C#.Net interview questions for freshers

1) What is an internal modifier?

- **Internal members can access only from within the same assembly (.dll).**
- **We can declare a class as internal, its member as internal or its fields as internal.**
- **Use internal keyword before the class declaration to create an internal class.**
- **Internal type can't access from external program.**
- **Classes defined within the current assembly can access internal classes.**

2) What are namespaces, and how they are used?

- **In .Net framework namespaces are used to manage classes.**
- **The Key difference between .Net namespaces and java packages is that namespace doesn't define the physical layout of source file while java packages do.**
- **Namespace define logical structure of the code.**
- **Namespaces can be utilized via using keyword, in .net framework, many class have their namespace defined such as System.Net.**
- **We can create our own C# source files which can relate to multiple projects.**

3) Why are strings in C# immutable?

- **When string values changes it means string is immutable, in this whenever we assign new value to string it occupies new memory reference for its new value by creating new string instance.**
- **String is declared by using String Keyword.**
- **Always use mutable string defined in System.Text.StringBuilder whenever its values will change.**
- **Inefficient use of memory and garbage collection resulted by Immutable string.**

4) What is boxing?

- **When Value type is explicitly converted into reference type is called as boxing.**
- **The reverse process is known as unboxing, when reference type is explicitly converted into value type.**
- **In boxing process, the Values of variable is stored in Stack is converted in an object reference is stored in heap.**
- **If the recipient reference type is equivalent to the unboxed type, then the value is copied to the heap.**
- **Unboxing called the vice versa process of boxing.**

5) How does C# differ from C++?

- **C# doesn't support multiple inheritances while C++ does.**
- **We can use switch statement with string values in C# while in C++ only Character and integer values supported.**
- **Casting is Safer in C# than C++.**
- **C# doesn’t require semicolon on completion of class definition while C++ use.**
- **In C#, Command line parameter acts differently as compared to C++.**

6) What is the difference between public, static and void?

- **public: Public Modifier in C# is most liberal among all access modifiers, it can access from anywhere inside or outside other class. There is no access restriction in public modifiers .public keyword is used just before the class keyword to declare class as public.**
- **static: Static method is used to declare main method global one and we do not need to create instance of that class. We can access methods of static class by using the class name followed by. Operator and method name .The compiler stores Static method’s address and uses this information before any object is created for execution.**
- **void: The void modifier tells that the Main method can't return any value.**

7)What is shadowing?

There are two ways of shadowing either through scope or through inheritance.

- **Hiding a method of child class and giving a new implementation is known as shadowing from inheritance.**
- **Shadowing through inheritance is the default when a derived class implements a method of base case which is not declared as overridden in the base class.**
- **Derived class member's signature, return type, access level may differ from base class.**
- **Shadowing can be obtained by using the new keyword.**
- **Shadowing is one of the polymorphism's concepts.**

8) What are the difference between Structure and Class?

- **Structures are Values types while Classes are Reference types.**
- **In structure values stored in stack while in class value's reference stored in heap.**
- **In structure direct values is stored while in class reference to a value is stored.**
- **Inheritance is supported in classes while structure doesn't support.**
- **We cannot declare destructor in structure whereas in class it is possible.**
- **We can't have explicitly parameter less constructors in structure whereas classes can have.**
- **Class can have protected members while structure can't have.**
- **Structure is declared by using struct keyword while class is declared by using Class keyword.**
- **Structures don't have memory management while classes have due to garbage collector.**
- **New operator works in classes while not in structure.**

9) How does assembly versioning work?

- **There is an assembly's version called the compatibility version.**
- **Version number is defined by four numeric parts (e.g. 5.5.3.11).**
- **Both the name and the version of the referenced assembly are included in each reference to an assembly.**
- **The two parts of assembly is normally seen as incompatible, but if third part is different than assemblies are deemed as 'may be compatible', and if the only fourth part of an assembly is different than assembly is compatible.**
- **We can specify version policy in the application configuration file.**
- **We cannot apply versioning to private assemblies; it is applied only to shared assemblies.**

10) What is Custom Control?

