**Q8.DLL (Class Library) in C# ?**

->

A Dynamic Link library (DLL) is a library that contains functions and codes that can be used by more than one program at a time. Once we have created a DLL file, we can use it in many applications. The only thing we need to do is to add the reference/import the DLL File. Both DLL and .exe files are executable program modules but the difference is that we cannot execute DLL files directly.

# Q.9Assemblies in .NET ?

# ->

An **assembly** is a collection of types and resources that are built to work together and form a logical unit of functionality. **Assemblies** take the form of executable (.exe) or dynamic link library (. dll) files, and are the building blocks of . NET applications.

**Q.10** **NET Core and .NET Framework ?**

**->**In .NET Core and .NET Framework, you can build an assembly from one or more source code files. In .NET Framework, assemblies can contain one or more modules. This allows larger projects to be planned so that several developers can work on separate source code files or modules, which are combined to create a single assembly.

# Q11.Application domains?

->Operating systems and runtime environments typically provide some form of isolation between applications. For example, Windows uses processes to isolate applications. This isolation is necessary to ensure that code running in one application cannot adversely affect other, unrelated applications.

Application domains provide an isolation boundary for security, reliability, and versioning, and for unloading assemblies. Application domains are typically created by runtime hosts, which are responsible for bootstrapping the common language runtime before an application is run.

# Q.12 Managed Code And Unmanaged Code?

# ->

# Code that runs under the control of the common language runtime (CLR) is called managed code, and code that runs outside the CLR is called unmanaged code.

**Q12. What is .Net ?**

**->**  
1).Net is a Platform Independent Language which is used for developing various applications like Windows Applications,Web Applications ,Mobile Applications .  
2).Net is a collection of more than 20 languages.Among them mostly used languages are C#.Net,VB.Net.

**Q13.What is CIL ?**  
->CIL is the Common Intermediate Language which is generated after the Compilation of the Code.  
After Compiling Vb.net code with its VB Compiler we get CIL Code.  
After compiling C# code with its C# Compiler we get CIL Code.  
  
CIL is the Intermediate Code which we get After the Compilation of the Source Code  
  
**Q14.What is Machine Code ?**  
->Each Machine whether it is Windows or Unix consists of its own Software and Hardware . It consists of its own MicroProcessor. Every Machine will have some Instructions which can be understood by its own MicroProcessor .Those Instructions are nothing but Machine code(Machine Instructions)  
  
**Q15.What is CLR ?**  
->Machine cannot Understand CIL Code which we get after the Compilation of a Program  
CLR is used for converting the CIL Code into Corresponding Machine Code (Machine Instructions) which can be understand by its own MicroProcessor.

**Q16.What is Language Interoperability ?**  
->

Code which is written in any .NET Language can be consumed in other .NET Language.This is called as Language Interoperability .   
 Every .Net Language after Compiling generates CIL Code which can be used by any other .Net Language i.e within .Net Each Langauage can perform Operations with other language with the help of CIL. This feature of.Net is nothing but Interoperability.   
  
For Example , the CIL code which we get after compiling C# Language can be used by VB Language and the CIL Code which we get after compilation of VB code can be used by C#

**Q17.How is this Possible ?**  
->Though .Net is a collection of languages ,Every language has its own Data Types.C# has its own Data Types and VB.Net has its own Data Types but their sizes are same .After compilation any .Net language CIL Code will be generated which can be used by other Languages of .Net  
  
Examples: After Compiling VB Code, Data Types in that Language will be Converted into CIL Type and it can be used C# and Vice-Versa  
 **VB Datatypes converts to CIL which can be used by C#  
  
VB --------> CIL --------> C#  
  
Integer --> Int32 --------> int  
  
Single --> Single --------> float  
  
Boolean --> Boolean --------> bool  
  
C# Datatypes converts to CIL which can be used by VB  
  
C# ----> CIL --------> VB  
  
Int --> Int32 --------> integer  
  
Float --> Single --------> single  
  
Bool --> Boolean --------> boolean**  
  
**Language Interoperability is possible because even though DataTypes names are Different in Different .Net Languages their Sizes are same for all .Net Languages but after Compiling they will generate same CIL Code which can be used by any other .Net Languages**  
  
Every .Net Language after Compiling Generates CIL Code which can be used in other .Net Langauges.

