

Visual Programming with C#

BICTE. 465

Prepared By: Bhesh Raj Pokhrel

1.1 .NET framework: Introduction

- The **.NET Framework** is a software development framework that is designed and developed by Microsoft.
- The first version(1.0) of the .Net framework was released in the year **2002**.
- In another words, it is a ***virtual machine*** for compiling and executing programs written in different languages like [C#](#), VB.Net, etc.
- It provides a ***runtime*** environment and a set of ***libraries*** and **tools** for building and running applications on Windows operating systems.
- It is used to develop **Form-based** applications (desktop), **Web-based** applications, **mobile**, and gaming applications.
- .NET Framework supports more than **60** programming languages of which **11** programming languages are designed and developed by Microsoft.
- .NET stands for “*Network enabled technology*”.

1.2.Features of .NET framework

1. Language Interoperability:

- Language Interoperability means that **code written in one programming language can be used in another language without problems.**
- For example:
 - We can write a class in **C#** and then use it in a program written in **VB.NET**.
 - Or, we can write a method in **F#** and call it from a **C#** project.
- Language interoperability in **.NET Framework** works only among the languages that are **supported by .NET** (like **C#, VB.NET, F#, JScript.NET, etc.**) because they all compile into the same **Intermediate Language (IL)** and run on the **Common Language Runtime (CLR)**.

2. Platform Independent:

- In general, **platform independence** means that the same program can run on different systems (hardware or operating systems) **without needing to be rewritten.**
- For example:
 - In **Java**, platform independence is achieved through the **Java Virtual Machine (JVM)**.
 - In **.NET**, it is achieved through the **Common Language Runtime (CLR)** and the **Intermediate Language (IL)**.
- This means programs written in different .NET languages are **portable inside the .NET environment** and can run on any system that has the .NET runtime installed.

1.2.Features of .NET framework

3. Object Oriented Programming:.

- NET Framework supports a **Object oriented programming language**. That means without class we can not develop a single application in .NET. A language which supports the feature of .NET framework is Object oriented programming language.

4. Flexible Data Access.

- Flexible Data Accessibility can be **provided by the ADO.NET technology**. ADO is database technology which is used to access the data. That means we can store the data from front-end to backend and retrieve data from backend to frontend.

5. Parallel Computing:

- The NET **Framework 4.0** introduces a new programming model for writing ***multithreaded*** and asynchronous code that greatly simplifies the work of application and library developers.

1.2 .Features of .NET framework

6. Automatic memory management:

- In many programming languages, programmers are responsible for allocating and releasing memory and for handling object lifetimes. In .NET Framework apps, the CLR provides these services on behalf of the app..NET provides the ***garbage collector*** and it takes care of freeing unused objects at appropriate intervals.. The ***Garbage Collector is*** responsible for collecting the objects no longer referenced by application.

7. Security:

- Windows platform was always criticized for poor security mechanisms. Microsoft has taken great efforts to make .NET platform safe and secure for enterprise applications Features such as ***type safety, code access security etc.***

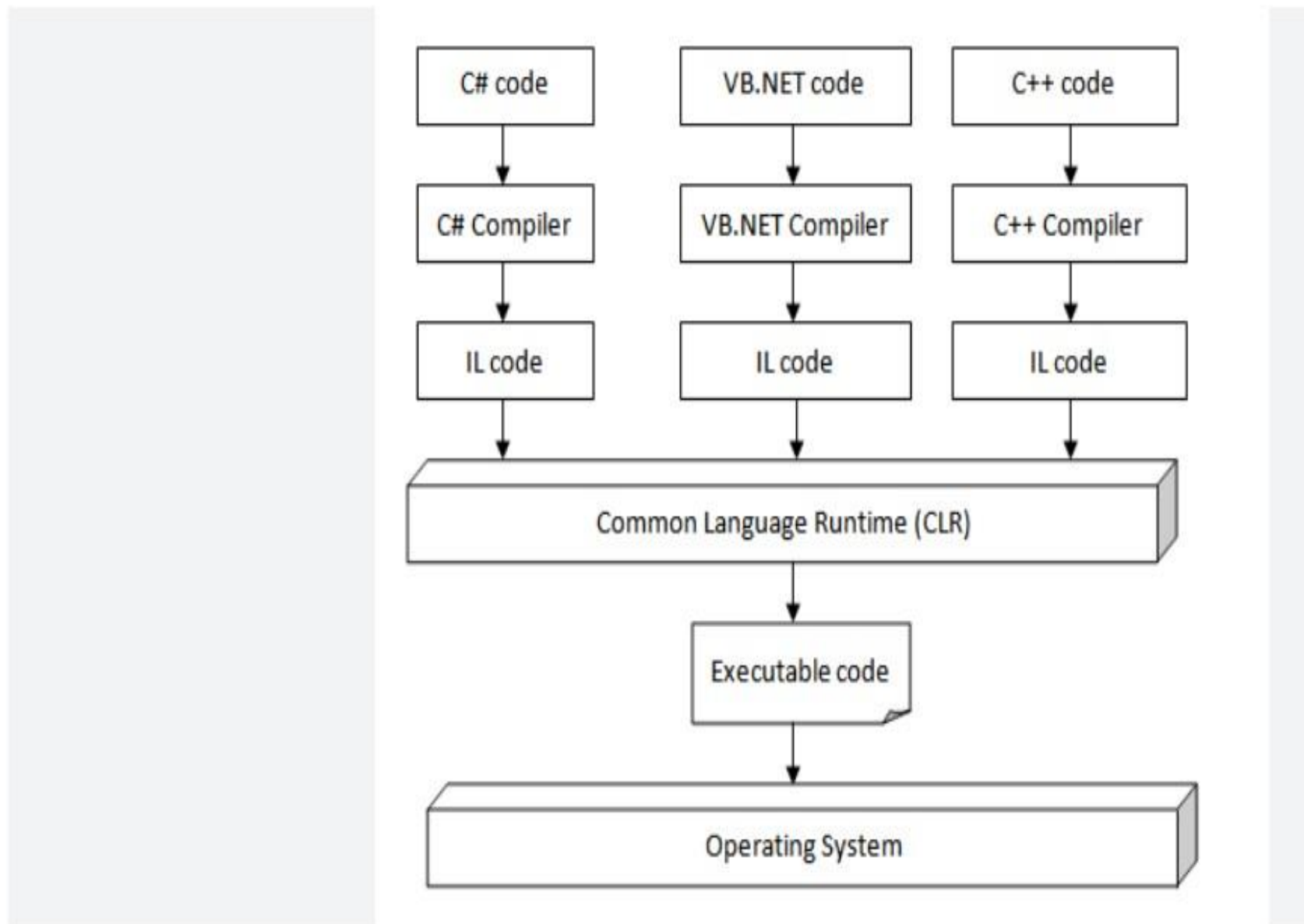
8. Dynamic web pages and web services.

- The .NET framework provides an environment for building, deploying and running **web services** and other application.

