

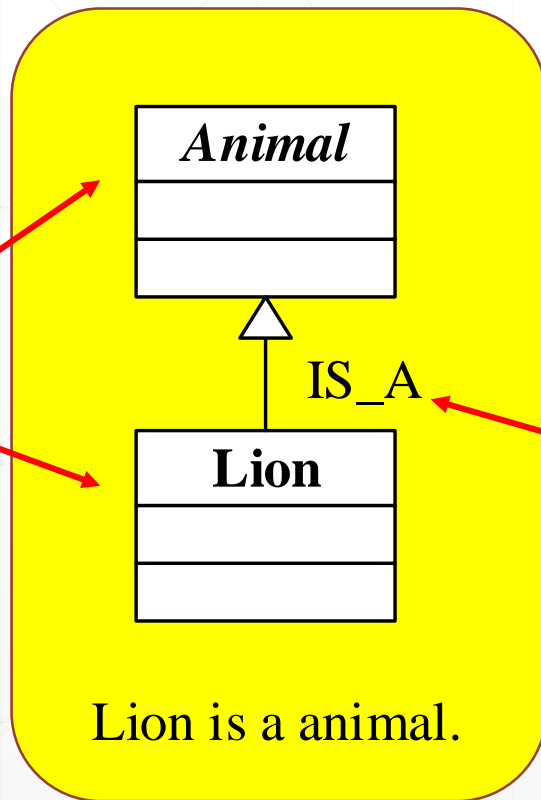
继承

北京理工大学计算机学院
金旭亮

继承是对现实生活中的“**分类**”概念的一种模拟。

示例：狮子**是**一种动物。

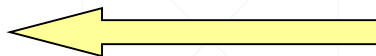
狮子拥有动物的一切基本特性，但同时又拥有自己的独特的特性，这就是“继承”关系的重要性质。



形成继承关系的两个类之间，是“**IS_A**”关系

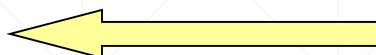
在C#中实现继承

```
class Animal  
{  
}
```



父类 (parent class) /
基类 (base class)

```
class Lion : Animal  
{  
}
```



子类 (child class)

从外部使用者角度来看，子类“自动”拥有了父类声明为 **public** 和 **protected** 的成员，这就是继承的最重要特性之一。

父类

```
class Parent
{
    public int Pi = 100;
    public void Pf()
    {
        Console.WriteLine("Parent.Pf()");
    }
    protected int Pj = 200;
    protected void Pg()
    {
        Console.WriteLine("Parent.Pg()");
    }
    private int k = 300;
}
```

子类中的代码可以直接访问父类保护级别的成员，但外界不能通过对象变量来直接访问声明为保护级别的类成员。

子类

```
class Child : Parent
{
    public void cf()
    {
        Pg(); //子类可以访问父类的保护成员
        Pj += 200;
        Console.WriteLine("Child.cf()");
        Console.WriteLine("Parent.Pj={0}", Pj);
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Child c = new Child();
        //可以通过子类变量访问定义在基类的公有成员
        c.Pi = 300;
        Console.WriteLine("Parent.Pi={0}" , c.Pi);
        c.Pf();
        //c.Pj = 1000; //Error! 不能访问保护级别的成员
        c.cf(); //可以通过子类定义的公有方法访问基类保护级别的成员
    }
}
```

更进一步：继承环境下的字段访问规则

同一类中的实例方法可以访问所有字段。

子类实例方法可以访问父类中的protected和public的字段，但不能访问private的字段。

变量同名时，“离得最近”、“关系最密切”的变量起作用。

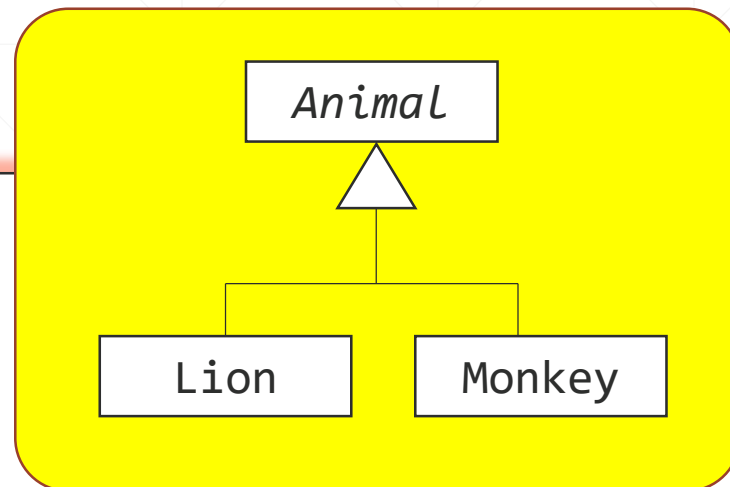
参看VariableScope示例。

“父”与“子”



