

抽象类与接口

北京理工大学计算机学院
金旭亮

抽象类

抽象类与抽象方法

- 在一个类前面加上“**abstract**”关键字，此类就成为了**抽象类**。
- 一个方法前面加上“**abstract**”关键字，此方法就成为了**抽象方法**。

```
abstract class Fruit    //抽象类
{
    public abstract void GrowInArea(); //抽象方法
}
```

1

抽象方法不包容任何实现代码。

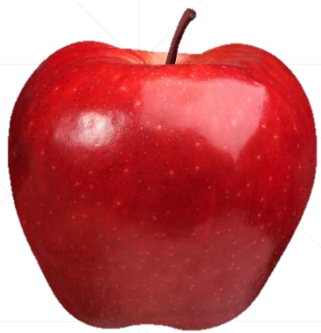
2

无法使用new关键字直接创建抽象类的对象。

3

包含抽象方法的类一定是抽象类，但抽象类中的方法不一定是抽象方法，抽象类中可以包容“普通的”方法。

实例背景



中国的南方、北方均可以种植苹果



菠萝是一种亚热带水果，
主要在中国的南方种植

抽象类的定义

```
3 个引用
abstract class Fruit    //抽象类
{
    4 个引用
    public abstract void GrowInArea(); //抽象方法
}

1 个引用
class Apple : Fruit //苹果
{
    4 个引用
    public override void GrowInArea()
    {
        Console.WriteLine("我是苹果，南方北方都可以种植我。");
    }
}

1 个引用
class Pineapple : Fruit //菠萝
{
    4 个引用
    public override void GrowInArea()
    {
        Console.WriteLine("我是菠萝，喜欢温暖，只能在南方看到我。");
    }
}
```

定义抽象基类

子类重写抽象基类的抽象方法

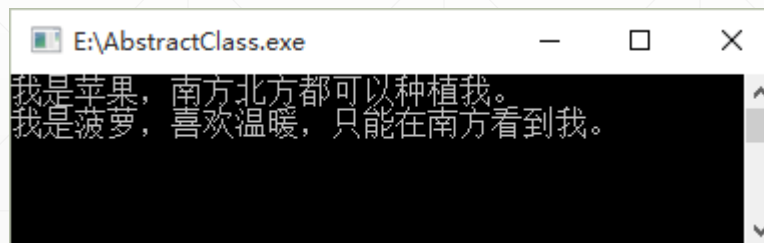
“抽象类”怎么用？

不能创建抽象基类的对象，只能用它来引用子类的对象。

抽象类名 变量名 = new 继承自此抽象类的具体子类名();

```
0 个引用
class Program
{
    0 个引用
    static void Main(string[] args)
    {
        Fruit f;
        f = new Apple();
        f.GrowInArea();
        f = new Pineapple();
        f.GrowInArea();
        Console.ReadKey();
    }
}
```

