### Herencia

La herencia permite crear una nueva clase que reutiliza, extiende y modifica el comportamiento que se define en otra clase.

La clase cuyos miembros se heredan se denomina **clase base** y la clase que hereda esos miembros se denomina **clase derivada**.

Sin embargo, todas las clases de C# heredan implícitamente de la clase **Object** que admite la jerarquía de clases .NET y proporciona servicios de bajo nivel a todas las clases.

C# solo admite solo **herencia simple**. **No admite herencia múltiple**

Para que una clase herede a otra se coloca un signo de dos puntos después del nombre de la clase al declararla.

Sintaxis

<modificador De Visibilidad> **class** <nombre **subclase o hija**> **:** <**superclase o clase Madre**>

{

}

La nueva clase (la clase derivada) obtiene todos los **datos no privados** y el comportamiento de la clase base, además de todos los demás datos y comportamientos que define para sí misma.

La nueva clase tiene dos tipos efectivos: el tipo de la nueva clase y el tipo de la clase que hereda.

Ejemplo

public class Persona

{

//Código

}

public class Alumno : Persona

{

//Código

}

#### This y base

La palabra clave **base** se utiliza para obtener acceso a los miembros de la clase base desde una clase derivada.

El acceso a una clase base sólo se permite en un constructor, en un método de instancia o en un descriptor de acceso a una propiedad de instancia.

La palabra clave **this** hace referencia a la instancia actual de la clase y también se utiliza como modificador del primer parámetro de un método de extensión.

public class ClaseBase

{

public string Metodo()

{

return "Metodo de la base";

}

}

public class ClaseDerivada : ClaseBase

{

public string MetodoDerivada()

{

base.Metodo();

this.Metodo();

this.MetodoDerivada();

return "";

}

}

Clase madre 🡪 Persona atributos (nombre, apellido, dni)

**Clase Hija Estudiante** 🡪 atributos (nombre, apellido, dni, matricula, carrera, materias):base(nombre, apellido, dni)