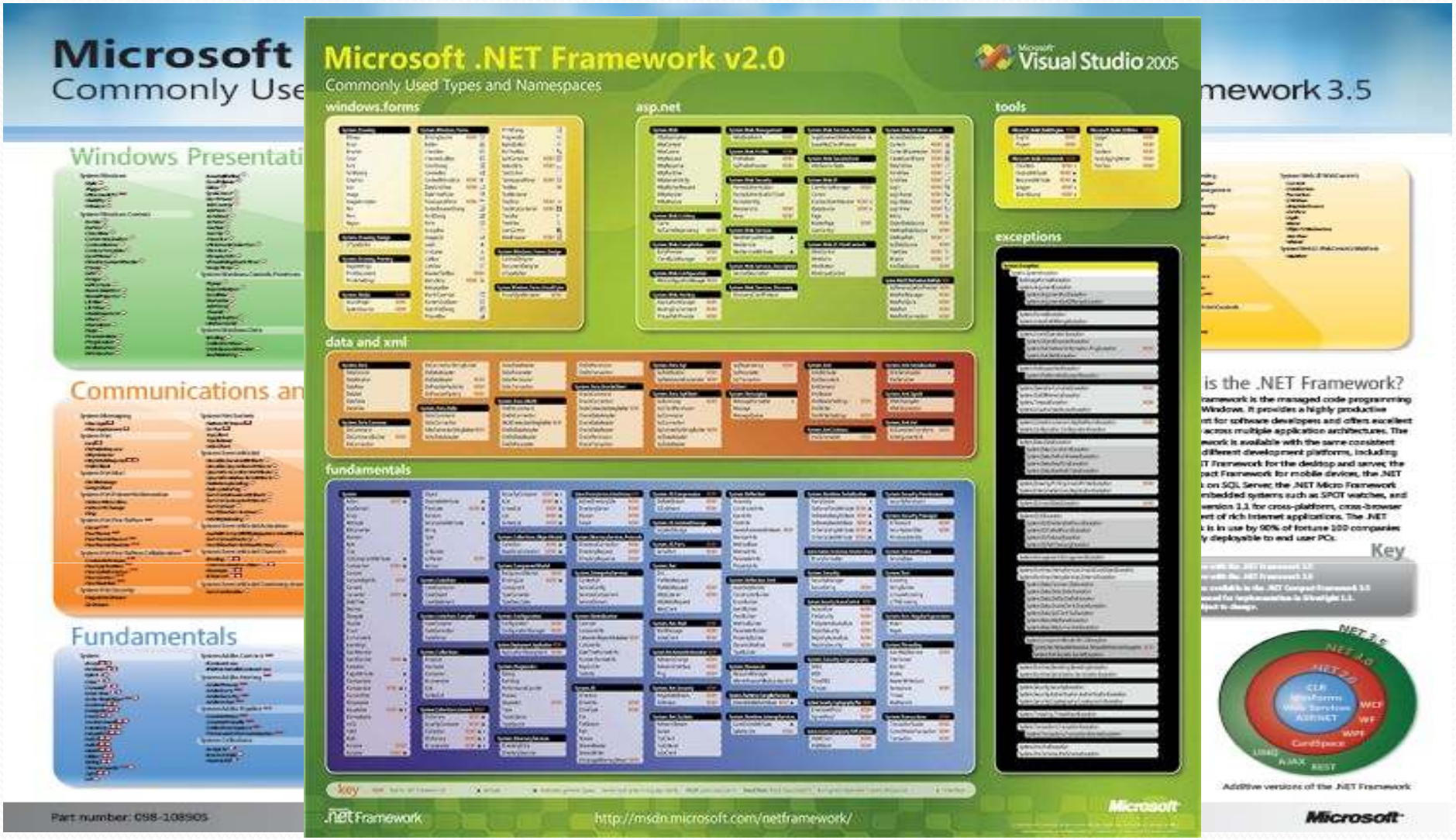
.NET History

Reference :
https://en.wikipedia.org/wiki/.NET_Framework_version_history

Enjoy the Video :
https://youtu.be/FFCn_z7dn_A

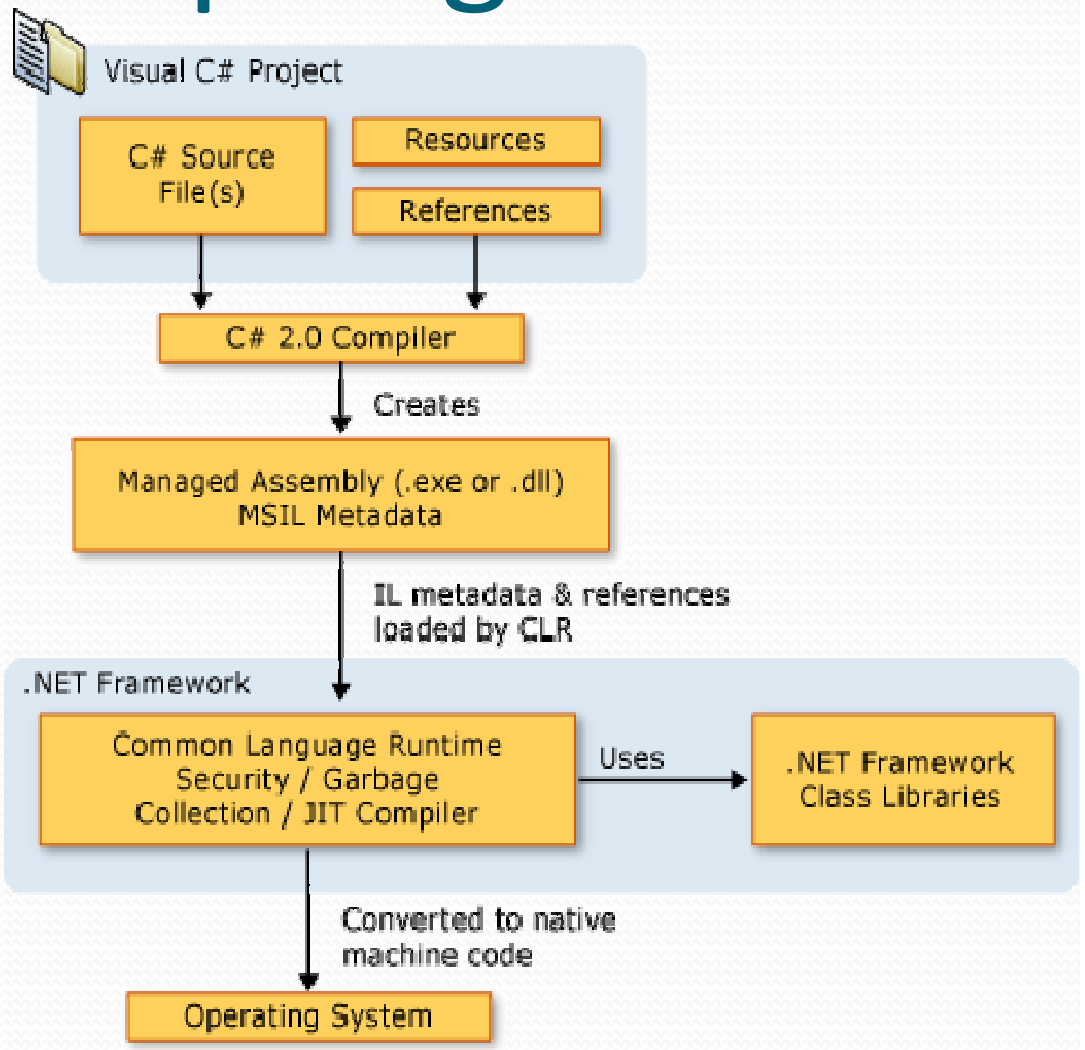


The Class Libraries



6. CLR and JIT compiling.

- C#, like Java, is executed indirectly through an abstract computer architecture called the CLR.
  - CLR => Common Language Runtime.
  - Abstract, but well defined.
- C# programs are compiled to an IL.
  - Also called MSIL, CIL (Common Intermediate Language) or bytecode.



http://msdn2.microsoft.com/en-us/library/z1zx9t92(VS.80).aspx

CLR and JIT compiling.