9. Rapid application development.

10. Easy and rich debugging support.

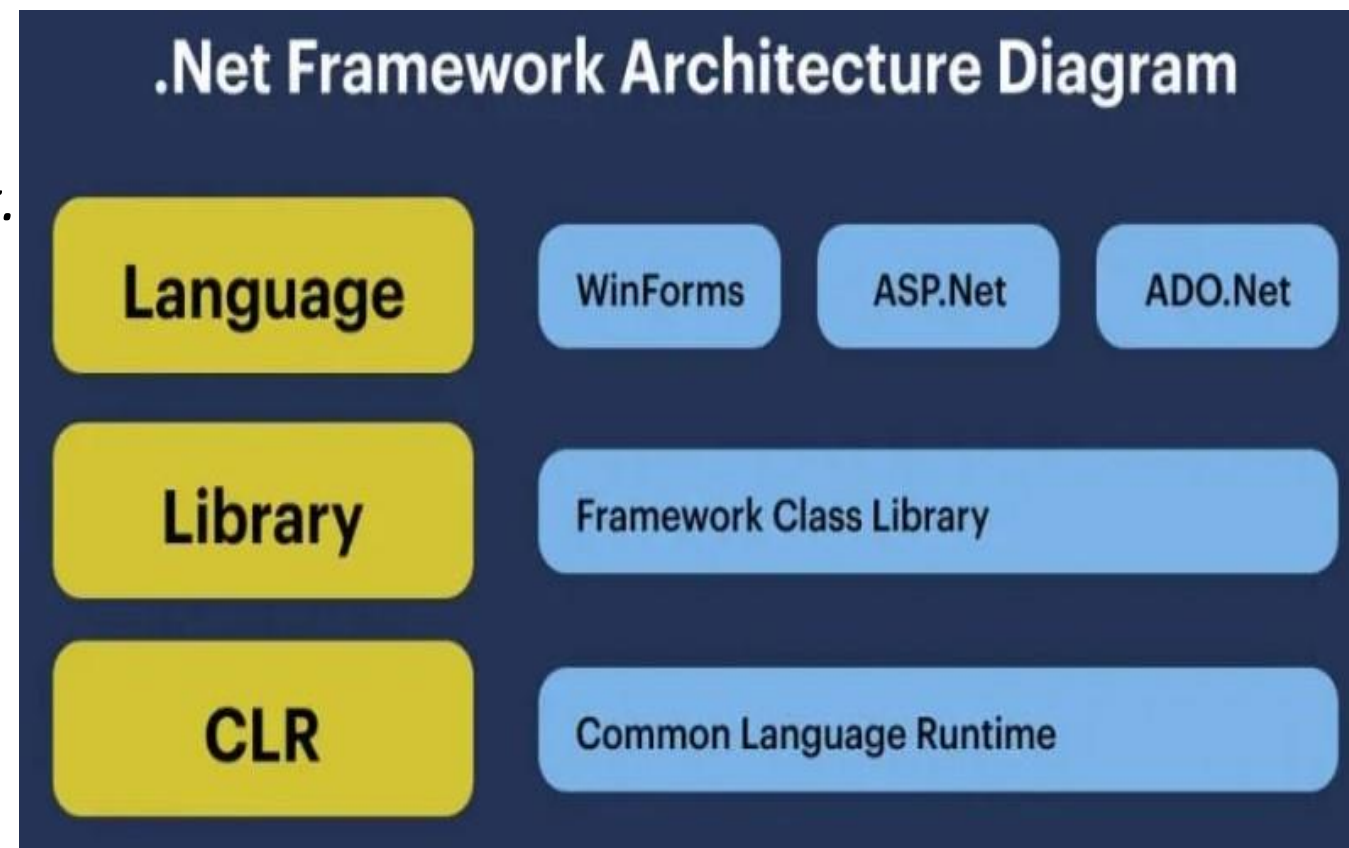
1.3. Architecture of .Net



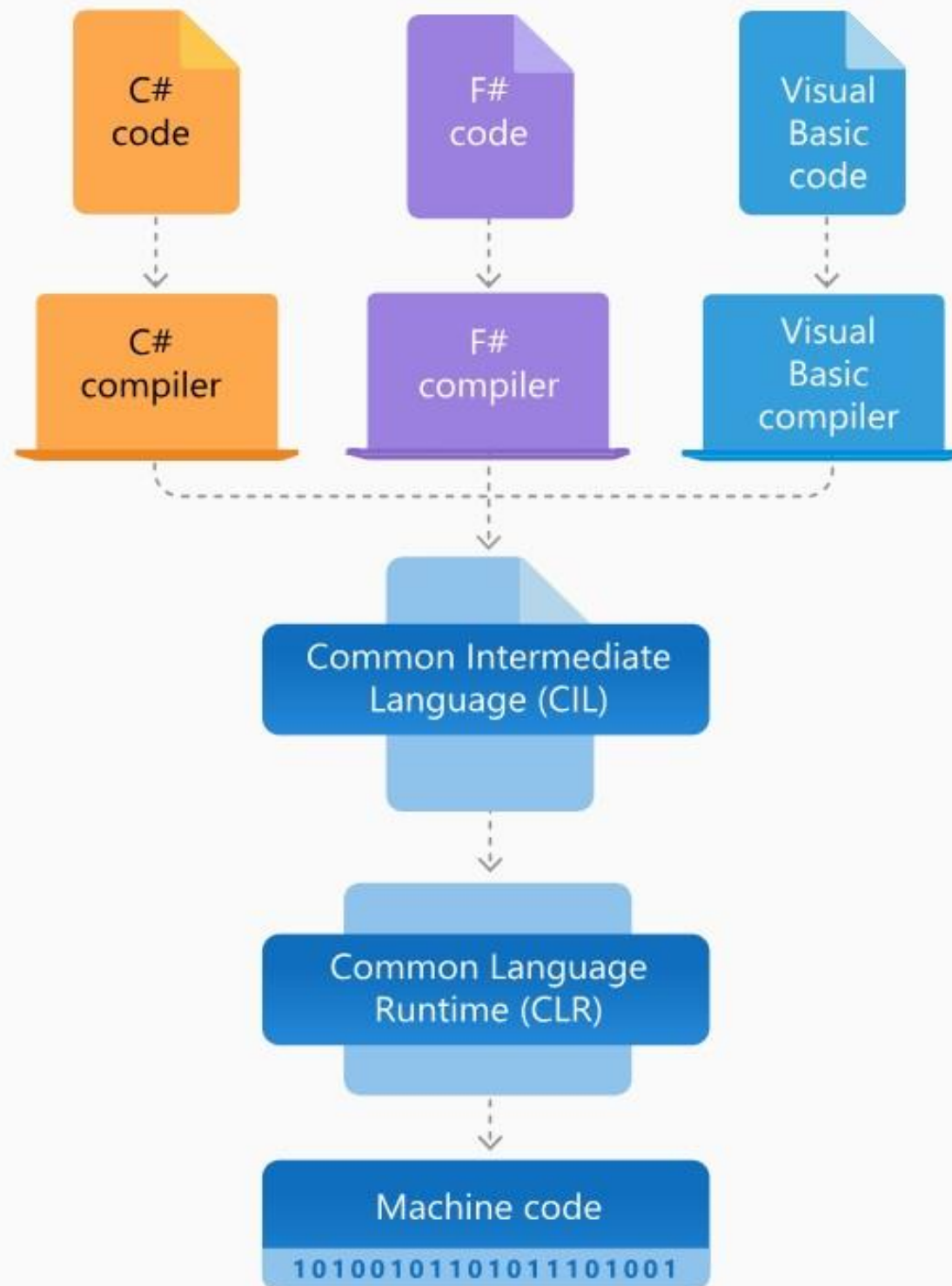
MICROSOFT VISUAL BASIC .NET : Architecture of .Net:

1.3.NET Framework Architecture

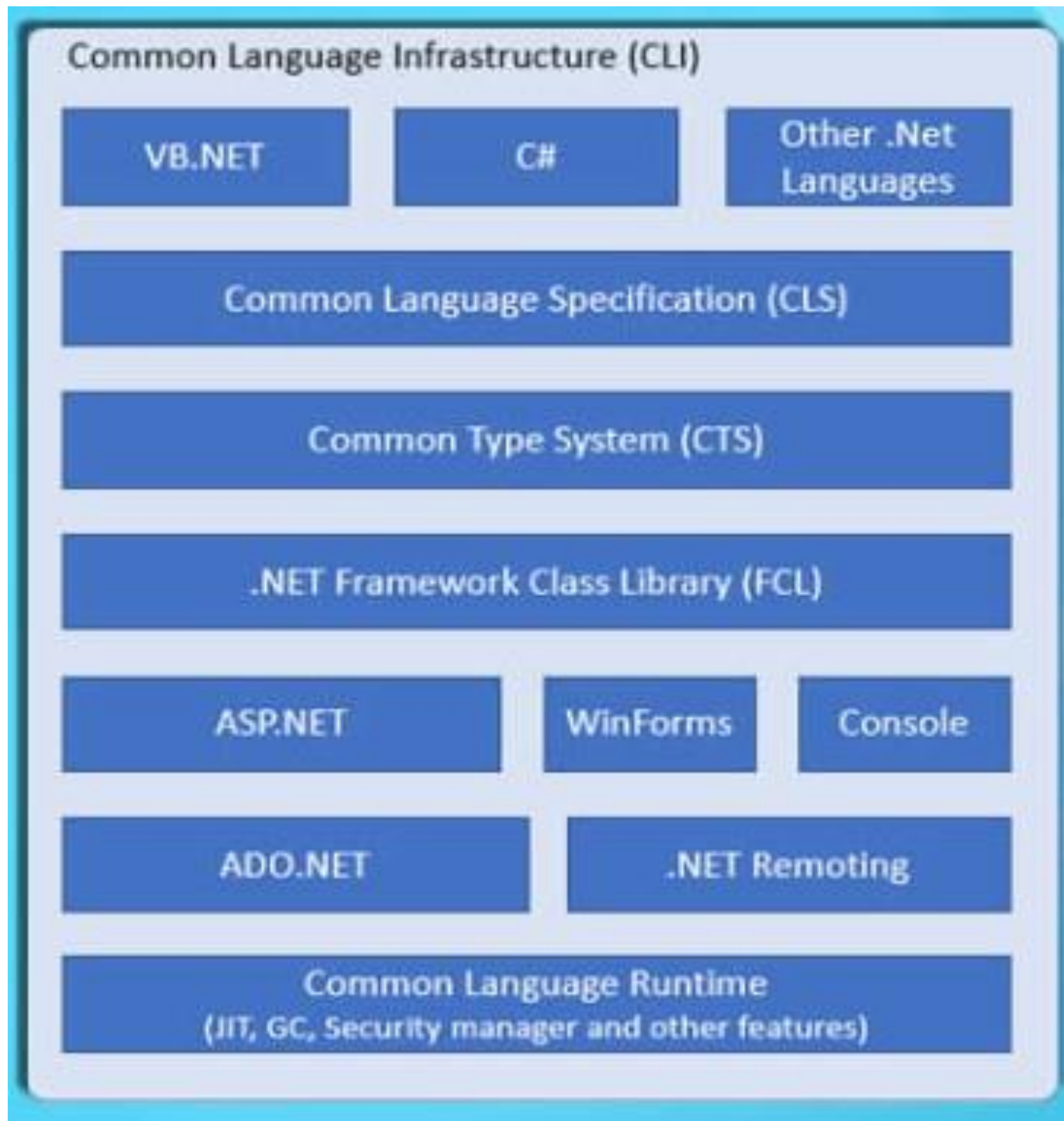
- Net Architecture is a **software architecture** used for building applications that run on Microsoft's .NET platform.
- The .NET Framework consists of a **wide range of components** that can be used to build applications for any computer platform.
- It provides: a set of **libraries & API (for Web, Desktop, mobile application)**, **tools** such as Visual Studio (IDE) and **runtime** environment.
- There are **three** essential parts to .NET architecture:
 - **Languages:** C#,VB, C++, ASP.NET, ADO.NET etc.
 - **Library:** Framework Class Library/ Base Class Library
 - **CLR:** Common Language Runtime.



1.3.NET Framework Architecture



1.3. NET Framework Architecture

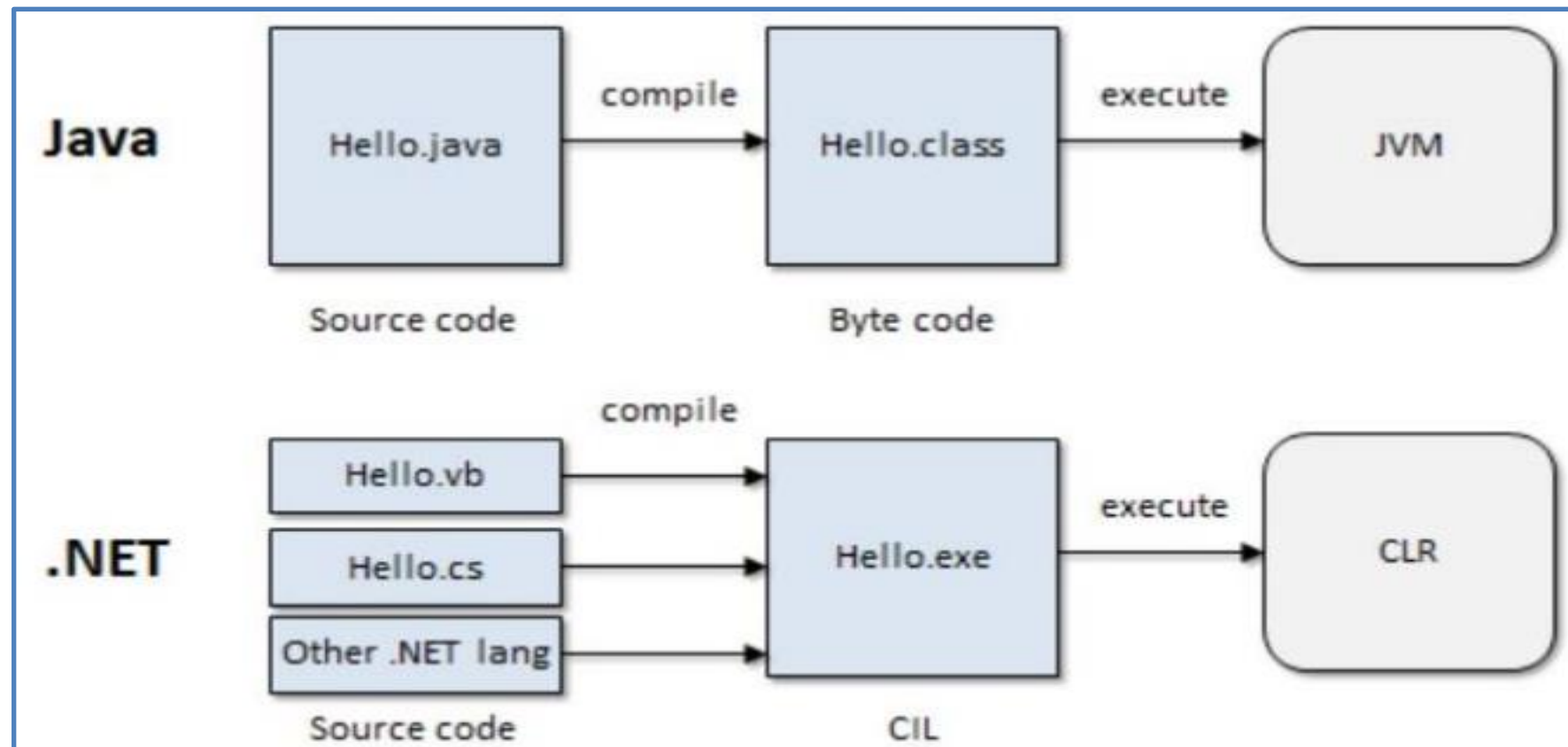


1.3. NET Architecture:

1. Common Language Runtime (CLR)

- The **Common Language Runtime (CLR)** is the main part of the .NET Framework that runs all .NET programs. It is often called the “heart” of .NET because every program works inside it.
- When a .NET program is compiled, the code is compiled into an **Intermediate Language (IL)**. Compiled code is stored in ***assemblies files*** (packages) with a **.dll** or **.exe** file extension.
- When an app runs, the **CLR** takes the assembly (package) and uses a **Just-In-Time (JIT) compiler** to convert this IL code into machine code that the computer can understand.
- Besides running the code, the CLR also takes care of important tasks like thread management, garbage collection, type-safety, exception handling, and more.

