Article Overview

Background

Here is a list of the most popular OOPS interview questions and answers explained. These OOPS interview questions are for both beginners and professional C# developers.

Questions

1. What is Object?
2. What is Encapsulation?
3. What is Abstraction?
4. Which are Access Specifiers?
5. What is Inheritance?
6. How can you implement multiple inheritance in C#?
7. Are private class members inherited to the derived class?
8. What is Polymorphism?
9. What is method Overloading?
10. When and why to use method Overloading?
11. What is method Overriding?
12. What is Constructor?
13. Describe some of the key points regarding the Constructor.
14. What is Private Constructor?
15. Can you create object of class with private constructor in C#?
16. What is the use of private constructor in C#?
17. What is the use of static constructor in C#?
18. What is Destructor?
19. What is Namespaces?
20. What are Virtual, Override, and New keywords in C#?
21. What is the difference between Struct and Class in C#?

24. What is Implicit interface implementation?
25. What is Explicit interface implementation?

28. What is the difference between Abstraction and Encapsulation?
29. Can Abstract class be Sealed in C#?
30. Can abstract class have Constructors in C#?
31. Can you declare abstract methods as private in C#?
32. Can abstract class have static methods in C#?
33. Does Abstract class support multiple Inheritance?
34. Abstract class must have only abstract methods. Is it true or false?
35. When do you use Abstract Class?
36. Why can Abstract class not be Instantiated?
37. Which type of members can you define in an Abstract class?
38. What is Operator Overloading?
39. Is it possible to restrict object creation in C#?
40. Can you inherit Enum in C#?
41. Is it possible to achieve Method extension using Interface?
42. Is it possible that a Method can return multiple values at a time?
43. What is Constant?
44. What is Readonly?
45. What is Static?
46. What is Static ReadOnly?
47. Can “this” be used within a Static Method?
48. What is Design Patterns in .Net?
49. Which are the Types of Design Pattern?
50. Which ate the key Benefits of using Design Patterns?
51. What is the difference between Static class and Singleton instance?
52. Can you serialize Hashtable?
53. Why is Singleton pattern considered an Anti-pattern?
54. What is Encapsulation and data hiding in C#?
55. What is the use of Yield keyword in C#?
56. How to catch multiple exceptions at once in C#?
57. What is use of the IDisposable interface in C#?
58. What is Property in C#.net?
59. What are accessors?
60. What is Partial Class?
61. What is Sealed Class?
62. What is Sealed Methods and Properties?
63. How to call base class constructor from derived class in C#?
64. What is base keyword?

67. What is Constructor Chaining in C#?
68. What is the Difference between this and base in C#?

- A class or struct definition is like a blueprint that specifies what the type can do.
- An object is basically a block of memory that has been allocated and configured according to the blueprint.
- A program may create many objects of the same class.
- Objects are also called instances, and they can be stored in either a named variable or in an array or collection.

2. What is Encapsulation?

Encapsulation is a process of binding the data members and member functions into a single unit.

3. What is Abstraction?

Abstraction is a process of hiding the implementation details and displaying the essential features.

4. What are Access Specifiers?

There are 5 access specifiers,

- Public can be accessible outside the class through object reference.
- Private can be accessible inside the class only through member functions.
- Protected can be just like private but accessible in derived classes also through member functions.
- Internal can be visible inside the assembly. Accessible through objects.
- Protected Internal can be visible inside the assembly through objects and in derived classes outside the assembly through member functions.

5. What is Inheritance?

Inheritance is a process of deriving the new class from an already existing class.

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

Using Interfaces, you can implement multiple inheritance in C#.

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

- Trying to access a private base class member in the derived cl compile time error.

8. What is Polymorphism?

When a message can be processed in different ways it is called polymorphism.

- Polymorphism means many forms.
- There are 2 types of polymorphism:
  - Compile time polymorphism also know as Overloading
  - Run time polymorphism also know as Overriding

9. What is method Overloading?

Creating multiple methods in a class with the same name but with different parameters and different types is called method overloading.

10. When and why to use method Overloading?

- You have to use method overloading in situations where you want a class to be able to do something, but there is more than one possibility for what information is supplied to the method that carries out the task.
- You should consider overloading a method when you need a couple of methods that take different parameters, but conceptually do the same thing.

