**1. What is an Object?**

Object in an instance of a Class. Technically, it is just a block of memory allocated that can be stored in the form of Variables, Array or a Collection

**2. What is Encapsulation?**

Encapsulation : The Internal representation of an object is hidden from the view outside object’s definition. Only the required information can be accessed whereas the rest of the data implementation is hidden.

**3. What is Abstraction?**

Abstraction : It is a process of identifying the critical behavior and data of an object and eliminating the irrelevant details.

**4. Which are Access Specifiers?**

Access modifiers in C# are used to specify the scope of accessibility of a member of a class or type of the class itself. For example, a public class is accessible to everyone without any restrictions, while an internal class may be accessible to the assembly only.

**5. What is Inheritance?**

Inheritance : It is the ability to create new classes from another class. It is done by accessing, modifying and extending the behavior of objects in the parent class.

**6. How can you implement multiple inheritance in C#?**

Since, C# does not support multiple class inheritance. To overcome this problem we use interfaces to achieve multiple class inheritance. With the help of the [interface](https://www.geeksforgeeks.org/c-sharp-interface/), inheriting class can get the features of any class’s.

**7. Are private class members inherited to the derived class?**

The derived class doesn't "inherit" the private members of the base class in any way - it can't access them, so it doesn't "inherit" them.

**8. What is Polymorphism?**

Polymorphism : The name means, one name, many forms. It is achieved by having multiple methods with the same name but different implementations.

**9. What is method Overloading?**

Having two or more methods with same name but different in parameters, is known as method overloading in C#.

The advantage of method overloading is that it increases the readability of the program because you don't need to use different names for same action.

You can perform method overloading in C# by two ways:

By changing number of arguments

By changing data type of the arguments

**10.When and why to use method Overloading?**

When user need to implement function by defining two or more functions in a class sharing the same name. And if we need to do the same kind of the operation in different ways for different inputs we use Method Overloading.

**11.What is method Overriding?**

if derived class defines same method as defined in its base class, it is known as method overriding in C#. It is used to achieve runtime polymorphism. It enables you to provide specific implementation of the method which is already provided by its base class.

To perform method overriding in C#, you need to use virtual keyword with base class method and override keyword with derived class method

**12.What is Constructor?**

In C#, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C# has the same name as class or struct.

There can be two types of constructors in C#.

Default constructor

Parameterized constructor

A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

**13.Describe some of the key points regarding the Constructor.**

* A class can have any number of constructors.
* A constructor doesn't have any return type, not even void.
* A static constructor can not be a parametrized constructor.
* Within a class, you can create one static constructor only.

**14.What is Private Constructor?**

Is a special instance constructor, and it is useful in classes that contain only static members. If a class contains one or more private constructors and no public constructors, then the other classes are not allowed to create an instance for that particular class except nested classes.

**15.Can you create object of class with private constructor in C#?**

No, object of a class having private constructor cannot be instantiated from outside of the class.

**16.What is the use of private constructor in C#?**

It is used to stop object creation of a class.

It is used to stop a class to be inherited.

It is used in singleton design patterns, to make sure that the only one instance of a class can ever be created.

**17.What is the use of static constructor in C#?**

It is used to initialize any static data, or to perform a particular action that needs to be performed only once. It is called automatically before the first instance is created or any static members are referenced.

**18.What is Destructor?**

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.

**19.What is Namespaces?**

They are used to organize large code projects. “System” is the most widely used namespace in C#. We can create our own namespace and use one namespace in another, which are called Nested Namespaces.

They are denoted by the keyword “namespace”.

**20.What are Virtual, Override, and New keywords in C#?**

-The virtual keyword is used to modify a method, property, indexer, or event declared in the base class and allow it to be overridden in the derived class.

-The override keyword is used to extend or modify a virtual/abstract method, property, indexer, or event of base class into derived class.

-The new keyword is used to hide a method, property, indexer, or event of base class into derived class.

**21.What is the difference between Struct and Class in C#?**

Basically, a class combines the fields and methods(member function which defines actions) into a single unit. A structure is a collection of variables of different data types under a single unit. It is almost similar to a class because both are user-defined data types and both hold a bunch of different data types.

**22.What is Interface?**

An Interface is a class with no implementation. The only thing that it contains is the declaration of methods, properties, and events.

**23.Why to use Interfaces in C#?**

It is used to achieve multiple inheritance which can't be achieved by class. It is used to achieve fully abstraction because it cannot have method body.

**24.What is Implicit interface implementation?**

Implicit implementations don't include the name of the interface being implemented before the member name, so the compiler infers this. The members will be exposed as public and will be accessible when the object is cast as the concrete type.

**25.What is Explicit interface implementation?**

An explicit interface implementation is a class member that is only called through the specified interface. Name the class member by prefixing it with the name of the interface and a period.

**26.What is Abstract class?**

Abstract classes are the way to achieve abstraction in C#. Abstraction in C# is the process to hide the internal details and showing functionality only. Abstraction can be achieved by two ways Abstract class and interface, both can have abstract methods which are necessary for abstraction.

