.NET Framework Interview Questions & Answers

.NET4.x

By Shailendra Chauhan

Microsoft MVP, Founder & CEO - Dot Net Tricks

DotNetTricks

.NET Framework Interview Questions & Answers

Release History

- Initial Release 1.0.0 - 21st June 2019

About Dot Net Tricks

Dot Net Tricks is founded by Shailendra Chauhan (Microsoft MVP), in Jan 2010. Dot Net Tricks came into existence in the form of a blog post over various technologies including .NET, C#, SQL Server, ASP.NET, ASP.NET MVC, JavaScript, Angular, Node.js and Visual Studio etc.

The company which is currently registered by a name of Dot Net Tricks Innovation Pvt. Ltd. came into the shape in 2015. Dot Net Tricks website has an average footfall on the tune of 300k+ per month. The site has become a cornerstone when it comes to getting skilled-up on .NET technologies and we want to gain the same level of trust in other technologies. This is what we are striving for.

We have a very large number of trainees who have received training from our platforms and immediately got placement in some of the reputed firms testifying our claims of providing quality training. The website offers you a variety of free study material in the form of articles.

Dot Net Tricks Courses

Master in-demand job skills with our step by step and project-based courses. Learn to start a new career, with our curated learning paths tailored to today's developers and technology needs. Learn to code, prepare yourself for interviews, and get hired!

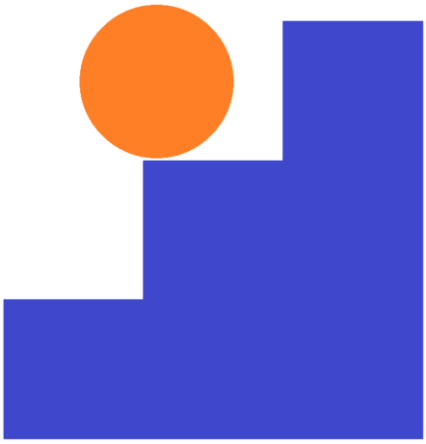
We offer the eBooks in the following categories:

- .NET Development
- Frond-end Development
- Cloud
- DevOps
- Programming Languages
- Database - SQL and NoSQL
- Mobile Development and many more...

You can start learning free from here: https://www.dotnettricks.com/courses

Dot Net Tricks Pro

DotNetTricks Pro unlocks the access of DotNetTricks premium features like unlimited access to all courses, source codes, assessments. Get help over email or phone. Upgrade your skills with curated learning paths tailored to today's developers and technology needs. Learn new skills and discover the world of possibilities with step-by-step guidance.



Start your journey today to learn coding. Because learning to code is the first step and foreword to advance your career. The detail about Dot Net Tricks Pro can be found here: https://www.dotnettricks.com/pro-membership

Dot Net Tricks Live Training

Instructor-led Training Programs

For a beginner who needs regular guidance, we have a fully packed Master Courses. They are almost equal to semester courses taught in engineering colleges when it comes to length, breadth of content delivery, the only difference instead of 5-6 months, they take approx. 16-weekend classes (2 months).

The detail about Master courses can be found here: https://www.dotnettricks.com/instructor-led-courses

Corporate Training

Dot Net Tricks having a pool of mentors who help the corporate to enhance their employment skills as per changing the technology landscape. Dot Net Tricks offers customized training programs for new hires and experienced employees through online and classroom mode. As a trusted and resourceful training partner, Dot Net Tricks helps the corporate to achieve success with its industry-leading instructional design and customer training initiatives.

Apart from these, we also provide on-demand boot camps and personalized project consultation.

The detail about Corporate Training can be found here: https://www.dotnettricks.com/corporate-training

Dot Net Tricks eBooks

Dot Net Tricks offer a wide range of eBooks on technical interviews Q&A. All eBooks are written by industry experts and coaches. These eBooks will help you to prepare yourself for your next job within a short time. We offer the eBooks in the following categories:

- .NET Development
- Frond-end Development
- Cloud

- DevOps
- Programming Languages
- Database - SQL and NoSQL
- Mobile Development and many more….

You can buy other eBooks from here: https://www.dotnettricks.com/books

Technical Recruiting

We provide full technical staffing service which suits our client needs. Our technical recruiters search across the world to find highly skilled professionals that will fit our clients need. If you are looking for a job change, do share your resume at hr@dotnettricks.com. Dot Net Tricks will help you to find your dream job in MNCs.

Join us today, learn to code, prepare yourself for interviews, and get hired!

Dedication

My mother Mrs Vriksha Devi and my wife Reshu Chauhan deserve to have their name on the cover as much as I do for all their support made this possible. I would like to say thanks to all my family members Virendra Singh(father), Jaishree and Jyoti(sisters), Saksham and Pranay(sons), friends, to you and to readers or followers of my articles at https://www.dotnettricks.com/mentors/shailendra-chauhan to encourage me to write this book.

-Shailendra Chauhan

Introduction

Writing a book has never been an easy task. It takes a great effort, patience and consistency with strong determination to complete it. Also, one should have a depth knowledge over the subject is going to write.

So, what where my qualification to write this book? My qualification and inspiration come from my enthusiasm for and the experience with the technology and from my analytic and initiative nature. Being a trainer, analyst, consultant and blogger, I have thorough knowledge and understandings of .NET technologies. My inspiration and knowledge have also come from many years of my working experience and research over it.

What This Book Is

.NET is a software development platform developed by Microsoft..NET provides tools and libraries that allow developers to develop applications and services much easily, faster and secure by using a convenient way.

This book is all about .NET 4.x and its components including version history, BCL, CLR, CTS, MSIL, COM, Assembly, DLL and Garbage collector. Learn to crack your technical Interview on .NET. It includes most commonly asked questions with their answers including diagram and code snippets.

What You'll Learn

This book is for freshers or .NET developers who are looking for a change or want to make a career in .NET. This book covers the interview questions on the following topics:

- .NET 4.x features details.
- Base Class Library (BCL).
- CLR and its components.
- CTS and CLS with their relationships.
- MSIL and JIT compiler.
- Managed and Unmanaged Code.
- Stack and Heap memory management.
- Assembly and GAC.
- Garbage Collector and its methods.
- Garbage Collector algorithm.

Our best wishes always with you for your learning and growth!

About the Author

Shailendra Chauhan - An Entrepreneur, Author, Architect, Corporate Trainer, and Microsoft MVP

He is the **Founder and CEO** of DotNetTricks which is a brand when it comes to e-Learning. DotNetTricks provides training and consultation over an array of technologies like **Cloud, .NET, Angular, React, Node and Mobile Apps development**. He has been awarded as **Microsoft MVP** three times in a row (2016-2018).

He has changed many lives from his writings and unique training programs. He has a number of most sought-after books to his name which have helped job aspirants in **cracking tough interviews** with ease.

Moreover, and to his credit, he has delivered **1000+ training sessions** to professionals worldwide in Microsoft .NET technologies and other technologies including JavaScript, AngularJS, Node.js, React and NoSQL Databases. In addition, he provides **Instructor-led online training**, **hands-on workshop** and **corporate training** programs.

Shailendra has a strong combination of **technical skills and solution development for complex application architecture with proven leadership and motivational skills** have elevated him to a world-renowned status, placing him at the top of the list of most sought-after trainers.

"I always keep up with new technologies and learning new skills to deliver the best to my students," says Shailendra Chauhan, he goes on to acknowledge that the betterment of his followers and enabling his students to realize their goals are his prime objective and a great source of motivation and satisfaction.

Shailendra Chauhan - **"Follow me and you too will have the key that opens the door to success"**

How to Contact Us

Although the author of this book has tried to make this book as accurate as it possible but if there is something strikes you as odd, or you find an error in the book please drop a line via e-mail.

The e-mail addresses are listed as follows:

- mentor@dotnettricks.com
- info@dotnettricks.com

We are always happy to hear from our readers. Please provide your valuable feedback and comments!

You can follow us on YouTube, Facebook, Twitter, LinkedIn and Google Plus or subscribe to RSS feed.

Table of Contents

1
Introducing .NET

Q1. What is .NET?

Ans. .NET is a software development platform developed by Microsoft. It runs on Microsoft Windows OS. .NET provides tools and libraries that allow developers to develop applications and services much easily, faster and secure by using a convenient way.

Q2. Why to use .NET?

Ans. There are following reasons you should use .NET.

- Using .NET, there is no need to learn new programming language. It supports 48 programming languages like as C++, C#, J#, VB etc. Hence, by using .NET platform; you can do programming in any programming language in which you feel comfortable.
- .NET is the best platform for delivering better, faster, cheaper, and more secure applications and services.

Q3. Explain evolution history of .NET Framework 4.x?

Ans. The evolution history of .NET Framework is given below:

.NET Version	Release Date	Introduced with IDE	Features Detail
4.8	2019	Visual Studio 2019	• Support for High-DPI • Performance updates, and security enhancements
4.7	2017	Visual Studio 2017	• Enhanced Cryptography • Improve TLS support • More support for touch in WPF • New print APIs for WPF
4.6	2015	Visual Studio 2015	• Ryu JIT • Support for TLS 1.1 and TLS 1.2 in WCF

4.5.1	2013	Visual Studio 2013	• Includes performance and debugging improvements • Support for automatic binding redirection • Expanded support for Windows Store apps
4.5	2012	Visual Studio 2012	• Features Enhancements to CLR 4.0 • Async Support • Support for building Windows Store apps • Features Enhancement to WPF, WCF, WF, and ASP.NET
4.0	2010	Visual Studio 2010	• Introduced CLR 4.0 • Managed Extensibility Framework (MEF) • Dynamic Language Runtime (DLR) • Task Parallel Library
3.5	2007	Visual Studio 2008	• Built-In AJAX Support • LINQ • Dynamic Data • Multi-targeting Framework Support
3.0	2006	Visual Studio 2005	• Windows Presentation Foundation (WPF) • Windows Communications Foundation (WCF) • Windows Workflow Foundation (WF), and CardSpace
2.0	2005	Visual Studio 2005	• Introduced CLR 2.0 • Generics and generic collections • Partial classes • Nullable types • Anonymous methods • Introduced many new controls and features to ASP.NET
1.1	2003	Visual Studio .NET 2003	• Introduced CLR 1.1 • Features Enhancement to ASP.NET and ADO.NET • Built-in support for mobile ASP.NET controls. • Security Enhancement • Built-in support for ODBC and databases • Internet Protocol version 6 (IPv6) support
1.0	2002	Visual Studio .NET	• Introduced CLR 1.0 • Support for Object-oriented Web application development • Use of DLL class libraries

Note: CLR 3.0 is skipped by Microsoft.

Q4. Explain .NET Framework 4.5 Architecture?