(德) 埃·奥·卜劳恩 (绘)

```
Animal an = null ;  
Lion lion = new Lion ();  
an = lion ;    // 正确  
lion = an;    // 编译时错误  
lion = (Lion)an; // 正确  
Monkey m = (Monkey)an; // 运行时错误
```



子类对象可以赋值给父类（基类）变量，这实际上是“IS_A”关系的体现。

当子类、父类的方法名相同时，有两种情况：

Overload（重载）

```
class Parent {  
    public void OverloadF() {  
    }  
}  
  
class Child : Parent {  
    public void OverloadF(int i) {  
    }  
}
```

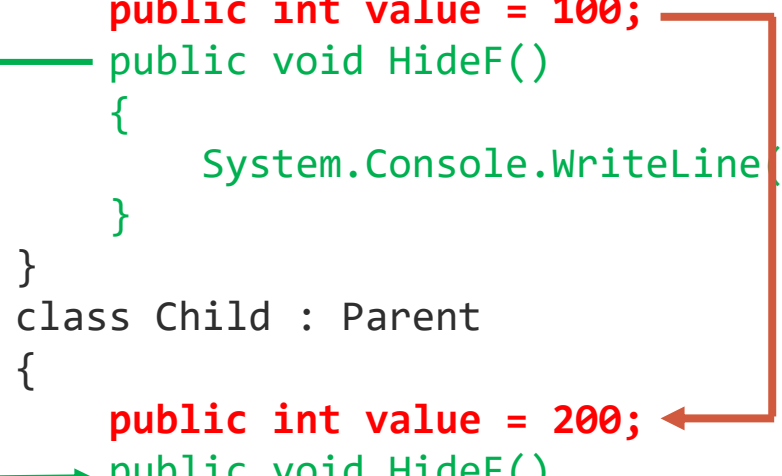
Override（重写/覆盖）

```
class Parent {  
    public virtual void OverrideF() {  
    }  
}  
  
class Child : Parent {  
    public override void OverrideF() {  
    }  
}
```

请仔细观察OverloadAndOverride示例的输出结果并细心体会其特点。

子类父类方法/字段“一模一样”时

```
class Parent
{
    public int value = 100;
    public void HideF()
    {
        System.Console.WriteLine("Parent.HideF()");
    }
}
class Child : Parent
{
    public int value = 200;
    public void HideF()
    {
        System.Console.WriteLine("Child. HideF()");
    }
}
```



当分别位于父类和子类的两个方法/字段完全一样时，调用哪个由对象变量**定义时**的类型决定，除非你使用了“类型强制转换”。

```
0个引用
class Program
{
    0个引用
    static void Main(string[] args)
    {
        Parent p = new Parent();
        Child c = new Child();

        p.HideF();
        c.HideF();

        p = c; //基类变量引用子类对象
        p.HideF(); //会输出什么结果？
        (p as Child).HideF(); //会输出什么结果？

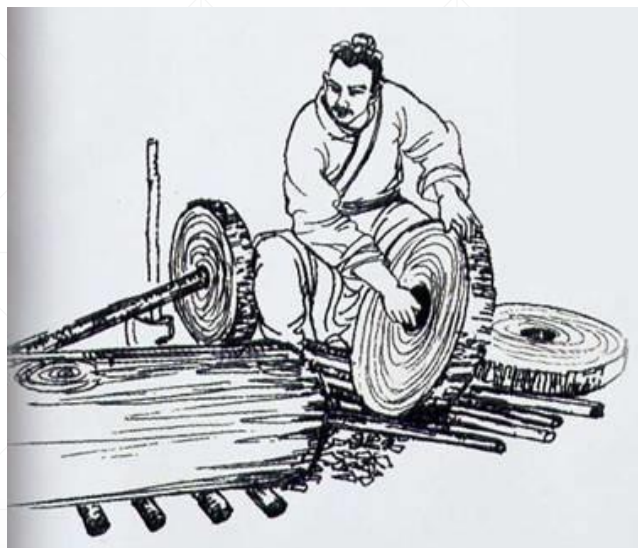
        //以下代码，输出哪个值？
        Console.WriteLine(p.value);
        Console.WriteLine((p as Child).value);

        Console.ReadKey();
    }
}
```


