# Herencia

* Es una de las principales características de la Programación Orientada a Objetos.
* Una nueva clase es creada absorbiendo los miembros de una clase existente.

La herencia permite crear clases que reutilizan, extienden y modifican el comportamiento definido en otras clases.

## Herencia simple

C# solo soporta herencia simple, esto es que cada clase solo puede heredar de una y sólo una clase.

## Jerarquías

La clase cuyos miembros se heredan se denomina *clase base* y la clase que hereda esos miembros se denomina *clase derivada*. Una clase derivada solo puede tener una clase base directa, pero la herencia es transitiva. Si ClassC se deriva de ClassB y ClassB se deriva de ClassA, ClassC hereda los miembros declarados en ClassB y ClassA.

En C#, es posible heredar campos y métodos de una clase a otra. Agrupamos el "concepto de herencia" en dos categorías:

**Clase derivada (hijo):** la clase que hereda de otra clase.

**Clase base (padre):** la clase de la que se hereda.

## Superclase (padre)

* Clase de la que desciende o deriva una clase.
* Las clases hijas (descendientes) heredan (incorporan) automáticamente los atributos, propiedades y métodos de la clase padre.

## Subclase (hijos)

* Clase descendiente de otra.
* Hereda automáticamente los atributos, propiedades y métodos de su superclase.
* Es una especialización de otra clase.
* Admiten la definición de nuevos atributos y métodos para aumentar la especialización de la clase.

Para indicar que una clase hereda de otra, después del nombre ponemos : y el nombre de la clase de la que hereda.

class Empleado : Persona

{

}

**Ejemplo:**

La clase Persona tiene como atributos Nombre y Edad, tiene el método saludar que puede ser sobreescrito, dos constructores, uno vacío y otro donde se inicializan las dos propiedades. La clase Empleado hereda de la clase Persona.

La clase Empleado además de las propiedades heredadas tiene los atributos Puesto y Sueldo, se sobreescribe el método Saludar de la clase Persona, se le añade el método Trabajar, e igualmente tiene un constructor que inicializa todas las propiedades. En ambas clases se sobreescribe el método ToString para reflejar la información de la clase.

**Program.cs**

using System;

namespace Herencia

{

class Program

{

static void Main(string[] args)

{

// instancia de clase persona

Persona p = new Persona("Carlos", 38);

p.Saludar();

// instancia de clase empleado

Empleado e = new Empleado("John", 29, "Ventas", 10000);

e.Saludar();

e.Trabajar();

// se ejecuta el método ToString

Console.WriteLine(e);

Console.ReadKey();

}

}

}

**Persona.cs**

using System;

namespace Herencia

{

class Persona

{

public string Nombre { get; set; }

public int Edad { get; set; }

public Persona() { }

public Persona(string nombre, int edad) {

Nombre = nombre;

Edad = edad;

}

// virtual indica que puede ser modificado por una subclase

public virtual void Saludar()

{

Console.WriteLine("Hola, mi nombre es {0}", Nombre);

}

// virtual se escribe en la superclase

// override se escribe en la subclase

public override string ToString()

{

return string.Format("{0}, {1} años", Nombre, Edad);

}

}

}

**Empleado.cs**

using System;

namespace Herencia

{

class Empleado : Persona

{

public string Puesto { get; set; }

public decimal Sueldo { get; set; }

// base se refiere a un método o a un constructor de la clase padre

// En el proceso de construcción siempre se inicializan primero las propiedades de la superclase, y después las de la subclase

public Empleado(string nombre, int edad, string puesto, decimal sueldo) : base(nombre, edad)

{

Puesto = puesto;

Sueldo = sueldo;

}

// Sobreescribir el metodo Saludar de la clase padre

public override void Saludar()

{

Console.WriteLine("Hola, soy {0} y mi puesto es {1}", Nombre, Puesto);

}

public void Trabajar()

{

Console.WriteLine("{0} está trabajando... ", Nombre);

}

public override string ToString()

{

return string.Format("{0}, {1} años, {2}, sueldo: ${3}", Nombre, Edad, Puesto, Sueldo);

}

}

}

# Bibliografía

<http://csharp-facilito.blogspot.com/2013/07/ejercicios-de-herencia-en-c-sharp.html>

<https://docs.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/inheritance>

<https://www.w3schools.com/cs/cs_inheritance.php>

<https://www.udemy.com/course/c-sharp-para-principiantes/learn/lecture/4618940>