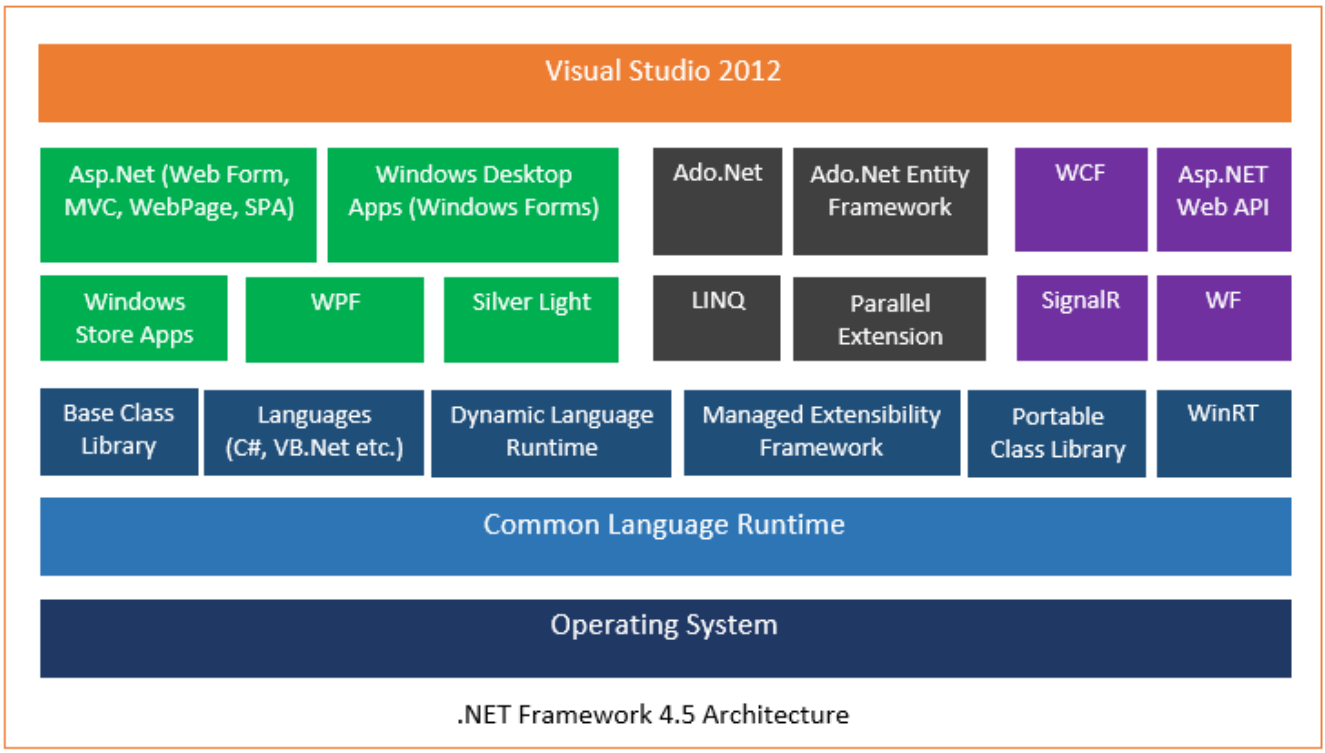
Ans. .NET Framework is an integrated component of windows operating system that supports development and execution of next generation applications, Windows store apps and services.

.NET Framework 4.5 at its core consists of Common Language Runtime (CLR), Dynamic Language Runtime (DLR), Base Class Library (BCL), Portable Class Library, Parallel Extension and WinRT.

With .NET Framework 4.0 there are various new additions in BCL which includes DLR, MEF, Parallel Extension, Entity Framework, WCF Data Services and ability to host .NET 4 runtime with .NET 3.5, 3.0, 2.0 runtimes side by side under the same hosting process.

Components of .NET Framework 4.5 Architecture are given below –

1. **Common Language Runtime (CLR) –** This acts as the execution engine for the .NET Framework. All .NET programs execute under the supervision of CLR.
2. **Base Class Library (BCL) -** This is a library of functionalities which are available to all languages using the .NET Framework. It consists of classes, interfaces of reusable types that integrates with CLR
3. **Portable Class Library (PCL) -** The Portable Class Library project in Visual Studio 2012 allows you to develop and build managed assemblies that work on multiple .NET Framework platforms. Using a Portable Class Library project, you choose the platforms (such as Windows Phone and .NET for Windows Store apps) to target.
4. **Managed Extensibility Framework (MEF) -** MEF is a library for creating lightweight, extensible applications. It allows application developers to discover and use extensions with no configuration required.
5. **Dynamic Language Runtime (DLR) -** This provides the runtime environment for dynamic languages like python etc. for executing under the full control of CLR.
6. **WinRT –** WinRT or Windows Runtime APIs provides the user interface elements for building Windows Store apps, and provides access to Windows 8 or Windows RT OS features. WinRT supports development in C and other managed languages C# and VB.NET, as well as JavaScript and TypeScript.



.NET Framework 4.5 Architecture

7. **ASP.NET –** This is used to build rich internet-based web application.
8. **Windows Store Apps (Metro Style Apps) -** A Windows Store app is a new type of application that runs on Windows 8 devices and can take advantage of new WinRT APIs. These can only be distributed in the Windows 8 store.
9. **Desktop Apps (Windows Forms) -** A Windows Desktop app is traditional Windows Forms application with a new name. Software developed for Windows XP, Windows Vista and Windows 7 will be categorized as a Windows Desktop app when running in Windows 8. Examples of Windows Desktop apps are Microsoft Office family's products, notepad etc.
10. **WPF -** WPF is used to create applications with a rich user experience. It includes application UI, 2D graphics, 3D graphics and multimedia. It takes advantage of hardware acceleration of modern graphic cards. WPF makes the UI faster, scalable and resolution independent.
11. **Silver Light -** This is a cross-browser web-based technology which allows designers and developers to deliver Rich Internet Applications (RIA) embedded in Web pages.
12. **ADO.NET –** This is used to create Data Access Layer to query and manipulate data from underlying data source like SQL Server, Oracle, and DB2 etc.
13. **LINQ -** This allows you to query the data from the various data sources (like SQL databases, XML documents, ADO.NET Datasets, Various Web services and any other objects such as Collections, Generics etc.) using a SQL Query like syntax with .NET Framework framework languages like C# and VB.
14. **ADO.NET Entity Framework –** This is used to query and store data into to the relational databases (like SQL Server, Oracle, DB2 etc.) in ORM fashion.
15. **Parallel Extension -** This allows you to distribute your work code across multiple processors to take advantage of the hardware.

16. **WCF -** This is used for building and developing services based on WS-* standards.
17. **ASP.NET WebAPI -** ASP.NET WebAPI is a framework for building HTTP services that can be consume by a broad range of clients including browsers, mobiles, iPhone and tablets.
18. **SignalR -** ASP.NET SignalR is a library that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.
19. **WF -** This is used to build process-oriented business workflow and rules engine.
20. **Visual Studio 2012 -** The Visual Studio IDE offers a set of tools that help you to write and modify the code for your programs, and also detect and correct errors in your programs. Using Visual Studio 2012 you can build Windows Store apps, desktop apps, mobile apps, ASP.NET web apps, and web services.

Q5. What is Base Class Library (BCL)?

Ans. The .NET Framework base class library is a library of classes, interfaces and value types that provides functions and features which can be used by any programming language which runs under .NET Framework. BCL is the foundation on which .NET Framework applications, components and controls are built.

Q6. What are the commonly used namespaces in BCL?

Ans. Some Namespaces from BCL are given below –

- System
- System.Collections
- System.Data
- System.Threading

2
CLR

Q1. What is CLR?

Ans. CLR stands for **Common Language Runtime**. It acts as the execution engine for the .NET Framework. All .NET programs execute under the supervision of CLR.

Q2. What are the functions of CLR?

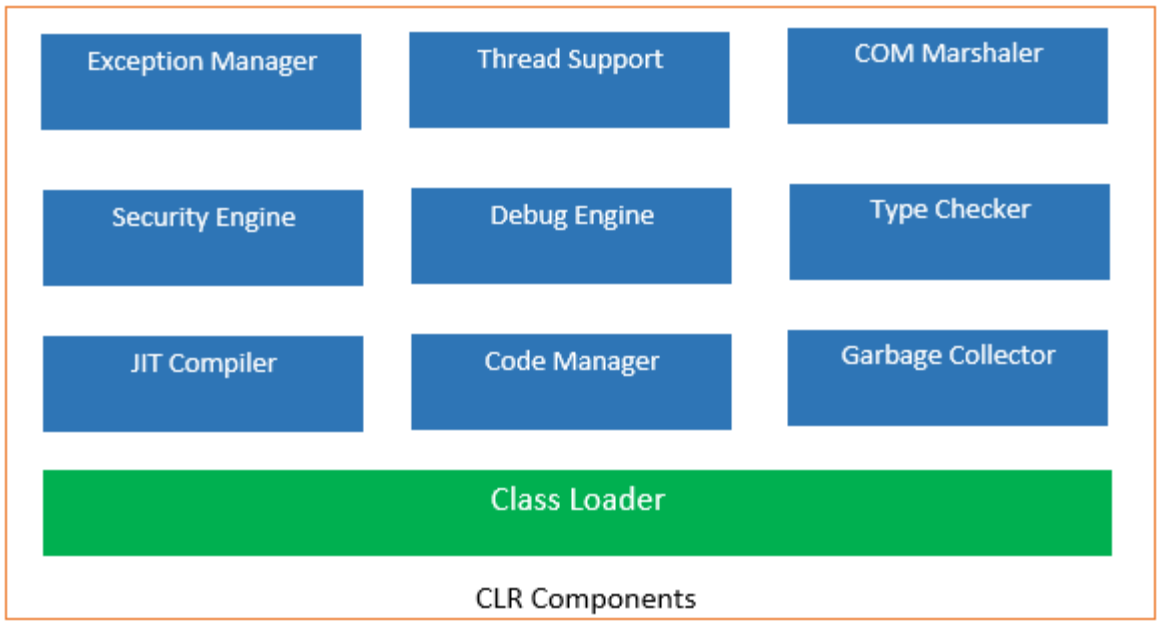
Ans. There are following functions of CLR –

- Code Management (loading and execution).
- Code Verification or Type Safety.
- Code Compilation and conversion of MSIL to native code.
- Memory Management using Garbage Collection.
- Exception Management.
- Thread Management.
- Security Management.
- Supports developer services like debugging, profiling etc.

Q3. What are the components of CLR?

Ans. Main components of CLR are given below in fig.

- **Class Loader –** Class loader is responsible to verify what are the class libraries referred by the .NET application and load all those class libraries into memory and make them available for .NET application. Class loader is not responsible for loading application code in to memory.
- **JIT Compiler –** JIT compiler is responsible for converting MSIL code to CPU native code which is understandable by processor.



