.Net Framework

Components

- MSIL – Microsoft Intermediate Language
- CLR – Common Language Runtime
- CLS – Common Language Specification
- CTS – Common Type Specification
- FCL – Framework Class Library

MSIL

- Microsoft Intermediate Language
- Also called CIL – Common Intermediate Language
- Low level, object-oriented, human readable language
- Platform independent and language independent set of instructions
- Two tools are associated with MSIL – MSIL Assembler (Ilasm.exe) and the MSIL Disassembler (Ildasm.exe).
- The former generates a portable executable file from IL code.
- The latter converts a portable executable file back to a text file, for viewing and modification.

Format of IL for adding two numbers

```
.method private hidebysig static void  Main(string[] args) cil managed
{
  .entrypoint
  // Code size       109 (0x6d)
  .maxstack  2
  .locals init ([0] int32 fnum,
           [1] string fstrnum,
           [2] int32 snum,
           [3] int32 ans,
           [4] bool CS$4$0000)
  IL_0000:  nop
  IL_0001:  ldstr      "Hello World"
  IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_000b:  nop
  IL_000c:  ldstr      "Enter First number:"
  IL_0011:  call       void [mscorlib]System.Console::WriteLine(string)
  IL_0016:  nop
  IL_0017:  call       string [mscorlib]System.Console::ReadLine()
  IL_001c:  stloc.1
  IL_001d:  br.s       IL_0025
  IL_001f:  call       string [mscorlib]System.Console::ReadLine()
  IL_0024:  stloc.1
  IL_0025:  ldloc.1
  IL_0026:  ldloca.s   fnum
  IL_0028:  call       bool [mscorlib]System.Int32::TryParse(string,
                                                             int32&)
```

MSIL

- With MSIL , compiler also produces **Metadata,** both kept in **PE : Portable Executable** files.
- MSIL performs many **task** and contains instruction useful for loading, initializing, storing, control flow, exception handling, object creation & manipulation, type conversion and many other operations.

Metadata

- Binary code that contains the self-description of the program.
- When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata. The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file .
- When the program gets executed the CLR loads the metadata into memory and references it to discover the information of the code.

Metadata

- Metadata contains the following information:-
  - Definition of each type used in your code
  - The signatures of data members used in types
  - The members that are referred by code
  - Data required by CLR to execute the program

CLR

- Virtual machine component of .NET
- Supervises the execution of a .NET program
- Also called the execution engine of the .NET Framework
- The MSIL code is converted into machine code at the time of execution with the help of the JIT(just-in-time) compiler

Components of CLR

- Components provided by CLR:-
  - **JIT compiler** – converts MSIL code to native code
  - **Memory Manager** – manages the resources that are required by the code at the time of execution
  - **Garbage Collector** – Provides automatic memory management. It automatically releases the objects that are no longer in use.
  - **Security Engine** – Enforces security restrictions to a .Net code by imposing CAS or Code Access Security. It prevents unauthorized access to protected resources and operations.
  - **Class Loader** – loads the classes of a .Net language in the runtime into RAM on demand.

Components of CLR

- Components provided by CLR(continued):-
  - **Type Checker** – performs strict type checking of .Net program
  - **Thread execution support** – provides multithreading support by .Net applications
  - **Exception Manager** – provides a mechanism to handle the runtime exceptions
  - **Debugger** – Specifies a debug engine that helps you to debug different type of applications

