060010407-GUI Programming

Unit-1 Introduction to .NET Framework and C# Fundamentals

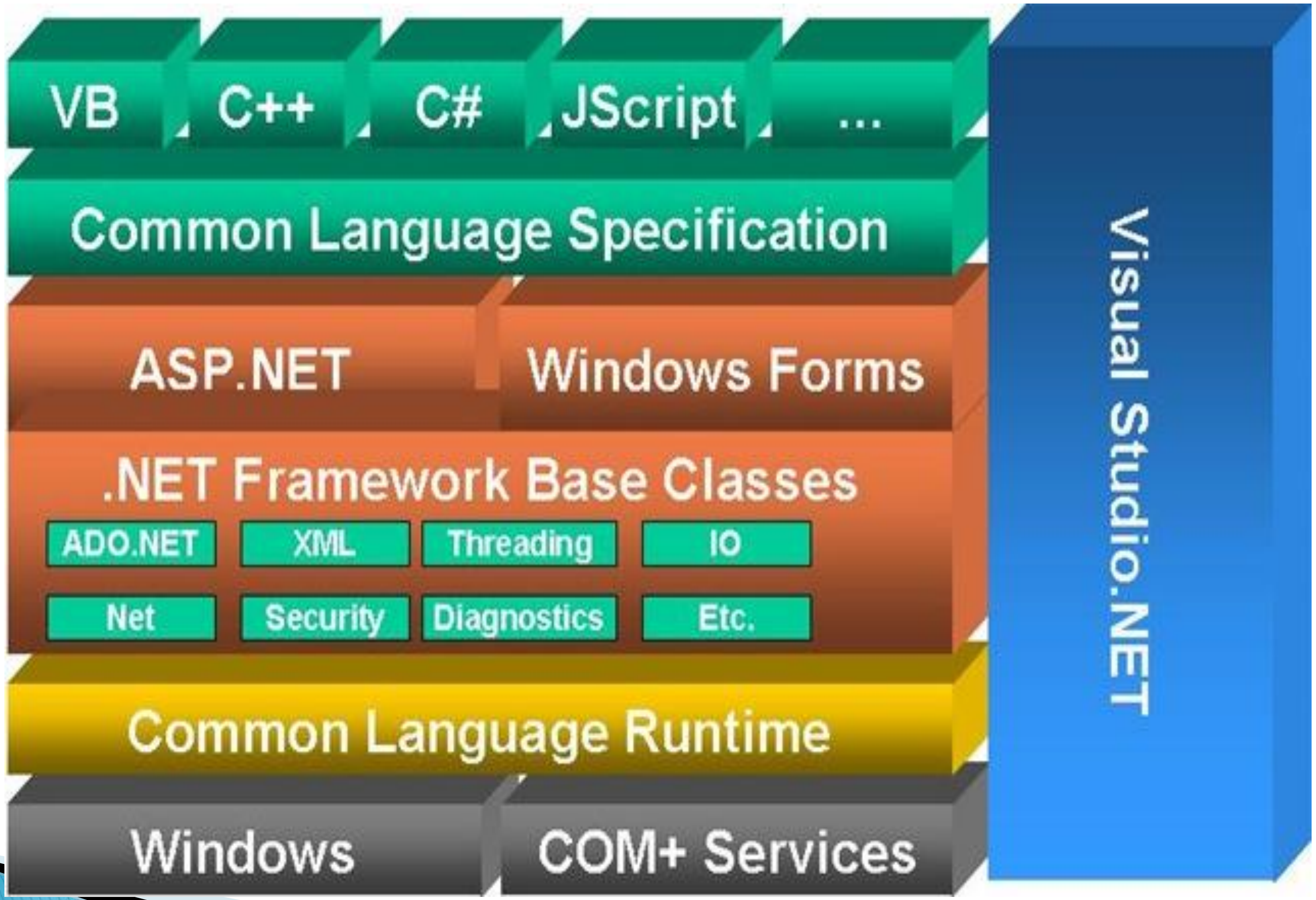
.NET framework & architecture

- The .NET Framework is a new and revolutionary platform created by Microsoft for developing applications.
- platform for application developers
- Framework that supports Multiple Language and Cross language integration.
- has IDE (Integrated Development Environment).
- set of utilities or can say building blocks of your application system.
- provides GUI in a GUI manner.

