.NET Tutorial

Microsoft .NET

- .NET is the Microsoft Web services strategy to connect information, people, systems, and devices through software.

- Integrated across the Microsoft platform, .NET technology provides the ability to quickly build, deploy, manage, and use connected, security-enhanced solutions with Web services.

- .NET was originally called NGWS (Next Generation Windows Services).

- Web services are self-describing software modules, semantically encapsulating discrete functionality, wrapped in and accessible via standard Internet communication protocols such as XML and SOAP.

Microsoft .NET

- The Microsoft. NET strategy was presented in June 2000:
  - .NET is Microsoft's new Internet and Web strategy
  - .NET is NOT a new operating system
  - .NET is a new Internet and Web based infrastructure
  - .NET delivers software as Web Services
  - .NET is a framework for universal services
  - .NET is a server centric computing model
  - .NET will run in any browser on any platform
  - .NET is based on the newest Web standards
- The motivations driving this vision are:
  - Object-oriented programming
  - Compiled once and run everywhere
 - Service-oriented application

.NET Standard

- A standard distributed application should run on almost any computer in the world. .NET have been standardized by ECMA (http://www.ecma-international.org/).

- Clients must be Standard Internet Browsers.
  - All clients use standard Internet browsers like Firefox, Internet Explorer, or Netscape running on Windows, Linux, or Mac computers.

- Servers must be Standard Internet Servers.
  - Standard Internet servers such as Apache, Tomcat, or Internet Information Services (IIS).

- Applications must use Internet Communication
  - Servers should be requested with standard stateless Internet HTTP requests. Servers should respond with a standard stateless Internet HTTP response.

.NET Infrastructure

- .NET Infrastructure
  - **Development Tools**: The .NET framework is a programming environment.
  - **Web Services**: .NET provides a standard syntax for the language defined for sites providing **web services**.
  - **Specialized servers** (.NET servers): Severs that work with .NET such as SQL Server 2000 is described as .NET enabled.
  - **Devices** that are running on .NET.
- .NET implementation
  - Windows: Microsoft .NET
  - Linux: Mono, DotGNU
  - MacOS X: Mono

.NET Infrastructure

- Development Tools
- Web services
- Specialized servers
- Clients
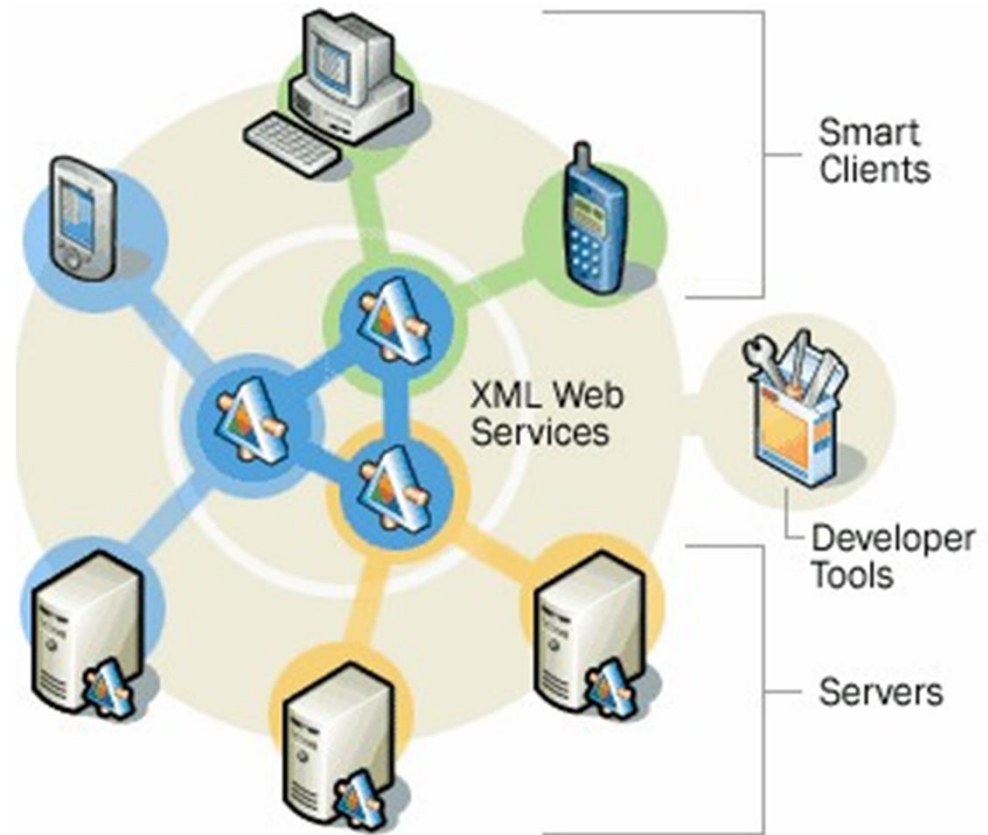
.NET Framework

- The .NET Framework is a common environment for building, deploying, and running Web Services and Web Applications (Microsoft JDK?). It contains:
  - The Common Language Runtime,
  - The .NET Framework Classes, and
  - higher-level features like ADO.NET, ASP.NET and Window Forms for developing desktop applications.

- The Common Language Runtime (CLR) (Microsoft JRE?) manages the execution of code compiled for the .NET platform. The CLR has two features:
  - Its specification has been opened up so that it can be ported to non-Windows platforms.
  - Any number of different languages can be used to manipulate the .NET framework classes, and the CLR will support them.

.NET Framework Standard

- The CLI (Common Language Infrastructure) is the definition of the fundamentals of the .NET framework:
  - the Common Type System (CTS) and metadata,
  - the Virtual Execution Environment (VES) and
  - its use of intermediate language (IL), and
  - the support of multiple programming languages via the Common Language Specification (CLS).
- The CLI is documented through ECMA - see http://msdn.microsoft.com/net/ecma/ for more details.
- The CLR (Common Language Runtime) is Microsoft's primary implementation of the CLI.