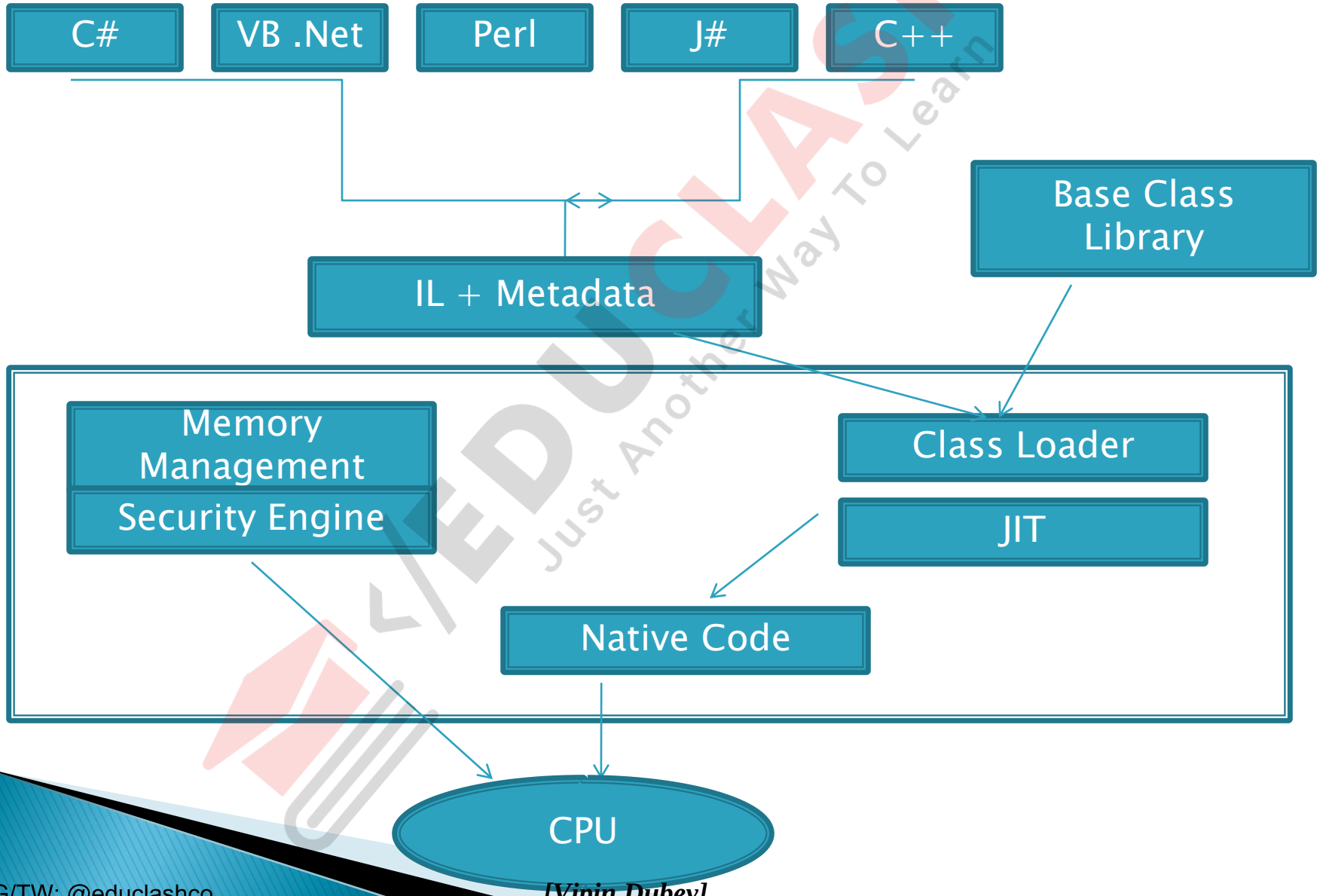
Managed Code vs Unmanaged Code

- In .Net, the source code is compiled into MSIL code and then the MSIL code is again converted into native code with the help of JIT compiler. This is called managed code. This code goes through CLR services like type checking, security and automatic garbage collection.
- Unmanaged code directly compiles to machine code and runs on the machine where it has been compiled. It does not have services like security and memory management.

Just in Time(JIT) compiler

- Converts MSIL code to native code.
- During the code execution time, the Managed Code is compiled only when it is needed, that is it converts the appropriate instructions to the native code for execution just before when each function is called.
- This process is called Just In Time (JIT) compilation, also known as Dynamic Translation .

How CLR works



Common Language Specification

- set of basic language features that .NET language need to develop applications and services
- Ensures interoperability among applications regardless of the language used to create the applications
- Is a subset of Common Type System

Common Type System

- Defines how types are declared and managed in Common Language Runtime
- Helps in cross language integration, type safety and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
- Provides a library that contains the primitive data types (such as Boolean, Byte, Char, Int32, and UInt64) used in application development.

Common Type System

- Five categories of types supported by CTS:-
  - Classes
  - Structures
  - Enumerations
  - Interfaces
  - Delegates