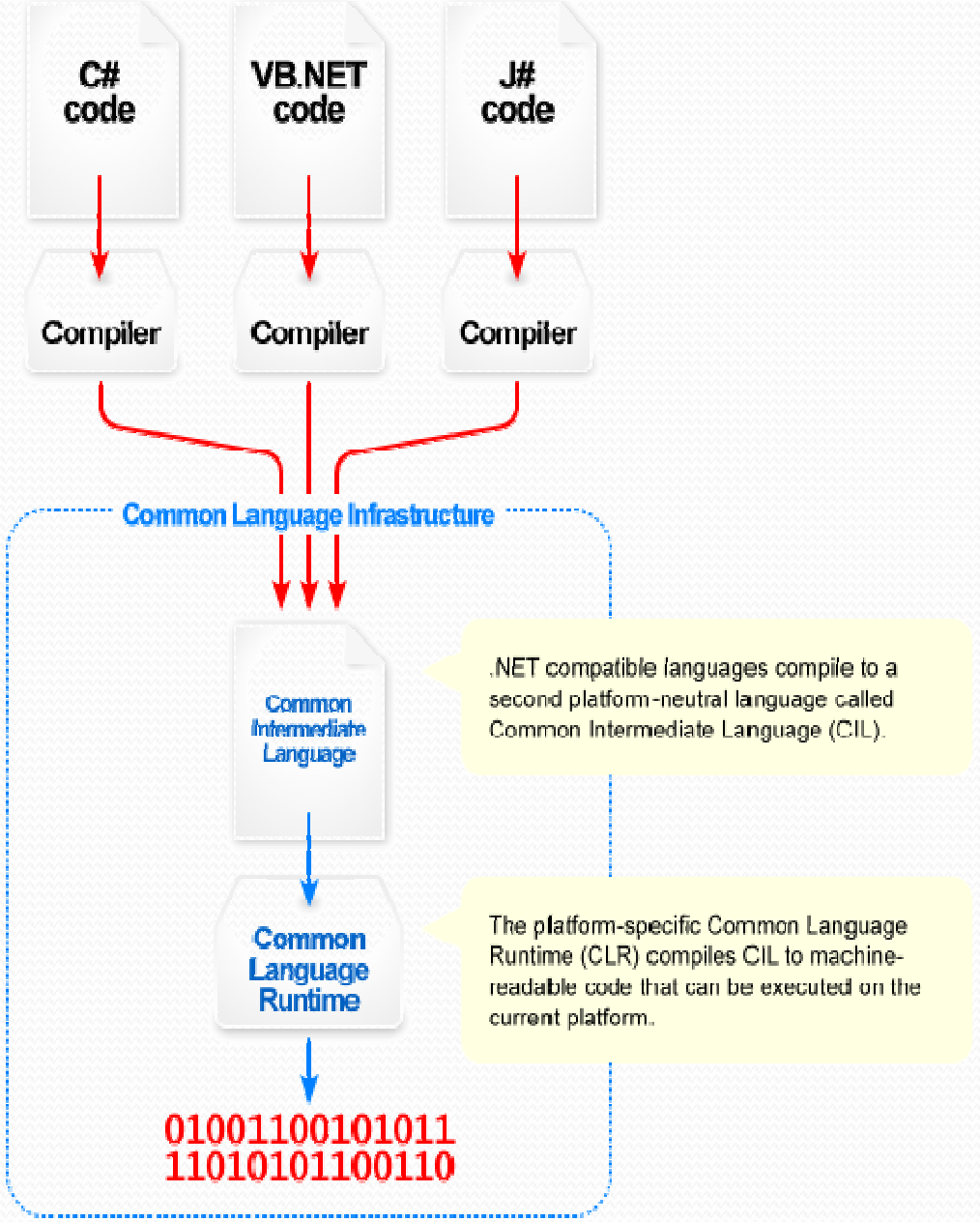
- The CLR transforms the CIL to assembly instructions for a particular hardware architecture.
  - This is termed jit'ing or Just-in-time compiling.
  - Some initial performance cost, but the jitted code is cached for further execution.
  - The CLR can target the specific architecture in which the code is executing, so some performance gains are possible.

