1.What is an Object?

Object, in C#, is an instance of a class that is created dynamically. Object is also a keyword that is an alias for the predefined type System. ... The unified type system of C# allows objects to be defined. These can be user-defined, reference or value type, but they all inherit directly or indirectly from System

2.What is Encapsulation?

In c#, Encapsulation is a process of binding the data members and member functions into a single unit. In c#, the class is the real-time example for encapsulation because it will combine various types of data members and member functions into a single unit.

3.What is Abstraction?

Abstraction in C# is the process to hide the internal details and showing only the functionality. The abstract modifier indicates the incomplete implementation. The keyword abstract is used before the class or method to declare the class or method as abstract.

4Which are Access Specifiers?

private, public, internal, protected and protected Internal.

5.What is Inheritance?

inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which is defined in other class.

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

we use interfaces to achieve multiple class inheritance.

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

When talking about values of private members: Of course they are inherited. A derived class always also is of the type of the base class. If the base class holds a private value to store some data, the derived class will do that, too - only that you can't access that value from the derived class.

8.What is Polymorphism?

Polymorphism means providing an ability to take more than one form, and it's one of the main pillar concepts of object-oriented programming after encapsulation and inheritance. Generally, polymorphism is a combination of two words, poly, and another one is morphs.

9 .What is method Overloading?

Overloading happens when you have two methods with the same name but different signatures (or arguments). In a class we can implement two or more methods with the same name. Overloaded methods are differentiated based on the number and type of parameter passed as arguments to the methods.

10.When and why to use method Overloading?

We Use it when you actually do need multiple methods with different parameters,

11.What is method Overriding?

Method Overriding in C# is similar to the virtual function in C++. Method Overriding is a technique that allows the invoking of functions from another class (base class) in the derived class. Creating a method in the derived class with the same signature as a method in the base class is called as method overriding.

12. What is Constructor?

It 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.

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?

A private constructor is **a special instance constructor**. It is generally used in classes that contain static members only. If a class has one or more private constructors and no public constructors, other classes (except nested classes) cannot create instances of this class.

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#?

A static constructor 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?

Destructors in C# are **methods inside the class used to destroy instances of that class when they are no longer needed**. The Destructor is called implicitly by the . NET Framework's Garbage collector and therefore programmer has no control as when to invoke the destructor.

19.What is Namespaces?

Namespaces are used in **C# to organize and provide a level of separation of codes**. They can be considered as a container which consists of other namespaces, classes, etc. A namespace can have following types as its members: Namespaces (Nested Namespace).

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

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#?

Structs are value types while classes are reference types. Structs can be instantiated without using a new operator. A **struct cannot inherit from another struct** or class, and it cannot be the base of a class. All structs inherit directly from System.

22.What is Interface?

Interface in C# is **a blueprint of a class**. It is like abstract class because all the methods which are declared inside the interface are abstract methods. It cannot have method body and cannot be instantiated. It is used to achieve multiple inheritance which can't be achieved by class.

23.Why to use Interfaces in C#?

Why And When To Use Interfaces? 1) **To achieve security - hide certain details and only show the important details of an object (interface)**. 2) C# does not support "multiple inheritance" (a class can only inherit from one base class).

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?

With C#, interfaces can be implemented implicitly or explicitly. ... With explicit implementations, in the class **the interface members are not declared as public members and cannot be directly accessed using an instance of the class**, but a cast to the interface allows accessing the members.

26.What is Abstract class?

An abstract class is **a special type of class that cannot be instantiated**. An abstract class is designed to be inherited by subclasses that either implement or override its methods. In other words, abstract classes are either partially implemented or not implemented at all.

27.Describe Abstract class in detail

Abstract class: is **a restricted class that cannot be used to create objects** (to access it, it must be inherited from another class). Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class (inherited from).

28.What is the difference between Abstraction and Encapsulation?

Encapsulation is the mechanism by which the abstraction is implemented.  
...  
Difference between Abstraction and Encapsulation.

Abstraction is set focus on the object instead of how it does it.

Encapsulation means hiding the internal details or mechanics of how an object does something.

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#?

An abstract **method cannot be private** as in the following, abstract class Demo() { private abstract void Call();

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

Static methods cannot be inherited or overridden, and that is why **they can't be abstract**. Since static methods are defined on the type, not the instance, of a class, they must be called explicitly on that type. So when you want to call a method on a child class, you need to use its name to call it

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?

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?

An abstract class cannot be instantiated **because it may contain members that are abstract and have no implementation**.

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

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. ... The function of the operator is declared by using the operator keyword.

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

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. Note : In the above sample, we have created the static variable count, which will hold the incremented count value while creating the 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

42.Is it possible to achieve Method extension using Interface?

**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.

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 (excluding System. ... In this example, the constant Months is always 12, and it cannot be changed even by the class itself.

44.What is Readonly?

The readonly keyword is a modifier that can be used in four contexts: In a field declaration, readonly indicates **that assignment to the field can only occur as part of the declaration** or in a constructor in the same class. ... A readonly field can't be assigned after the constructor exits.

45.What is Static?

In C#, 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.

44.

29.