**Inheritance**

Inheritance is the process by which object of one class can access the other class property.

Inheritance is one of the fundamental attributes of object-oriented programming. It allows you to define a child class that reuses (inherits), extends, or modifies the behavior of a parent class. The class whose members are inherited is called the base class. The class that inherits the members of the base class is called the derived class.

**Parent🡪 Child  
Base🡪 Derived**

C# and .NET support single inheritance only. That is, a class can only inherit from a single class. However, inheritance is transitive, which allows you to define an inheritance hierarchy for a set of types. In other words, type D can inherit from type C, which inherits from type B, which inherits from the base class type A. Because inheritance is transitive, the members of type A are available to type D.

Not all members of a base class are inherited by derived classes. The following members are not inherited:

* [Static constructors](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/static-constructors), which initialize the static data of a class.
* [Instance constructors](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors), which you call to create a new instance of the class. Each class must define its own constructors.
* [Finalizers](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/destructors), which are called by the runtime's garbage collector to destroy instances of a class.

While all other members of a base class are inherited by derived classes, whether they are visible or not depends on their accessibility. A member's accessibility affects its visibility for derived classes as follows:+

* [Private](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/private) members are visible only in derived classes that are nested in their base class. Otherwise, they are not visible in derived classes.
* [Protected](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected) members are visible only in derived classes.
* [Internal](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected) members are visible only in derived classes that are located in the same assembly as the base class. They are not visible in derived classes located in a different assembly from the base class.
* [Public](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/protected) members are visible in derived classes and are part of the derived class' public interface. Public inherited members can be called just as if they were defined in the derived class. In the following example, class A defines a method named Method1, and class B inherits from class A. The example then calls Method1 as if it were an instance method on B.

Derived classes can also override inherited members by providing an alternate implementation. In order to be able to override a member, the member in the base class must be marked with the [virtual](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/virtual) keyword. By default, base class members are not marked as virtual and cannot be overridden. Attempting to override a non-virtual member, as the following example does, generates compiler error CS0506: " cannot override inherited member because it is not marked virtual, abstract, or override.

## **Multiple Inheritance in C#**

**C# does not support multiple inheritance**. However, you can use interfaces to implement multiple inheritance.

Ref:  
<https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/inheritance>