In short, the CLR runs the code, manages memory, cleans up unused data, handles errors, and ensures security making it the backbone of all .NET applications.



1.3. NET Architecture:

2. Common Language Infrastructure (CLI)

- The **CLI** is a **specification** developed by Microsoft that describes the **runtime environment for executing programs written in different programming languages**.
- It is the foundation of the .NET Framework. It **defines rules** for how **code, data types, and metadata** should be represented so that different .NET languages can **work together easily**.
- The CLI consists of several key components, including the **Common Language Runtime (CLR)**, the **Common Type System (CTS)**, the **Common Language Specification (CLS)**, and the **Base Class Library (BCL)**.
- Essentially, the CLI provides the rules and environment that allow .NET to support multiple languages and enable programs to execute efficiently and securely.

1.3. NET Architecture:

3. The Common Language Specification (CLS)

- The **CLS** is a set of rules and standards that all .NET programming languages must follow to ensure they can work together smoothly.
- It is responsible for converting the different .NET programming language **syntactical rules and regulations** into **CLR understandable format**.
- i.e. CLS defines basic programming concepts, such as **how data types, methods, and classes** should be declared so that any .NET language that follows these rules can interact with code written in other .NET languages without problems.
- Its main purpose is to provide **language interoperability**, , which means that code written in one .NET language, like C#, can be used in another language, like VB.NET or F#.

4. Common Type System (CTS)

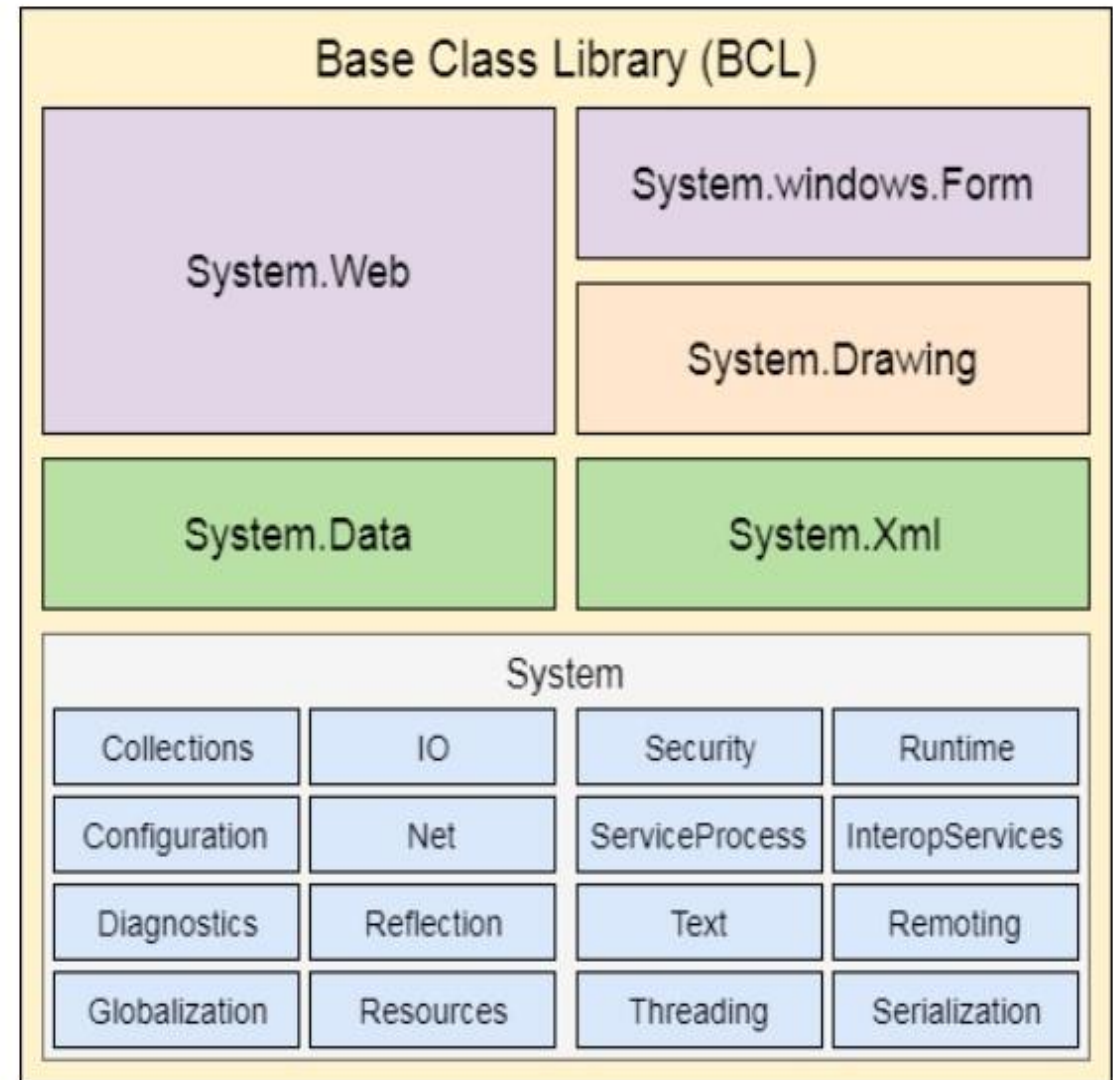
- It provides rules for data **type definitions, type safety, and memory layout**, so that all languages in the .NET environment can share data and objects seamlessly. For example, an int in C# and an Integer in VB.NET are treated the same under the CTS.
- By enforcing a **consistent type system**, the CTS enables **language interoperability**, reliable program execution, and reduces errors caused by type mismatches.

1.3. NET Architecture:

5. Base Class Library / Framework Class Library.

- The **Base Class Library (BCL)** is a large collection of **pre-built, reusable classes and functions** provided by the .NET Framework.
- It offers **ready-made tools** for common programming tasks, such as **file handling** (reading and writing files), working with **collections** like arrays, lists, and dictionaries, **networking, database access, threading, security**, and much more.
- By providing these resources, the BCL allows developers to **save time and effort**, as they do not need to write these basic functionalities from scratch.

FCL (Framework Class Library



1.3. NET Architecture:

6. Application Models

- The .NET Framework provides different **application models** for building various types of software.
 - **WinForms** is used to create desktop applications with graphical user interfaces, such as calculators, notepads, or inventory management systems.
 - **ASP.NET (Active Server Pages)** allows developers to build web applications that run in browsers, including e-commerce websites, online portals, and social networking sites.
 - **ADO.NET (ActiveX Data Object)** for database-driven applications, It provides tools to connect to, read from, and update databases like SQL Server or Oracle.
 - **WPF (Windows Presentation Foundation)** is designed for creating rich desktop applications with advanced graphics, animations, and multimedia support.
 - **WCF (Windows Communication Foundation)** enables applications to exchange data over networks, making it useful for web services and messaging systems.
 - Finally, the **Entity Framework** simplifies data access by using **object-relational mapping (ORM)**, allowing developers to work with databases in an object-oriented way.
- Together, these models provide a **flexible and powerful framework** for building a wide variety of .NET applications.