- **Code Manager –** Code manager is responsible for loading .NET application code into memory and manages it until .NET application is closed. The .NET application code that is loaded in to memory and runs under the full control of code manager is called managed code and the .NET application code that is loaded into memory but not runs under the control of code manager but runs under the control of OS is called as unmanaged code.
- **Garbage Collector (GC) -** This is responsible for automatic memory management for the .NET application. Memory allocation and de-allocation for the variable and objects created in .NET application is managed by Garbage Collector.
- **Security Engine -** This is responsible for managing **Role Based Security** or **Code Access Security**.
- **Debug Engine -** This is responsible to debug your application and trace the execution of code.
- **Type Checker -** This does not allow unsafe cast or uninitialized variables. MSIL can be verified to guarantee type safety.
- **Exception Manager -** This is responsible for exception handling.
- **Thread Support -** This provides classes and interfaces that enable multithreaded programming.
- **COM Marshaler -** This is responsible for providing marshalling to and from COM.

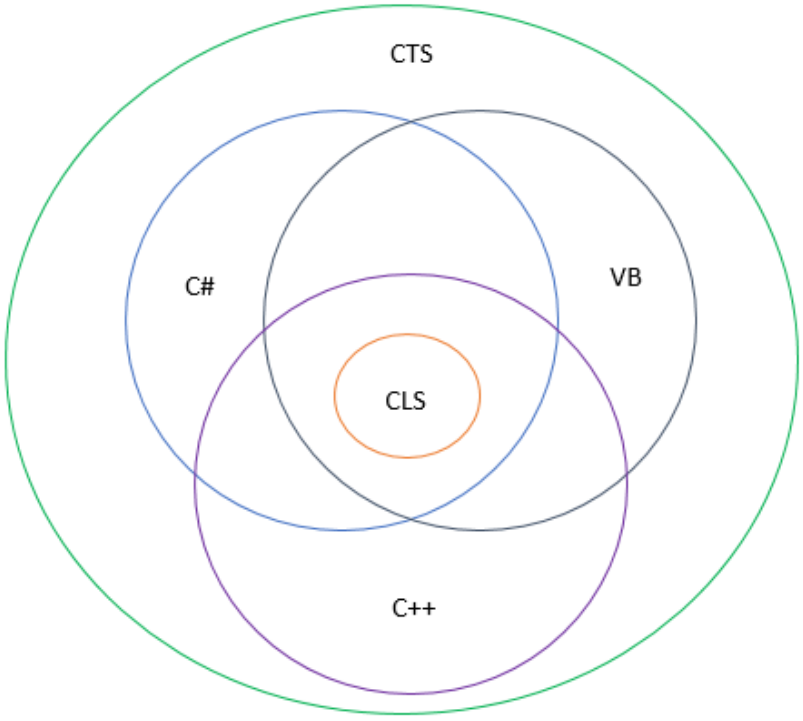
Q4. What is CTS and CLS? How both are related to each other?

Ans. **CTS -** CTS stands for Common Type System. It defines the rules which **Common Language Runtime** follows when declaring, using and managing types.

The common type system performs the following functions:

1. It enables cross-language integration, type safety and high-performance code execution.
2. It provides an object-oriented model for implementation of many programming languages.
3. It defines rules that every language must follow which runs under .NET framework. It ensures that objects written in different .NET Languages like C#, VB.NET, F# etc. can interact with each other.

CLS - CLS stands for Common Language Specification and it is a subset of CTS. It defines a set of rules and restrictions that every language must follow which runs under .NET framework. The languages which follow these set of rules are said to be CLS compliant. In simple words, CLS enables cross-language integration.



CTS and CLS

For example, one rule is that you cannot use multiple inheritance within .NET Framework. As you know C++ supports multiple inheritances but; when you will try to use that C++ code within C#, it is not possible because C# doesn't support multiple inheritance.

One another rule is that you cannot have members with same name with case difference only i.e. you cannot have add() and Add() methods. This easily works in C# because it is case-sensitive but when you will try to use that C# code in VB.NET, it is not possible because VB.NET is not case-sensitive.

Q5. Why CTS is called Common Type System?

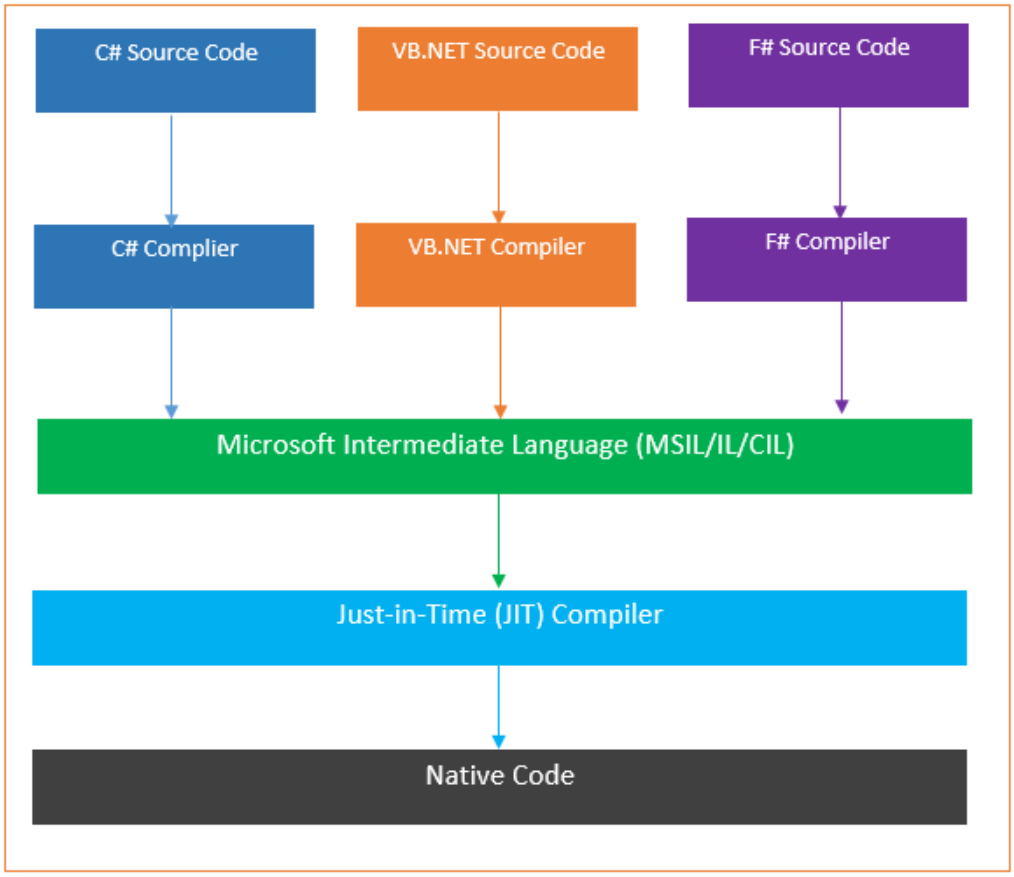
Ans. In .NET, every data type is internally represented by a class or structure. All the classes and structures related to data types are collectively known as CTS. As you know every language provides its own keywords for data types but internally all the languages which run under .NET framework use the classes and structures available in CTS.

For example, C# has *int* Data Type and VB.NET Framework has *Integer* Data Type. Hence a variable declared as *int* in C# or *Integer* in vb.net, finally after compilation, uses the same structure *Int32* from CTS.

All the structures and classes available in CTS are common for all .NET languages and purpose of these is to support language independence in .NET. Hence, it is called CTS.

Q6. What is MSIL or IL or CIL?

Ans. MSIL stands **for Microsoft Intermediate Language** or CIL stands **for Common Intermediate Language**. In .NET, when code is compiled, language compiler translates your source code MSIL, which is a CPU-independent set of instructions.



It includes instruction for loading, storing, initializing and methods calling on objects as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations.

Further, MSIL is converted to CPU native code by a just-in-time (JIT) compiler; which is understandable by processor.

Q7. What is JIT complier? Why it is called just-in-time?

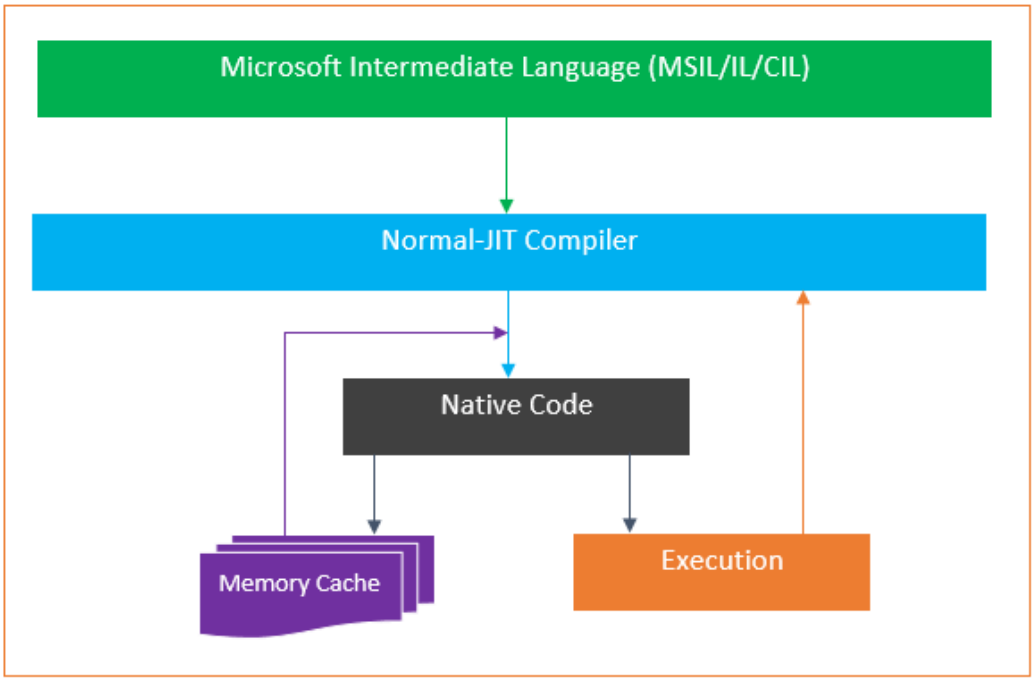
Ans. JIT stands for just-in-time compiler. It converts the MSIL code to CPU native code as it is needed during code execution. It is called just-in-time since it converts the MSIL code to CPU native code; when it is required within code execution otherwise it will not do nothing with that MSIL code.

Q8. What are different types of JIT?

