Abstract Class and Abstract Methods in C#

What is an Abstract Class in C#?

A class that is declared by using the keyword **abstract** is called an abstract class. An abstract class is a partially implemented class used for implementing some of the operations of an object which are common for all next-level subclasses and the remaining abstract methods to be implemented by the next-level subclasses. So it contains both abstract methods and concrete methods including variables, properties, and indexers.

It is always created as a superclass next to the interface in the object inheritance hierarchy for implementing common operations from an interface. An abstract class may or may not have abstract methods. But if a class contains an abstract method then it must be declared as abstract.

What is the abstract method in C#?

A method that does not have a body is called an abstract method. It is declared with the modifier abstract. It contains only a **declaration/signature** and does not contain the implementation/body/definition of the method. An abstract function should be terminated with a semicolon. Overriding an abstract function is compulsory.

Points to Remember while working with an abstract class in C#

1.An abstract class can contain both abstract methods and non-abstract (concrete) methods.
2.It can contain both static and instance variables.
3.The abstract class cannot be instantiated but its reference can be created.
4.If any class contains abstract methods then it must be declared by using the keyword abstract.
5.An abstract class can contain sealed methods but an abstract method or class cannot be declared as sealed.
6.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.

When should a class be declared as abstract?

A class should be declared as abstract

1.If the class has any abstract methods

2.If it does not provide implementation to any of the abstract methods it inherited

3.If it does not provide implementation to any of the methods of an interface

Rules Of Abstract Method and Abstract Class in C#:

Rule1: If a method does not have the body, then it should be declared as abstract using the abstract modifier else it leads to a compile-time error: "**must declare a body because it is not marked abstract, extern, or partial**"

Rule2: If a class has an abstract method it should be declared as abstract by using the keyword abstract else it leads to a compile-time error: '**Example.m1()' is abstract but it is contained in non-abstract class 'Example**'.

Rule3: If a class is declared as abstract it cannot be instantiated violation leads to compile-time Error.

Rule4: The sub-classes of an abstract class should override all the abstract methods or it should be declared as abstract else it leads to the compile-time error:

What is the need for abstract classes in application development?

The concepts of abstract methods and abstract classes are an extension to the inheritance wherein inheritance we have been discussing that with the help of a parent class we can provide property to the child class that can be consumed by the child classes which gives us re-usability.

Along with the parent providing property to the children, the parent can also impose the restriction on the children with the help of abstract methods so that all the child classes have to full fill the restriction without failing.

```
public abstract class MyClass
{
public abstract void calculate(double x);
}
class Sub1 : MyClass
{
public override void calculate(double x)
{
Console.WriteLine("SQUARE ROOT IS " + Math.Sqrt(x));
}
}
public class Sub2 : MyClass
{
public override void calculate(double x)
{
Console.WriteLine("SQUARE is :" + (x * x));
}
}
public class Sub3 : MyClass
{
public override void calculate(double x)
{
Console.WriteLine("CUBE is :" + (x * x * x));
}
}
```

```
class Test
{
static void Main(string[] args)
{
Sub1 obj1 = new Sub1();
Sub2 obj2 = new Sub2();
Sub3 obj3 = new Sub3();
obj1.calculate(9);
obj2.calculate(9);
obj3.calculate(9);
Console.ReadKey();
}
}
```

Differences between the interface and abstract class

Abstract class	Interface
It is a partially implemented class. It allows us to define both concrete and abstract methods.	It is a fully un-implemented class. It allows us to define only abstract methods.
It provides both reusability and forcibility	It provides the only forcibility
It should be declared as abstract by using the abstract keyword, abstract methods should also contain the abstract keyword.	It should be created by using the keyword interface. Declaring its methods as abstract is optional because by default the methods of an interface are abstract. The compiler places abstract keywords at the time of program compilation.
A class that contains one or more abstract functions is called abstract class.	The class which contains all the abstract functions is known as an interface.
Its member's default accessibility modifier is private and can be changed to any of the other accessibility modifiers.	Its member's default accessibility modifier is public and cannot be changed.
It is possible to declare data fields in an abstract class.	But it is not possible to declare any data fields in an interface.
An abstract class can contain the non-abstract function.	An interface cannot contain non-abstract functions.
An abstract class can inherit from another abstract class or from an interface.	An interface can inherit from only other interfaces but cannot inherits from the abstract class.
It can have inner classes	It can also have inner classes.
An abstract class cannot be used to implement multiple inheritances.	An interface can be used to implement multiple inheritances.
Abstract class members can have access modifiers.	Interface members cannot have access modifiers.