.NET Framework

- The Common Language Runtime – An execution environment providing
  - Garbage collection
  - Security
  - Exception Handling
  - Multithreading
  - Memory Management
  - Type Safety
- The .NET Framework Classes
  - A collection of reusable classes
  - Classes grouped into logical collections, called *namespaces* like System.IO
  - Classes grouped into physical collections, called *assemblies* like System.IO.dll
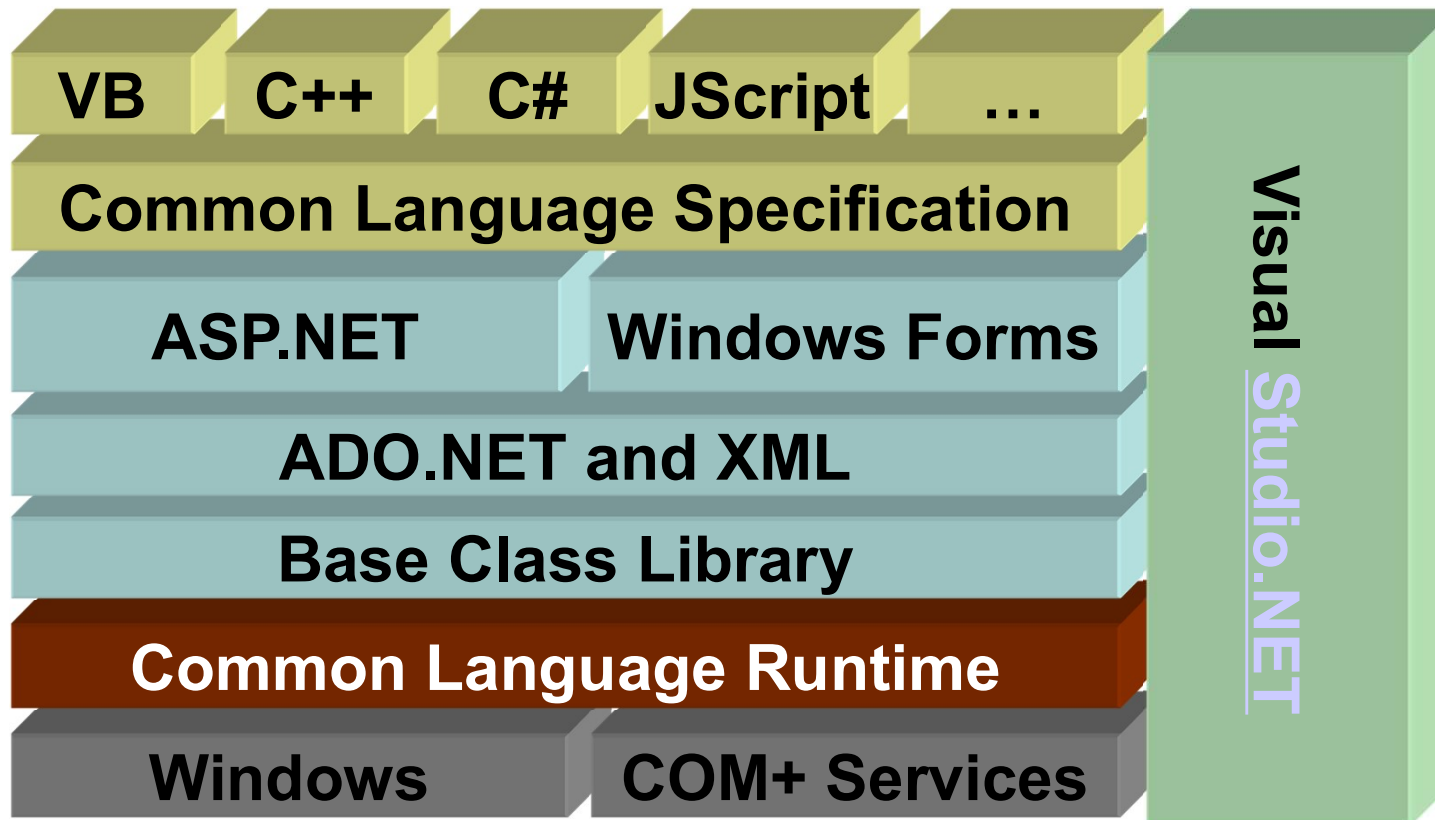
.NET Framework

- ADO.NET
  - A disconnected programming environment for working with data sources
  - Ability to create a pseudo database with tables and relationships in your applications to hold cached copies of data from different sources
  - Support for XML

- ADO.ASP
  - As easy to create Web Applications as it is to create Windows Applications
  - Full power of a programming language such as VB.NET and C# compared to VBScript in ASP 2.0
  - Many new features which previous ASP 2.0 programmers had to code themselves

.NET Framework

- The .NET Framework is language neutral.
  - Currently it supports C++, C#, Visual Basic, JScript (The Microsoft version of JavaScript) and COBOL.
  - Third-party languages such as Eiffel, Perl, Python and others are also available for building future .NET Framework applications.
- Not all of the supported languages fit entirely neatly into the .NET framework.
- The Visual Studio.NET is a common development environment for the new .NET Framework.
  - It provides a feature-rich application execution environment, simplified development and easy integration between a number of different development languages.

.NET Framework

VB	C++	C#	JScript	…

Common Language Specification

ASP.NET	Windows Forms

ADO.NET and XML

Base Class Library

Common Language Runtime

Windows	COM+ Services

Visual Studio.NET

.NET Software

- ## Windows .NET
  - Windows XP form the backbone of .NET.
  - Windows .NET is the new generation Windows. It will provide support for all the .NET building blocks and .NET digital media.
  - Windows .NET will be self-supporting with updates via Internet as users need them.
- ## Office .NET
  - A new version of Microsoft Office - Office .NET - will have a new .NET architecture based on Internet clients and Web Services.
  - With Office .NET, browsing, communication, document handling and authoring will be integrated within a XML-based environment which allow users to store their documents on the Internet.