- **A Custom control inherits from System.Windows.Controls.Control class.**
- **They are compiled code (Dlls), faster to use, difficult to develop, and can be placed in toolbox.**
- **Custom controls can be derived from different custom controls according to requirement.**
- **Custom controls can be reused on multiple places easily.**
- **Provide more flexibility in extending the control's behavior.**
- **Custom controls are loosely coupled control in respect to code an UI.**
- **Custom controls can be used just by drag and drop into the form.**
- **Custom controls have dynamic layout.**

11) What is User Control?

- **A User control inherits from System.Windows.Controls.UserControls class.**
- **User control defines UI as XMAL.**
- **User control has fixed UI and can't have different look in every project.**
- **They are tightly coupled controls.**
- **We can't add them to the toolbox.**
- **They are less flexible as compared to Custom controls.**

12) How can I produce an assembly?

An assembly can produce from a .Net compiler.

For Example, the following C# program:

```
public class CDemo
{
public CDemo()
{
System.Console.WriteLine( "Hello from CDemo" );
}
}
```

Can be compiled into a dll as:
Csc/t:library cdemo.cs

Now the contents of assembly can be view through opening the "IL Disassembler" tool of .Net SDK.

Also compiling source into modules and then using assembly linker (al.exe) for combining them to generate assembly.

13) What is an application domain?

- **An AppDomain is a lightweight activity.**
- **Goal of Appdomain is to isolate application from each other which is useful in hosting like Asp.net Application.**
- **The host can destroy the Appdomain without loosing others in the process.**
- **.Net runtime managed AppDomain memory to ensure that they don't access other's memory.**
- **AppDomain is useful in creation and destruction of types on the fly.**
- **AppDomain Created automatically on creation of new application by CLRHost.**

14) What are the features of C#?

- **C# is a powerful and simple programming language for writing applications.**
- **Developers can easily build the web services through any language, on any platform across the internet.**
- **C# is a hybrid of C++ and VB.**
- **C# has many C++ features in the area expressions, operators and statements.**
- **C# introduces improvement in boxing, unboxing, type safety, events, and versioning and garbage collections.**
- **It reduces programming error in the code due to fewer lines of code.**
- **C# has a key feature that can split up the implementation into logical pieces called region.**
- **It support multiline comment feature.**

15) What is an abstract class?

- **We can't create the instance of an abstract class, because abstract classes are incomplete.**
- **Abstract modifier doesn't supported by interface, Values type and static types.**
- **We can declare class as abstract by using abstract keyword in beggining of the class definition.**
- **Abstract class provides similar definition of base class which can be shared by multiple derived classes.**
- **Abstract class may define abstract methods by using the abstract keyword before the method definition.**
- **Abstract methods have no implementation, only they can be implemented in the derived class.**

For Example:

```
public abstract class Demo
{
public abstract void DoWork(int a);
}
```

16) What is Managed code?

- **Manage Code are the codes which require common language runtime for its execution.**
- **Managed code target the services of the common language runtime in .Net framework.**
- **Managed code must provide the metadata necessary for the runtime to provide services like memory management, cross-language integration, code access security, and automatic lifetime control of objects.**
- **Managed code provides services like cross-language integration, code access security, memory management and automatic lifetime control of objects by supplying necessary metadata for runtime.**
- **Codes executed by Microsoft intermediate language are Managed code.**

17) What is Un-Managed Code?

- **Code which is perfectly executed by operating system is known as un-managed code.**
- **C, C++, VB 6.0 are examples of unmanaged code.**
- **Un-managed code all the time dependent on computer architecture.**
- **Un-managed code always compile to native code, so require compilation of code for different platform again and again.**
- **Developers should take care of type safety, security, memory allocations.**

18) What is the difference between an event and a delegate?

