INSTITUTO TECNOLOGICO DEL SUR DE NAYARIT

" Programacio Orientada a Objetos "

Maestra: Cinthia Anahi Mata Bravo

Alumno: Juan Diego Flores Nava

INDICE

INTERFACES

En teoría de orientación a objetos, la interfaz de una clase es *todo lo que podemos hacer con ella*. A efectos prácticos: todos los métodos, propiedades y variables públicas (aunque no deberían haber nunca variables públicas, debemos usar propiedades en su lugar) de la clase conforman *su interfaz*.

Cuando se tienen varias clases que tienen que implementar el mismo método, es aconsejable utilizar “interfaces”. Son como las clases (propiedades, métodos y eventos) pero no contienen código.

La codificación se hará en la clase que implementa la interfaz. Una interfaz contiene definiciones para un grupo de funcionalidades relacionadas que una clase no abstracta o una estructura deben implementar. Una interfaz puede definir staticmétodos, que deben tener una implementación. Una interfaz puede proporcionar una implementación predeterminada para cualquiera o todos sus miembros de instancia declarados. 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 múltiples 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.

El nombre de una interfaz debe ser un nombre de identificador de C # válido. Por convención, los nombres de las interfaces comienzan con mayúscula 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 implemente IEquatable<T>para contener un Equalsmétodo 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 estáticos, campos, constantes u operadores. Para enlaces a ejemplos, vea secciones relacionadas. Una interfaz no puede contener campos de instancias, constructores de instancias o finalizadores. Los miembros de la interfaz son públicos por defecto.

PARA IMPLEMENTAR INTERFACES

Para implementar un miembro de 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 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.

Aquí se muestra una implementación de la interfaz IEquatable <T> . La clase implementadora Car, debe proporcionar una implementación del método Equals .

C#

```
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 e indexadores de una clase pueden definir accesores 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 a gety set accessor. Sin embargo, si la propiedad o el indexador usa una implementación explícita, los accesores deben coincidir. Para obtener más información sobre la implementación explícita, vea Implementación explícita de la interfaz y Propiedades de la interfaz . 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 en 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 a través de interfaces que otras interfaces heredan. 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 usted 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. Las implementaciones definidas en las interfaces son virtuales y la clase implementadora puede anular esa implementación. Una clase base también puede implementar miembros de la interfaz mediante el uso de miembros virtuales. En ese caso, una clase derivada puede cambiar el comportamiento de la interfaz anulando los miembros virtuales.

Resumen de interfaces

En un contexto más cotidiano y fácil de entender para mí, la interfaz es como un contrato entre personas, se deben seguir los parámetros ya definidos que pueden ser formas de trabajar o métodos como en el caso de c·.
En programación las interfaces tienen métodos y propiedades que declaran variables y funciones de los métodos establecidos. Las interfaces son clases "Abstractas que solo tienen métodos y propiedades abstractos sin cuerpo, que para acceder a estos métodos la interfaz debe ser heredada por otra clase para lo cual utilizamos ":" entre la clase padre y la heredada para indicar que queremos que se herede de la clase padre. Algunos puntos que son importantes en las interfaces no podemos crear objetos y sus miembros son por defecto abstractos y públicos (una clase al ser abstracta tiene el mismo propósito pero se usa cuando la clase base declara pocos métodos y la heredada implementa las funcionalidades) y algo que deriva de no poder crear objetos es que no podemos tener un constructor ya que no hay objetos, para declarar la interfaz en un proyecto se utiliza la palabra **interfaz** que va a ser publica por defecto

-Una interfaz es típicamente como una clase base abstracta con solo miembros abstractos. Cualquier clase o estructura que implemente la interfaz debe implementar todos sus miembros. Opcionalmente, una interfaz puede definir implementaciones predeterminadas para algunos o todos sus miembros.
-Una interfaz no puede ser instanciada directamente. Sus miembros son implementados por cualquier clase o estructura que implemente la interfaz. Una clase o estructura puede implementar múltiples interfaces. Una clase puede heredar una clase base y también implementar una o más interfaces.