**Herencia**

La herencia en la programación funciona del mismo modo que en la vida real, un padre puede dejar en herencia ciertas características y atributos a sus hijos.

Aprendamos a utilizar la herencia en nuestro código y veamos como una clase hijo puede heredar ciertas características o funciones de la clase padre, también veremos que estas funciones o métodos heredados se pueden sobrescribir.

Al momento de programar nuestras clases padre podemos usar la palabra clave **abstract** para que dicha clase solamente pueda ser heredada, pero nunca instanciada. Por el contrario, tenemos la palabra clave **sealed** permite generar instancias de la clase, pero no permite heredarla.

**Abstracto**. Ejemplo que esta clase es solo una idea

* Una clase abstracta (asbtract) puede ser heredada pero no instanciada
* Una **clase sellada (sealed) no puede ser heredada pero si instaciada**

**Polimorfismo**

En polimorfismo un objeto hijo que hereda de una clase padre puede ser tratado como un objeto padre, pero al ser convertido en objeto padre ya no se podrá acceder a los atributos del objeto hijo. Por otro lado, un objeto padre no puede tratarse como un objeto hijo a menos que el objeto padre estuviera guardando un objeto hijo.

**Lista de Objetos polimórfica**

En C# podemos realizar validaciones sobre el tipo de objeto que estamos manejando, usando las siguientes palabras clave:

* **is**, para verificar si un objeto es de un tipo en específico.
* **as**, para tratar un objeto como un tipo específico, en caso de no poder convertir el objeto entonces va a asignar un valor null.

**Implementación de interfaces**

En C# no existe la herencia múltiple, sin embargo, esto no es una limitante para hacer que un objeto se vea como otros objetos diferentes ya que podemos recurrir al uso de interfaces.  
Una interfaz es la definición de la estructura de un objeto. Las interfaces las podemos tomar como contratos en las que obligamos a que los objetos cumplan determinadas características.