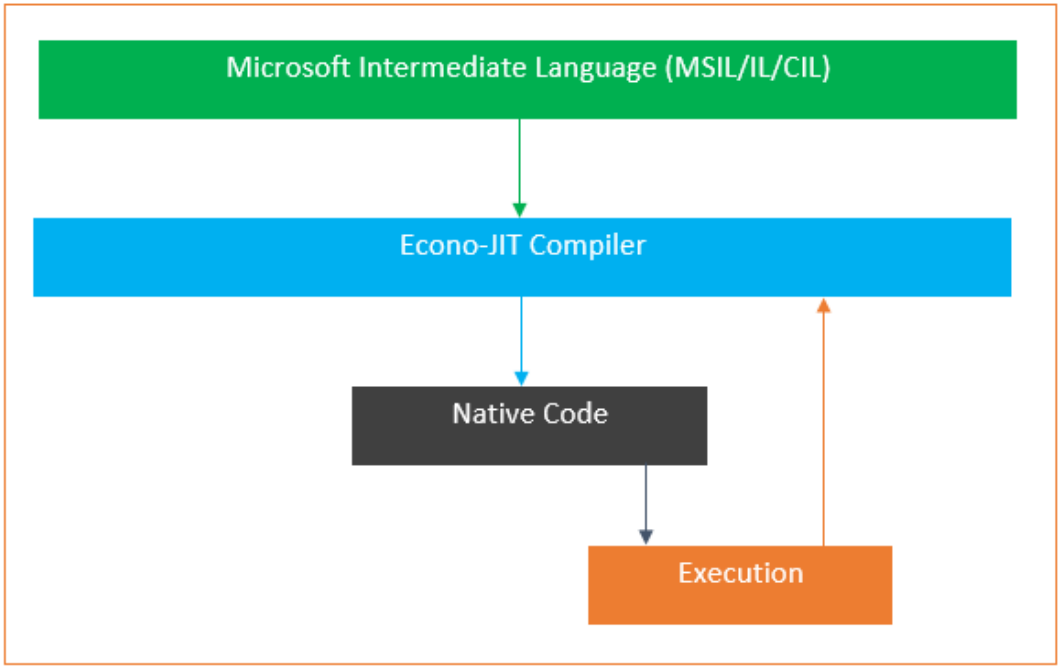
Ans. There are three types of JIT as given below –

1. **Normal JIT -** This complies only those methods that are called at runtime. These methods are compiled only first time when they are called, and then they are stored in memory cache. This memory cache is

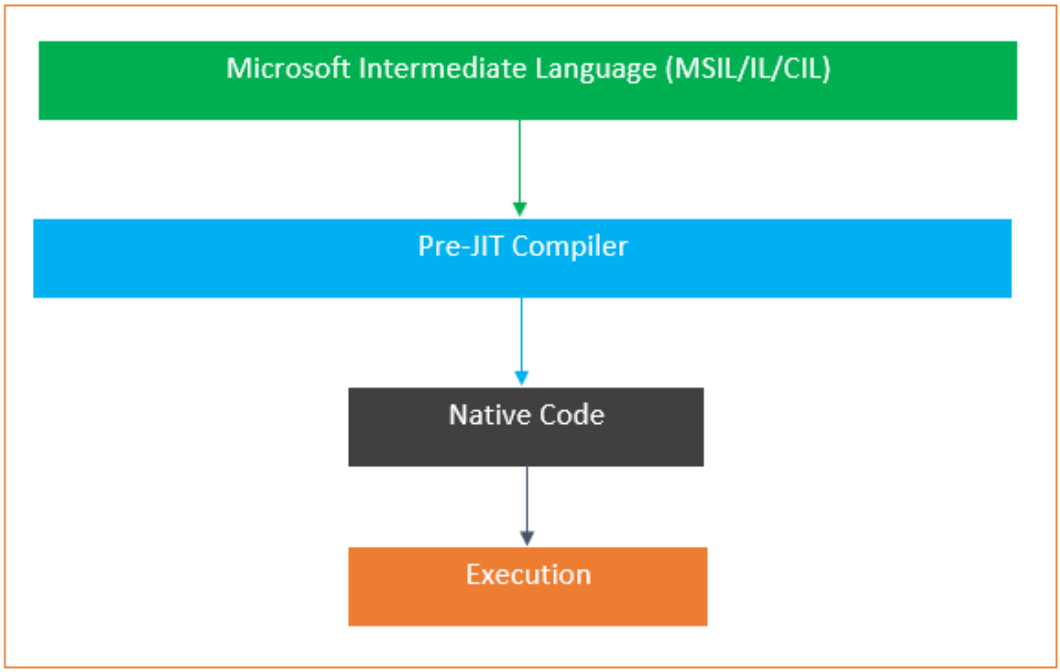
commonly called as *JITTED.* When the same methods are called again, the complied code from cache is used for execution.



2. **Econo JIT -** This complies only those methods that are called at runtime and removes them from memory after execution.



3. **Pre JIT -** This complies entire MSIL code into native code in a single compilation cycle. This is done at the time of deployment of the application.



Q9. What are managed code and unmanaged code?

Ans. **Managed Code -** A source code which runs under the management of CLR is called Managed Code. It takes advantages of CLR features like memory management, type safety, thread management and security. This is developed within .NET Framework by using .NET Languages like C#, Vb.NET, F# etc.

Unmanaged Code - A source code which does not run under the management of CLR is called Unmanaged Code. This is developed outside .NET Framework by using any programming languages like Visual Basic, C++ etc. Unmanaged code is executed with help of wrapper classes.

Q10. What is managed class?

Ans. A class which instances memory is managed by GC is called managed class. A C++ class can be marked as managed class by using _gc keyword. A managed C++ class can be inherited from a VB class.

```
public __gc class Block {};
```

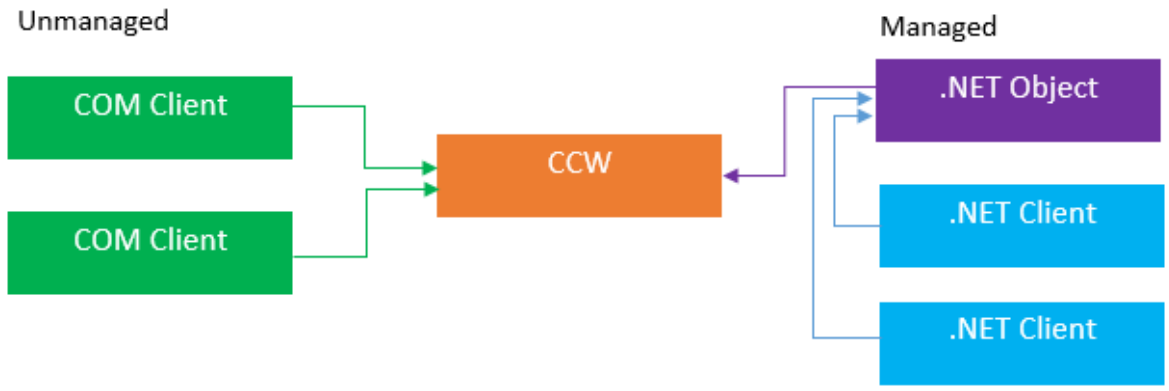
Q11. What is managed wrapper class or wrapper class in .NET framework?

Ans. When you need to access unmanaged code with in your managed code, you need to wraps the unmanaged code functionality into a managed class which is called managed wrapper class. This wrapper class allows you to access unmanaged code.

Q12. What are CCW (COM Callable Wrapper)?

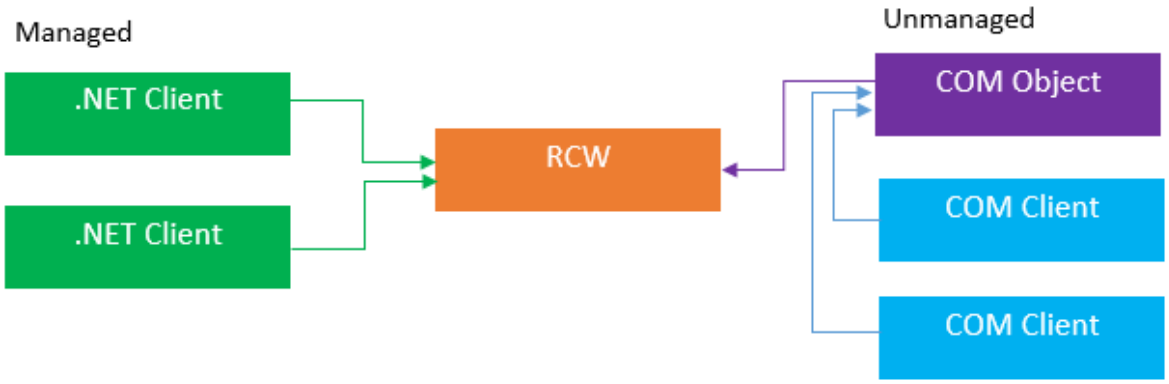
Ans. When a COM client calls a .NET object, the CLR creates the managed object and a COM callable wrapper (CCW) for the object. COM clients use the CCW as a proxy for the managed object.

The CLR creates exactly one CCW for a managed object and multiple COM clients can hold a reference to the CCW that exposes the managed object. Both COM and .NET clients can make requests on the same managed object simultaneously.



Q13. RCW (Runtime Callable Wrapper)?

Ans. When a .NET Framework client requires a COM object, CLR exposes COM objects through a proxy called the runtime callable wrapper (RCW). A RCW appears as an ordinary object to .NET clients.



The CLR creates exactly one RCW for each COM object and multiple .NET Framework clients can hold a reference to the RCW that exposes the COM object. Both COM and .NET clients can make requests on the same COM object simultaneously.

Q14. What is COM, COM+ and DCOM?

OR

Explain COM vs. COM+ vs. DCOM?

Ans. **COM:** COM refers to the Component Object Model. COM is a technology which enables software components to communicate with each other. COM is used by developers to create re-usable software components and link components. COM objects can be created with a variety of programming languages like as C++, Visual Basic etc. The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX controls. COM components can only perform inter-process communication across process boundaries on the same machine.

For example, COM is used in the Microsoft office family of products. COM OLE technology allows word documents to dynamically link to data in excel spreadsheets.

COM+: COM+ is an extension of Microsoft's COM technology. COM+ is combination of COM and Microsoft Transaction Server (MTS). COM+ handles many of the resource management tasks that you previously had to program yourself, such as thread allocation and security. COM+ also makes your applications more scalable by providing thread pooling, object pooling, and just-in-time object activation.

COM+ is designed especially for Microsoft Visual C++ and Microsoft Visual Basic developers.

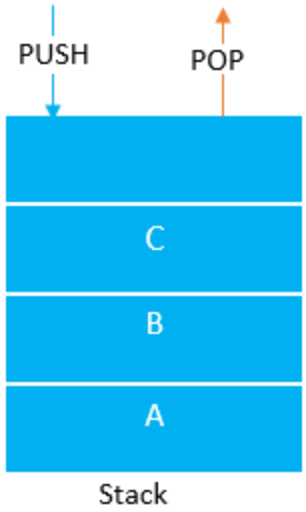
DCOM: DCOM is like as COM but it is specially designed for distributed applications. It enables communication among software components distributed across networked computers. DCOM, which originally was called "Network OLE", extends Microsoft's COM.

