# Conceptual Questions

1. What is Object?

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

1. What is Encapsulation?

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

1. What is Abstraction?

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

1. What are 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.

1. What is Inheritance?

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

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

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

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

* Yes, the private members are also inherited in the derived class but we will not be able to access them.
* Trying to access a private base class member in the derived class will report a compile time error.

1. What is Polymorphism?

* Polymorphism means many forms and it occurs when we have many classes that are related to each other by inheritance.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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 cannot be a parameterized constructor.
* Within a class, you can create one static constructor only.

1. 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.

1. 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.

1. 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.

1. 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.

1. 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.

1. What is Namespaces?

* Namespace allow creating a system to organize the code.

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

* 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.

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

* + Structs
* Structs are value types, allocated either on the stack or inline in containing types.
* Allocations and de-allocations of value types are in general cheaper than allocations and de-allocations of reference types.
* In structs, each variable contains its own copy of the data (except in the case of the ref and out parameter variables), and an operation on one variable does not affect another variable.
* Classes
* Classes are reference types, allocated on the heap and garbage-collected.
* Assignments of large reference types are cheaper than assignments of large value types.
* In classes, two variables can contain the reference of the same object and any operation on one variable can affect another variable.

1. 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.

1. Why to use Interfaces in C#?

* Extensibility
* Implementation Hiding
* Accessing object using interfaces
* Loose coupling

1. What is Implicit interface implementation?

* This is the most regular or obvious way to implement members of an interface.
* Here you do not specify the interface name of the members and implement implicitly.

1. 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.

1. What is Abstract class?

* Abstract class is a generic class used as a basis for creating specific objects that conform to its protocol, or the set of operations it supports. Abstract classes are not instantiated directly.

1. Describe Abstract class in detail.

* 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.
* A concrete class can not inherit more than one abstract class, in other words multiple Inheritance is not possible.

1. 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.
* Abstraction refers to showing only the necessary details to the intended user.

1. Can Abstract class be Sealed in C#?

* No, an abstract class cannot be a sealed class.
* Because, the sealed modifier prevents a class from being inherited and the abstract modifier requires a class to be inherited.

1. Can abstract class have Constructors in C#?

* Yes, Abstract class can have constructor in C#.

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

* No. Abstract methods cannot be private in C#.

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

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

1. Does Abstract class support multiple Inheritance?

* No, Abstract class does not support multiple Inheritance.

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

* False

1. 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.

1. 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.

1. 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.

1. What is Operator Overloading?

* Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined.
* Similar to any other function, an overloaded operator has a return type and a parameter list.

1. 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

1. Can you inherit Enum in C#?

* No, Because, Enums are by default sealed. So, you can not inherit them.

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

* Yes, it is possible Interfaces were actually one of the driving forces for the development of extension methods.

1. 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#

1. 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.

1. What is Readonly?

* Readonly is known as “readonly” keyword in C#.
* It is also known immutable values.
* They are known at compile and run time and do not change their values at run time like in any function for the life of application till the application is running.

1. What is Static?

* The static keyword is used to specify a static member.
* 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
* 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.

1. What is Static Read Only?

* Static Read only 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.