CLR and JIT compiling.

- All .NET languages compile to the same CIL.
- Each language actually uses only a subset of the CIL.
- The least-common denominator is the *Common Language Specification* (CLS).
- So, if you want to use your C# components in Visual Basic you need to program to the CLS.

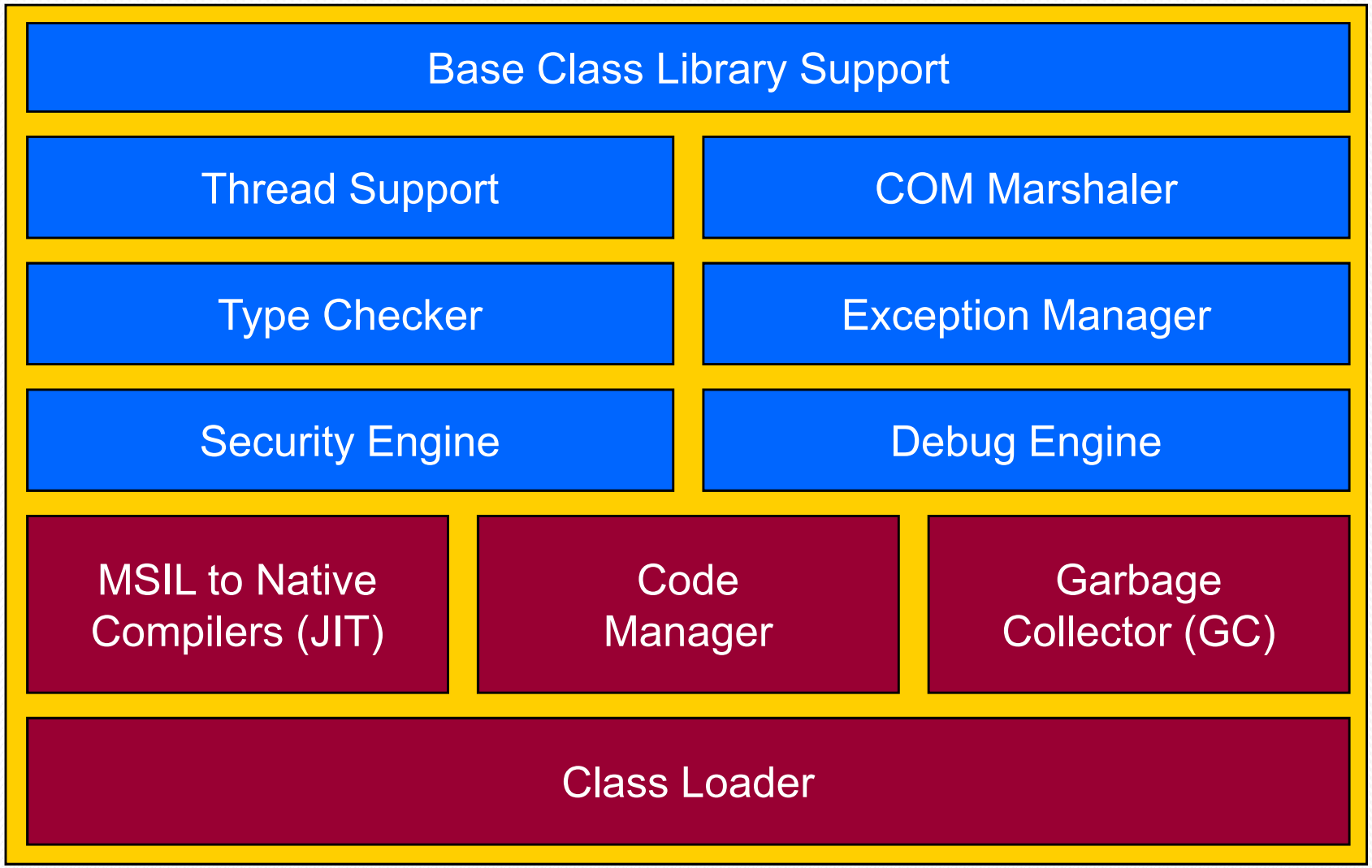
CLR versus CLI.