.NET framework architecture

- platform independent
- provides interoperability between languages i.e. Common Type System (CTS) .
- includes the .NET Common Language Runtime (CLR), which is responsible for maintaining the execution of all applications developed using the .NET library.
- consists gigantic library of code.

.NET framework & architecture

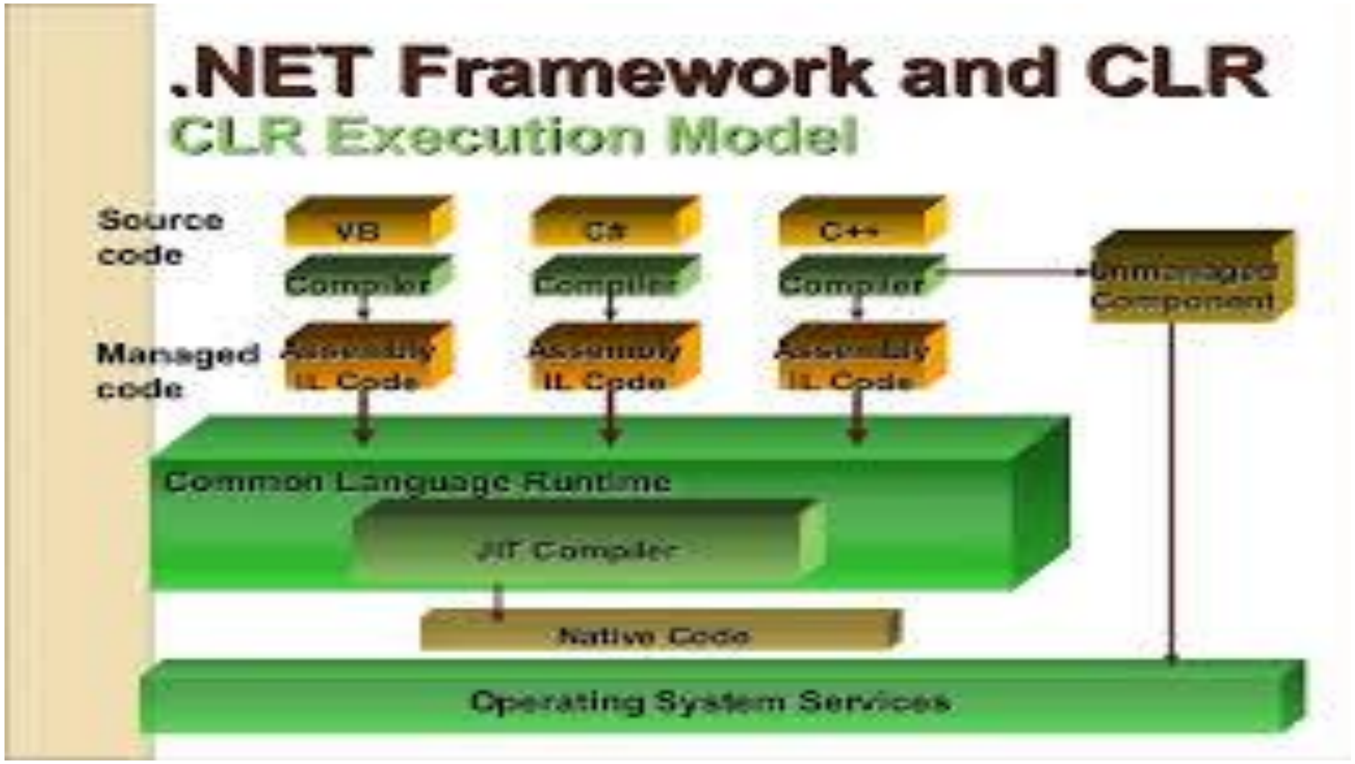


Common Language Runtime(CLR)

- .Net Framework has two core components.