Q15. What is Marshalling?

Ans. This is a process which acts as a bridge between managed code and unmanaged code communication. It carries messages from the managed to the unmanaged environment and vice versa. It is one of the core services offered by the CLR.

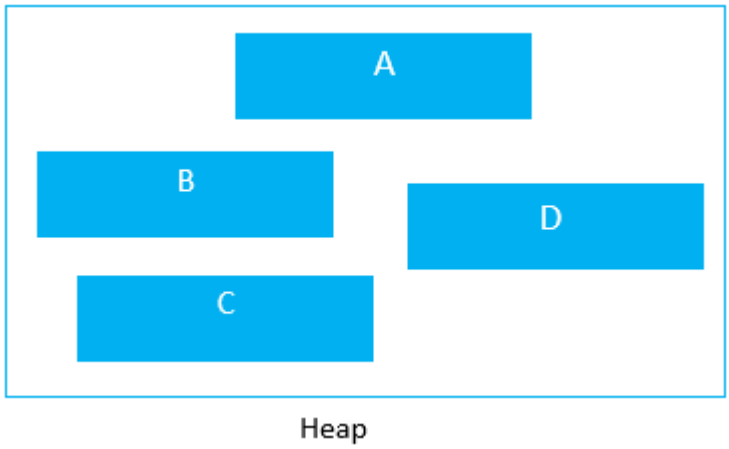
Q16. What is Stack in .NET?

Ans. A **s**tack is a data structure in memory which is used for storing all value types in a LIFO (Last-In-First-Out) manner. It contains local variables, passing parameters and the call stack. Variables created on the stack are automatically de-allocated when they will go out of scope. In stack memory allocation is fast as compared to heap.



Q17. What is Heap in .NET?

Ans. A heap is a data structure in memory which is used for storing all reference types (objects) in random manner. Whenever a new object is created, it is allocated on the heap, and a reference to that object is returned. .NET CLR has a garbage collector that periodically de-allocates objects from the heap.



Q18. What are different types of managed heap in .NET?

Ans. Managed heap can be divided into two parts – SOH and LOH.

Q19. What are LOH (Large Object Heap) and SOH (Small Object Heap)?

Ans. **SOH –** Small .NET objects are allocated onto the Small Object Heaps. These objects may be of **Generation 0**, **Generation 1**, and **Generation 2**. These are compacted and move to next generation based on their age.

LOH - .NET Objects larger than 85 KB are allocated onto the Large Object Heap. They aren't compacted, because of the overhead of copying large chunks of memory.

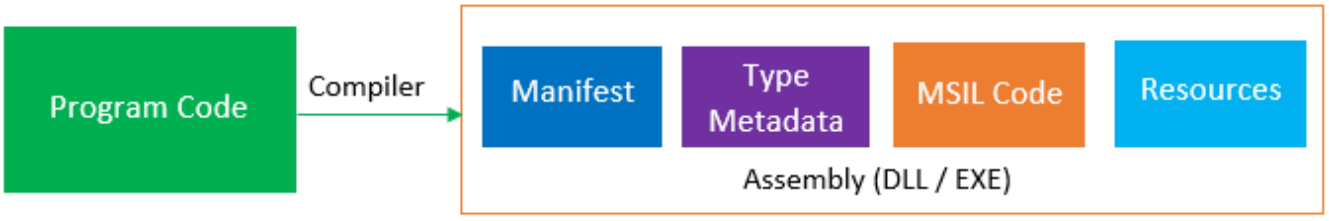
3
Assembly

Q1. What is assembly?

Ans. An assembly is compiled code library which contains manifest, metadata MSIL code and resources supporting that code. It is used for deployment, versioning, and security. Basically, an assembly can be a process assembly (EXE file) and library assembly (DLL file). A process assembly is self-executable and represents a process that can use classes defined in library assemblies.

Q2. Explain an Assembly structure?

Ans. An assembly can have of one or more files. An assembly can have code of a single programming language.



Structure of Assembly

- **Manifest -** It contains the assembly name, version number, culture and an optional strong name that uniquely identifying the assembly. It also contains information about modules in the assembly and references to other assemblies. This manifest information is used by the CLR.
- **Type Metadata -** It provides information about the code. It provides description of assembly code members like classes, structure, properties and methods etc. along with the data type for their parameters and returns values.
- **MSIL Code -** It is a CPU-independent set of instructions. It includes instruction for loading, storing, initializing and methods calling on objects as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations.
- **Resources -** These are non-executable parts or files required for the code execution like as images, icons or text files.

Q3. What are advantages of assembly?

Ans. There are following advantages of an assembly –

1. Solved versioning problem (DLL hell).
2. Provides better code management and maintenance.
3. Provides code security.
4. Improves application performance since it contains compiled code.

Q4. What are different types of assembly?

Ans. There are five types of assemblies in .NET as given below –

1. **Private Assembly -** An assembly that is used only by single application is called private assembly. It is created and stored in the application directory.

2. **Public or Shared Assembly -** An assembly that is used by more than one application is called shared application. It is stored in Global Assembly Cache (GAC).
   Each shared assembly has four parts, name including its face name, version, public key token and culture information.

3. **Satellite Assembly -** An assembly which contains localized resources for another assembly is called satellite assembly. This assembly is used for deploying an application globally for different languages.

4. **Static Assembly -** There assemblies are stored on the disk in portable executable (PE) files. These include .NET Framework classes, interfaces and resource files. These assemblies are not loaded directly from the memory instead they are directly loaded from the disk when CLR requests for them.

5. **Dynamic Assembly -** These assemblies are not stored on the disk before execution; but after execution they get stored on the disk. When .NET CLR calls them, they are directly loaded from the memory not from the disk. These are created in the memory by using System.Reflection.emit namespace.

Q5. What is the difference between DLL and EXE?

Ans. The difference between these two is given below –

DLL	EXE
DLL is the extension for a dynamic link library.	EXE is an extension used for executable files.
It is used by other applications.	It can be run independently.
A DLL shares the same process and memory space of the calling application.	An EXE creates its own separate process and memory space.
It has no entry point.	It has an entry point (Main function).
It has a version associated with itself.	It has no version associated with itself.
It is not self-executable.	It is self-executable.
It is reusable.	It is not reusable.

Q6. What is strong name in Assembly?

Ans. A strong name includes the name of the assembly, version number, culture identity, and a public key token. Strong names guarantee name uniqueness for an assembly. Assemblies with the same strong name are expected to be identical. Strong name also provides a strong integrity check.

Q7. How you can create a strong name for a .NET assembly?

Ans. You can generate a strong name for a .NET assembly with the help of strong name tool **sn.exe.**

Q8. What is namespace?

Ans. Namespace is a logical grouping of classes, interfaces, structures, enums, delegates or even another namespace. Each namespace must have a unique name but two different namespaces may have nested namespace with same name.

```
namespace MyNamespace
{
    interface MyInterface
    {
        void show();
    }

    enum MyEnum { mon, tue, wed, thu, fri, sat, sun }

    delegate void display();

    class MyClass
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Namespace Demo");
        }
    }

    namespace NestedNamespace
    {
        //Other classes, interfaces, enums and delegates
    }
}
```

Key points about Namespace:

1. Namespaces implicitly have public access which is not modifiable means you cannot define the access modifier along with a namespace.
2. Namespace can have only classes, interfaces, structures, enums, delegates or even another namespace.
3. A namespace can be spanned into multiple assemblies.

Q9. What is the purpose of a namespace?

Ans. A namespace is used to uniquely identify the classes, interfaces, structures, enums and delegates which it contains. Typically, it resolves the ambiguity between two or more classes, interfaces, structures, enums and delegates which have same name.

For Example,

```
namespace MyNamespace1
{
    class MyClass
    {
        public void Show()
        {
            Console.WriteLine("Hello");
        }
    }
}

namespace MyNamespace2
{
    class MyClass
    {
        public void Show()
        {
            Console.WriteLine("Bye");
        }
    }
}

//Using both the classes having same name
namespace MyNamespace3
{
    class ClassA
    {
        public void Display()
        {
            // MyNamespace1 uniquely identify the MyClass
            MyNamespace1.MyClass obj1 = new MyNamespace1.MyClass();
            obj1.Show(); //Hello

            // MyNamespace2 also uniquely identify the MyClass.
            // There is no ambiguity between MyClass of MyNamespace1 and MyNamespace2.
            MyNamespace2.MyClass obj2 = new MyNamespace2.MyClass();
            obj2.Show(); //Bye
        }
    }
}
```

Q10. What is the difference between namespace and assembly?

Ans. A Namespace is a logical grouping of classes, interfaces, structures, enums, delegates or even another namespace. An assembly is a physical grouping of namespaces or classes, interfaces, structures, enums, delegates etc.

An assembly contains manifest, metadata MSIL code and resources supporting that code. It is used for deployment, versioning, and security.

Q11. What is DLL hell problem?

Ans. DLL hell refers to a set of problems which are caused when a multiple applications share a common DLL. This problem arises when the shared DLL is modified/replaced by an application which causes incompatible with others dependent applications.

This is a common problem with .NET Framework 1.0 and 1.1.

Q12. How to solve DLL hell problem?

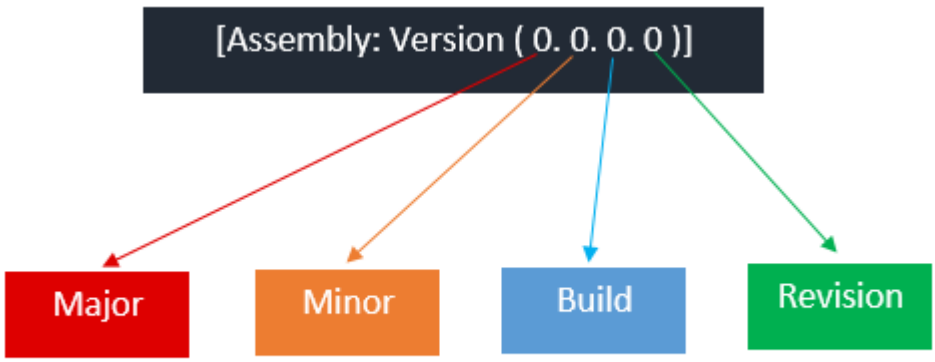
Ans. Version policy is the technique to provide a new version to the modified DLL and to prevent existing DLL from replacement. Hence, you will have the multiple versions a common shared DLL which provides compatibility across the applications.

Q13. What are parts or numbers of an assembly version number?

