C# Interview Question

1) What is C#?

C# is a simple, modern, general purpose programming language. It is an object oriented programming language developed by Microsoft. It is a safe and managed language that is compiled by .NET framework to generate Microsoft intermediate language (machine code).

2) What is the reason behind the invention of C#?

C# is designed for Common Language Infrastructure (CLI). It contains the executable code and runtime environment that makes the users able to use various high-level languages on different computer platforms and architectures.

3) What are the main reasons to use C# language?

These are top reasons to use C# language:

* Easy to learn
* General purpose and object oriented programming language
* Component oriented
* Structured language
* Can be compiled on variety of computer platforms
* Produces efficient programs
* Part of .net framework

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

You can access public declared variables anywhere in the application.

Static declared variables are globally accessible without creating an instance of the class.

Void is a type modifier that specifies that the method doesn't return any value.

5) What are constructors in C#?

A constructor is a member function in the class and has the same name as its class. Whenever the object class is created, the constructor is automatically invoked. It constructs the value of data members while initializing the class.

6) What are the different types of constructors in C#?

Basically, there are five types of constructors:

* Static constructor
* Private constructor
* Copy constructor
* Default constructor
* Parameterized constructor

7) What is static constructor?

Static constructor is used to initialize static data members as soon as the class is referenced first time.

8) What is method overloading in C#?

Method overloading is mechanism to create multiple methods with the same name and unique signature in the same class. When you go for compilation, the compiler uses overload resolution to determine the specific method to be invoked.

9) Is overriding of a function possible in the same class?

No

10) What is array?

Array is a set of related instances either value or reference types.

There are three types of array supported by C#:

* **Single Dimensional Array**: It contains a single row. It is also known as vector array.
* **Multi Dimensional Array**: It is rectangular and contains rows and columns.
* **Jagged Array**: It also contains rows and columns but it has an irregular shape.

11) What is ArrayList?

ArrayList is a dynamic array. You can add and remove the elements from an ArrayList at runtime. In the ArrayList, elements are not automatically sorted.

12) What is a collection?

A collection works as a container for instances of other classes. All classes implement ICollection interface.

13) What is an interface?

Interface is an abstract class that has only public abstract method. These methods only have declaration not the definition. These abstract methods must be implemented in the inherited classes.

14) What is the lock statement in C#?

Lock statement is used to ensure that one thread doesn?t enter a critical section of code while another thread is in the critical section. If another thread attempts to enter a locked code it will wait, block, until the object is released.

15) What is serialization?

If you want to transport an object through network then you have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called serialization.

16) How to declare a property in a class?

1. **int** m\_PersonID = 0;
2. **public** **int** PersonID
3. {
4. get { **return** m\_PersonID; }
5. set { m\_PersonID = value; }
6. }

17) What is the difference between early binding and late binding in C#?

Early binding and late binding are the concept of polymorphism. There are two types of polymorphism in C#.

* **Compile Time Polymorphism**: It is also known as early binding.
* **Run Time Polymorphism**: It is also known as late binding or method overriding or dynamic polymorphism.

18) Which are the access modifiers available in C#?

Following are the access modifiers generally used for accessibility:

* **Public**: If you define an attribute or method as public, it can be accessed from any code of the project.
* **Private**: A private defined attribute or method can be accessed by any code within the containing class only.
* **Protected**: If you define the method or attribute as protected it can be accessed by any method in the inherited classes and any method within the same class.
* **Internal**: If you define an attribute or a method as internal, it is restricted to classes within the current position assembly.
* **Protected internal**: If you define an attribute or method as protected internal, access is restricted to classes within the current project assembly or types derived from the containing class.

19) What is the difference between abstract class and interface in C#?

Abstract class can have abstract and concrete methods whereas interface has only abstract methods.

20) What is the difference between dispose() and finalize() methods in C#?

The dispose() method is explicitly called by user to free unmanaged resources such as files, database connections etc whereas finalize() method is implicitly called by garbage collector to free unmanaged resources like files, database connections etc.

The dispose() method belongs to IDisposable interface whereas finalize() method belongs the Object class.

21) What is the difference between method overloading and method overriding in C#?

Method parameters must be different in method overloading whereas it must be same in method overriding.

Inheritance is not required in method overloading, it occurs within the same class. But inheritance is required in method overriding.

22) What is object pool in .Net?

Object pool is a container of ready to use objects. It reduces the overhead of creating new object.

23) What is delegate in C#?

A delegate in C# is an object that holds the reference to a method. It is like function pointer in C++.

24) What is Hashtable?

A Hashtable is a collection of key/value pairs. It contains values based on the key.