11. What is method Overriding?

- Overriding means to change the functionality of a method without changing the signature.
- You can override a method in base class by creating a similar method in derived class.
- This can be done by using virtual/override keywords.
- Base class method has to be marked with virtual keyword and you can override it in derived class using override keyword.
- Derived class method will completely overrides base class method i.e. when you refer base class object created by casting derived class object a method in derived class will be called.

12. What is Constructor?

- Constructor is a special method of the class that will be automatically invoked when an instance of the class is created.
- The main use of constructors is to initialize private fields of the class while creating an instance for the class.
- When you have not created a constructor in the class, the compiler will automatically create a default constructor in the class.
- The default constructor initializes all numeric fields in the class to zero and all string and object fields to null.

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

- A static constructor can not be a parameterized constructor.
- Within a class you can create only one static constructor.

  - Default Constructor
  - Parameterized Constructor
  - Copy Constructor
  - Static Constructor
  - Private Constructor

14. What is Private Constructor?

- Private constructor is a special instance constructor which is used in a class that contains static member only.
- If a class has one or more private constructor and no public constructor then other classes is not allowed to create instance of this class this mean you can neither create the object of the class nor it can be inherit by other class.
- The main purpose of creating private constructor is to restrict the class from being instantiated when it contains every member as static.

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

No, object of a class having private constructor can not 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 in Singleton class.
- It is used to stop a class to be inherited.

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

- Static constructor is a special constructor that gets called before the first object of the class is created.
- It is used to initialize any static data, or to perform a particular action that needs to be performed once only.
- The time of execution of static constructor is not known. But, it is definitely before the first object creation – may be at the time of loading assembly.

18. What is Destructor?

- A Destructor is automatically invoked when an object is finally destroyed.
- The name of the Destructor is the same as class and prefixed with a tilde (~).
- A Destructor is used to free the dynamic allocated memory and release the resources.

Namespace allow creating a system to organize the code.

- Virtual is used to modify a method, property, indexer, or event declared in the base class and allows it to be overridden in the derived class.
- Override is used to extend or modify a virtual/abstract method, property, indexer, or event of the base class into the derived class.
- New is used to hide a method, property, indexer, or event of the base class into the derived class.

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

Struct and Class both are the user defined data type.

Category	Struct	Class
Type	It is value type	It is reference type
Inherits from	It inherits from System.Value type	It inherits from System.Object type
Used for	Usually used for smaller amounts of data	Usually used for large amounts of data
Inherited to	It can not be inherited to other type	It can be inherited to other class
Abstract	It can not be abstract	It can be abstract type
New keyword	No need to create object by new keyword	Can not use an object of a class with using new keyword
Default constructor	Do not have permission to create any default constructor	You can create a default constructor

22. What is Interface?

- An interface looks like a class, but it has no implementation.
- The only thing it contains is declarations of events, indexers, methods and/or properties.
- The reason interfaces only provide declarations is because they are inherited by structs and classes, which must provide an implementation for each interface member declared.

23. Why to use Interfaces in C#?

- Interfaces are mainly used in c# for,
    - Extensibility
    - Implementation Hiding
    - Accessing object using interfaces
    - Loose coupling

24. What is Implicit interface implementation?

- This is the most regular or obvious way to implement members of an interface.

25. What is Explicit interface implementation?

- This is another way to implement members of an interface.

The constraint with explicit implementation is that an explicitly implemented member can not be accessed using a class instance, but only through an instance of the interface.

26. What is Abstract class?

- An abstract class is a special kind of class that can not be instantiated.
- So, the question is why do you need a class that can not be instantiated?
- An abstract class is only to be sub-classed (inherited from).
- In other words, it only allows other classes to inherit from it but can not be instantiated.
- The key advantage is that it enforces certain hierarchies for all the sub classes.
- In simple words, it is a kind of contract that forces all the sub classes to carry on the same hierarchies or standards.

27. Describe Abstract class in detail.

- You can not create an object of Abstract Class.
- An inheritance between abstract to abstract classes is possible. You do not need to implement abstract methods of the base abstract class into a derived abstract class. You can implement it later in concrete classes.
- An abstract class can never be sealed or static.
- An abstract class can have abstract as well as non abstract methods.
- The abstract keyword can be used with class, methods, properties, indexers and events.
- Abstract members can only be declared inside an abstract class.
- An abstract member can not be static or private.
- An abstract method can not be marked virtual.
- A concrete class can not inherit more than one abstract class, in other words multiple Inheritance is not possible.
- Without an abstract class, you can not implement the Template Method Pattern.