- CLR is actually an **implementation** by Microsoft of the CLI (Common Language Infrastructure) .
- CLI is an open *specification*.
- CLR is really a platform specific implementation.



Reference : *wikipedia.org*

The CLR Architecture



From MSDN

Common Language Infrastructure.

- CLI allows for cross-language development.
- Four components:
  - Common Type System (CTS)
  - Meta-data in a language agnostic fashion.
  - Common Language Specification – behaviors that all languages need to follow.
  - A Virtual Execution System (VES).

7. Common Type System (CTS)

- A specification for *how* types are *defined* and how they *behave*.
  - no syntax specified
- A type can contain zero or more members:
  - Field
  - Method
  - Property
  - Event
- We will go over these more throughout the quarter.

Common Type System (CTS)

- CTS also specifies the rules for visibility and access to members of a type:
  - Private
  - Family
  - Family and Assembly
  - Assembly
  - Family or Assembly
  - Public
- We will go over these more throughout the quarter.

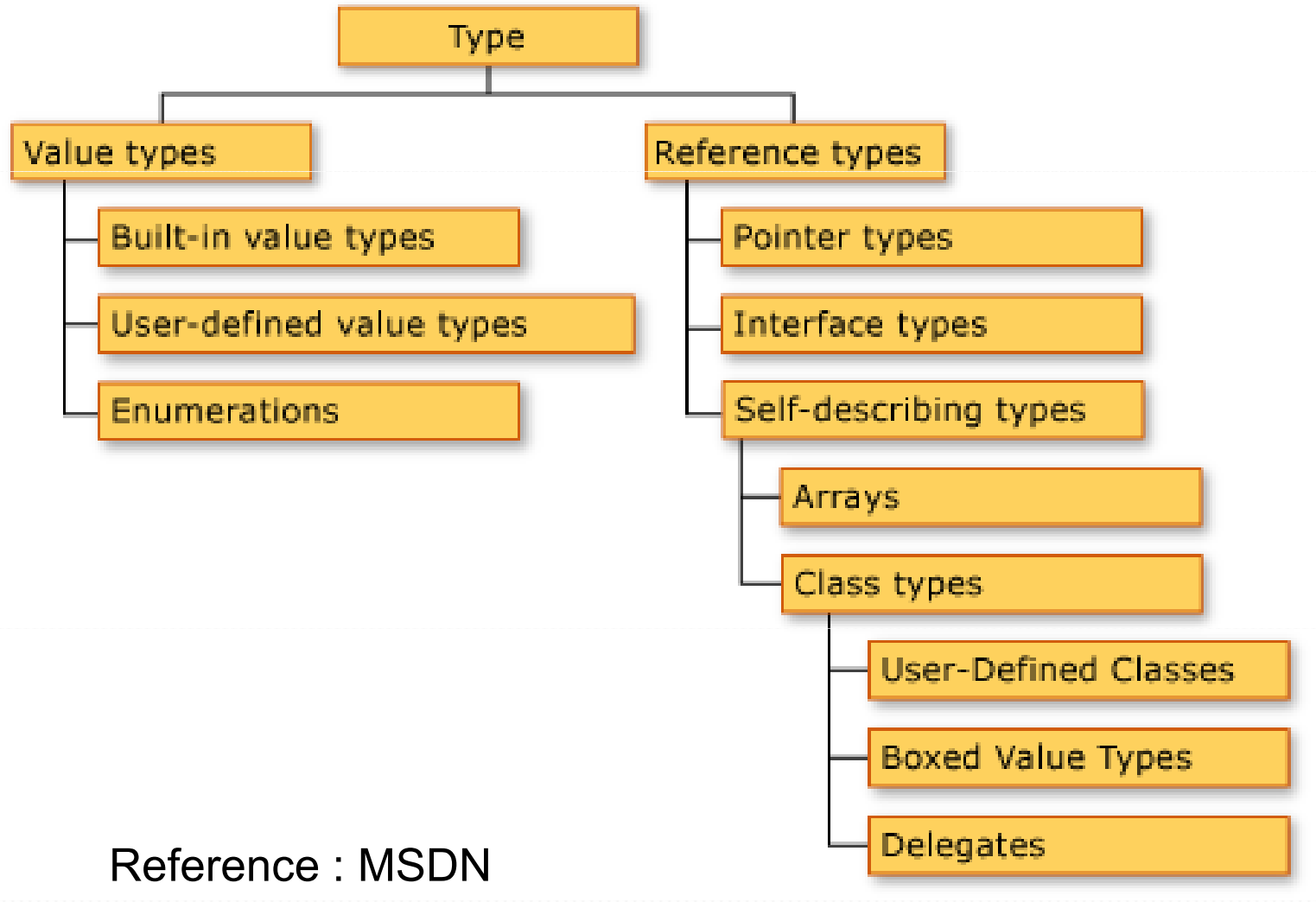
Common Type System (CTS)

- Other rules
  - Object life-time
  - Inheritance
  - Equality (through System.Object)

Common Type System (CTS)

- Languages often define aliases
- For example
  - CTS defines System.Int32 – 4 byte integer
  - C# defines *int* as an alias of System.Int32
  - C# aliases System.String as *string*.

Common Type System (CTS)