1. CLR(Common language runtime).
2. Base Class Library.

Common Language Runtime(CLR)



Common Language Runtime(CLR)

- In order for C# code to execute, it must be converted into a language that the target operating system understands, known as *native code*.
- This conversion is called *compiling* code, an act that is performed by a *compiler*.
- Under the .NET Framework, this is a two-stage process.

CIL and JIT

- When you compile code that uses the .NET Framework library, you don’t immediately create operating-system-specific native code. Instead, you compile your code into *Common Intermediate Language (CIL)* code.
- This code isn’t specific to any operating system (OS) and isn’t specific to C# or Other .NET languages — Visual Basic .NET.
- That is the job of a just-in-time (JIT) compiler, which compiles CIL into native code that is specific to the OS and machine architecture being targeted.

Assemblies

- When you compile an application, the CIL code created is stored in an *assembly*.
- include both executable application files that you can run directly from Windows without the need for any other programs (.exe extension) and libraries (.dll extension) for use by other applications.
- include *meta* information (that is, information about the information contained in the assembly, also known as *metadata*) and optional *resources* (additional data used by the CIL, such as sound files and pictures).

Assemblies

- The meta information enables assemblies to be fully self-descriptive.
- It is often useful to place the reusable code in a place accessible to all applications.
- In the .NET Framework, this is the *global assembly cache (GAC)*. Placing code in the GAC is simple — you just place the assembly containing the code in the directory containing this cache.

Managed and Unmanaged Code

- Code written using the .NET Framework is *managed* when it is executed (a stage usually referred to as *runtime*).
- This means that the CLR looks after your applications by managing memory, handling security, allowing cross-language debugging, and so on.
- By contrast, applications that do not run under the control of the CLR are said to be *unmanaged,* and certain languages such as C++ can be used to write such applications
- for example, access low-level functions of the operating system.

