Guía de Estudio: Herencia

1. **¿Qué nombre recibe la clase que hereda y qué nombre recibe la clase que es heredada?**

La clase que hereda recibe el nombre de hija, derivada o subclase y la clase que es heredada recibe el nombre de padre o clase base.

1. **¿Qué significa que la herencia es transitiva?**

Quiere decir que la herencia se transfiere de una clase a sus derivadas. Si C hereda de B y B hereda de A, entonces, C hereda de A.

1. **¿Se heredan los constructores?**

Los constructores no se heredan.

1. **¿Se heredan los miembros private de la clase base?**

Se heredan los miembros private de la clase base pero no son visibles en las clases derivadas.

1. **¿Qué es herencia múltiple? ¿Es posible en C#?**

Es cuando una clase puede heredar de una o más clases base. No es posible en C#.

1. **¿Que nos permite la herencia? ¿Cuál es su propósito?**

Es una relación entre clases en la cual una clase comparte la estructura y el comportamiento definido en otra clase.

Cada clase que hereda de otra posee los atributos de la clase base y soporta todos o algunos de los métodos de la clase base.

El propósito de la herencia es organizar mejor las clases que comparten una determinada realidad y poder agruparlas en función de atributos y comportamientos comunes. A la vez que cada una se especializa según sus particularidades.

1. **¿Una clase pública puede heredar de una clase privada?**

No, una clase derivada no puede ser más accesible que su clase base.

1. **¿Qué es una clase sellada (sealed)?**

Es una clase de la que no se puede heredar.

1. **¿Una clase sellada puede heredar de otras clases? (Ser clase derivada)**

Sí, una clase sellada puede heredar de otras, pero no se puede heredar de ella.

1. **¿Cómo actúa el modificador protected en los miembros de la clase base para una clase derivada y cómo para una clase no-derivada?**

Para una clase derivada, el modificador protected es equivalente a la palabra public.

Para una clase no derivada, los miembros protegidos se comportan como miembros privados para la otra clase

1. **¿Qué pasa si la clase derivada no hace una llamada explícita a un constructor de la clase base? En esta situación, ¿qué pasa si la clase base declaró explícitamente un constructor con parámetros de entrada?**

En ese caso, el compilador de C# usará implícitamente un constructor de la forma : base().

El compilador no creará un constructor por defecto si la clase base tiene su propio constructor explícito. El compilador generará un mensaje de error si el constructor indicado no coincide con ningún constructor de la clase base.