Az interfészek C#-ban egy fontos programozási eszközt jelentenek az objektum-orientált programozásban. Egy interfész lényegében egy absztrakt szerződés, amely meghatározza, hogy egy osztály milyen metódusokat, tulajdonságokat vagy eseményeket kell implementáljon anélkül, hogy azok részleteit megadná. Egy osztály több interfészt is implementálhat, ami rugalmasságot ad a tervezésben.

Alapvető interfész példa

```csharp
// Definiálunk egy interfészt
public interface IAnimal
{
    void MakeSound();
    string Name { get; }
}

// Osztály, amely implementálja az interfészt
public class Dog : IAnimal
{
    public string Name { get; private set; }

    public Dog(string name)
    {
        Name = name;
    }

    public void MakeSound()
```

```
    {
        Console.WriteLine("Woof!");
    }
}

// Másik osztály, amely szintén implementálja az interfészt
public class Cat : IAnimal
{
    public string Name { get; private set; }

    public Cat(string name)
    {
        Name = name;
    }

    public void MakeSound()
    {
        Console.WriteLine("Meow!");
    }
}

// Az interfész használata
class Program
{
    static void Main(string[] args)
    {
        IAnimal dog = new Dog("Buddy");
```

```
        IAnimal cat = new Cat("Whiskers");

        Console.WriteLine($"The dog's name is {dog.Name} and it says:");
        dog.MakeSound();

        Console.WriteLine($"The cat's name is {cat.Name} and it says:");
        cat.MakeSound();
    }
}
```

Kimenet:

```
The dog's name is Buddy and it says:
Woof!
The cat's name is Whiskers and it says:
Meow!
```

Több interfész implementálása egy osztályban

Egy osztály több interfészt is implementálhat, ami hasznos lehet különböző képességek szétválasztására.

```csharp
public interface IAnimal
{
```

```
    void MakeSound();
}

public interface IPet
{
    void Play();
}

public class Dog : IAnimal, IPet
{
    public void MakeSound()
    {
        Console.WriteLine("Woof!");
    }

    public void Play()
    {
        Console.WriteLine("The dog is playing with a ball.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Dog dog = new Dog();
        dog.MakeSound();
```

```
        dog.Play();
    }
}
```

Interfész és polimorfizmus

Az interfészek segítenek a polimorfizmus elérésében. Egy interfész típusú változó különböző osztályok példányaira mutathat, amelyek implementálják az adott interfészt.

```csharp
public interface IShape
{
    double CalculateArea();
}

public class Circle : IShape
{
    public double Radius { get; set; }

    public Circle(double radius)
    {
        Radius = radius;
    }

    public double CalculateArea()
```

```
    {
        return Math.PI * Radius * Radius;
    }
}

public class Rectangle : IShape
{
    public double Width { get; set; }
    public double Height { get; set; }

    public Rectangle(double width, double height)
    {
        Width = width;
        Height = height;
    }

    public double CalculateArea()
    {
        return Width * Height;
    }
}

class Program
{
    static void Main(string[] args)
    {
        IShape circle = new Circle(5);
```

```
        IShape rectangle = new Rectangle(4, 6);


        Console.WriteLine($"The area of the circle is {circle.CalculateArea()}");
        Console.WriteLine($"The area of the rectangle is {rectangle.CalculateArea()}");
    }
}
```

Kimenet:

```
The area of the circle is 78.53981633974483
The area of the rectangle is 24
```

Összegzés

Az interfészek fontos szerepet játszanak a kód rugalmasságának növelésében, lehetővé téve az implementáció részleteinek elválasztását a szerződéstől. Ezáltal a kód könnyebben bővíthető, fenntartható és tesztelhető lesz.