25) What is Reflection?

Reflection allows us to get metadata and assemblies of an object at runtime.

26) What is Garbage Collection?

Garbage Collection is a process of releasing memory automatically occupied by objects which are no longer accessible.

27) Explain Abstraction.

**Abstraction** is one of the OOP concepts. It is used to display only the essential features of the class and hides the unnecessary information.

28) Explain Polymorphism?

Programmatically, **Polymorphism** means same method but different implementations.

It is of 2 types, Compile-time and Runtime.

**Compile time polymorphism** is achieved by operator overloading.

**Runtime polymorphism** is achieved by overriding. Inheritance and Virtual functions are used during Runtime Polymorphism.

**For Example**, If a class has a method Void Add(), polymorphism is achieved by Overloading the method, that is, void Add(int a, int b), void Add(int add) are all overloaded methods.

29) How is Exception Handling implemented in C#?

**Exception handling is done using four keywords in C#:**

* **try** – Contains a block of code for which an exception will be checked.
* **catch** – It is a program that catches an exception with the help of exception handler.
* **finally** – It is a block of code written to execute regardless whether an exception is caught or not.
* **Throw** – Throws an exception when a problem occurs.

30) What is a Destructor in C#?

Ans: A Destructor is used to clean up the memory and free the resources. But in C# this is done by the garbage collector on its own. System.GC.Collect() is called internally for cleaning up. But sometimes it may be necessary to implement destructors manually.

For Example:

~Car()

{

Console.writeline(“….”);

}

31) What are Boxing and Unboxing?

Ans: Converting a value type to reference type is called Boxing.

For Example:

int Value1 -= 10;

//————Boxing——————//

object boxedValue = Value1;

Explicit conversion of same reference type (created by boxing) back to value type is called Unboxing.

For Example:

//————UnBoxing——————//

int UnBoxing = int (boxedValue);

32) Explain Copy constructor in C#?

If the constructor contains the same class in the constructor parameter then it is called as copy constructor.

class MyClass  
{  
 public string prop1, prop2;  
 public MyClass(string a, string b)  
 {  
 prop1 = a;  
 prop2 = b;  
}

public MyClass(MyClass myobj) // Copy Constructor  
 {  
 prop1 = myobj.prop1;  
 prop2 = myobj.prop2;  
 }  
}

33) Explain Static constructor in C#?

If the constructor is declared as static then it will be invoked only once for all number of instances of a class. Static constructor will initialize the static fields of a class.

class MyClass  
{  
 public string prop1, prop2;  
 public MyClass(string a, string b)  
 {  
 prop1 = a;  
 prop2 = b;  
 }

Static MyClass()  
 {  
 Console.WriteLine(“Static Constr Test”);  
 }  
 public MyClass(MyClass myobj) // Copy Constructor  
 {  
 prop1 = myobj.prop1;  
 prop2 = myobj.prop2;  
 }  
}

34) What are the collection types can be used in C#?

Below are the collection types in C# -

* ArrayList
* Stack
* Queue
* SortedList
* HashTable
* Bit Array

**35) What is an Object?**  **What is a class ?**

36)Different types of overloading in C#

* Constructor overloading
* Function overloading
* Operator overloading

37) Explain Inheritance in C# ?

In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects. In inheritance, there will be two classes - base class and derived classes. A class can inherit attributes and methods from existing class called base class or parent class. The class which inherits from a base class is called derived classes or child class. For more clarity on this topic, let us have a look at 2 classes shown below. Here Class Car is Base Class and Class Ford is derived class.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

}

**38) Explain the use of Virtual Keyword in C# ?**

When we want to give permission to a derived class to override a method in base class, Virtual keyword is used. For example. lets us look at the classes Car and Ford as shown below.

class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public virtual void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public Ford()

{

Console.WriteLine("Derived Class Ford");

}

public void Price()

{

Console.WriteLine("Ford Price : 100K $");

}

public override void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

39) What is Sealed Classes in c# ?

If a class is defined as Sealed, it cannot be inherited in derived class. Example of a sealed class is given below.

public sealed class Car

{

public Car()

{

Console.WriteLine("Base Class Car");

}

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}

40) What is Method Hiding in C# ?

If the derived class doesn't want to use methods in the base class, derived class can implement it's own version of the same method with same signature. For example, in the classes given below, DriveType() is implemented in the derived class with same signature. This is called Method Hiding.

class Car

{

public void DriveType()

{

Console.WriteLine("Right Hand Drive");

}

}

class Ford : Car

{

public void DriveType()

{

Console.WriteLine("Right Hand ");

}

}