Ans. An assembly version number is composed from four numbers as given below –

1. **Major or Minor -** The major or minor numbers specify incompatible change(s) with in the assembly. Major or minor changes to assembly may include a change to the data types of some methods' parameters or the removal of parameters or methods altogether.

   For example, an assembly having version 4.0.0.0 or 3.1.0.0 would be considered incompatible with version 3.0.0.0



Assembly Version Parts

2. **Build -** The Build number is typically used to distinguish between daily builds or smaller compatible releases.
   For example, an assembly having version 4.0.1.0 would be considered compatible with version 4.0.0.0

3. **Revision -** The Revision number is typically reserved for an incremental build which fix a particular bug. Sometimes this also referred to as the "emergency bug fix" number.

For example, an assembly having version 4.0.0.1 would be considered full compatible with version 4.0.0.0 and a minor bug fixing.

Q14. What is Application Domain?

Ans. An application domain provides an isolated environment where an application executes. Application domains are typically created by runtime hosts. Application domain provides security and reliability.

Each application is loaded into a separate process, which isolates the application from other applications running on the same computer. The applications are isolated because memory addresses are process-relative. An application running inside a one application domain cannot directly access the code running inside another application domain. To access the code running in another application domain; application need to use proxy.

Q15. What is Global Assembly Cache (GAC)?

Ans. Each machine where the CLR is installed has a machine-wide code cache; called the global assembly cache. GAC is a special directory (C:\Windows\assembly - for CLR 2.0 and C:\Windows\Microsoft.NET\assembly - for CLR 4.0) located in Windows OS which contains shared assemblies among several applications on the computer. You can share an assembly by installing it into the global assembly cache.

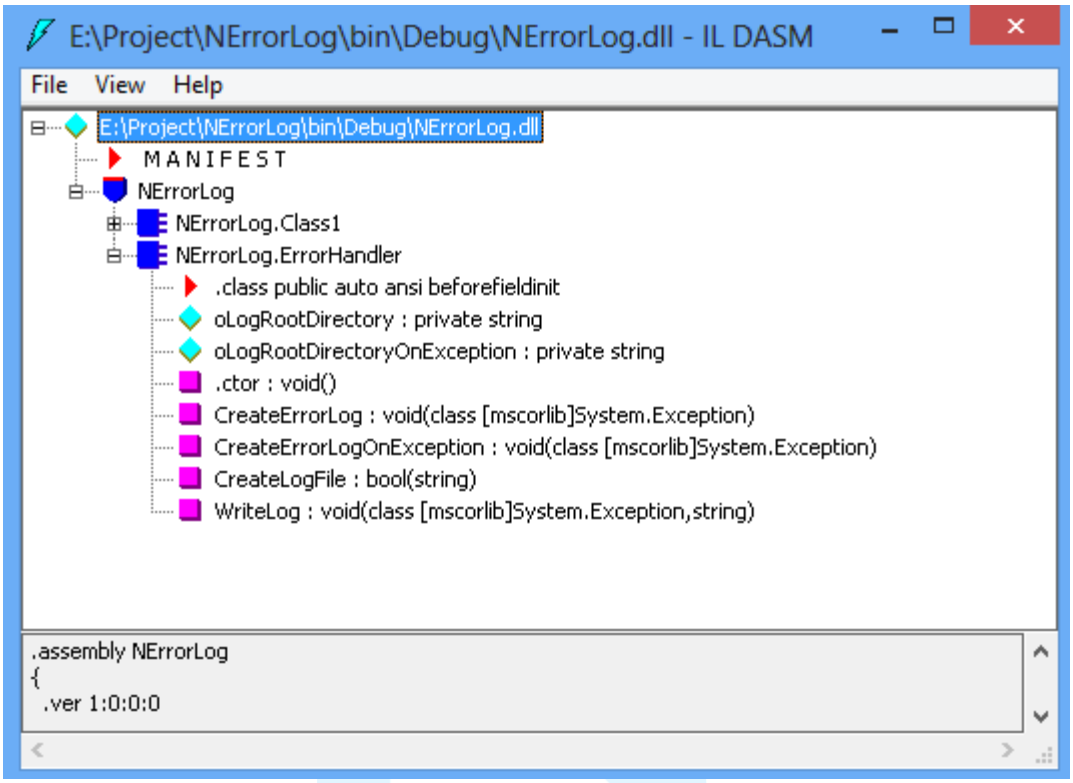
Q16. What is PE (Portable Executable) file?

Ans. PE is a file format that all EXE, object code and DLLs use to be loaded and executed by Windows OS. PE file format is derived from the Microsoft Common Object File Format (COFF).

The PE file format was defined to provide the best way for the Windows OS to execute code and also to store the essential data which is needed to run a program.

Q17. What is ILDASM (IL Disassembler) or MSILDASM?

Ans. ILDASM is a toll which used to inspect an assembly. It converts the entire assembly into the understandable instructions. This tool is automatically installed with Visual Studio and with the Windows SDK. To open ILDASM tool open Visual Studio command prompt and type command *ILDASM.*



Since metadata for .NET programs is stored in DLL files. Hence you can open DLL or EXE files in IL Disassembler and look at the MSIL code embedded in them.

4

Garbage Collector

Q1. What is GC (Garbage Collector)?

Ans. Garbage collector/collection is a CLR features that is executed as a part of your .NET program and responsible for reclaiming the memory of no longer used objects. Garbage collector automatically frees the memory for objects that are no longer referenced and keeps the memory for future allocations.

Advantage of Garbage Collector:

1. Allow us to develop an application without having worry to free memory.
2. Allocates memory for objects efficiently on the managed heap.
3. Reclaims the memory for no longer used objects and keeps the free memory for future allocations.
4. Provides memory safety by making sure that an object cannot use the content of another object.

Q2. What are garbage collection methods?

Ans. There are following methods used by garbage collection during memory reclaim process-

GC.Collect() - This method forces the garbage collection to be executed for all the generations. You can also force a garbage collection of a particular generation by passing an integer value for generation 0, 1, 2.

```
public static void Collect();
public static void Collect(int i);
```

GC.KeepAlive() - This method prevents the object from collection and extends the life time of an object passed to it as a parameter.

```
public static void KeepAlive(object objToKeepAlive);
```

GC.SupressFinalize() - This method suppresses the finalization of an object by preventing the system from calling *Finilize()* method. This method should be used when dispose() method is called by the client.

```
public static void SupressFinalize(object obj);
```

GC.GetGeneration() - This returns the current generation of an object.

```
GC.GetGeneration(object obj);
```

GC.WaitForPendingFinalizers() - This method suspend the current thread until the thread has emptied the finalization Queue.

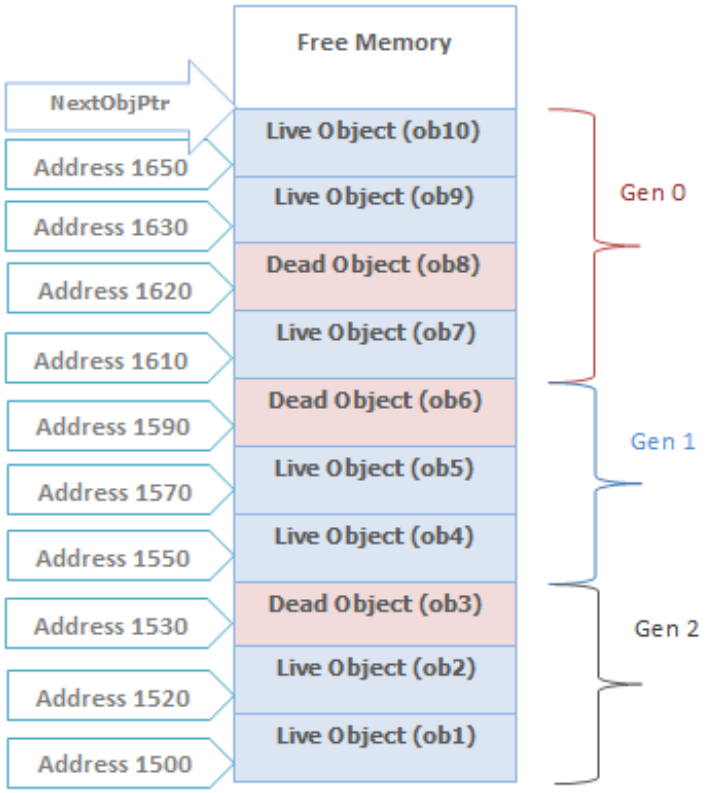
```
public static void WaitForPendingFinalizers();
```

Q3. Can you force garbage collection to run at certain time?

Ans. .NET runtime performs automatic garbage collection, However you can force the garbage collection to run at a certain point of our code by calling *System.GC.Collect()* method.

Q4. What is generation in garbage collector or managed heap?

Ans. The managed heap is organized into three generations so that it can handle short lived and long lived objects efficiently. Garbage collector first reclaims the short lived objects that occupy a small part of the heap.



Generation on Managed Heap

1. **Generation 0 -** This is the youngest generation and contains the newly created objects. Generation 0 has short-lived objects and collected frequently. The objects that survive the Generation 0 are promoted to Generation 1.
   For Example: A temporary object.

2. **Generation 1 -** This generation contains the longer lived objects that are promoted from generation 0. The objects that survive the Generation 1 are promoted to Generation 2. Basically this generation serves as a buffer between short-lived objects and longest-lived objects.

3. **Generation 2 -** This generation contains the longest lived objects that are promoted from generation 1 and collected infrequently.
   For Example: An object at application level that contains static data which is available for the duration of the process.