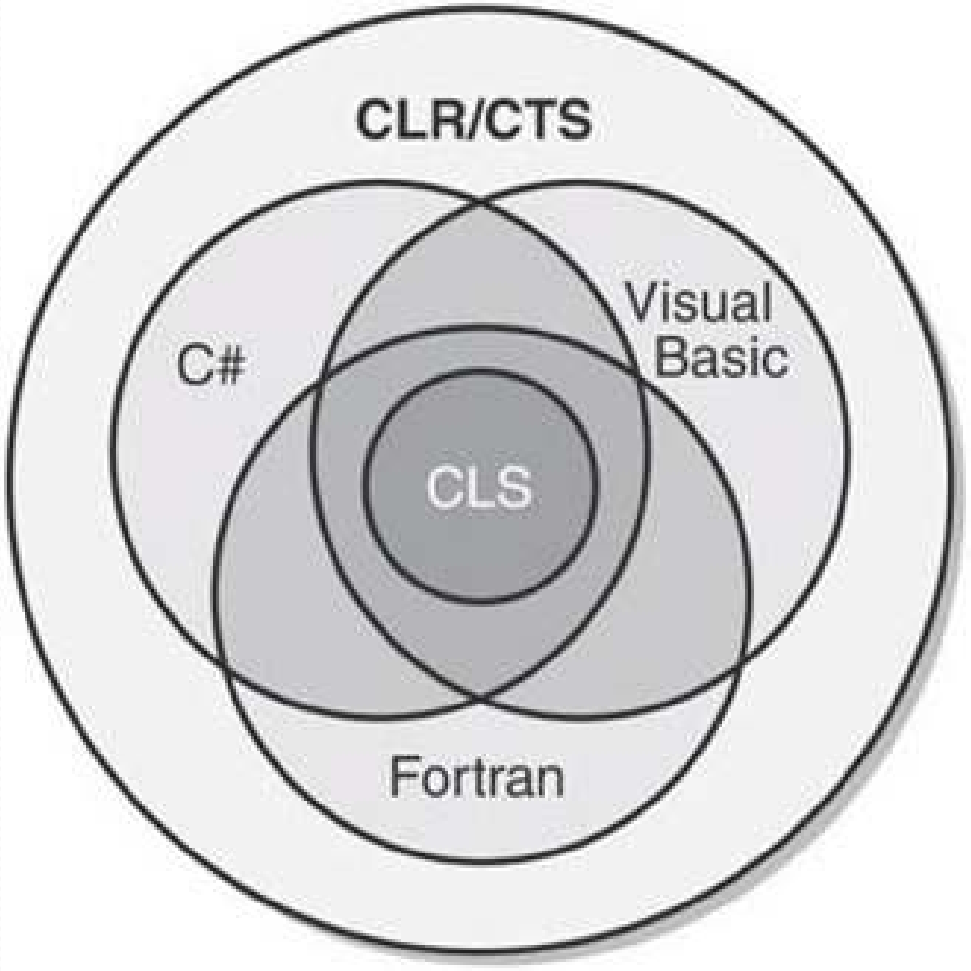


Reference : MSDN

Common Language System

- A specification of language features
  - how methods may be called
  - when constructors are called
  - subset of the types in CTS which are allowed
- For example
  - Code that takes UInt32 in a public method
  - UInt32 is not in the CLS
- Can ***mark*** classes as CLS-compliant
  - not marked is assumed to mean not compliant

CLS versus CLR



CLR via C#, *Jeffrey Richter*

8. Built-in Types

C#	***CTS type (FCL name)***	***CLS compliant***
int	System.Int32	yes
uint	System.UInt32	no
sbyte	System.SByte	no
byte	System.Byte	yes
short	System.Int16	yes
ushort	System.UInt16	no
long	System.Int64	yes
ulong	System.UInt64	no
float	System.Single	yes
double	System.Double	yes
decimal	System.Decimal	yes
char	System.Char	yes
string	System.String	yes
object	System.Object	yes

9. C# Assemblies

- Code contained in files called “assemblies”
  - code and ***metadata***
  - .exe or .dll as before
  - Executable needs a class with a “Main” method:
    - `public static void Main(string[] args)`
  - types
    - local: local assembly, not accessible by others
    - shared: well-known location, can be ***GAC***
    - strong names: use crypto for signatures
      - then can add some versioning and trust

10. First C# Program

```
using System;
namespace Program1
{
    class Progr1Class
    {
        static void Main()
        {
            System.Console.WriteLine("Hello, World...");
        }
    }
}
```

Program Explanation…..

- `using`
  - like import in Java, #inclinde in C: bring in namespaces
- `namespace`
  - disambiguation of names
  - like Internet hierarchical names and C++ naming
- `class`
  - Define a class as in C++ or Java
  - single inheritance up to object

Program Explanation…..

- static void Main()
  - Defines the entry point for an assembly.
  - Four different overloads – taking string arguments and returning int's.
- Console.Write(Line)
  - Takes a formatted string: "Composite Format"
  - Indexed elements: e.g., {0}
    - can be used multiple times
    - only evaluated once
  - {index [,alignment][:formatting]}

References:

- https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history
- http://en.wikipedia.org/wiki/List_of_CLI_languages
- https://en.wikipedia.org/wiki/.NET_Framework_version_history
- https://youtu.be/FFCn_z7dn_A
- https://www.w3schools.com/cs/