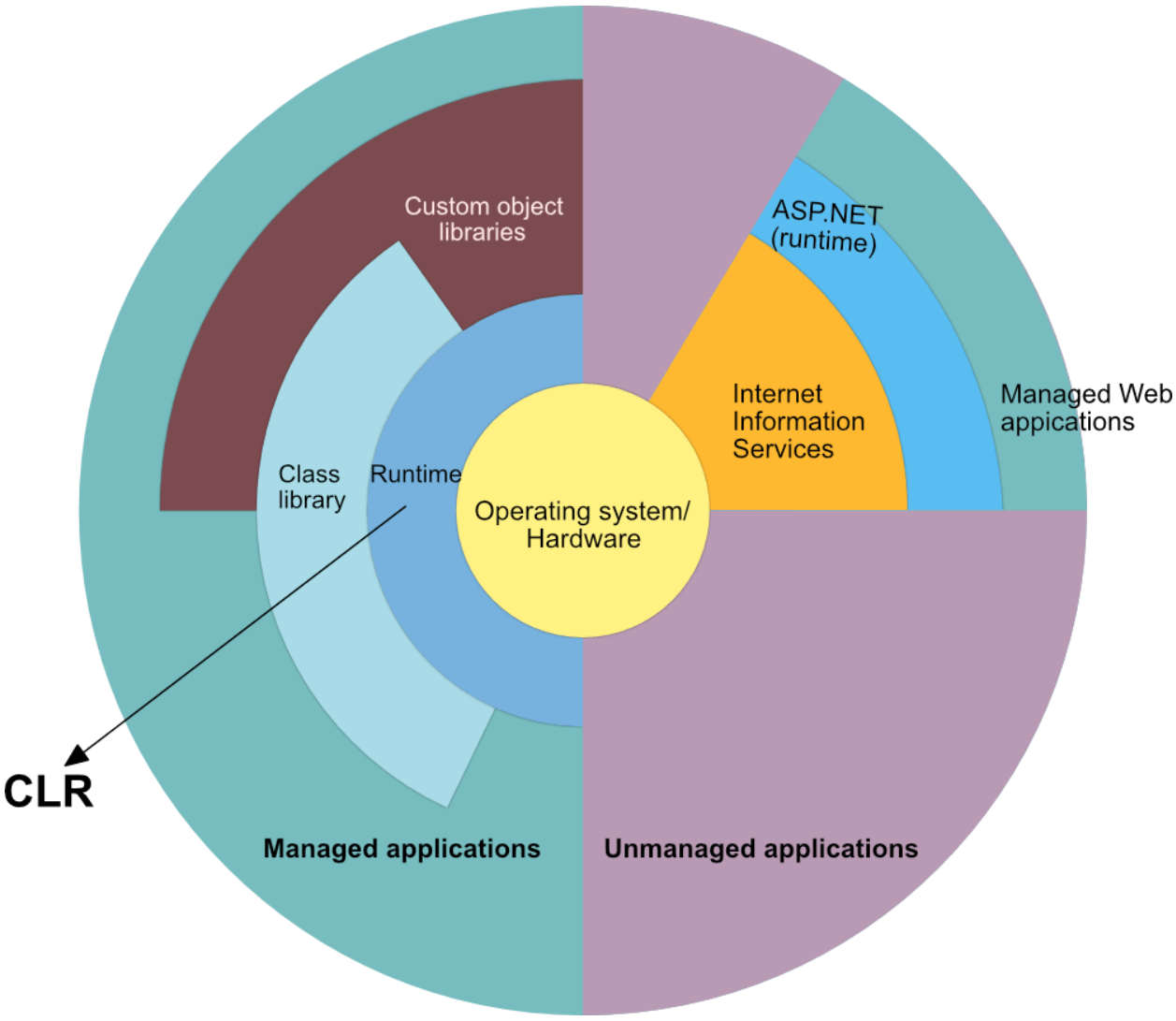
Common Language Runtime (CLR) in C#

CLR is the basic and Virtual Machine component of the **.NET Framework**. It is the **run-time environment in the .NET Framework** that runs the codes and helps in making the development process easier by providing the various services. Basically, it is responsible for managing the execution of *.NET programs* regardless of any *.NET* programming language. Internally, CLR implements the *VES(Virtual Execution System)* which is defined in the Microsoft's implementation of the *CLI(Common Language Infrastructure).*
The code that runs under the Common Language Runtime is termed as the Managed Code. In other words, you can say that CLR provides a managed execution environment for the *.NET* programs by improving the security, including the cross language integration and a rich set of class libraries, etc. CLR is present in every .NET framework version. Below table illustrate the CLR version in .NET framework.

CLR Versions	.NET Framework Versions
1.0	**1.0**

CLR Versions	.NET Framework Versions
1.1	**1.1**
2.0	**2.0**
2.0	**3.0**
2.0	**3.5**
4	**4**
4	**4.5(also 4.5.1 & 4.5.2)**
4	**4.6(also 4.6.1 & 4.6.2)**
4	**4.7(also 4.7.1 & 4.7.2)**

Below diagram illustrate how CLR is associated with the operating system/hardware along with the class libraries. Here, the runtime is actually CLR.

