**INTERFACES IN C#**

An interface is a contract between itself and any class that implements it. This contract states that any class that implements the interface will implement the interface's properties, methods and/or events. An interface contains no implementation, only the signatures of the functionality the interface provides. An interface can contain signatures of methods, properties, indexers & events.

The interface defines the **'what'** part of the syntactical contract and the deriving classes define the **'how'** part of the syntactical contract.

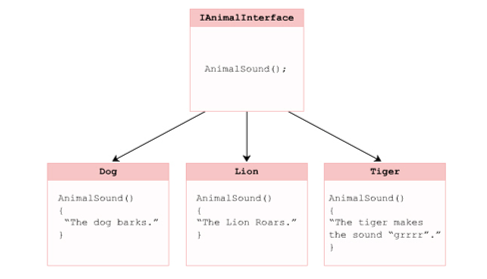
Interfaces define properties, methods, and events, which are the members of the interface. Interfaces contain only the declaration of the members. It is the responsibility of the deriving class to define the members.

One of the main reason to introduce interfaces is that it can be used in multiple inheritance.

* An interface contains only abstract members, just like classes interface also contains properties, methods, delegates or events, but only declarations, no implementations.
* An interface cannot be instantiated but can only be inherited by classes or other interfaces.
* Interface cannot have fields.
* An interface is declared using the keyword **interface**.
* In C#, by default, all members declared in an interface have public as the access modifier. They don’t allow explicit access modifiers.

**IMPLEMENTING AN INTERFACE**

* An interface is implemented by a class in a way similar to inheriting a class.
* When implementing an interface in a class, implement all the abstract methods declared in the interface. If all the methods are not implemented, the class cannot be compiled.
* The methods implemented in the class should be declared with the same name and signature as defined in the interface.



**INTERFACE INHERITANCE IN C#**

* Interface can inherit from other interfaces.
* A class that inherits this interface must provide implementation for all interface members in the entire interface inheritance chain.
* Interface reference variable can have the reference of their child class.