TECNOLOGICO DEL SUR DE NAYARIT

PROGRAMACION ORIENTADA A OBJETOS

CARLOS DANIEL ELIAS SOTO

“INVESTIGACIÓN DE INTERFAZ”

2 SEMESTRE

191140021

ITIC

INDICE

INTERFAZ EN C#

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.

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 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.

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 get y 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. Otra estructura o clase.

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. Para obtener más información sobre miembros virtuales, consulte Polimorfismo.

RESUMEN PERSONAL

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.

Las interfaces pueden heredar de una o más interfaces. 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. Otra estructura o clase.

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. Para obtener más información sobre miembros virtuales, consulte Polimorfismo.

Ejemplo:

Interfaz

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

Clase

```
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);
    }
}
```