.NET Software

- Active Server Pages - ASP .NET is the latest version of ASP. It includes Web Services to link applications, services and devices using HTTP, HTML, XML and SOAP. New in ASP .NET:
  - New Language Support
  - Programmable Controls
  - Event Driven Programming
  - XML Based Components
  - User Authentication
  - User Accounts and Roles
  - High Scalability
  - Compiled Code
  - Easy Configuration
  - Easy Deployment
  - Not ASP Compatible
  - Includes ADO .NET

.NET Software

- What is ADO?
    - ADO is a Microsoft technology
    - ADO stands for **A**ctiveX **D**ata **O**bjects
    - ADO is a Microsoft Active-X component
    - ADO is automatically installed with Microsoft IIS
    - ADO is a programming interface to access data in a database
- ADO.NET is the next evolutionary step in data access technology. Employing the aforementioned ADO technology, ADO.NET expands this by incorporating XML into a standard model to not only relational data models but also text based XML data.

.NET Software

- Visual Studio .NET
  - The latest version of Visual Studio - Visual Studio .NET - incorporates ASP .NET, ADO .NET, Web Services, Web Forms, and language innovations for Visual Basic.
  - The development tools have deep XML support, an XML-based programming model and new object-oriented programming capabilities.

- Visual Basic .NET
  - Visual Basic .NET has added language enhancements, making it a full object-oriented programming language.

.NET Web Services

- ## What are Web Services?
  - – Web services are small units of code
  - – Web services are designed to handle a limited set of tasks
  - – Web services use XML based communicating protocols
  - – Web services are independent of operating systems
  - – Web services are independent of programming languages
  - – Web services connect people, systems and devices

- ## Small Units of Code
  - – Web services are small units of code designed to handle a limited set of tasks.
  - – An example of a web service can be a small program designed to supply other applications with the latest stock exchange prices. Another example can be a small program designed to handle credit card payment.

.NET Web Services

- ## XML Based Web Protocols
  - Web services use the standard web protocols HTTP, XML, SOAP, WSDL, and UDDI.

- ## HTTP
  - HTTP (Hypertext Transfer Protocol) is the World Wide Web standard for communication over the Internet. HTTP is standardized by the World Wide Web Consortium (W3C).

- ## XML
  - XML (eXtensible Markup Language) is a well known standard for storing, carrying, and exchanging data. XML is standardized by the W3C.

.NET Web Services

- ## SOAP
  - SOAP (Simple Object Access Protocol) is a lightweight platform and language neutral communication protocol that allows programs to communicate via standard Internet HTTP. SOAP is standardized by the W3C.

- ## WSDL
  - WSDL (Web Services Description Language) is an XML-based language used to define web services and to describe how to access them. WSDL is a suggestion by Ariba, IBM and Microsoft for describing services for the W3C XML Activity on XML Protocols.

- ## UDDI
  - UDDI (Universal Description, Discovery and Integration) is a directory service where businesses can register and search for web services.

.NET Servers

- ## SQL Server
  - SQL Server is a fully web-enabled database.
  - SQL Server has strong support for XML and HTTP which are two of the main infrastructure technologies for .NET.
  - Some of the most important new SQL Server features are direct access to the database from a browser, query of relational data with results returned as XML, as well as storage of XML in relational formats.
  - The current version is SQL Server 2000. SQL Server 2005 is going to be released soon.

- ## Internet Information Services
  - IIS has strong support for more programming to take place on the server, to allow the new Web Applications to run in any browser on any platform.
  - The current version is SQL IIS 6.0.

.NET Servers

- ## Internet Storages
  - .NET offers secure and addressable places to store data and applications on the Web. Allowing all types of Internet devices (PCs, Palmtops, Phones) to access data and applications.

- ## Internet Dynamic Delivery
  - Reliable automatic upgrades by demand and installation independent applications.
  - .NET will support rapid development of applications that can be dynamically reconfigured.

- ## Internet Identity
  - NET supports many different levels of authentication services like passwords, wallets, and smart cards.

.NET Servers

- Internet Messaging
  - NET supports integration of messaging, e-mail, voice-mail, and fax into one unified Internet Service, targeted for all kinds of PCs or smart Internet devices.

- Internet Calendar
  - .NET supports Internet integration of work, social, and private home calendars. Allowing all types of Internet devices (PCs, Palmtops, Phones) to access the data.

- Internet Directory Services
  - .NET supports a new kind of directory services that can answer XML based questions about Internet Services, far more exactly than search engines and yellow pages.
  - These services are built on the UDDI standard.

.NET Component Model

- .NET allows all classes to be reused at the binary level. You simply write a .NET class, which then becomes a part of an assembly and supports plug-and-play.

- An assembly is the basic unit of deployment and versioning, consisting of a manifest, a set of one or more modules (usually DLL), and an optional set of resources

- Microsoft .NET provides a simpler way to build and deploy components.

Interoperability

- In distributed systems interoperability is a major issue. A number of standards and architectures have been developed to address these issues. For example:

  - **Representation standards,** such as External Data Representation (XDR) and Network Data Representation (NDR), address the issue of passing data types between different machines.

  - **Architecture standards,** such as the Distributed Computing Environment's (DCE) Remote Procedure Call (RPC), the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA), and Microsoft's Component Object Model (COM), address the issue of calling methods across language, process, and machine boundaries.

  - **Language standards** such as ANSI C allow the distribution of source code across compilers and machines.

  - **Execution environments** such as virtual machines (VM) of SmallTalk and Java, allow code to execute on different physical machines by providing a standardized environment for execution.

Language interoperability

- Language interoperability is a problem that is not well solved.
  - Language interoperability does not just refer to a standardized calling model, such as COM and CORBA, but a schema that allows classes and objects in one language to be used as first class citizens in another language.
  - For example, it should be possible for Python code to instantiate a C++ object that inherits from an Eiffel class.