1.3. NET Architecture:

7. Languages

- The .NET Framework supports more than **60 programming languages**, including popular ones like **C#, VB.NET**, and **F#**.
- All these languages are designed to **work within the .NET environment** and **follow the framework's standards**.
- When a program is written in any of these languages, it is first **compiled into an Intermediate Language (IL)**, which is a platform-independent, low-level code.
- This IL code is then executed by the **Common Language Runtime (CLR)**, which converts it into machine code that the computer can understand.

8. Intermediate Language (IL) & JIT Compiler

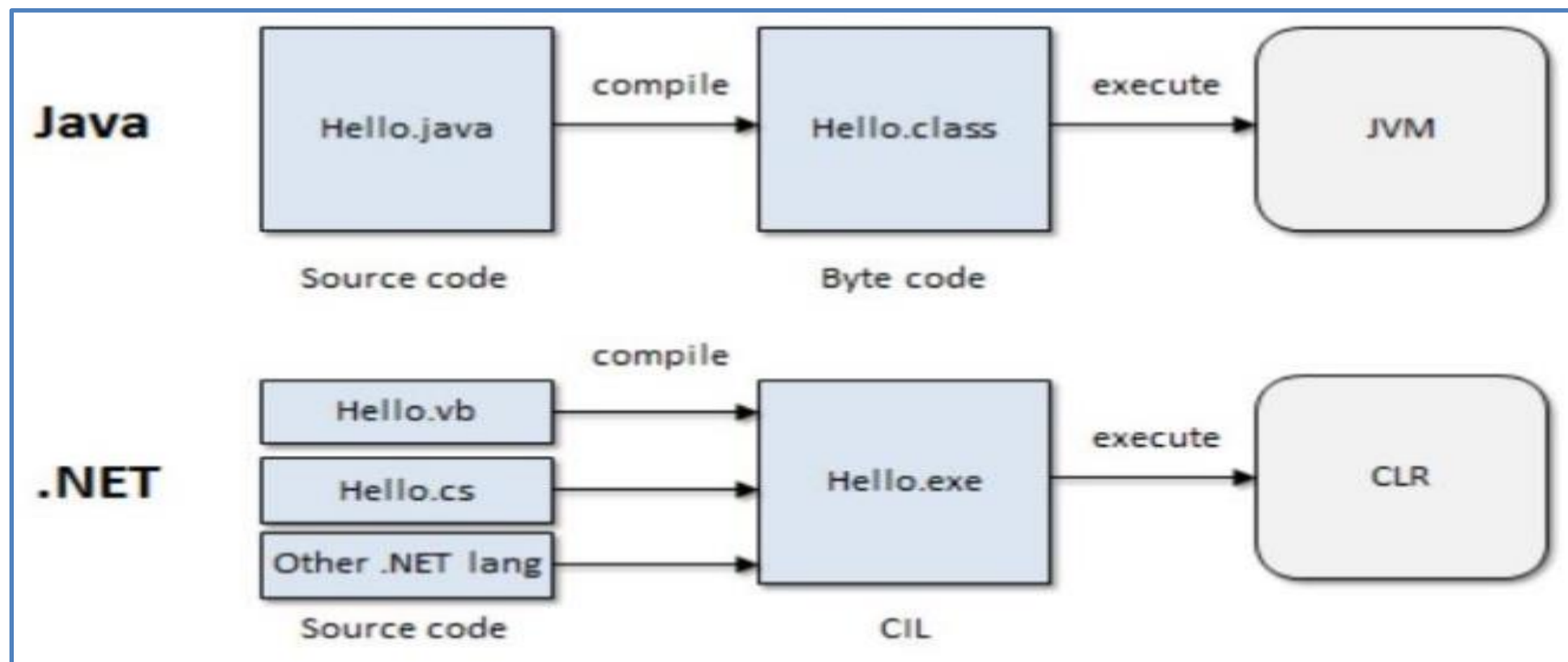
- When a .NET program is compiled, it is not immediately converted into machine code. At first it transformed into an **Intermediate Language (IL)**, which is a platform-independent, low-level code understood by the .NET Framework.
- At runtime, the **Just-In-Time (JIT) compiler** takes this IL code and converts it into **machine code** that is specific to the computer's CPU and operating system.
- This process allows .NET programs to be **portable and platform-independent** within the .NET environment while still running efficiently on any machine with the appropriate runtime.

1.4: .NET Components

- The two major components of .NET Framework are the Common Language Runtime and the .NET Framework Class Library.
- The *Common Language Runtime (CLR)* : is the **execution engine** that handles running applications. It provides services like thread management, garbage collection, type-safety, exception handling, and more.
- The *Class Library*: provides a set of **APIs** and **types** for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.

1.4.1:Common Language Runtime (CLR)

- The CLR is the **core of the** .NET framework and is responsible for loading and running C# programs. In another word CLR is the ***heart and soul*** of the .NET Framework.
- As the name suggests, CLR is a **runtime environment** in which program written in C# and other .NET languages are executed. It also supports cross-language interoperability.



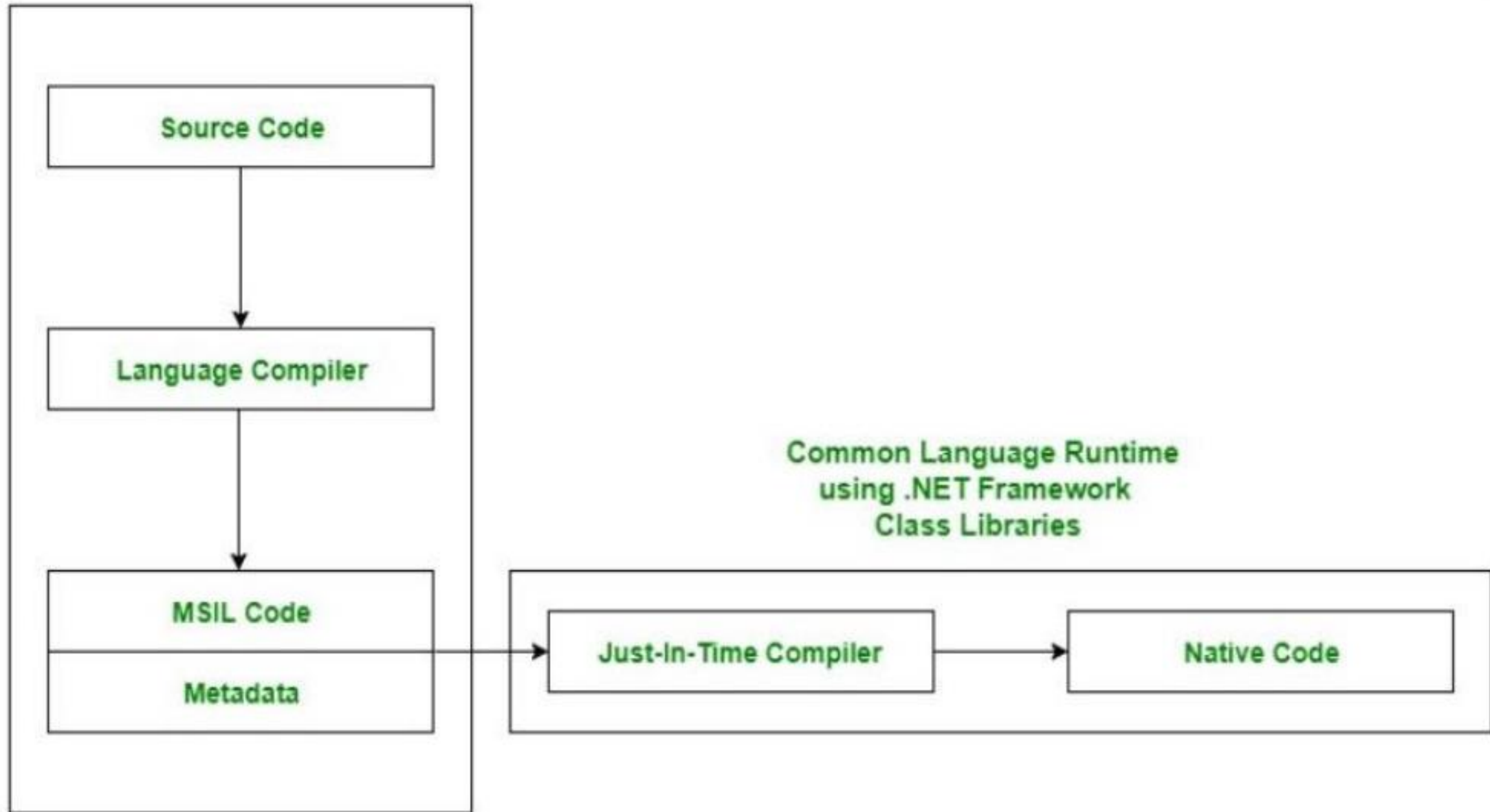
- It is a program ***execution engine*** that loads and executes the program. It converts the program into ***native code***. It acts as an ***interface*** between the framework and operating system.

