Interfaces

Una **interfaz** contiene definiciones para un grupo de funcionalidades relacionadas que debe implementar una clase o estructura no abstracta .

Una interfaz puede definir `static` métodos, que deben tener una implementación. A partir de C # 8.0, una interfaz puede definir una implementación predeterminada para los miembros.

Una interfaz **no puede** declarar datos de instancia como campos, propiedades implementadas automáticamente o eventos similares a propiedades.

Mediante el uso de interfaces, puede, por ejemplo, incluir el comportamiento de varias fuentes en una clase.

Esa capacidad es importante en **C #** porque el lenguaje **no admite la herencia múltiple** de clases. Además, debe usar una interfaz **si desea simular la herencia de estructuras**, porque en realidad no pueden heredar de otra estructura o clase.

```
interface IEquatable<T>
{
    bool Equals(T obj);
}
```

El nombre de una interfaz debe ser un nombre de identificador de C # válido . **Por convención**, los nombres de **las interfaces comienzan con mayúscula** `I`.

Cualquier clase o estructura que implemente la interfaz IEquatable <T> debe contener una definición para un método Equals que coincida con la firma que especifica la interfaz. Como resultado, puede contar con una clase que se implementa `IEquatable<T>`para contener un `Equals` método con el que una instancia de la clase puede determinar si es igual a otra instancia de la misma clase.

La definición de `IEquatable<T>`no proporciona una implementación para `Equals`.

Una **clase o estructura** puede implementar múltiples interfaces, pero **una clase solo puede heredar de una sola clase**.

Las interfaces pueden contener métodos de instancia, propiedades, eventos, indexadores o cualquier combinación de esos cuatro tipos de miembros.

Las interfaces pueden contener constructores, campos, constantes u operadores estáticos.

Una interfaz no puede contener campos de instancia, constructores de instancia ni finalizadores. Elementos de interfaz son públicos por defecto, y se puede especificar explícitamente modificadores de accesibilidad,
como `public`, `protected`, `internal`, `private`, `protected internal`, o `private protected`.

Un `private` miembro debe tener una implementación predeterminada.

Para implementar un miembro de la interfaz, el miembro correspondiente de la clase de implementación debe ser público, no estático y tener el mismo nombre y firma que el miembro de la interfaz.

Cuando una clase o estructura implementa una interfaz, la clase o estructura debe proporcionar una implementación para todos los miembros que la interfaz declara pero no proporciona una implementación predeterminada. Sin embargo, si una clase base implementa una interfaz, cualquier clase derivada de la clase base hereda esa implementación.

El siguiente ejemplo muestra una implementación de la interfaz IEquatable <T> . La clase de implementación `Car`, debe proporcionar una implementación del método Equals .

```
public class Car : IEquatable<Car>
{
    public string Make {get; set;}
    public string Model { get; set; }
    public string Year { get; set; }

    // Implementation of IEquatable<T> interface
    public bool Equals(Car car)
    {
        return (this.Make, this.Model, this.Year) ==
            (car.Make, car.Model, car.Year);
    }
}
```

Las propiedades y los indexadores de una clase pueden definir descriptores de acceso adicionales para una propiedad o indexador que se define en una interfaz.

Por ejemplo, una interfaz puede declarar una propiedad que tiene un descriptor de acceso get . La clase que implementa la interfaz puede declarar la misma propiedad con un descriptor de acceso `get` y set .

Sin embargo, si la propiedad o el indexador utiliza una implementación explícita, los descriptores de acceso deben coincidir.

Las interfaces pueden heredar de una o más interfaces.

La interfaz derivada hereda los miembros de sus interfaces base.

Una clase que implementa una interfaz derivada debe implementar todos los miembros de la interfaz derivada, incluidos todos los miembros de las interfaces base de la interfaz derivada.

Esa clase se puede convertir implícitamente a la interfaz derivada o cualquiera de sus interfaces base.

Una clase puede incluir una interfaz varias veces a través de clases base que hereda o mediante interfaces que heredan otras interfaces.

Sin embargo, la clase puede proporcionar una implementación de una interfaz solo una vez y solo si la clase declara la interfaz como parte de la definición de la clase (`class ClassName : InterfaceName`).

Si la interfaz se hereda porque heredó una clase base que implementa la interfaz, la clase base proporciona la implementación de los miembros de la interfaz.

Sin embargo, la clase derivada puede volver a implementar cualquier miembro de la interfaz virtual en lugar de utilizar la implementación heredada. Cuando las interfaces declaran una implementación predeterminada de un método, cualquier clase que implemente esa interfaz hereda esa implementación.