开发建议：不要自找麻烦！

在实际开发时，不要在子类中定义与父类一模一样的成员（包括字段、属性和方法）！

提升软件开发效率的法宝——重用

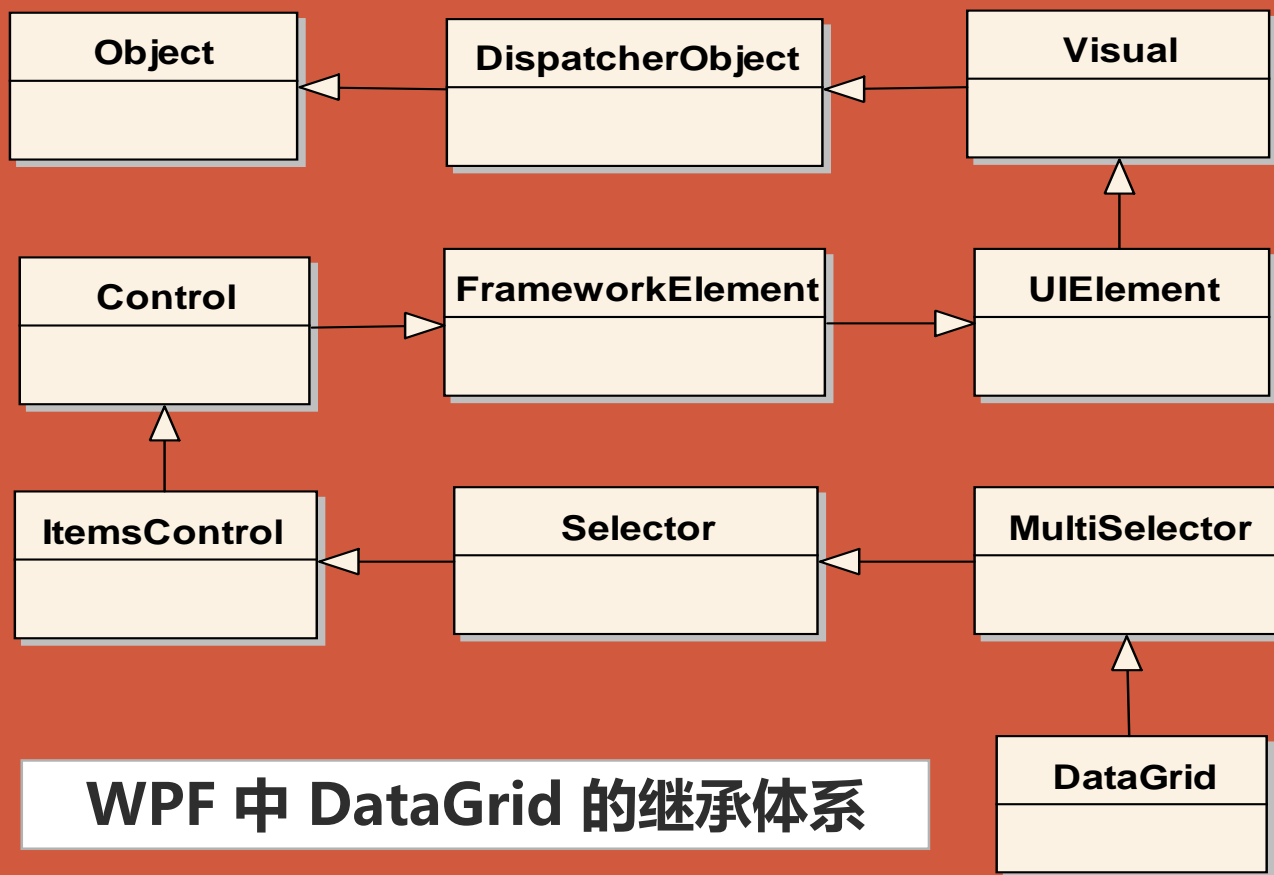


不要重新发明轮子

在面向对象思想发展的初期，**通过继承复用代码**曾经被认为是面向对象最重要的目标之一。

很遗憾，实践中人们发现在开发中滥用继承后患无穷.....

可怕的“一字长蛇阵”



代码间强耦合，拥有极深的类型继承树，上层基类一改，所有子类均受影响，并且这种变动所带来的影响很难预计……



牵一发而动全身

结论：

慎用继承！