1.4.1:Common Language Runtime (CLR)

The CLR provides a number of services that include:

1. Loading and execution of programs.
2. Memory isolation for applications.
3. Verification of type safety.
4. Compilation of **IL** into native executable code.
5. Providing metadata.
6. Memory management (automatic garbage collection).
7. Enforcement of security.
8. Interoperability with other systems.
9. Managing exceptions and errors.
10. Supports for task such as debugging and profiling (characterize/describe) .

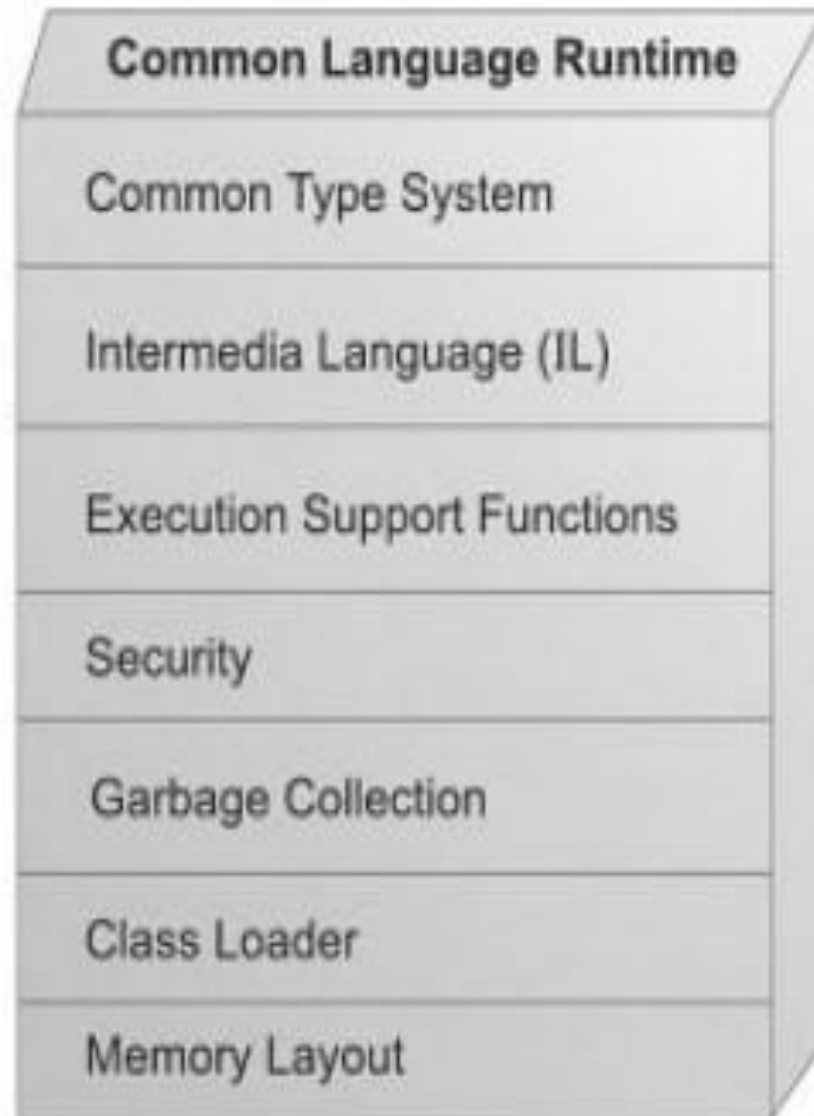
1.4.1. Common Language Runtime (CLR)



1.4.1:Common Language Runtime (CLR)

- **Language specific compiler** compiles the source code into the **MSIL**(Microsoft Intermediate Language) which is also know as the **CIL**(Common Intermediate Language) or **IL**(Intermediate Language) along with its metadata.
- **Metadata** includes the all the types, actual implementation of each function of the program. MSIL is machine **independent** code.
- Compiled code is stored in assemblies—files with a **.dll or .exe** file extension.
- When an app runs, the **CLR** takes the assembly and uses a just-in-time compiler (**JIT** Internally included in CLR) to turn it into **machine code** that can execute on the specific architecture of the computer it is running on.
- During execution, the **IL code** and any requirement from the **base class library** are brought together by the **class loader**.
- The combined code is tested for **type-safety** and then compiled by the **JIT** compiler to produce native code, which is sent to the runtime manager for execution.

Component of CLR



Main components of CLR:

1. Common Language Specification (CLS)
2. Common Type System (CTS)
3. Garbage Collection (GC)
4. Just In – Time Compiler (JIT)

Fig. 2.5 *Component of CLR*

Component 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 the Language Interoperability.
- **Common Type System (CTS)** : Every programming language has its own data type system, so CTS is **responsible for understanding all the data type** of system.
- **Garbage Collection (GC)**: It is used to provide the **Automatic Memory Management** feature.
- **Just In – Time Compiler (JIT)** : It is **responsible for converting the CIL(Common Intermediate Language) into machine code** or native code using the Common Language Runtime environment.

1.5: .NET Framework, .NET Core and .NET standard

- There are multiple ***variants*** of .NET, each supporting a different type of app. The reason for multiple variants is part historical, part technical.

.NET implementations:

- **.NET Framework** -- The ***original*** .NET. It provides access to the broad capabilities of ***Windows*** and Windows Server. It is actively supported, in ***maintenance***.
- **Mono** -- The original community and ***open source*** .NET. A ***cross-platform*** implementation of .NET Framework. Actively supported for Android, iOS, and WebAssembly.
- **.NET (Core)** -- ***Modern .NET***. A ***cross-platform*** and ***open source*** implementation of .NET, rethought for the cloud age while remaining significantly compatible with .NET Framework. Actively supported for ***Linux, macOS, and Windows***.

1.5.1.NET Framework

- .NET framework is a ***virtual machine*** for compiling and executing programs written in different languages like C#, VB.Net, etc.
- It is used to develop ***form-based*** applications, ***web-based*** applications, and web services.
- .NET Framework is ***compatible with*** the **windows operating** system. Although, it was developed to support software and applications on all operating systems.
- .NET Framework supports ***more than 60*** programming languages in which ***11 programming languages*** are designed and developed by Microsoft. The remaining Non-Microsoft languages which are supported by .NET Framework but not designed and developed by Microsoft.