- The Microsoft .NET Framework is intended to address the issue of language interoperability.

- Common language runtime (CLR) is core component of the .NET Framework for the language interoperability.
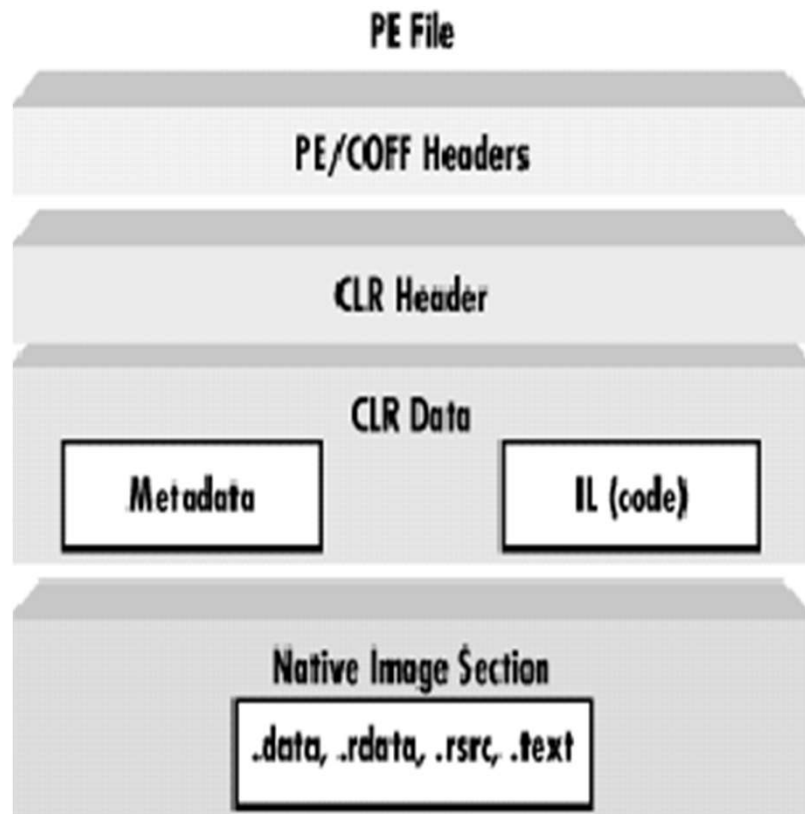
Common Language Runtime

- The CLR manages and executes code written in .NET languages and is the basis of the .NET architecture, similar to the Java Virtual Machine (JVM).

- The CLR activates objects, performs security checks on them, lays them out in memory, executes them, and garbage-collects them.

- The CLR executables or **Portable Executable** (PE) files, which are .NET assemblies or units of deployment.

  – The CLR is the runtime engine that loads required classes, performs just-in-time compilation on needed methods, enforces security checks, and accomplishes a bunch of other runtime functionalities.