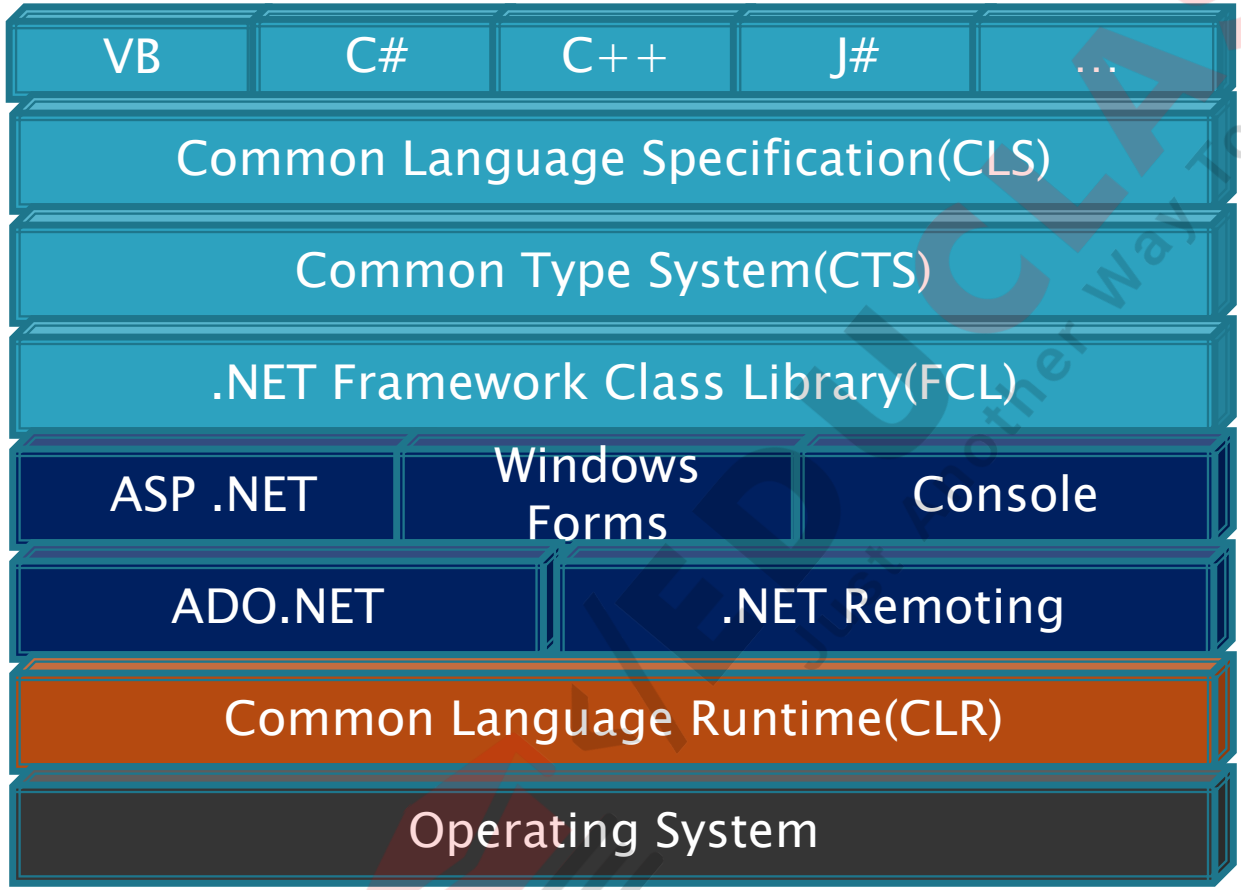
Framework Class Library(FCL)

- Library of classes and interfaces that provide access to system functionality
- Foundation on which .NET framework applications, components and controls are built.
- Example: System, System.Net, System.Linq

FCL

- Base Class Library – irrespective of the type of application
- Console Libraries
- Winform Libraries
- Webform Libraries
- Mobile Libraries

.Net Framework Architecture



Assemblies

- Compiled and versioned code library that contains a collection of code and metadata.
- Building blocks of .Net Framework applications.
- smallest unit of deployment of a .net application
- Can include both .dll or .exe file
- During compile time metadata is created with MSIL and stored in a file called Assembly Manifest.
- Every assembly that is created can have one or more program files and a Manifest

Assemblies

- Assemblies are self-describing.
- The Assembly Manifest contains the version number of the assembly.
- Multiple assemblies can be loaded side by side.
- Assembly Structure
  - Assembly Metadata
  - Type Metadata
  - CIL Code
  - Resources

Assemblies

- There are two types of Assemblies:
  - Private Assembly
  - Shared Assembly

A private assembly is used by a single application and is usually stored in the application's install directory.

A shared assembly is one that can be referenced by more than one application. If multiple applications need to access an Assembly, we should add the Assembly to the Global Assembly Cache (GAC).