28. What is the difference between Abstraction and Encapsulation?

- Encapsulation is wrapping, it's just hiding properties and methods.
- Encapsulation is used for hiding the code and data in a single unit to protect the data from the outside world.
- Class is the best example of encapsulation.
- Abstraction refers to showing only the necessary details to the intended user.

29. Can Abstract class be Sealed in C#?

- No, an abstract class cannot be a sealed class.

30. Can abstract class have Constructors in

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

No. Abstract methods cannot be private in C#.

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

Yes, Abstract class can have static methods in C#.

33. Does Abstract class support multiple Inheritance?

No, Abstract class does not support multiple Inheritance.

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

False. Abstract class can have both abstract and non abstract methods.

35. When do you use Abstract Class?

When you have a requirement where your base class should provide the default implementation of certain methods whereas other methods should be open to being overridden by child classes that time you have to use abstract classes.

36. Why can Abstract class not be Instantiated?

- Because, it has not fully implemented the class as its abstract methods can not be executed.
- If the compiler allows us to create the object for the abstract class, then you can invoke the abstract method using that object which cannot be executed by CLR at runtime.
- Hence, to restrict the calling of abstract methods, the compiler does not allow you to instantiate an abstract class.

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

You can define all static and non-static members including properties, fields, indexers and also abstract methods.

- Overloaded operators are functions with special names the key- Operator followed by the symbol for the operator being defined.

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

Yes, it is possible to restrict object creation in C# by using following,

- Abstract Class
- Static Class
- Private or Protected Constructor

40. Can you inherit Enum in C#?

- No, you cannot inherit Enum in C#.
- Because, Enums are by default sealed. So, you can not inherit them.

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

- Yes, it is possible to achieve Method extension using Interface.
- Most of the LINQ is built around interface extension methods.
- Interfaces were actually one of the driving forces for the development of extension methods.
- Since they can not implement any of their own functionality and extension methods are the easiest way of associating actual code with interface definitions.

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

- Yes, it is it possible that a Method can return multiple values at a time in C# by using the following:
    - KeyValue pair
    - Ref or Out parameters
    - Struct or Class
    - Tuple

43. What is Constant?

- Constant is known as “const” keyword in C#.
- It is also known as immutable values.
- Which are known at compile time and do not change their values at run time like in any function or constructor for the life of application till the application is running.

44. What is Readonly?

- Readonly is known as “readonly” keyword in C#.
- It is also known immutable values.

- You can assay their value by constructor when we call construct word.

45. What is Static?

- It means static members are common to all the objects and they do not get tied to a specific object.
- Static keyword can be used with classes, fields, methods, properties, operators, events, and constructors
- But, static can not be used with indexers, destructors, or types other than classes.
- Key points about Static keyword,
    - If the static keyword is applied to a class, all the members of the class must be static.
    - Static methods can only access static members of same class.
    - Static properties are used to get or set the value of static fields of a class.
    - Static constructor cannot be parameterized.
    - Access modifiers cannot be applied on static constructor. Because, it is always a public default constructor which is used to initialize static fields of the class.

46. What is Static ReadOnly?

- Static Readonly type variable value can be assigned at runtime or at compile time and can be changed at runtime.
- Such variable's value can only be changed in the static constructor and can not be changed further.
- It can change only once at runtime.

47. Can “this” be used within a Static Method?

- No, “this” cannot be used within a static method.
- Because, keyword 'this' returns a reference to the current instance of the class containing it.
- Static methods (or any static member) do not belong to a particular instance.
- They exist without creating an instance of the class and call with the name of a class not by instance so you can not use this keyword in the body of static methods.

48. What is Design Pattern in .Net?

- Design Patterns in the object oriented world is a reusable solution to common software design problems that occur repeatedly in real-world application development.
- It is a template or description for how to solve problems that can be used in many situations.

49. Which are the Types of Design Pattern?

- Creational Patterns: It mainly deals with creation of Objects and Classes.
- Structural Patterns: It deals with Class and Object Composition.