  – The CLR executables are either EXE or DLL files that consist mostly of metadata and code.
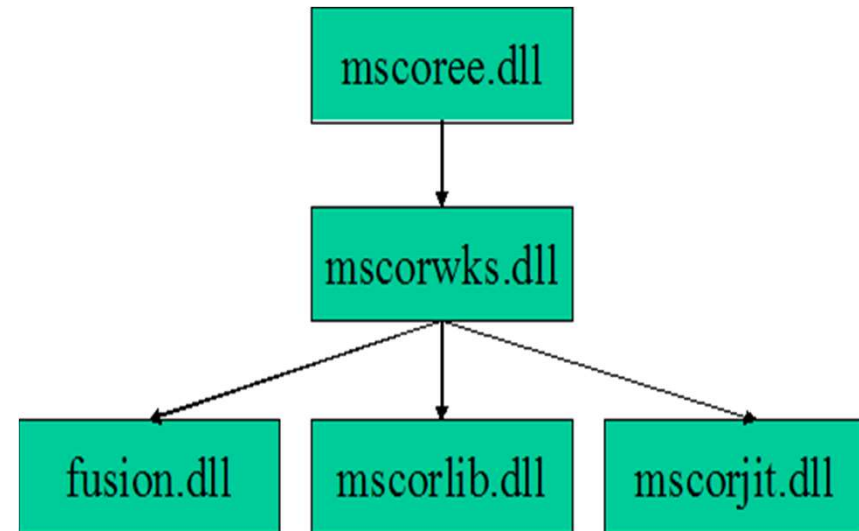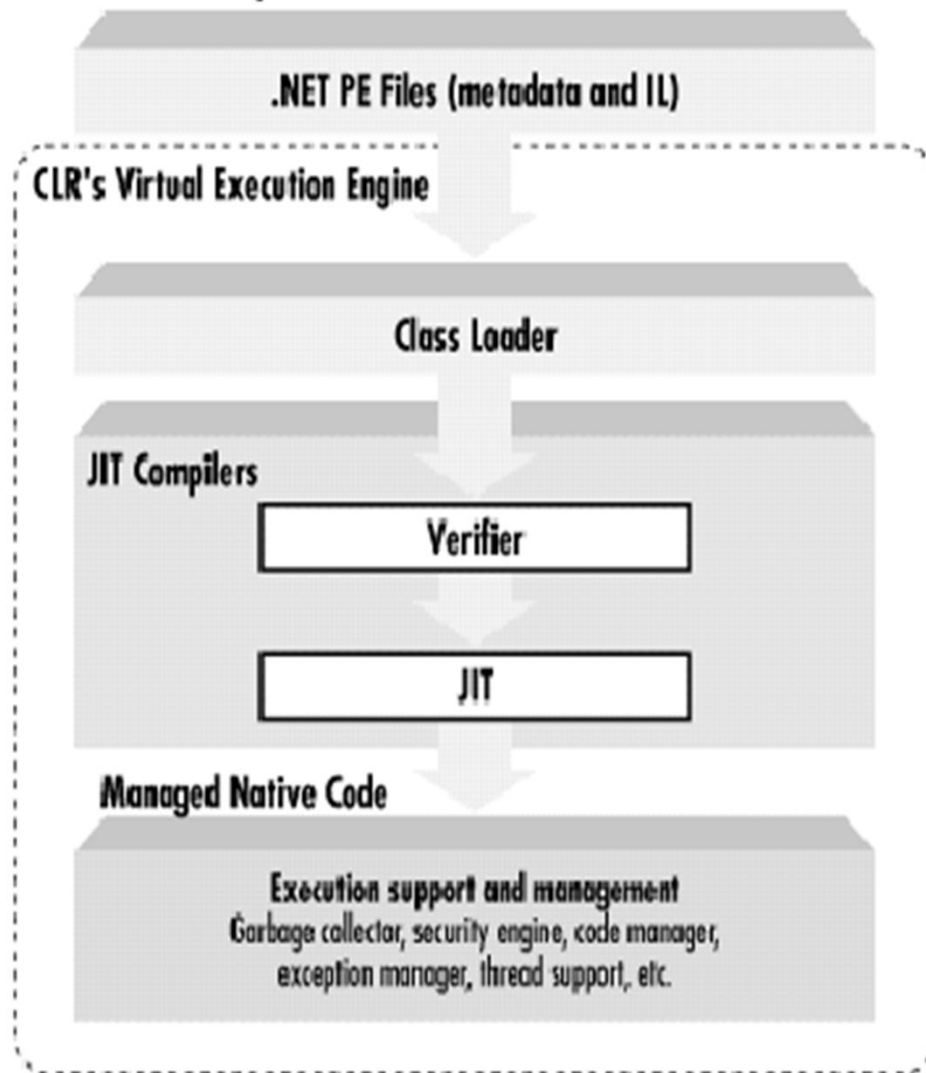
Common Language Runtime
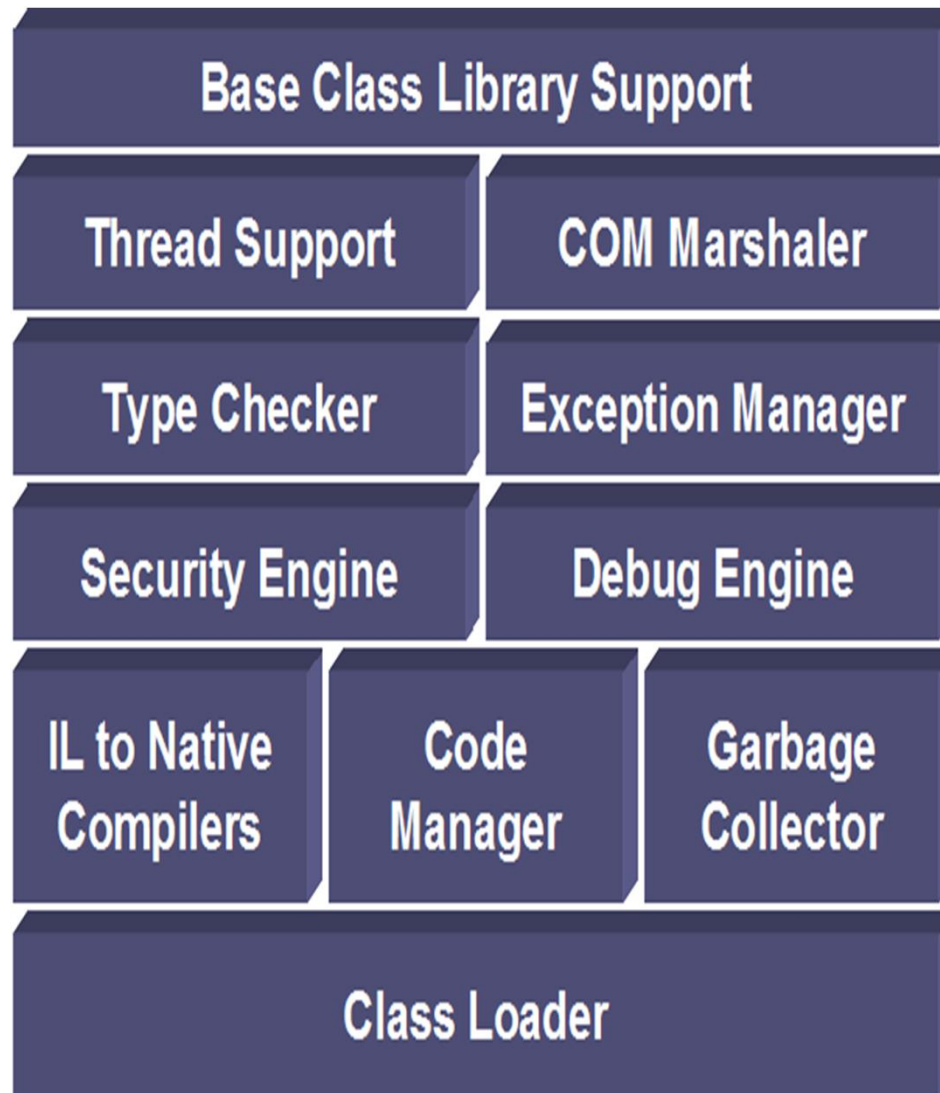
- Manages running code
  - Verifies type safety
  - Provides garbage collection, error handling
  - Code access security for semi-trusted code
- Provides common type system
  - Value types – integer, float, user defined, etc)
  - Reference types – Objects, Interfaces
- Provides access to system resources
  - Native API, COM, etc.

.NET Portable Executable (PE) Files

PE File

PE/COFF Headers

CLR Header

CLR Data

| .Metadata | IL (code) |

Native Image Section

| .data, .rdata, .rsrc, .text |

- A Windows executable, EXE or DLL must conform to the PE file format, which is derivative of the Microsoft **Common Object File Format** (COFF).

- Any compiler that wants to generate Windows executables must obey the PE/COFF specification.

- A .NET PE file consist of 4 parts as shown in the Figure.

