Instituto Tecnológico Nacional de México

Instituto Tecnológico del Sur de Nayarit

20-3-2020

Programación Orientada a Objetos
Tema IV "Herencia y Polimorfismo"

Alumno: Meza Rosales Luis Antonio Ismael

No. De control:191140008

Docente: Cinthia Anahí Mata Bravo

Segundo semestre

Conceptos de interfaces c#

Índice

INVESTIGACIÓN

Una interfaz contiene las definiciones de un grupo de funcionalidades relacionadas que una clase o una estructura no abstracta deben implementar. Una interfaz puede definir métodos static, que deben tener una implementación. Una interfaz puede proporcionar una implementación predeterminada para cualquiera de sus miembros de instancia declarados o para todos ellos. Una interfaz no puede declarar datos de instancia, como campos, propiedades implementadas automáticamente o eventos similares a las propiedades.

Inclusiones

Mediante las interfaces puede incluir, por ejemplo, un comportamiento de varios orígenes en una clase. Esta 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 no pueden heredar de otra estructura o clase.

Para definir una interfaz, deberá usar la palabra clave I interface, tal y como se muestra en el ejemplo siguiente.

```
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 interfaz comienzan con una letra I mayúscula.

Implementación de IEquatable

Cualquier clase o estructura que implementa la interfaz IEquatable<T> debe contener una definición para un método Equals que coincida con la firma que la interfaz especifica. Como resultado, puede contar con una clase que implementa IEquatable<T> para contener un método Equals 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 varias interfaces, pero una clase solo puede heredar de una sola clase.

Información que puede contener

Las interfaces pueden contener propiedades, eventos, indizadores o métodos de instancia, o bien cualquier combinación de estos cuatro tipos de miembros. Las interfaces pueden contener constructores estáticos, campos, constantes u operadores. Para obtener vínculos a ejemplos, vea Secciones relacionadas. Una interfaz no puede contener campos de instancia, constructores de instancias ni finalizadores. Los miembros de la interfaz son públicos de forma predeterminada.

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 sin proporcionar ninguna implementación predeterminada para ellos. Sin embargo, si una clase base implementa una interfaz, cualquier clase que se derive de la clase base hereda esta implementación.

En el siguiente ejemplo se 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 indizadores de una clase pueden definir descriptores de acceso adicionales para una propiedad o indizador que estén definidos en una interfaz. Por ejemplo, una interfaz puede declarar una propiedad que tenga 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 indizador usan una implementación explícita, los descriptores de acceso deben coincidir. Para obtener más información sobre la implementación explícita.

Herencia

Las interfaces pueden heredar de una o varias interfaces. La interfaz derivada hereda los miembros de sus interfaces base. Una clase que implementa una interfaz derivada debe implementar todos los miembros de esta, incluidos los de las interfaces base. Esa clase puede convertirse implícitamente en la interfaz derivada o en cualquiera de sus interfaces base. Una clase puede incluir una interfaz varias veces mediante las clases base que hereda o mediante las 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 se 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 usar la implementación heredada. Cuando una interfaz declara una implementación predeterminada de un método, cualquier clase que implemente la interfaz en cuestión hereda esa implementación. Las implementaciones definidas en interfaces son virtuales y es posible que la clase que realice la implementación las invalide.

Polimorfismo

Una clase base también puede implementar miembros de interfaz mediante el uso de los miembros virtuales. En ese caso, una clase derivada puede cambiar el comportamiento de la interfaz reemplazando los miembros virtuales. Para obtener más información sobre los miembros virtuales, vea Polimorfismo.

RESUMEN

Una interface contiene las definiciones de un grupo de funciones específicas que están relacionadas a una clase o una estructura similar no abstracta. Además una interface no puede declarar datos, como campos, propiedades implementadas automáticamente o eventos similares. Mediante las interfaces se puede incluir, por ejemplo, un método. Esta capacidad es importante en C# porque el lenguaje no admite que varias clases tengan herencia múltiple. Una interface se puede usar para simular la herencia de estructuras, porque no pueden heredar de otra estructura o clase. El nombre de una interface debe ser un nombre de identificador como las demás partes de los programas que hemos, en este caso los nombres de interface comienzan con una letra I mayúscula y posteriormente el nombre de la interface (INombreDeLaInterfaz).

Las interfaces pueden contener propiedades, eventos, indizadores o métodos de instancias, así mismo la implementación de todos los miembros antes mencionados simultáneamente. Las interfaces pueden contener constructores estáticos, campos, constantes u operadores. Las interfaces no pueden contener campos de instancia, constructores de instancias ni finalizadores. Los miembros de la interface son públicos de forma predeterminada. Cuando una clase o estructura implementa una interfaz, la clase o estructura debe proporcionar una implementación para todos los miembros que la interface declara sin proporcionar ninguna implementación predeterminada para ellos. Sin embargo, si una clase base implementa una interface, cualquier clase que se derive de la clase base hereda esta implementación.

Las interfaces pueden heredar de una o varias interfaces simultáneamente. La interface derivada hereda los miembros de sus interfaces bases, si una interface contiene algún método y otra interface es hija de esa interface, esa nueva interface también contendrá el método de la interface papá. Una clase que implementa una interfaz derivada debe implementar todos los miembros de esta, incluidos los de las interfaces bases, es decir, la clase que hereda de una interface debe incluir en esta todo lo contenido en la interfaz debe contener aparte de los métodos, variables, entre otro, propios de dicha clase que posteriormente pueden ser heredados a otra clase. 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, es decir determinar con que la clase hereda de la interfaz como se muestra a continuación (class ClassName:InterfaceName) .Una clase base también puede implementar miembros de interfaz mediante el uso de clases abstractas, es decir clases abstractas con su contenido abstracto, por ejemplo:

```
abstrac ClassName1:InterfaceName

{

Public abstract void NombreDelMetodo();

}
```

En ese caso, una clase derivada puede cambiar el comportamiento de la interfaz reemplazando los miembros virtuales, implementando un override en el nuevo método de la nueva clase como se muestra a continuación.

```
ClassName2: ClassName1

{

Public override void NombreDelMetodo()

{

  Y en ese sitio colando las operaciones que va a realizar el método

}

}
```