50. Which are the key Benefits of using Design Patterns?

They are language neutral and so can be applied to any language that supports object orientation.

- They aid communication by the very fact that they are well documented and can be researched if that is not the case.
- They have a proven track record as they are already widely used and thus reduce the technical risk to the project.
- They are highly flexible and can be used in practically any type of application or domain.

51. What is the difference between Static class and Singleton instance?

- In C# a static class can not implement an interface. While a single instance class needs to implement an interface for some business reason or IoC purpose, you can use the Singleton pattern without a static class.
- You can clone the object of Singleton but, you cannot clone the static class object.
- Singleton object stores in heap but, static object stores in stack.
- A singleton can be initialized lazily or asynchronously while a static class is generally initialized when it is first loaded.

52. Can you serialize Hashtable?

- No, you can not serialize Hashtable.
- Because, the .NET Framework does not allow serialization of any object that implements the IDictionary interface.

53. Why is Singleton pattern considered an Anti-pattern?

- Singletons are not easy to handle with unit tests. You cannot control their instantiation and they may retain state across invocations.
- Memory allocated to a Singleton can not be freed.
- In multithreaded environment, access to the singleton object may have to be guarded (e.g. via synchronization).
- Singletons promote tight coupling between classes, so it is hard to test.

54. What is Encapsulation and data hiding in C#?

- Encapsulation is a process of hiding the members from outside of class and implemented using access specifiers.
- Encapsulation is also called as data (information) hiding.
- Encapsulation provides a way to preserve the integrity of state data. Rather than defining public fields, private data fields should be defined.
- Well-encapsulated class should hide its data and the details of how it operates on data from the outside world.

55. What is the use of Yield keyword in C#?

There are two scenarios where "yield" keyword is useful;

- It helps to provide custom iteration without creating temp collections.
- It helps to do state-full iteration.

56. How to catch multiple exceptions at once in C#?

You can catch multiple exceptions using condition statements in C#.

57. What is use of the IDisposable interface in C#?

- The primary use of the IDisposable interface is to clean up unmanaged resources.
- "unmanaged" means things like database connections, sockets, etc.

58. What is Property in C#.net?

- Properties are members that provide a flexible mechanism to read, write or compute the values of private fields.
- In other words by property we can access private fields.
- A property is a return type function/method with one parameter or without a parameter.
- These are always public data members.
- It uses methods to access and assign values to private fields called accessors.

59. What are accessors?

The Get and Set portions or blocks of a property are called accessors.

60. What is Partial Class?

- A partial class is only used to split the definition of a class in two or more classes in a same source code file or more than one source files.
- You can create a class definition in multiple files but it will be compiled as one class at run time and also when you'll create an instance of this class. So, you can access all the methods from all source file with a same object.
- Partial Class can be created in the same namespace and it does not allowed to create a partial class in different namespace.
- You can use "partial" keyword with all the class name which you want to bind together with the same name of class in same namespace.

61. What is Sealed Class?

- A class, which restricts inheritance for security reason is declar lass.
- Sealed class is the last class in the hierarchy.

functionality and here you are restricting it to inherit.
- If you have ever noticed, structs and enum are sealed.

62. What is Sealed Methods and Properties?

- Sealed method is used to define the overriding level of a virtual method.
- Sealed keyword is always used with override keyword.

63. How to call base class constructor from derived class in C#?

Use base keyword for this type of initialization.

64. What is base keyword?

- The base keyword is used to access members of the base class from within a derived class.
- Call a method on the base class that has been overridden by another method.
- Specify which base-class constructor should be called when creating instances of the derived class.
- It is an error to use the base keyword from within a static method.

65. What are the Benefits of Three Tier Architecture?

- Reusability: You can reuse the middle layer with different user interfaces like ASP.NET, windows etc.
    - You can also reuse your DAL with different projects.
- Maintainability: When you change in one layer due to the modular approach it does not have ripple effect on other layers.
    - You have to do less amount of changes in other layer when you change logic of one layer.

66. What is the Difference between Design Patterns and Architectural Patterns?

- Design Patterns
    - They are well known patterns for solving technical problems in a way that has proven it self many times.
    - They are common design structures and practices that make for creating reusable Object-Oriented software.
    - Design pattern examples are Factory Pattern, Singleton, Facade, State, etc.