第六课：继承和接口

继承（Inheritance）

子类继承父类的属性和方法，避免重复代码。

```
// 父类
class Animal
{
    public string name;
    public void Eat()
    {
        Console.WriteLine($"{name} 在吃东西");
    }
}

// 子类
class Dog : Animal
{
    public void Bark()
    {
        Console.WriteLine($"{name} 在叫: 汪汪!");
    }
}
```

```
Dog dog = new Dog();
dog.name = "旺财";
dog.Eat();   // 继承自 Animal
dog.Bark();  // 自己独有的
```

方法重写（override）

```
// 父类 virtual 方法
class Animal
{
    public string name;
    public virtual void Speak()
```

```
    {
        Console.WriteLine($"{name} 发出了声音");
    }
}

// 子类重写
class Cat : Animal
{
    public override void Speak()
    {
        Console.WriteLine($"{name} 喵喵喵~");
    }
}
```

```
Animal a = new Cat();
a.name = "小猫";
a.Speak();  // 输出: 小猫 喵喵喵~
```

base 关键字

子类调用父类的方法/构造函数：

```
class Dog : Animal
{
    public string breed;

    public Dog(string name, string breed) : base(name)
    {
        this.breed = breed;
    }

    public override void Speak()
    {
        base.Speak();  // 先调用父类方法
        Console.WriteLine($"{name} 汪汪!");
    }
}
```

访问修饰符 protected

修饰符	任何地方	继承访问
`private`	✗	✗
`protected`	✗	✓

```
class Animal
{
    protected string name;  // private 的话子类也访问不到
}

class Dog : Animal
{
    public void Test()
    {
        name = "旺财";  // protected 可以访问
    }
}
```

接口（Interface）

接口 = 约定/契约，定义必须实现的方法。

```
// 定义接口
interface IFlyable
{
    void Fly();           // 不写实现
    int MaxAltitude { get; }  // 可以有属性
}

// 实现接口
class Bird : IFlyable
{
    public int MaxAltitude => 1000;

    public void Fly()
    {
        Console.WriteLine("鸟儿在飞翔");
    }
}
```

```
class Plane : IFlyable
{
    public int MaxAltitude => 10000;

    public void Fly()
    {
        Console.WriteLine("飞机在飞翔");
    }
}
```

接口 vs 继承

	继承	接口
关键词	`class A : B`	`class A : IB`
只能继承一个父类	✓	可以实现多个接口
继承获得代码	✓	只获得"约定"
何时用	代码复用	约定行为

```
// 一个类只能继承一个父类
class Dog : Animal { }

// 但可以实现多个接口
class Duck : Animal, IFlyable, ISwimable
{
    public void Fly() { }
    public void Swim() { }
}
```

例子：设备驱动架构

```
// 设备接口
interface IDevice
```

```
{
    void Connect();
    void Disconnect();
    bool IsConnected { get; }
}

// 麦克风设备
class Microphone : IDevice
{
    public bool IsConnected { get; private set; }

    public void Connect()
    {
        Console.WriteLine("麦克风连接");
        IsConnected = true;
    }

    public void Disconnect()
    {
        Console.WriteLine("麦克风断开");
        IsConnected = false;
    }
}
```