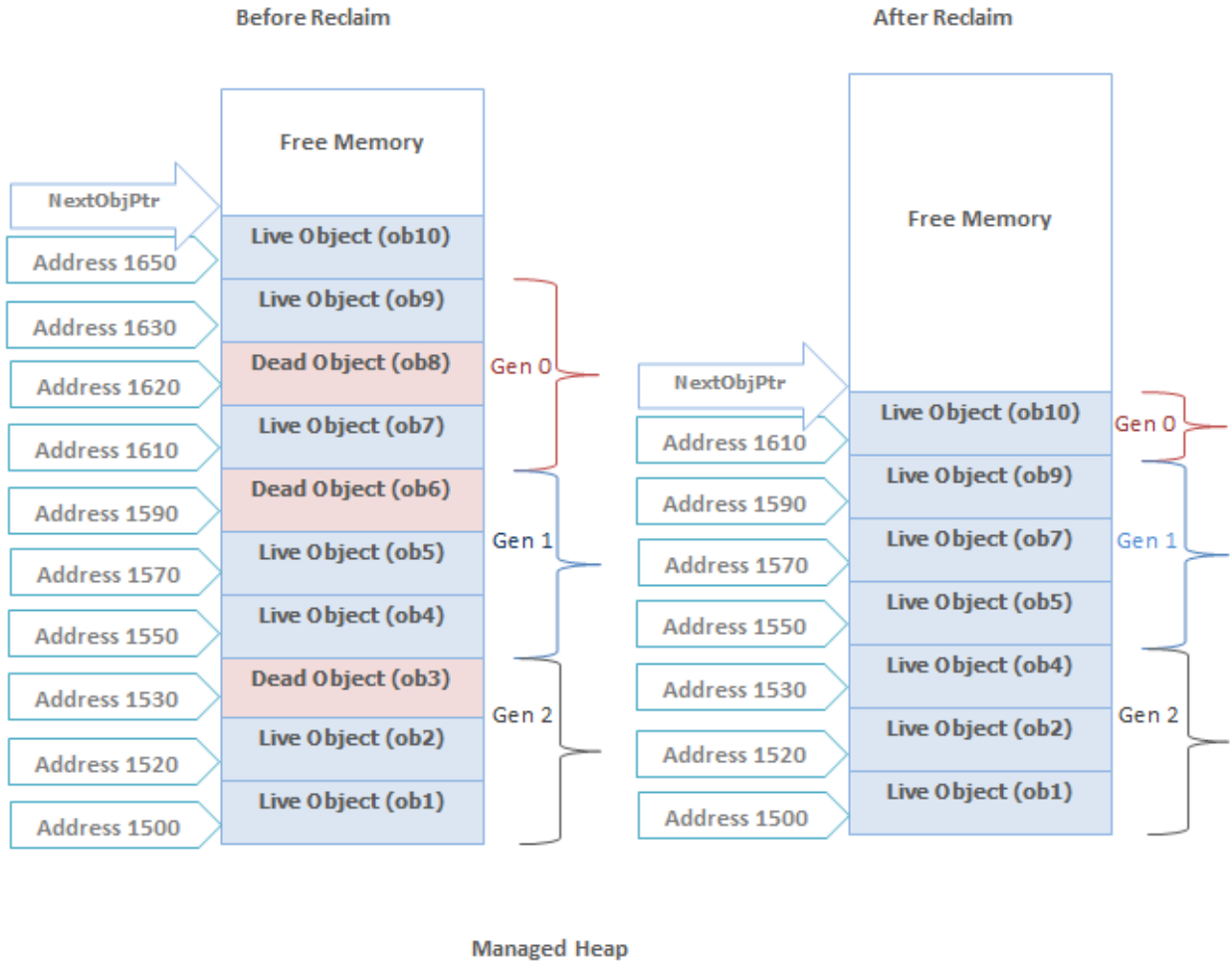
Q5. How Garbage Collector works?

Ans. Garbage Collector works in three phases as given below-

1. **Marking Phase -** In this phase garbage collector finds and creates a list of all live objects.

2. **Relocating Phase -** In this phase garbage collector updates the references to the objects that will be compacted.

3. **Compacting Phase -** In this phase garbage collector reclaims the memory occupied by the dead objects and compacts the surviving objects. The compacting phase moves the surviving objects toward the older end of the memory segment.



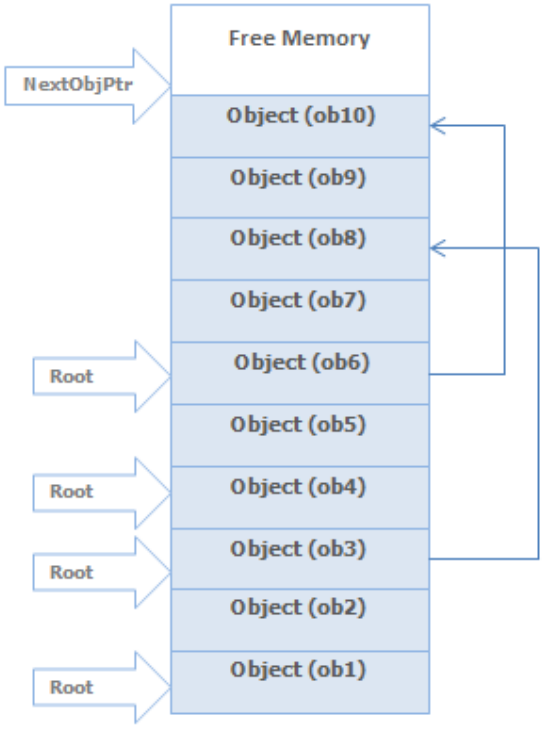
Q6. Explain garbage collector algorithm?

Ans. Garbage collector determines whether any object in the heap is dead or not being used by the application. If such objects exist then memory used by these objects can be reclaimed. But how garbage collectors know about these objects?

Each and every application has a set of roots and these identify the storage locations for the objects on the managed heap. The list of active roots is maintained by the JIT compiler and CLR, and is made accessible to the garbage collector's algorithm.

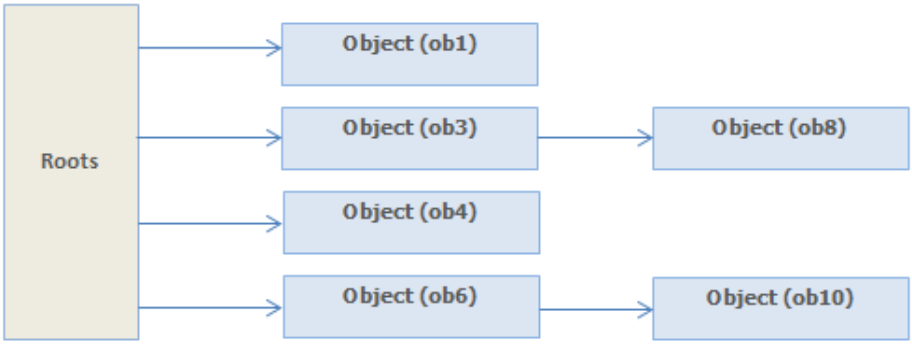
Memory Reclaim Process:

Now, the garbage collector starts go through the roots and make a graph of all the objects reachable from the roots. The below given figure shows a heap with allocated objects. In this heap the application roots directly refer to the objects 1,3,4,6 and object 3 & 6 refers to the objects 8 & 10. Hence all these objects will become the part of the live objects graph.



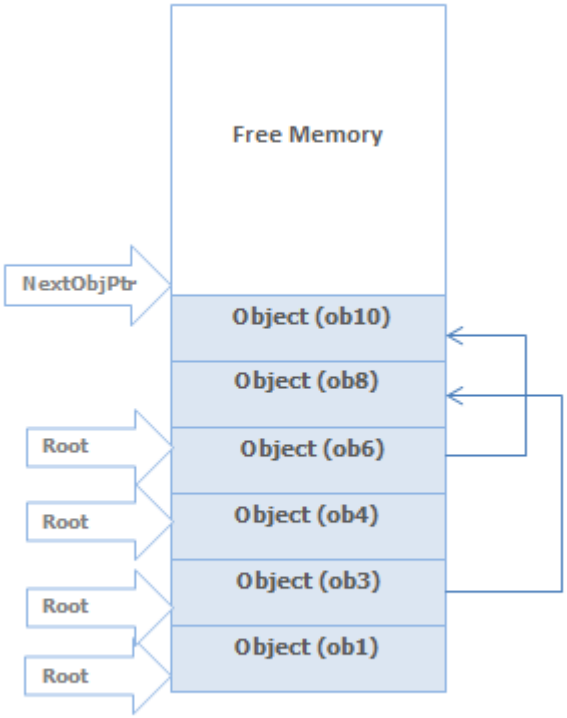
Objects in Managed Heap

The objects which are not reachable from application's roots are considered as garbage since these are not accessible by the application. In above heap objects 2,5,7,9 will be considered as dead objects.



Live objects Graph

The garbage collector then remove the dead objects from the heap and live objects will move toward the older end of the memory segment as shown in below fig. Garbage collector also updates all the references (including root references) to the moving objects in the heap.



Managed Heap After Collection

Q7. What are application roots?

Ans. Many objects in managed heap are linked to one another and make a chain, depending on which object is referenced by whom. There might be multiple such chains in the heap. The reference to the first object, i.e. the memory location to any such chain is called a Root.

All the global, static objects pointers and all the local variable/ parameter object pointers on the thread's stack in the application are considered part of the application's roots. More over any CPU registers containing pointers to objects in the managed heap are also considered a part of the application's roots.

All these roots are stored in the application stack and identify storage locations, which refer to objects on the managed heap or to objects that are set to null. The list of active roots is maintained by the JIT compiler and CLR, and is made accessible to the garbage collector's algorithm.

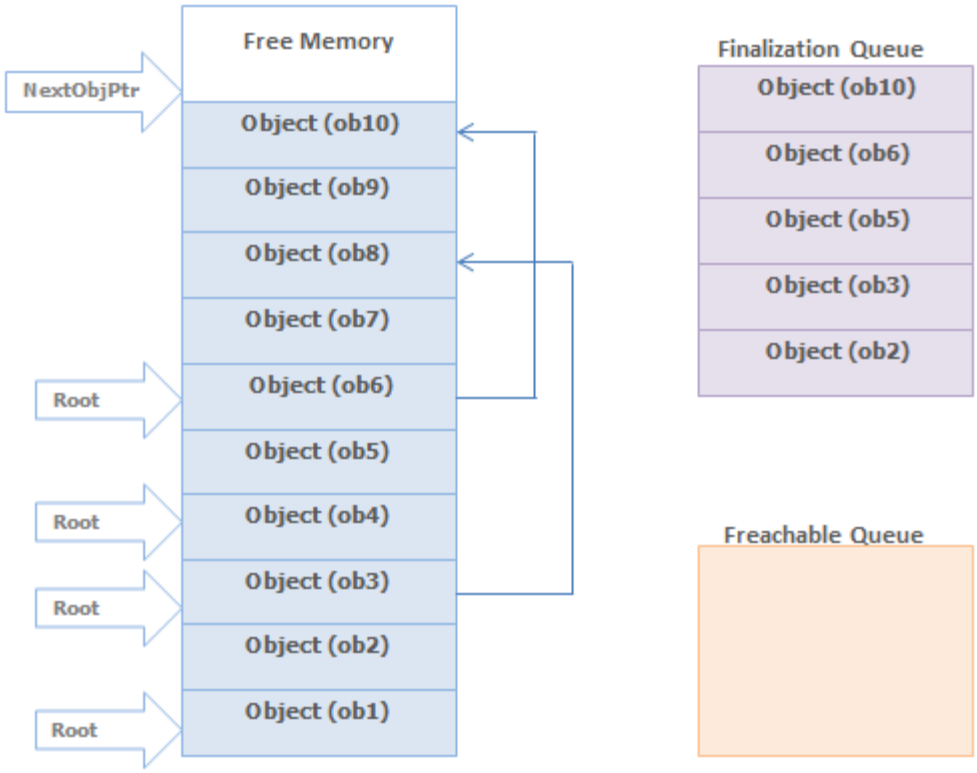
Q8. What is finalization queue and freachable queue?

