Interfaces

18

Programación II y Laboratorio de Computación II

Edición 2018

Interfaces

- Es un contrato que establece una clase en el cual esta clase asegura que implementará un conjunto de métodos.
- Son una manera de describir qué debería hacer una clase sin especificar el cómo.
- Es la descripción de uno o más métodos que posteriormente alguna clase puede implementar.

Generalidades

- C# no permite especificar atributos en las interfaces.
- Todos los métodos son públicos (no se permite especificarlo).
- Todos los métodos son como “abstractos” ya que no cuentan con implementación (no se permite especificarlo).
- Se pueden especificar propiedades (sin implementación).
- Las clases pueden implementar varias interfaces.
- Las interfaces pueden “simular” algo parecido a la herencia múltiple.

Definir una interfaz

```
[modificadores] interface INombreInterface
{
    //Miembros de la interface
}
```

- Se utiliza la palabra reservada **interface**.
- Por convención, los nombres de las interfaces comienzan con la letra I seguida del identificador.

Definir una interfaz

```
[modificadores] class NombreClase : INombreInterface
{
    //Miembros de la interface
}
```

- Para que una clase implemente una interface se emplea el operador dos puntos (:).
- Para implementar una interface a una clase derivada, primero hay que indicar la clase base, luego la interface separadas por una coma (,).
- Para sobrescribir los miembros de las interfaces **NO** se emplea la palabra **override**, ya que no fueron declaradas como *virtual* o *abstract* en la interface.

```
[modificadores] class NombreClase : IMiInterface1, IMiInterface2
{ }
```

Implementación explícita

- Los miembros implementados explícitamente sirven para ocultar la implementación de miembros de interfaces a las clases que lo implementan.
- También sirve para evitar la ambigüedad cuando, por ejemplo, una clase implementa dos interfaces las cuales poseen un miembro con la misma firma.
- Las clases derivadas de clases que implementan interfaces de manera explícita no pueden sobrescribir los métodos definidos explícitamente.
- Sintácticamente la implementación de una interfaz de manera explícita e implícita es igual, lo único que cambia es la firma del miembro en la clase que implementa la interfaz.

Interfaz explícita

```
void INombreInterface.NombreMetodo()
{
    Console.WriteLine("Hola");
}
```

- **Se coloca el nombre de la interfaz adelante del nombre del método que se esta implementando explícitamente y además no se le indica la visibilidad.**

Interfaz explícita

```
public interface IMiInterfaz
{
    string MiMetodo();
}

public class PruebaInterfazImplicita : IMiInterfaz
{
    string IMiInterfaz.MiMetodo()
    {
        return "Hola";
    }
}
```

Interfaces

```
public static void Main()
{
    PruebaInterfazImplicita obj = new PruebaInterfazImplicita();

    // Produce un error de compilación de que la clase
    //no contiene ese método
    //obj.MiMetodo();

    // Debemos castear el objeto
    string str = ((IMiInterfaz)obj).MiMetodo();
    Console.WriteLine(str);
    Console.ReadKey();
}
```