1.5.1.NET Framework

- .NET Framework consists of two major components: the ***common language runtime (CLR)***, which is the *execution engine* that handles running apps, and the ***.NET Framework Class Library***, which provides a library of tested, reusable code that developers can call from their own apps.
- The services that .NET Framework provides to running apps include the following:
 - Memory management.
 - A common type system.
 - An extensive class library.
 - Development frameworks and technologies.
 - Language interoperability.
 - Version compatibility
 - Side-by-side execution

1.5.1.NET Framework

- From approximately **2002-2014**, .NET Framework was the Microsoft framework of choice for all kinds of development.
- .NET Framework versions span from **1.0 to 4.8** and 4.8 will be the last version ever produced.
- .NET Framework is **limited** to Windows operating systems.
- Starting in 2012, Microsoft decided to go **open-source** and **cross-platform** with their next implementation of .NET, called .NET Core.

1.5.2.NET Core

- .NET is a **free, cross-platform, open-source** development platform for building many kinds of applications.
- It operates across **several platforms** and has been **rebuilt** to make .NET fast, scalable, and **modern**.
- It can run programs written in multiple languages with **c#** being the most popular.
- .NET Core is one of Microsoft's big contributions and it offers the following features:
 - ❑ **Cross-Platform:** (Windows, Linux, and macOS operating systems)
 - ❑ **Open Source**
 - ❑ **High Performance**
 - ❑ **Multiple environments** and development mode etc.

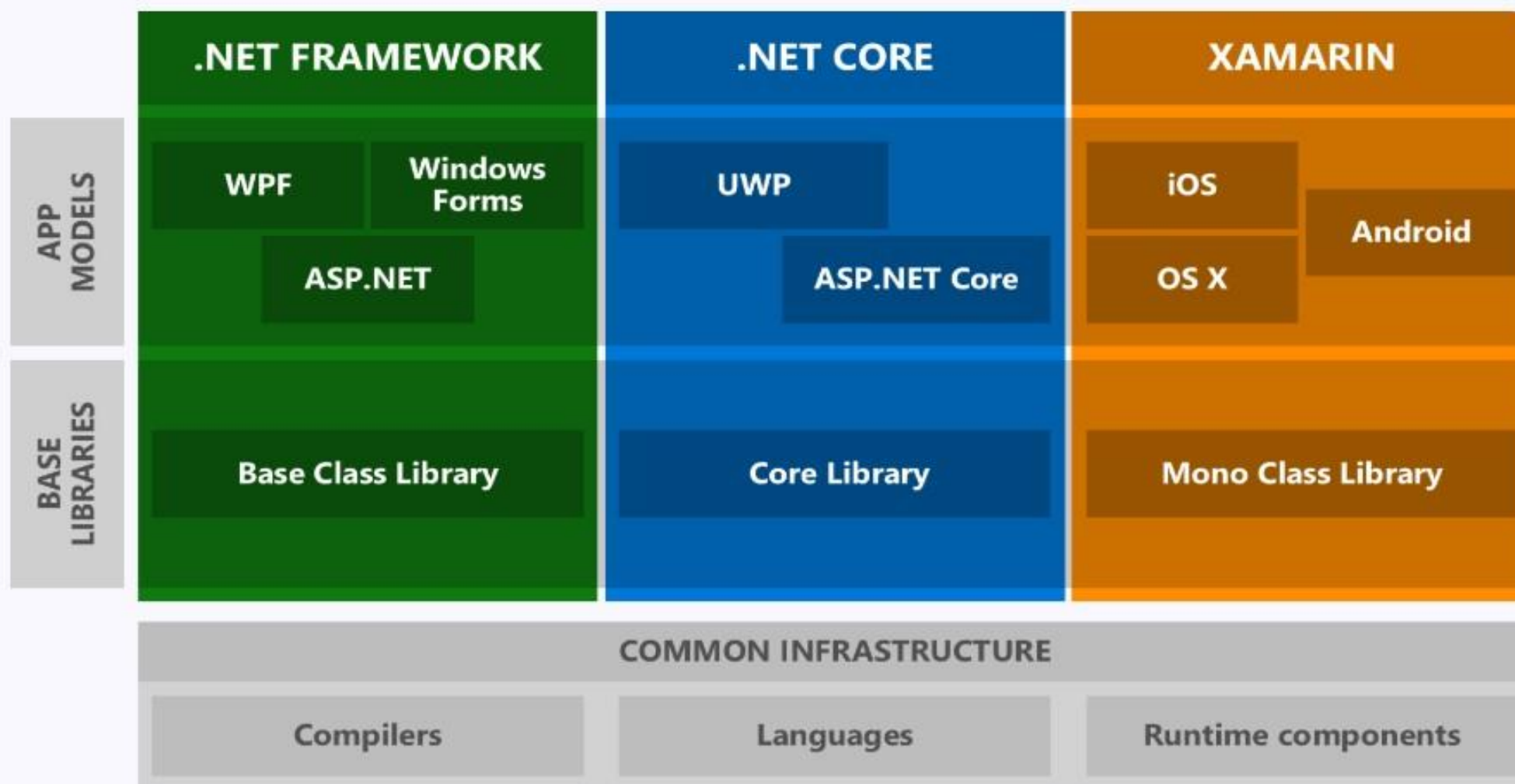
1.5.2.NET Core

- .NET Core was introduced *in 2014* as a response to the issue of a lack of unified framework contracts that can be implemented *across different platforms and devices*.
- NET Core has spanned from version *1.0 to 3.1* with 3.1 being the last.
- The main focus of .NET Core was to make .NET run on *more operating* systems than just Windows and to go *open source*.
- In November of *2020*, a new implementation of .NET arrived that dropped the *Core* naming scheme entitled *.NET 5*.

1.5.3.NET Standard

- The **divide** between .NET Framework and .NET Core would be a major problem for *code sharing and reuse*. This is where .NET Standard fits in.
- .NET Standard is a **specification** for a set of .NET APIs that are available to both .NET Framework and .NET Core.
- .NET Standard is also a powerful tool for migration from .NET Framework to .NET Core.
- The .NET Standard was introduced in **2016** as a unified collection of ***specifications*** that are tailored to have cross-platform implementations.
- This allows us to use the ***same stack*** to develop for many different operating systems and devices.
- The motivation behind .NET Standard was to establish greater ***uniformity*** in the .NET ecosystem.
- **.NET 5** and later versions adopt a different approach to establishing uniformity that eliminates the need for .NET Standard in most scenarios.