## Q18.Features of C# Language

->There are many important features of C# language that make it more useful and unique compared to other languages.

* Fast Speed
* Simple
* Object-Oriented
* Modern Programming Language
* Type-Safe
* Interoperability
* Scalable and Updateable
* Structured Programming Language
* Rich Library
* Component Oriented

### Fast Speed

C# language is very fast, its compilation, and execution time is too quick.

### Simple

C# is a simple language. It gives a structured approach to breaking the problem into parts. Also, It has a rich set of library functions and data types. C# language code does not require header files. Its code is written inline.

### Object-Oriented

C# language is an object-oriented programming language. Similarly, Oops makes development and maintenance easier as compare to Procedure-oriented programming language.

However, it is too difficult to manage if the code grows as project size grows. Besides, C# programming supports Data Encapsulation, inheritance, polymorphism, interfaces.

### Modern Programming Language

C# language is one of the modern programming languages because it is based upon the current trend. However, it is very simple, powerful for building scalable, interoperable and robust applications.

### Type-Safe

C# language is type-safe code that can only access the memory location and has permission to execute. Therefore, it improves the security of the program.

In C# language, you can’t perform unsafe casts like convert double to a Boolean. Its value types (primitive types) are initialized to zeros and reference types (objects and classes) are initialized to null by the compiler automatically.

### Interoperability

Interoperability is the process that enables the C# programs to do almost anything that a native C++ application can do. In brief, language interoperability is the ability of code to interact with code that is written using a different programming language. It can help maximize code reuse and, therefore, improve the efficiency of the development process.

C# language provides support for using COM objects, no matter what language was used to author them. However, it also supports a special feature that enables a program to call out any native API.

### Scalable and Updateable

C# language is a computerized scalable and update-able programming language. However, one important thing is, that to update your .Net framework. you have to kill your old files and update them with the new one.

### Structured Programming Language

C# language is a structured programming language. However, structured programming languages is a subset of procedural programming that enforces a logical structure on the program being written to make it more efficient and easier to understand and modify.

In other words, to solve large problems C# programming divides the problem into smaller modules called functions or procedures each of which handles a particular responsibility that’s why C# language called a structured programming language.

### Rich Library

C# language is rich in the library. So that it provides a lot of inbuilt functions that make development fast.

### Component Oriented

C# language is a component-oriented programming language and supports component-oriented programming through the concepts of methods, properties, events, and attributes (or metadata), allowing self-contained and self-describing components of functionality called assemblies.

In the above section, we talked about the C# programming language, now we discuss the advantages of C#.

### Object-oriented

In the above section, we already discussed that C# programming language is a pure object-oriented language so that it allows you to create modular maintainable applications and reusable codes. This is one of the biggest advantages of C# over C++ languages.

### Cross-Platform

The most important requirement for C# programming is the NET framework. Your machine has to install the NET Framework to run your application well.

### Automatic Garbage Collection

In C# programming, a very efficient system installed that collects and erases garbage automatically present on the system. However, we called that C# language is very efficient in managing the system because it doesn’t create a mess in the system, and the system doesn’t get hanged during execution.

### Avoid the problem of memory leak.

The major benefit of C# language is its strong memory backup. C# programming language contains high memory backup so that memory leakage problem and other such types of problem is not occurring as it happens in the case of C++ language.

### Easy-to-Development

C# language has a rich class of libraries that make many functions easy to be implemented. The C# programming language influences most of the programmers of the world and has a history in the programming world.

### Better Integration

An application written in .NET will have better integration and interpret-ability as compared to other NET Technologies. C# programming runs on C.L.R that making it easy to integrate with components written in other languages.

### Cost-benefit

The maintenance cost is less and is safer to run as compared to other languages. C# language can develop iOS, Android and Windows Phone native apps, with the help of the Xamarin framework.

### Familiar syntax

It is pretty easy to pick up and work productively with a working knowledge of languages like C, C++, Java because its core syntax is similar to C-style languages.

### Programming support

You can buy support from Microsoft in C# programming. If any issue occurs you can solve it with the support of Microsoft.

