What is abstract class?

A class with abstract modifier indicate that class is abstract class. An abstract class cannot be instantiated. The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share.

```
using System;

namespace ObjectOrientedProgramming
{
    public abstract class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Position { get; set; }

        public abstract void EmployeeDetails();
    }

    public class ParmanentEmployee:Employee
    {
        public int Salary { get; set; }

        public override void EmployeeDetails()
        {
            Id = 1;
            Name = "Farhan Ahmed";
            Position = "Software Engineer";
            Salary = 80000;

            Console.WriteLine("Employee ID:{0}",Id);
            Console.WriteLine("Employee Name:{0}",Name);
            Console.WriteLine("Position:{0}",Position);
            Console.WriteLine("Salary:{0}",Salary);
        }
    }

    public class ContractEmployee:Employee
    {
        public int HourlyRate { get; set; }

        public override void EmployeeDetails()
        {
            Id = 2;
            Name = "Abdul Jabbar";
            Position = "Web Developer";
            HourlyRate = 500;

            Console.WriteLine("Employee ID:{0}",Id);
            Console.WriteLine("Employee Name:{0}",Name);
            Console.WriteLine("Position:{0}",Position);
            Console.WriteLine("Salary:{0}",HourlyRate);
        }

        public double CalculateSalary(double Hour)
        {
            return Hour * HourlyRate;
        }
    }

    class MyClass
    {
        static void Main()
        {
            //PERMANENT EMPLOYEE
            var PE = new ParmanentEmployee();
            PE.EmployeeDetails();

            Console.WriteLine("----------------------");

            //CONTRACT EMPLOYEE
            var CE = new ContractEmployee();
            CE.EmployeeDetails();
            Console.WriteLine("Monthly Income:"+CE.CalculateSalary(50));
            Console.ReadLine();
        }
    }
}
```

Features of abstract class

1. An abstract class cannot be instantiated.
2. An abstract class may contain abstract methods and accessors.
3. An abstract class cannot be sealed. The **sealed** modifier prevents a class from being inherited and an abstract class requires to be inherited.
4. A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors.

What is abstract methods?

A method with abstract modifier indicate that method is abstract method. Abstract methods have no implementation, so the method definition is followed by a semicolon instead of a normal method block. Derived classes of the abstract class must implement all abstract methods.

```
public abstract class AbstractClass
   {
       public abstract void MyAbstractMethod();
   }
```

Features of abstract method

1. An abstract method is implicitly a virtual method.
2. Abstract method declarations are only permitted in abstract classes.
3. An abstract method declaration provides no actual implementation, there is no method body; the method declaration simply ends with a semicolon and there are no curly braces ({ }) following the signature.
4. The implementation is provided by a method override, which is a member of a non-abstract class.
5. It is an error to use the static or virtual modifiers in an abstract method declaration.

When to use Abstract class in C#?

If various implementations are of the same kind and use common behavior or status then abstract class is better to use.

What is interface?

Interface is a contract. An interface contains only the declaration of the methods, properties, and events, but not the implementation. It is left to the class that implements the interface by providing implementation for all the members of the interface. Interface makes it easy to maintain a program.

```
using System;

namespace Interface_Demo
{
    public interface IEmployeeBonus
    {
        Double CalculateBonus();
    }
}

using System;

namespace Interface_Demo
{
    class Employee : IEmployeeBonus
    {
        public string firstName;
        public string lastName;
        public int salary;
        public decimal CalculateBonus()
        {
           return salary * 0.10M;
        }
    }
}
using System;

namespace Interface_Demo
{
    class Program
    {
        static void Main(string[] args)
        {
            var employee = new Employee();
            employee.firstName = "Farhan";
            employee.lastName = "Ahmed";
            employee.salary = 50000;
            employee.CalculateBonus();

            Console.WriteLine("Employee Name:"+employee.firstName + " " +
employee.lastName);
            Console.WriteLine("Employee Salary:"+employee.salary);
            Console.WriteLine("Employee Bonus:"+employee.CalculateBonus());
            Console.ReadLine();
        }
    }
}
```

Features of interface

1. An interface is like an abstract base class. Any class or struct that implements the interface must implement all its members.
2. An interface can't be instantiated directly. Its members are implemented by any class or struct that implements the interface.
3. Interfaces can contain events, indexers, methods, and properties.
4. Interfaces contain no implementation of methods.
5. A class or struct can implement multiple interfaces. A class can inherit a base class and also implement one or more interfaces.

Advantages of interface

1. Interfaces facilitate parallel application development.
2. They are great for implementing Inversion of Control or Dependency Injection.
3. Interfaces enable mocking for better unit testing.
4. Interfaces allow us to develop very loosely coupled systems.
5. Interfaces also allow us to implement polymorphic behavior.

If a class inherits an interface, what are the 2 options available for that class?

Option 1: Provide Implementation for all the members inherited from the interface.
Option 2: If the class does not wish to provide Implementation for all the members inherited from the interface, then the class has to be marked as abstract.

When to use interface?

If various methods share only methods signature then it better to use interface. Interface allow multiple inheritance.

Can a class implement interfaces with same method name?

```
using System;

namespace InterviewQuestions
{
    public interface IOne
    {
        void InterfaceMethod();
    }

    public interface ITwo
    {
        void InterfaceMethod();
    }

    public class A : IOne, ITwo
    {
        void IOne.InterfaceMethod()
        {
            Console.WriteLine("I am interface one method");
        }

        void ITwo.InterfaceMethod()
        {
        Console.WriteLine("I am interface two method");
        }
    }

    class Interface_Demo
    {
        static void Main()
        {
            IOne iOne = new A();
            iOne.InterfaceMethod();

            ITwo iTwo = new A();
            iTwo.InterfaceMethod();
        }
    }
}
```

What are the difference between abstract class and interface?

Abstract Class	Interface
Abstract classes can have implementations for some of its members.	interface can't have implementation for any of its members
An abstract class can contain fields.	Interfaces cannot contain fields
An abstract class can inherit from another class or another interface.	An interface can inherit from another interface only and cannot inherit from any class.
A class cannot inherit from multiple classes at the same time.	A class and Struct can inherit from more than one interface at the same time
Abstract class members can have access modifiers	Interface members cannot have access modifiers. It is by default public.
A class that inherit from abstract class must provide implementations for all inherited abstract members or class don't with implements.	Just like classes interfaces also contain properties, methods, delegates or events, but only declaration no implementations.
We cannot create an instance of abstract class, but abstract class reference variable can point to derived class object.	A class that inherit an interface must provide implementation of all interface otherwise we will get compiler error.
	We cannot create an instance of interface, but an interface reference variable can point to a derived class object