**27.Describe Abstract class in detail.**

A method which is declared abstract and has no body is called abstract method. It can be declared inside the abstract class only. Its implementation must be provided by derived classes. An abstract method in C# is internally a virtual method so it can be overridden by the derived class.

Abstract class is a class which is declared abstract. It can have abstract and non-abstract methods. It cannot be instantiated. Its implementation must be provided by derived classes. Here, derived class is forced to provide the implementation of all the abstract methods.

**28.What is the difference between Abstraction and Encapsulation?**

-Abstraction allows us to represent complex real world in simplest manner. It is process of identifying the relevant qualities and behaviors an object should possess, in other word represent the necessary feature without representing the back ground details. Abstraction is a process of hiding work style of an object and showing only those information which are required to understand the object. Abstraction means putting all the variables and methods in a class which are necessary.

- Encapsulation is a process of hiding all the internal details of an object from the outside real world. The word Encapsulation, like Enclosing into the capsule. It restrict client from seeing its internal view where behavior of the abstraction is implemented. In Encapsulation, generally to hide data making it private and expose public property to access those data from outer world. Encapsulation is a method for protecting data from unwanted access or alteration. Encapsulation is the mechanism by which Abstraction is implemented.

**29.Can Abstract class be Sealed in C#?**

The abstract method or class cannot be declared as sealed. A subclass of an abstract class can only be instantiated if it implements all of the abstract methods of its superclass. Such classes are called concrete classes to differentiate them from abstract classes.

**30.Can abstract class have Constructors in C#?**

Yes, an abstract class can have a constructor. In general, a class constructor is used to initialize fields. Along the same lines, an abstract class constructor is used to initialize fields of the abstract class.

**31.Can you declare abstract methods as private in C#?**

If a method of a class is private, you cannot access it outside the current class, not even from the child classes of it. But, incase of an abstract method, you cannot use it from the same class, you need to override it from subclass and use. Therefore, the abstract method cannot be private.

**32.Can abstract class have static methods in C#?**

Yes, abstract class can have Static Methods. The reason for this is Static methods do not work on the instance of the class, they are directly associated with the class itself.

**33.Does Abstract class support multiple Inheritance?**

An abstract class cannot be inherited by structures. It can contains constructors or destructors. It can implement functions with non-Abstract methods. It cannot support multiple inheritance.

**34.Abstract class must have only abstract methods. Is it true or false?**

False, An abstract class is a class that is declared abstract —it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. However, if it does not, then the subclass must also be declared abstract

**35.When do you use Abstract Class?**

Generally, we use abstract class at the time of inheritance. A user must use the override keyword before the method which is declared as abstract in child class, the abstract class is used to inherit in the child class. An abstract class cannot be inherited by structures. It can contains constructors or destructors.

**36.Why can Abstract class not be Instantiated?**

The sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited. A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors.

**37.Which type of members can you define in an Abstract class?**

The abstract keyword enables you to create classes and class members that are incomplete and must be implemented in a derived class.

The [sealed](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/sealed) keyword enables you to prevent the inheritance of a class or certain class members that were previously marked [virtual](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual).

**38.What is Operator Overloading?**

Operator overloading gives the ability to use the same operator to do various operations. It provides additional capabilities to C# operators when they are applied to user-defined data types.

**39.Is it possible to restrict object creation in C#?**

Yes, We can limit the number of object creation of class in C# using the static variable. Static variable is used to share the value to all instance of that class.

**40.Can you inherit Enum in C#?**

Nope. it is not possible. Enum can not inherit in derived class because by default Enum is sealed.

**41.Is it possible to achieve Method extension using Interface?**

Yes, You can use extension methods to extend a class or interface, but not to override them. An extension method with the same name and signature as an interface or class method will never be called. At compile time, extension methods always have lower priority than instance methods defined in the type itself.

**42.Is it possible that a Method can return multiple values at a time?**

No, you can't return multiple values from a function in C# (for versions lower than C# 7), at least not in the way you can do it in Python. However, there are a couple alternatives: You can return an array of type object with the multiple values you want in it.

**43.What is Constant?**

Constants are immutable values which are known at compile time and do not change for the life of the program. Constants are declared with the const modifier. Only the C# [built-in types](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types) (excluding [System.Object](https://docs.microsoft.com/en-us/dotnet/api/system.object)) may be declared as const. User-defined types, including classes, structs, and arrays, cannot be const. Use the [readonly](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/readonly) modifier to create a class, struct, or array that is initialized one time at run time (for example in a constructor) and thereafter cannot be changed.

**44.What is Readonly?**

The readonly keyword is a modifier that can be used in four contexts: In a field declaration, In a readonly struct type definition, In an instance member declaration within a structure type, and In a [ref readonly](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/readonly#ref-readonly-return-example) method return.

**45.What is Static?**

Static means something which cannot be instantiated. You cannot create an object of a static class and cannot access static members using an object. C# classes, variables, methods, properties, operators, events, and constructors can be defined as static using the static modifier keyword.

**46.What is Static ReadOnly?**

A Static Readonly type variable's value can be assigned at runtime or assigned at compile time and changed at runtime. But this variable's value can only be changed in the static constructor. And cannot be changed further. It can change only once at runtime.