How CLR acts as a Virtual Machine?

.NET PE Files (metadata and IL)

CLR's Virtual Execution Engine

Class Loader

JIT Compilers

Verifier

JIT

Managed Native Code

Execution support and management
Garbage collector, security engine, code manager,
exception manager, thread support, etc.

mscoree.dll

mscorwks.dll

fusion.dll   mscorlib.dll   mscorjit.dll

mscoree.dll (execution engine)

mscorwks.dll (does most initialization)

mscorjit.dll (contains JIT)

mscorlib.dll (BCL)

fusion.dll (assembly binding)

Common Language Runtime

- The Common Language Runtime (CLR) consists of three main components:
  - A **type system**, which supports many of the types and operations found in modern programming languages.
  - A **metadata system**, which allows metadata to be persisted with types at compile time and then interrogated by the execution system at run time.
  - An **execution system**, which executes .NET Framework programs, utilizing the metadata system information to perform services such as memory management.

Common Language Runtime

- The following figure shows one view of the relationship between elements of the runtime.
  - At the top of diagram, the source file may hold a definition of a new type in any .NET languages, such as C#.
  - When this file is compiled by a .NET Framework C# compiler, the resulting Microsoft Intermediate Language (MSIL) is saved with the new type's metadata. The metadata format used is independent of the programming language in which the type was defined.