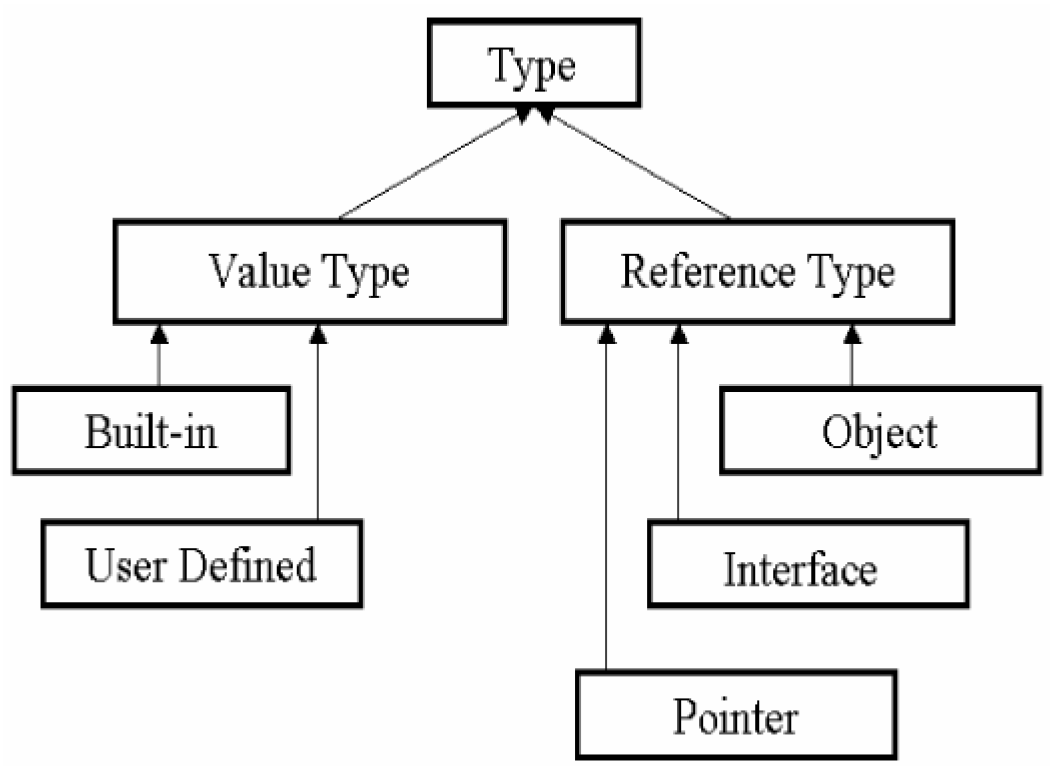
Garbage Collection

- .NET method of making sure that the memory used by an application is freed up completely when the application is no longer in use.
- Prior to .NET this was mostly the responsibility of programmers.
- NET garbage collection works by periodically inspecting the memory of your computer and removing anything from it that is no longer needed.
- There is no set time frame for this, it might happen thousands of times a second, once every few seconds.

Common type system.

- This type system permits .NET to support multiple languages-a reasonable expectation, considering that languages have great difficulty interoperating if they are based on dissimilar types.
- This multi language support must take into account both procedural and object-oriented languages.
- There are two root types available
- Value type
- Reference type

Common type system.

- Value types are allocated on the stack, while reference types are allocated on the heap.
- 

Common type System.

- The built-in value types consist of primitive types such as integers and longs. User-defined value types are types such as structures and enumerations.
- An object reference type can be said to be a class.

Common Language Specification.

- defines the minimum requirements that must be realized before a language is considered .NET compliant.
- primarily of use to language and framework designers.
- specifies a minimum subset of CTS types to be supported and a set of usage rules.

Common intermediate language(MSIL)

- CIL or MSIL or simply IL, is a CPU-independent instruction set that allows .NET to support multiple languages.
- operating system- independent as long as the platform can host an implementation of the CLI
- The major difference between IL and Java's byte code is that IL is only an intermediate step on the path to execution. The Java Virtual Machine runs an application by interpreting its byte codes.
- .NET compilers, on the other hand, produce an assembly that is loaded by the CLR at runtime.

.NET application runs

- Once the CLR has been loaded and initialized and an application domain created, the application is ready to run.
- At this point, the CLR loads the application in the domain and then verifies that the IL compiled code is type safe.
- If verification fails, an exception is thrown.
- Once verification succeeds or is skipped, the JIT compiler compiles the IL to native code.
- The resulting native code is cached and used for subsequent calls of the method. It does this by marking the method, which in turn points subsequent calls to the binary code.

Similarities with java

- All Objects are References.
- Garbage Collection
- Both C# and Java are Type-Safe Languages
- Both C# and Java Are "Pure" Object-Oriented Languages
- Single Inheritance
- Built-in Thread and Synchronization Support
- Formal Exception Handling
- Built-in Unicode Support

Differences with java

- C# includes more primitive types and the functionality to catch arithmetic exceptions.
- C# includes a large number of notational conveniences over Java.
- Event handling is a "first class citizen"—it is part of the language itself.
- C# implements properties as part of the language syntax.
- C# allows switch statements to operate on strings.

Differences with java

- C# allows anonymous methods providing closure functionality.
- C# allows iterator that employs co-routines via a functional-style yield keyword.
- C# has support for output parameters, aiding in the return of multiple values.
- C# has the ability to alias namespaces.
- C# provides integration with COM.
- C# has "Explicit Member Implementation" which allows a class to specifically implement methods of an interface, separate from its own class methods.

Differences with java

- Java supports checked exceptions for better enforcement of error trapping and handling.

THANK YOU