### Properties and Indexers

C# programming has features like Properties and Indexers which are not available in Java language.

### Most useful

It can develop iOS, Android and Windows Phone native apps, with the help of the Xamarin framework. However, it is also greatly used for developing a Windows app (Mobile, Desktop).

### Most Powerful

C# language is the most powerful programming language for the .NET Framework.

### Motivate towards work

We already discussed that .NET applications work on Windows platforms only and Microsoft keeps retiring support for old Windows platforms. So always you would need to upgrade your .Net framework.

But after the upgrade, this could be an advantage or a disadvantage as well. Hence, it always motivates you to work hard and excel in your field and this is a good thing in my point of view.

## ****Q18.Value type****

Value types are generally (not always) stored on the stack and are passed by copying.  
  
The way in which a variable assignment works differs between reference and value types.

A Value Type stores its contents in memory allocated on the stack. When you created a Value Type, a single space in memory is allocated to store the value and that variable directly holds a value. If you assign it to another variable, the value is copied directly and both variables work independently. Predefined datatypes, structures, enums are also value types, and work in the same way. Value types can be created at compile time and Stored in stack memory, because of this, Garbage collector can't access the stack.

1. **class** Program
2. {
3. **static** **void** Main(**string**[] args)
4. {
5. A obj1 = **new** A(12);
6. **int** v1 = 12;
7. **int** v2 = 22;
8. v2 = v1;
9. Console.WriteLine(v2);
10. Console.ReadLine();
11. }
12. }

## ****Reference Type****

A value type is basically stored on the heap and passed by creating a reference.

Reference Types are used by a reference which holds a reference (address) to the object but not the object itself. Because reference types represent the address of the variable rather than the data itself, assigning a reference variable to another doesn't copy the data. Instead it creates a second copy of the reference, which refers to the same location of the heap as the original value. Reference Type variables are stored in a different area of memory called the heap. This means that when a reference type variable is no longer used, it can be marked for garbage collection. Examples of reference types are Classes, Objects, Arrays, Indexers, Interfaces etc.

1. **using** System;
2. **class** A {
3. **public** **int** value
4. {
5. **get**;
6. **set**;
7. }
8. **public** A(**int** passbyref)
9. {
10. **this**.value = passbyref;
11. }
12. }
13. **class** Program
14. {
15. **static** **void** Main(**string**[] args)
16. {
17. A v1 = **new** A(12);
18. A v2 = **new** A(22); //Breakpoint
19. v2 = v1;
20. Console.WriteLine(v1.value);
21. Console.WriteLine(v2.value);
22. Console.ReadLine();
23. }
24. }

## Stack and Heap

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM.

## Class and Struct

Class is pass-by-reference and Struct is pass-by-copy, it means that, Class is a reference type and its object is created on the heap memory where as structure is a value type and its object is created on the stack memory.

## Dynamic Data Type

The dynamic keyword brings exciting new features to C# 4. Dynamic Type means that you can store any type of value in the dynamic data type variable because type checking for dynamic types of variables takes place at run-time.

# Q19.Boxing and Unboxing ?

# -> Boxing:-

# Boxing is used to store value types in the garbage-collected heap. Boxing is an implicit conversion of a [value type](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types) to the type object or to any interface type implemented by this value type. Boxing a value type allocates an object instance on the heap and copies the value into the new object.

**UnBoxing:-**   
Unboxing is an explicit conversion from the type object to a [value type](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types) or from an interface type to a value type that implements the interface. An unboxing operation consists of:

* Checking the object instance to make sure that it is a boxed value of the given value type.
* Copying the value from the instance into the value-type variable.

# Boxing is the process of converting a [value type](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/value-types) to the type object or to any interface type implemented by this value type. When the common language runtime (CLR) boxes a value type, it wraps the value inside a [System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) instance and stores it on the managed heap. Unboxing extracts the value type from the object. Boxing is implicit; unboxing is explicit. The concept of boxing and unboxing underlies the C# unified view of the type system in which a value of any type can be treated as an object.

**Boxing Example:**-

int i = 123;

// The following line boxes i.

object o = i; //Boxing

**UnBoxing Example:**-

o = 123;

i = (int)o; // unboxing