  - Once the MSIL for this new type exists, other source files, possibly written in other languages, such as C++, Eiffel, Python, or Visual Basic®, can import this file.
  - At runtime, the execution system compiles, loads, and starts executing an MSIL file. References to a type defined in a different MSIL file cause that file to be loaded, its metadata is read, and then instances of the new type can be exposed to the runtime.
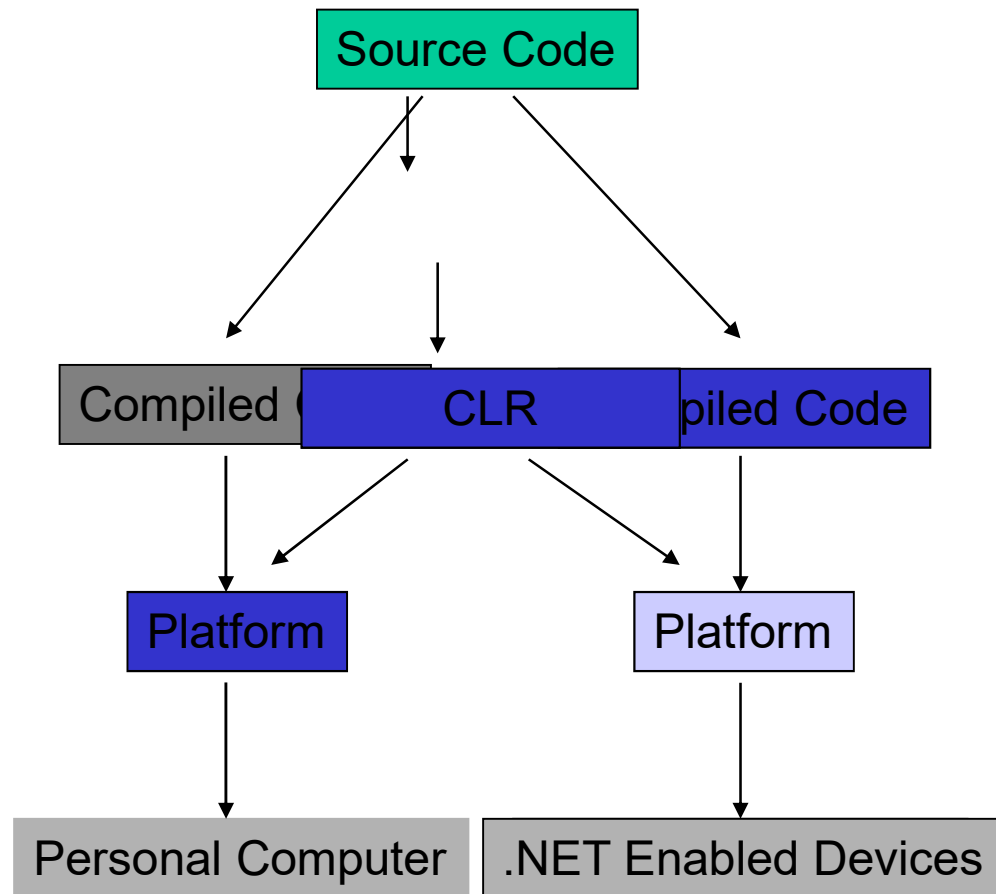
Common Language Runtime
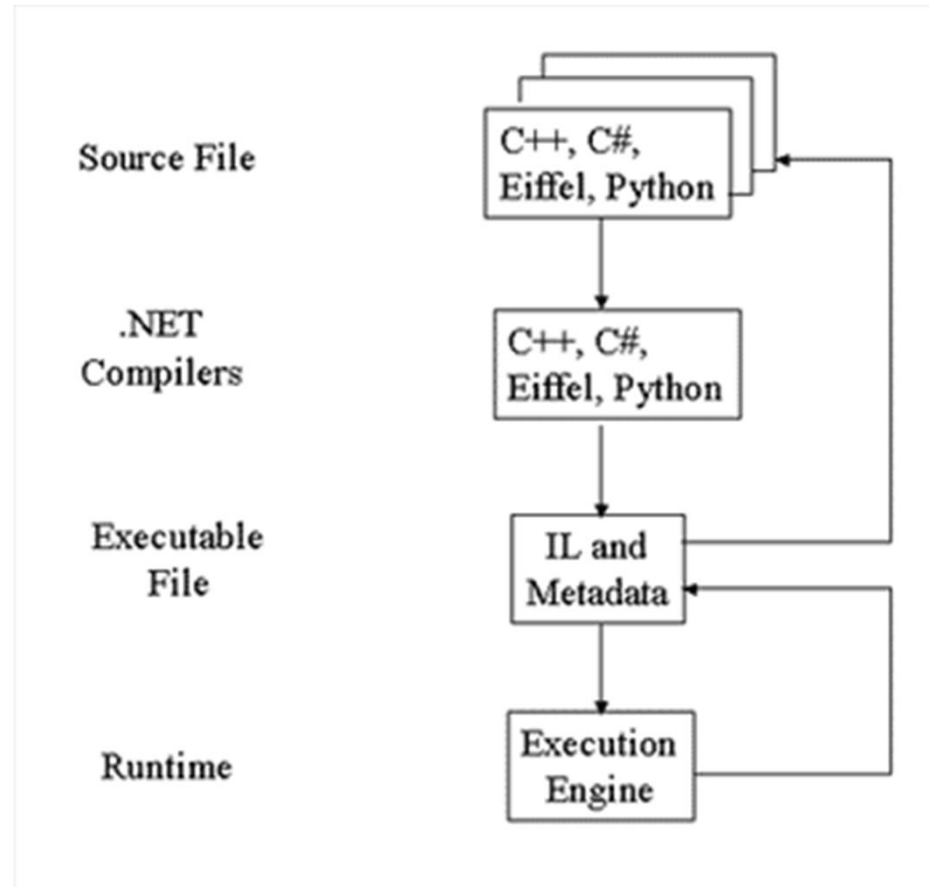
Execution scheme of .NET
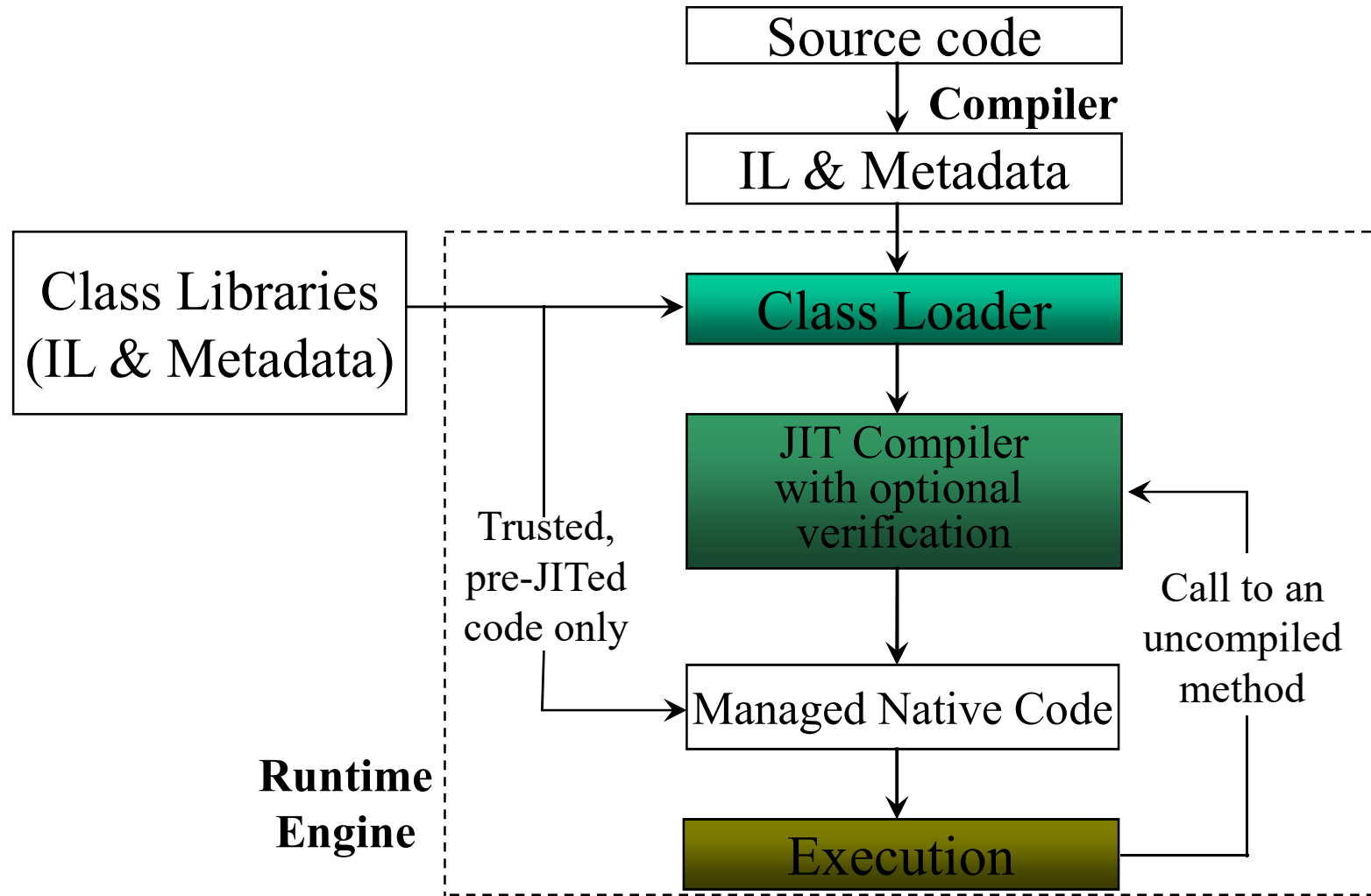
.NET Concept

- Source Code partially compiled to MSIL

- Source code written in various languages

- MSIL runs on CLR – an implementation of the CLS standard

- CLR available for various devices – the use of the Compact Framework

Language interoperability

- When striving for language interoperability, the basic issues remain. Some form of agreement about the representation of data types must be adopted.
  - In the CORBA world, the Object Management Architecture defines the concepts of object and type, and the CORBA specification quantifies these concepts.
  - In the .NET Framework, the common language runtime type system describes the types within the system.
- The common language runtime type system is divided into two subsystems:
  - **Value types**: A variable of value types directly contains only an object with the value.
  - **Reference types**: A variable of reference type directly contains a reference to an object. Another variable many contain a reference to the same object.