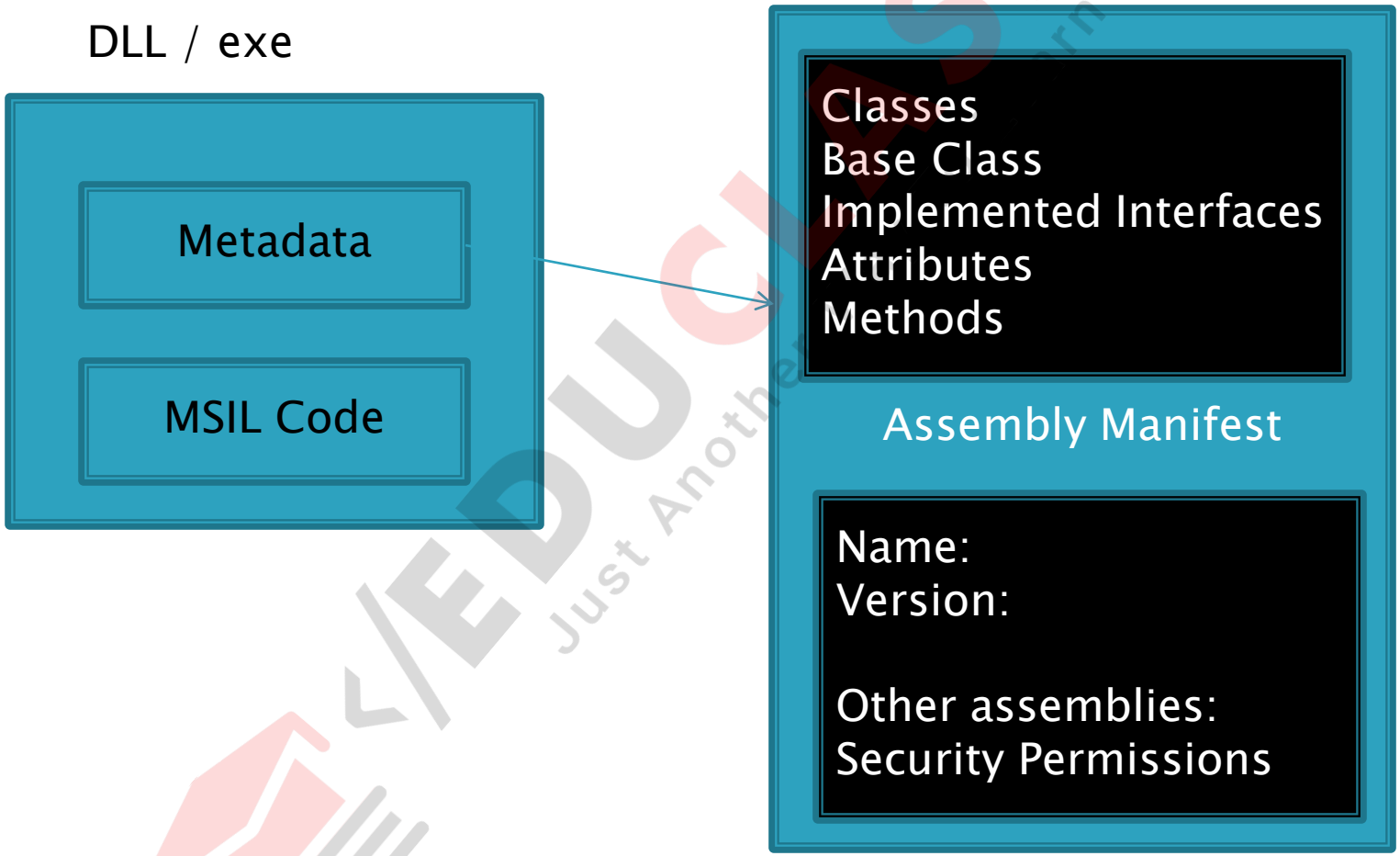
Assembly Manifest/Metadata

- Information stored in the Assembly Manifest
  - Assembly Name
  - Version Number
  - List of all files in the assembly
  - List of assemblies referenced by the assembly

Type Metadata

- Contains information about the types declared and used in assembly.
- Classes, interfaces, structs, events etc. are used.
- All information about types i.e. methods, attributes, properties etc.

Assembly Contents



GAC

- Global Assembly Cache
- Machine wide cache in a computer
- GAC exists is every computer where the CLR is installed.
- Assemblies must be made shareable by registering them in GAC.
- The GAC is simply the C:\Windows\assembly folder where all the assemblies which are shareable are stored.
- To share an assembly in the GAC the gacutil.exe is used.
  - gacutil –I <assemblyname>

Application Domain

- Mechanism to provide isolation to two or more executed software applications so that they do not affect each other.
- Isolation is provided to ensure that code running in one application does not affect other unrelated applications.
- Provided for security, reliability, versioning and for unloading assemblies.
- Light weight processes having its own set of code, data and configuration settings.
- Created by runtime hosts and invoked by CLR to load the applications that need to be executed

Features of .Net

- Interoperability –
  - every .Net compatible language can communicate with each other.
  - .Net components can communicate with existing COM components without migrating them to .Net.
- OOPs support – supports object oriented programming
- Multi language support
- Rich Functionality – variety of functionalities, drag and drop tools
- Automatic memory management– garbage collection takes place at the time of compilation i.e. frees the unused objects from memory
- Security

Features of .Net

- End of DLL Hell – 'DLL Hell' refers to the problems that were faced due to one single version of component across the machine. The problem arises when the version of the DLL on the computer is different than the version that was used when the program was being created. Since DLLs are not backward compatible, the programs using the previous version of DLL will crash in its attempt to use the DLL.
- When a .Net component is installed onto the machine, the Global Assembly Cache looks at its version, its public key, and its language information and creates a strong name for the component. The component is then registered in the repository and indexed by its strong name, so there is no confusion between different versions of the same component, or DLL.