- **Events are the reason of happening and delegates are the address of function.**
- **In .Net, Delegates looks like an interface with a single method and you can make a call to it by delegate's instance.**
- **By event's you can let other people know that something going on.**
- **Events is declared by using "event" keyword whereas delegate is declared by using "delegate" keyword as below**
  Public event <nameofth handler>;
  Public delegates <type> <delegatename>;
- **Adding a public multicast delegate field is same as adding a public event to a class.**

19) How does .NET remoting work?

- **.NET remoting is a process of sending messages through channels.**
- **Standard channels in remoting are HTTP and TCP.**
- **TCP is used for LANs only while HTTP can be used for LANs or WANs (internet).**
- **Support can provided for many message serialization formats. Examples are SOAP (XML-based) and binary.**
- **By default, the HTTP channel uses SOAP and the TCP channel uses binary. channela can use any serialization format.**
- **There are two styles of remote access:**

1) SingleCall. Every request from a client is serviced by a new object. The object is thrown away on completion of request.
2) Singleton. Single server object processed all request from clients.

20) How do I spawn a thread?

We need to create an instance of a System.Threading.Thread object and passing it an instance of a ThreadStart delegate that will be executed on the new thread.

For example:

```
class MyDemoThread
{
public MyDemoThread( string initData )
{
m_data = initData;
m_thread = new Thread( new ThreadStart(ThreadMain) );
m_thread.Start();
} // ThreadMain() is executed on the new thread.
private void ThreadMain()
{
Console.WriteLine( m_data );
}
public void WaitUntilFinished()
{
m_thread.Join();
private Thread m_thread;
private string m_data;
}
```

In this case creating an instance of the MyDemoThread class is sufficient to spawn the thread and execute the MyDemoThread.ThreadMain() method:

```
MyDemoThread t = new MyDemoThread( "Hello, world." );
t.WaitUntilFinished();
```

21) What are partial classes?

- **Derived classes focus more on important portions by splitting a class definition into multiple source files using partial classes.**
- **In partial class code is hidden in a file named Form.Designer.vb or Form.Designer.cs.**
- **Partial class files can be viewed by expending all files in solution explorer.**
- **We can enhance and extend auto-generated code by using partial classes.**
- **These multiple files get compiled into a single class during compilation.**

22) What are attributes?

- **Attributes describe a type, method, or property in a way that can be queried using a technique call reflection.**
- **Attributes specify which security privileges a class requires.**
- **Attributes provide a description, title, and copyright notice to explain an assembly.**
- **Attribute types derive from the System. Attribute base class and are declared using <> or [] notation.**
- **Visual studio automatically creates some standard attributes for your assembly when you create a project including title, description, company, guide and version.**
- **Attribute can also declare requirements or capabilities.**

23) What are the File System Classes?

- **The file system classes are separated into two types of classes: information and utility.**
- **Information classes derive from the FileSystemInfo base class.**
- **Information Classes exposes all the system information about file system objects like files, directories and drives and named as FileInfo and DirectoryInfo classes.**
- **DriveInfo class cannot derive from the FileSystemInfo class; it is an information class which represents a drive in the file system.**
- **The utility classes have static methods to perform function on file system objects such as file system paths and Directories.**
- **System.IO namespace contains collection of classes for files, drives and directories.**

24) Switch statement works differently than in C++, why?

C# does not support an explicit fall through for case blocks.

The following code is not legal and will not compile in C# :

```
switch(a)
{
case 0: // some code here
case 1: // some code here
default: // some code here
break;
}
```

To run above code in C#, the code must be modified as shown below

```
class Demo
{
public static void Main()
{
int a = 3;
switch(a)
{
case 0: // do some code here
goto case 1;
case 1: // do some code here
goto default;
default: // do some code here
break;
}
}
}
```

25) Why Trim function is used in C#?

The Trim function is used to remove white spaces from the end and start of the instance.

Trim provide following methods as:

1) Trim(): This method of Trim is used to remove both off spaces from start and the end.
2) TrimStart(): This method of Trim is used to remove white spaces from the start of String.
3) TrimEnd(): This method of Trim is used to remove white spaces from the end of the string.