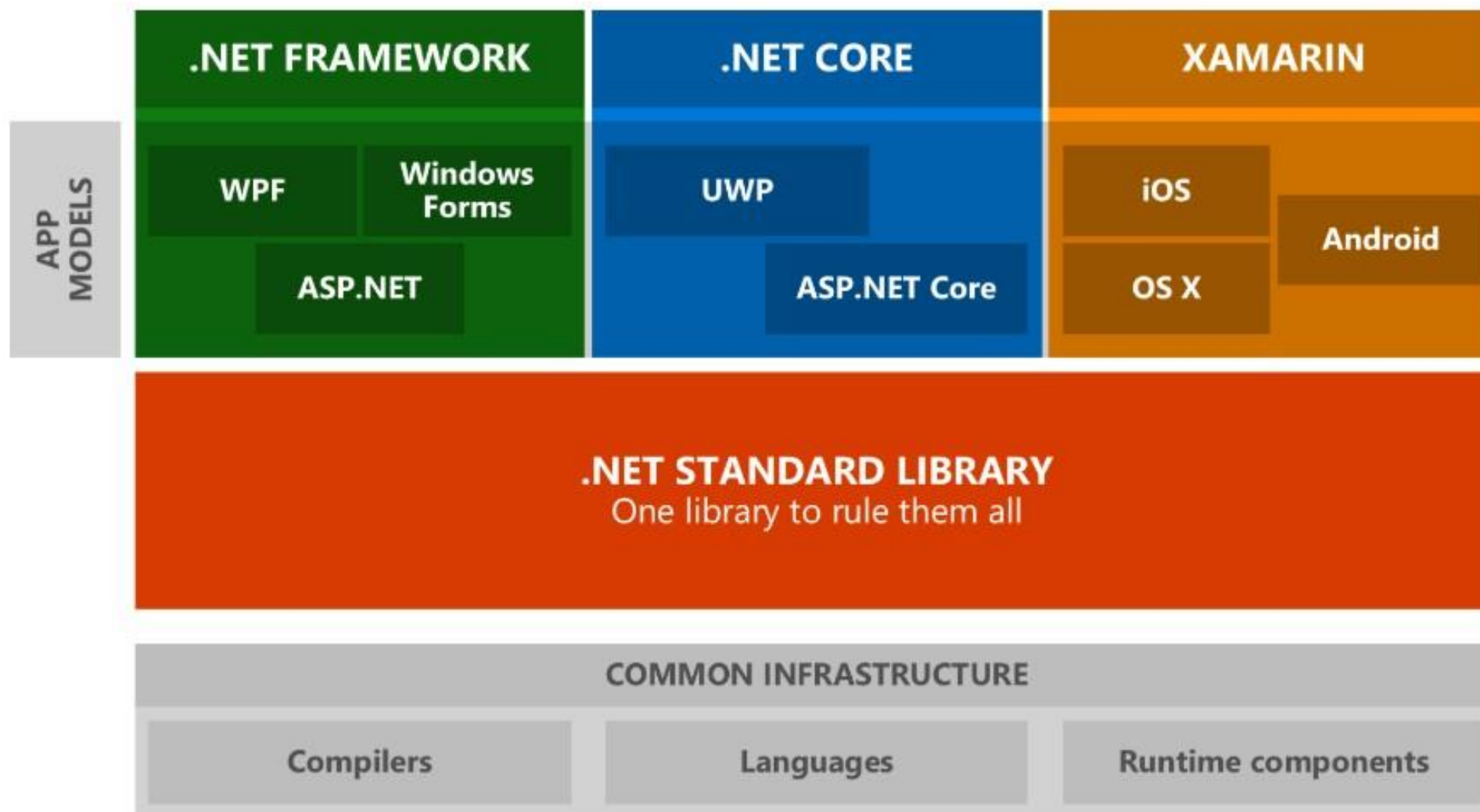
1.5.3.NET Standard



Before .NET Standard:

Basically there are ***three*** major flavors of .NET, which means we have to master/learn ***three different base class libraries*** in order to write code that works across all of them.

1.5.3.NET Standard



After .NET Standard:

.NET Standard makes sure that ***all .NET platforms share the same API*** for the base class library. Once we learn how to use it in our desktop application we know how to use it in our mobile application or other services.

1.6.NET Tools & Editors

 Visual Studio	Visual Studio :Fully-featured integrated development environment (IDE) on Windows for building every type of .NET application.
 Visual Studio Code	Visual Studio Code :Develop on Linux, macOS, or Windows to build cross-platform websites and services. Install the C# Dev Kit to get the best experience.
 OmniSharp	OmniSharp :Cross-platform .NET development in editors such as Atom, Brackets, Sublime Text, Emacs, and Vim.

1.6.NET Tools & Editors

 JetBrains Rider	JetBrains Rider :Cross-platform .NET IDE built using IntelliJ and ReSharper technology. It offers support for .NET and .NET Core applications on all platforms
 .NET CLI	.NET CLI :The <i>command-line interface</i> (CLI) comes with the .NET SDK for developing cross-platform websites and services on Linux, macOS, and Windows.
 Ionide	Ionide: A Visual Studio Code package suite for working with the <i>F# programming</i> language on Linux, macOS, or Windows.

1.6.1. Visual Studio

- Visual Studio is a powerful **developer tool** that we can use to complete the entire development cycle in one place.
- It's a comprehensive integrated development environment (**IDE**) that we can use to **write, edit, debug**, and build code. Then **deploy** our app.
- Visual Studio includes ***compilers, code completion tools, source control, extensions***, and many other features to enhance every stage of the software development process.
- Visual Studio supports different parts of the software development cycle. Such as :
 - ☐ Develop our code
 - ☐ Debug our code
 - ☐ Deploy our app
 - ☐ AI-assisted development
 - ☐ Test our code
 - ☐ Build our app
 - ☐ Version control

1.6.2.Why use Visual Studio?

- Visual Studio provides developers a feature ***rich development environment*** to develop high-quality code efficiently and collaboratively.
- ***Workload-based installer***:- We can install only what we need
- ***Powerful coding tools and features***:- everything we need to build our apps in one place
- ***Multiple language support*** :- code in C++, C#, JavaScript, TypeScript, Python, and more
- ***Cross-platform development*** :- build apps for any platform
- ***Version control integration*** :- collaborate on code with team mates
- ***AI-assisted development*** :- write code more efficiently with AI assistance

1.6.3.Install/Setup Visual Studio

To setup the visual studio we have to follow the following steps:

- **Step 1** - Make sure our computer is ready (have required configuration RAM,CPU, OS etc) for Visual Studio
- **Step 2** - Determine which version and edition of Visual Studio to install
- **Step 3** - Initiate the installation
- **Step 4** - Choose workloads
- **Step 5** - Choose individual components (optional)
- **Step 6** - Install language packs (optional)
- **Step 7** - Select the installation location (optional)
- **Step 8** - Sign in to your account (optional)
- **Step 9** - Start developing

1.6.4. Visual Studio Code

- Visual Studio Code is an ***open-source and lightweight*** source code editor which runs on Windows, Mac, and Linux.
- Visual Studio Code (also called VS Code) is like the ***mini version of Visual Studio***.
- It comes with **built-in support** for **JavaScript, TypeScript** and **Node.js** but we can use it to code in any language (such ***as C++, C#, Java, Python, PHP, Go, .NET***) by downloading the relevant extensions (API).
- Since it supports JavaScript, TypeScript, and Node JS by default, we get a ***debugger and intellisense (code-completion aid), too***. But to get intellisense, a compiler, and debuggers for other languages, you have to **download relevant extensions**.

1.6.4. Visual Studio Code

- Some of the extensions are made by **Microsoft**, but a lot of others are **third-party** extensions.
- Unlike Visual Studio, we don't need much space to download VS Code. We might **not need more than 200 MB** of disk space to download it.
- Now we know that Visual Studio is an **IDE** and Visual Studio Code is a **text editor**. So let's summarize their main differences as follows.

1.7. Difference between Visual Studio and Visual Studio Code

Visual Studio	Visual Studio Code
Visual Studio is an Integrated Development Environment , also known as an IDE.	Visual Studio Code is code editor . A developer can easily edit their code.
Visual Studio is slower when it comes to performing across different platforms. The processing speed is slower.	VS Code is comparatively faster .
Visual studio has free editor for developer to use but also comes with a better and paid IDE version .	VS Code is completely free of cost and is open-source.
Visual Studio engages the best and the most advanced IntelliSense .	IntelliSense is comparatively not up to the mark in VS Code.
The overall download size is pretty large .	Compared to visual studio, visual studio code is pretty lightweight . It doesn't require a heavy or large download.
VS requires more space to work better and smoother.	VS Code comparatively does not need a lot of space to run. It can easily run on 300 MBs of ram .
Visual Studio only runs on Windows and mac-OS .	VS Code can run on mac-OS, Windows as well as Linux
Not many professionally developed plugins are available for Visual Studio	VS Code comes with a wide range of professionally curated plugins and extensions to meet all kinds of editing and compiling needs.

The End