OR

Explain finalization process in garbage collector?

Ans. When a new object is created, the memory is allocated in the managed heap. If newly created object have a *Finalize()* method or a destructor then a pointer pointing to that object is put into the finalization queue. Basically, **finalization queue** is an internal data structure that is controlled and managed by the GC. Hence each pointer in finalization queue points to an object that have its Finalize method call before the memory is reclaimed.

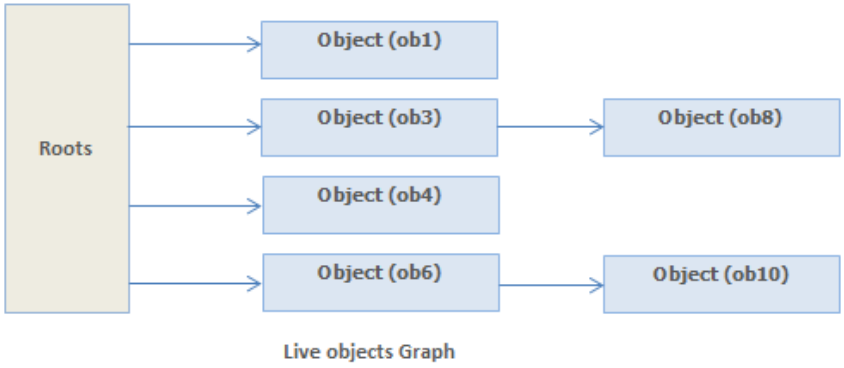
In the below fig. the managed heap contains 10 objects and objects 2, 3, 5, 6, and 10 also contains the Finalize method. Hence pointers to these objects were added to the finalization queue.



Managed Heap With Finalize Method

When the garbage collector starts go through the roots, it makes a graph of all the objects reachable from the roots. The below figure shows a heap with allocated objects. In this heap the application roots directly refer to

the objects 1,3,4,6 and object 3 & 6 refers to the objects 8 & 10. Hence all these objects will become the part of the live objects graph.

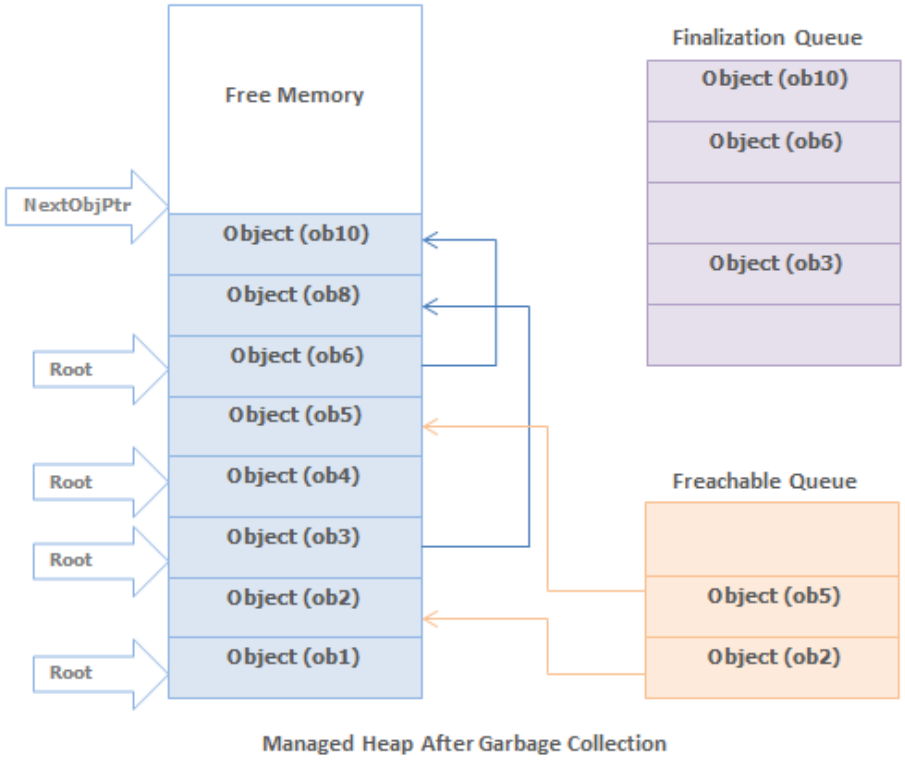


Live objects Graph

The objects which are not reachable from application's roots, are considered as garbage since these are not accessible by the application. In above heap objects 2,5,7,9 will be considered as dead objects.

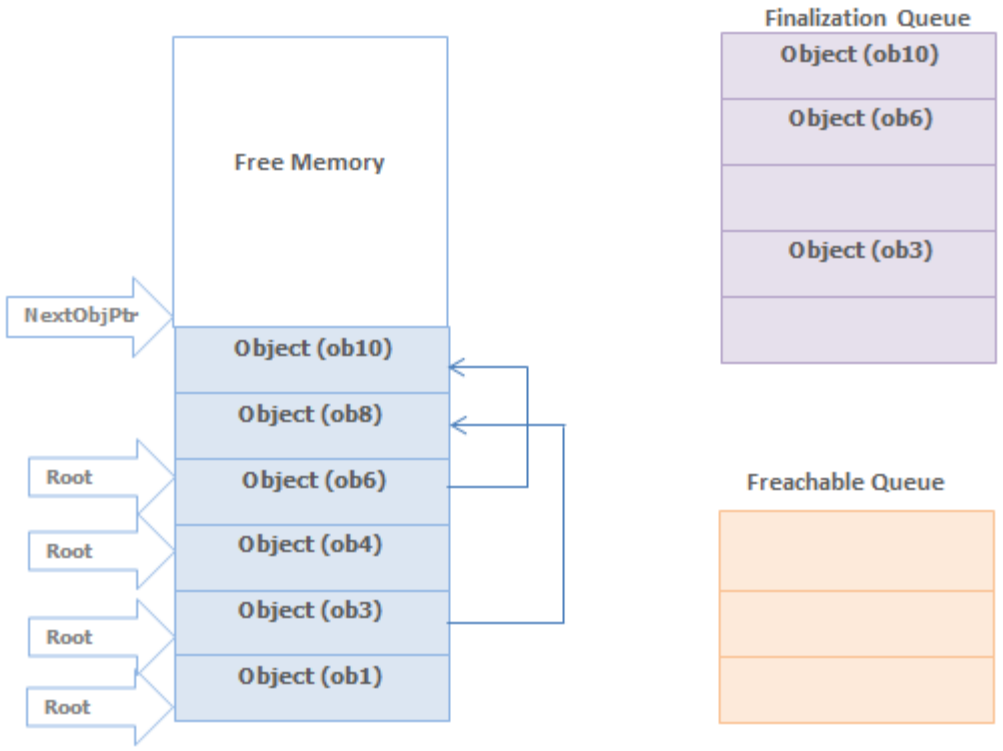
Before the collections for dead objects, the garbage collector looks into the finalization queue for pointers identifies these objects. If the pointer found, then the pointer is flushed from the finalization queue and append to the freachable queue .The **freachable queue** is also an internal data structure and controlled by the garbage collector. Now each and every pointer with in the freachable queue will identify an object that is ready to have its Finalize method called.

After the collection, the managed heap looks like below Fig. Here, you see that the memory occupied by objects 7 and 9 has been reclaimed because these objects did not have a Finalize method. However, the memory occupied by objects 2 and 5 could not be reclaimed because their Finalize method has not been called yet.



Managed Heap After Garbage Collection

When an object's pointer entry moves from the finalization queue to the freachable queue, the object is not considered garbage and its memory is not reclaimed. There is a special run-time thread that is dedicated for calling Finalize methods. When there is no entry in the freachable queue then this thread sleeps. But when there is entry in the freachable queue, this thread wakes and removes each entry from the queue by calling each object's Finalize method.



Managed Heap after Second Garbage Collection

The next time the garbage collector is invoked, it sees that the finalized objects are truly garbage, since the application's roots don't point to it and the freachable queue no longer points to it. Now the memory for the object is simply reclaimed.

Q9. What is the difference between finalize and dispose method?

Ans. Differences between these two are given below-

Finalize()	**Dispose()**
Used to free unmanaged resources like files, database connections, COM etc. held by an object before that object is destroyed.	It is used to free unmanaged resources like files, database connections, COM etc. at any time.
Internally, it is called by garbage collector and cannot be called by user code.	Explicitly, it is called by user code and the class implementing dispose method must implement IDisposable interface.
It belongs to Object class.	It belongs to IDisposable interface.
Implement it when you have unmanaged resources in your code, and want to make sure that these resources are freed when the garbage collection happens.	Implement this when you are writing a custom class that will be used by other users.
There is performance costs associated with finalize method.	There is no performance costs associated with dispose method.

For Example,

```
// Implementing Finalize method
public class MyClass
{
   //At    runtime    C#    destructor    i
automatically Converted to Finalize method
   ~MyClass ()
   {
     //TO DO: clean up unmanaged objects
   }
}
```

For Example,

```
// Implementing Dispose method
public class MyClass : IDisposable
{
   private bool disposed = false;

   //Implement IDisposable.
   public void Dispose()
   {
        Dispose(true);
   }

  protected virtual void Dispose(bool disposing)
    {
        if (!disposed)
        {
            if (disposing)
            {
              //TO DO: clean up managed objects
            }

            //TO DO: clean up unmanaged objects

            disposed = true;
        }
    }
}
```

Note -

1. You cannot override the finalize method in the C# or C++ languages. But you can override finalize method in VB.NET since it does not support destructor.
2. You should not implement a Finalize method for managed objects, because the garbage collector cleans up managed resources automatically.
3. A dispose method should call the *GC.SuppressFinalize()* method for the object of a class which has destructor because it has already done the work to clean up the object, then it is not necessary for the garbage collector to call the object's finalize method.

```
// Using Dispose and Finalize method
public class MyClass : IDisposable
{
    private bool disposed = false;

    //Implement IDisposable.
    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    protected virtual void Dispose(bool disposing)
    {
```

```
        if (!disposed)
        {
            if (disposing)
            {
                // TO DO: clean up  managed objects
            }

            // TO DO: clean up  unmanaged objects

            disposed = true;
        }
    }

    //At runtime C# destructor is   automatically Converted to Finalize method
    ~MyClass()
    {
        Dispose(false);
    }
}
```

References

This book has been written by referring to the following sites:

1. https://docs.microsoft.com/en-us/dotnet/framework- Microsoft Docs -.NET
2. https://stackoverflow.com/questions/tagged/.net - Stack Overflow -.NET
3. https://www.dotnettricks.com/learn/netframework - Dot Net Tricks -.NET