Common language runtime type system
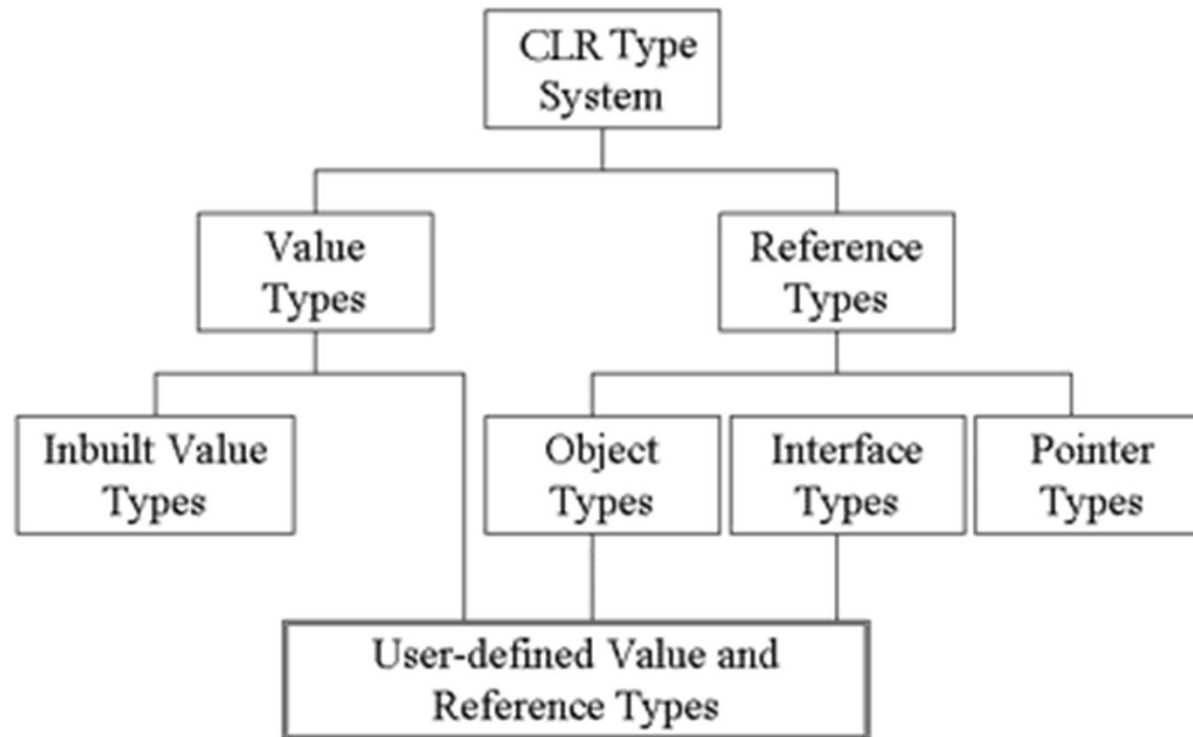
Metadata

- Metadata is the essential link that bridges the runtime type system and the execution engine. Currently, two significant problems exist with many component-based programming systems:
  - Information about the component (its metadata) is often not stored with the component. Rather, metadata is often stored in auxiliary files, such as interface definition language (IDL) files, type libraries, interface repositories, implementation repositories, and the Registry. The .NET common language runtime stores metadata with types to avoid this issue.
  - Metadata facilities are primitive. Most facilities only allow developers to specify the syntax of an interface, but not its semantics. The .NET Framework common language runtime addresses this problem by providing a standardized metadata extension system known as custom attributes.

Common Language Specification

- Compilers targeting the .NET Framework describe the types they produce with metadata for two reasons:
  - Metadata permits types defined in one language to be usable in another language. This facility ensures language interoperability in the common language runtime.
  - The execution engine requires metadata to manage objects. Managing objects includes requirements such as memory management.
- If the common language runtime can be described as the union of many programming language features, the common language specification (CLS) is a subset of these features. The subset is not the set of all features common to all languages. Rather it is a set of features common to many programming languages.

.NET security

- Security is an essential part of .NET.
- Means of ensuring safety:
  - Safe type system (type verification): .NET code is subject to verification.
  - Code authenticity check: .NET assembly is signed using 128-bit public key cryptography.
  - Resource access authorization
  - Declarative/Imperative security
  - Security policy:
    - Access policies
    - Roles (based on the roles)
- Increased security role for remote execution
- Cryptographic security methods available for embedding into user applications

.NET vs. COM

- COM provides a way for components to integrate. However, each component has to provide plumbing and objects could not directly interact.

- With .NET framework common language runtime, components are built on a common platform. No plumbing is needed and objects could directly interact.

- Though COM permits you to integrate binary components developed using any language, it does require you to obey the COM identity, lifetime, and binary layout rules and write the plumbing code that is required to create a COM component

- .NET class can be exported as a COM object

.NET vs. COM

- Eliminates COM plumbing
  - Registration ➔ self describing components
  - GUIDs (Globally Unique Identifier) ➔ hierarchical namespaces
  - .IDL files ➔ source code to metadata
  - HRESULTs (a coded numerical value assigned to a specific exception) ➔ structured exceptions
  - IUnknown (The IUnknown interface is the basis of all COM interfaces.) ➔ root object class
  - AddRef/Release ➔ garbage collector
  - CoCreateInstance (This function creates on the local system a single uninitialized object of the class associated with a specified class identifier.) ➔ new operator

CLR vs. JVM

CLR	JVM (CORBA IDL, CORBA ORB)
IL code and metadata	bytecode
JIT compiler/install time Code Gen	Interpreted/JIT compiler
Platform-specific, dependent on platform-specific class library; any computer architecture,	Any platform, any computer architecture
provides a set of framework classes and lets every language use it	Bytecode Support different language by native call

They are both a runtime infrastructure and platform differences are hiding from the users.