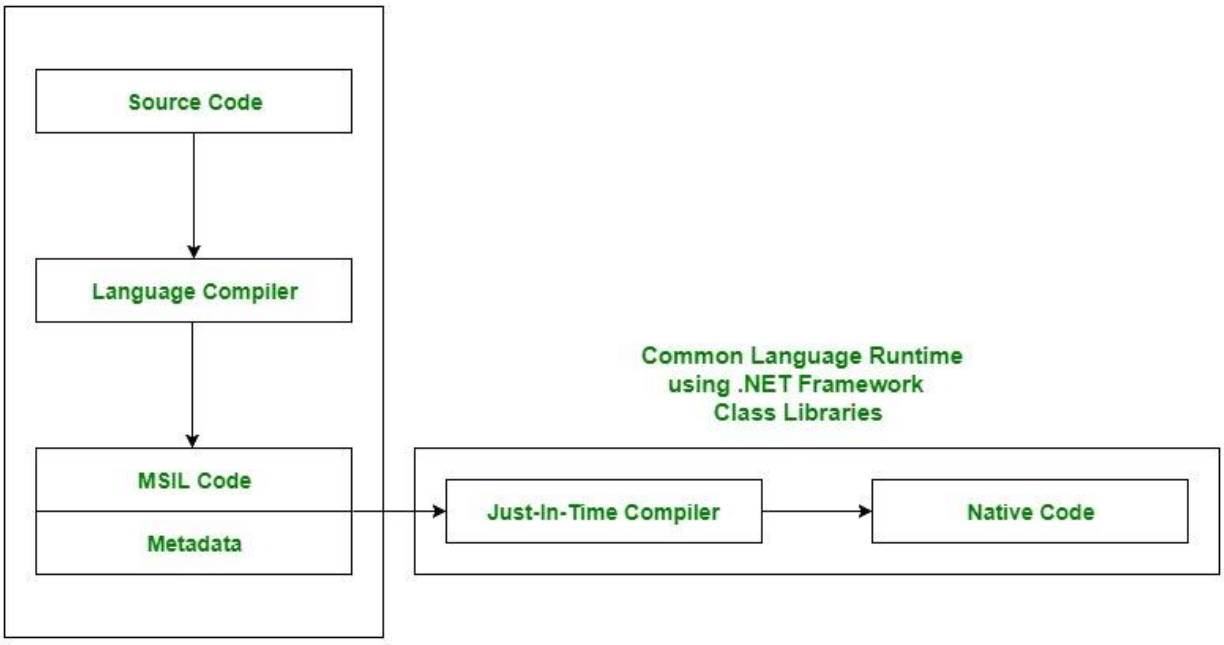


Role of CLR in the execution of a C# program

- Suppose you have written a C# program and save it in a file which is known as the Source Code.
- Language specific compiler compiles the source code into the ***MSIL(Microsoft Intermediate Language)*** which is also known as the ***CIL(Common Intermediate Language)*** or ***IL(Intermediate Language)*** along with its metadata. *Metadata* includes all the types, actual

implementation of each function of the program. MSIL is machine-independent code.

- Now CLR comes into existence. CLR provides the services and runtime environment to the MSIL code. Internally CLR includes the JIT(Just-In-Time) compiler which converts the MSIL code to machine code which further executed by CPU. CLR also uses the .NET Framework class libraries. Metadata provides information about the programming language, environment, version, and class libraries to the CLR by which CLR handles the MSIL code. As CLR is common so it allows an instance of a class that written in a different language to call a method of the class which written in another language.



Main Components of CLR

As the word specify, Common means CLR provides a common runtime or execution environment as there are more than 60 .NET programming languages.

Main components of CLR:

Common Language Specification (CLS):
It is responsible for converting the different .NET programming language syntactical rules and regulations into CLR understandable format. Basically, it provides Language Interoperability. Language Interoperability means providing execution support to other programming languages also in .NET framework.

Language Interoperability can be achieved in two ways :

1. **Managed Code:** The MSIL code which is managed by the CLR is known as the Managed Code. For managed code CLR provides **three** .NET facilities:
2. **Unmanaged Code:** Before .NET development, programming languages like.COM Components & Win32 API do not generate the MSIL code. So these are not managed by CLR rather managed by Operating System.

Common Type System (CTS):
Every programming language has its own data type system, so CTS is responsible for understanding all the data type systems of .NET programming languages and converting them into CLR understandable format which will be a common format.

There are 2 Types of CTS that every .NET programming language have :

1. **Value Types:** Value Types will store the value directly into the memory location. These types work with stack mechanisms only. CLR allows memory for these at Compile Time.
2. **Reference Types:** Reference Types will contain a memory address of value because the reference types won't store the variable value directly in memory. These types work with Heap mechanism. CLR allot memory for these at Runtime.

Garbage Collector:

It is used to provide the *Automatic Memory Management* feature. If there was no garbage collector, programmers would have to write the memory management codes which will be a kind of overhead on programmers.

JIT(Just In Time Compiler):

It is responsible for converting the CIL(Common Intermediate Language) into machine code or native code using the Common Language Runtime environment.

Benefits of CLR:

- It improves the performance by providing a rich interact between programs at run time.

- Enhance portability by removing the need of recompiling a program on any operating system that supports it.
- Security also increases as it analyzes the MSIL instructions whether they are safe or unsafe. Also, the use of delegates in place of function pointers enhance the type safety and security.
- Support automatic memory management with the help of Garbage Collector.
- Provides cross-language integration because CTS inside CLR provides a common standard that activates the different languages to extend and share each other's libraries.
- Provides support to use the components that developed in other .NET programming languages.
- Provide language, platform, and architecture independence.
- It allows easy creation of scalable and multithreaded applications, as the developer has no need to think about memory management and security issues.