相同的语句，在不同的环境下执行，得到不同的结果，这就是“多态”的一种具体表现形式。



E:\AbstractClass.exe

我是苹果，南方北方都可以种植我。
我是菠萝，喜欢温暖，只能在南方看到我。

接口

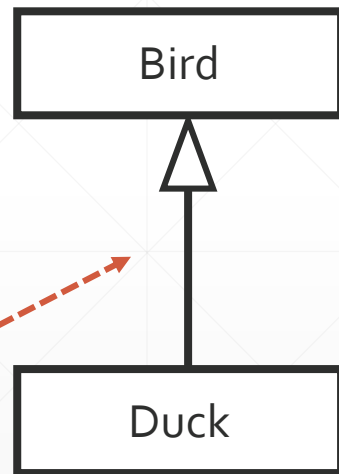
让我们先从“继承”聊起……

“继承”是对现实世界中“**是一种 (IS_A)**”关系的模拟。

鸭子“**是一种**”鸟。



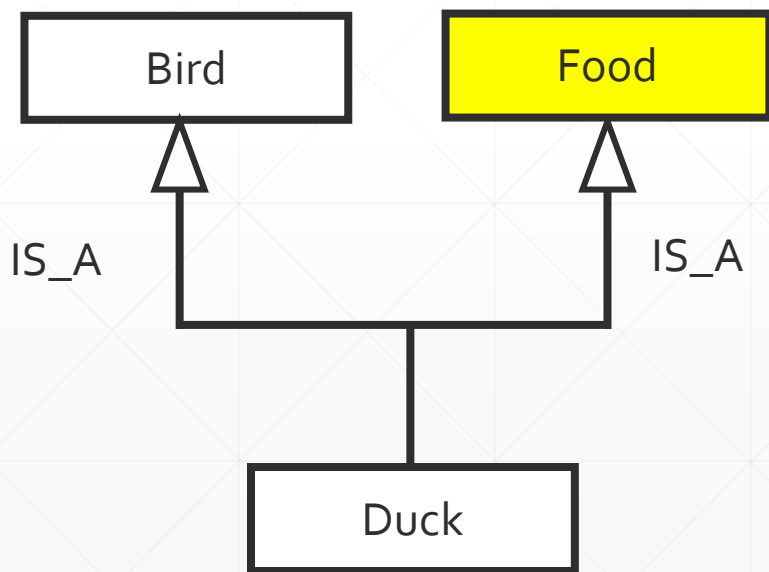
空心三角箭头符号代表继承



现在试着为以下一个场景建立一个面向对象的编程模型

鸭子**是一种**鸟，会游泳，同时**又是一种**食物。

一种设计方案是：



- “会游泳”这个方法放在哪个类中？

1. 并不是只有鸭子一种鸟会游泳。
2. 并不是所有鸟都会游泳。

- C#/Java等编程语言不支持多继承

解决方案

将事物的特性单独抽取出来

```
graph TD; A[将事物的特性单独抽取出来] --> B[IFood接口抽象出“可食用（即可烹调）”特性]; A --> C[ISwim接口抽象出“会游泳”特性];
```

IFood接口抽象出“可食用（即可烹调）”特性

```
public interface IFood
{
    void Cook();
}
```

ISwim接口抽象出“会游泳”特性

```
public interface ISwim
{
    void Swim();
}
```

C#中接口的特点

使用interface关键字定义接口

```
public interface IFood
{
    void Cook();
}
```

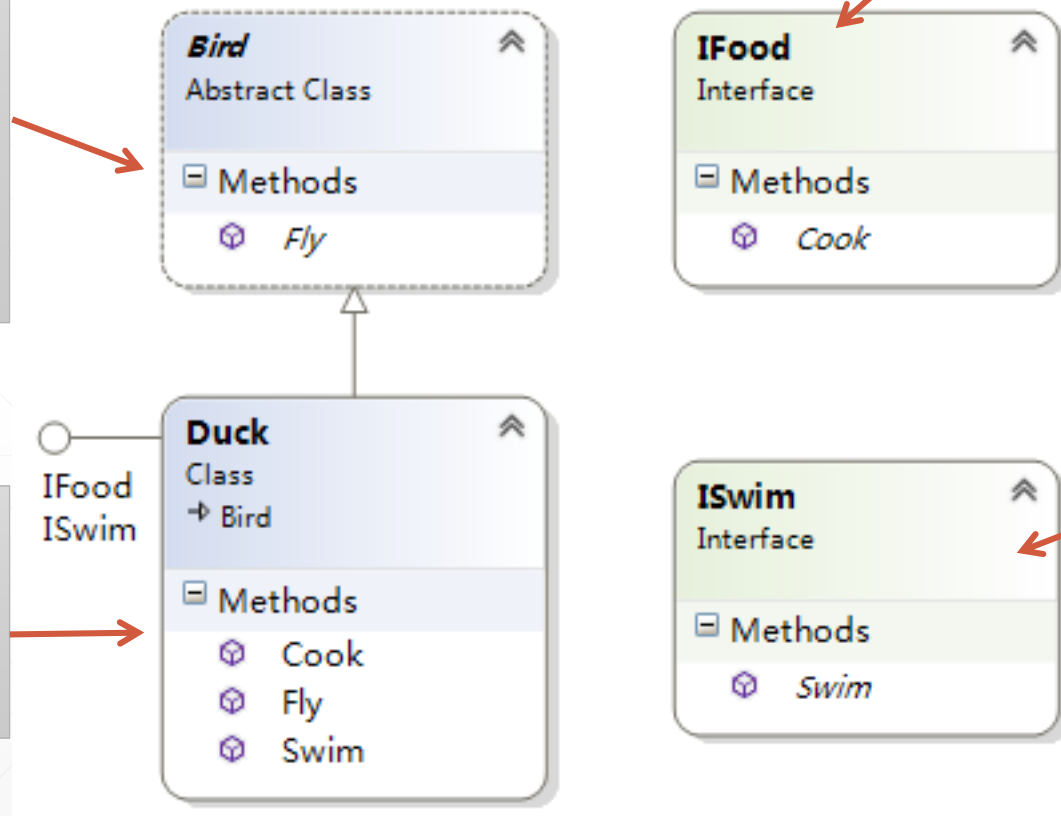
接口的名字通常以“I”打头。

接口中的方法只有声明，不包含任何代码

解决“鸭子”的建模问题

鸟是一种抽象的概念，所以设计为抽象类，同时，鸟会飞，不同鸟的飞翔方式不一样，所以fly方法是Bird类的抽象方法

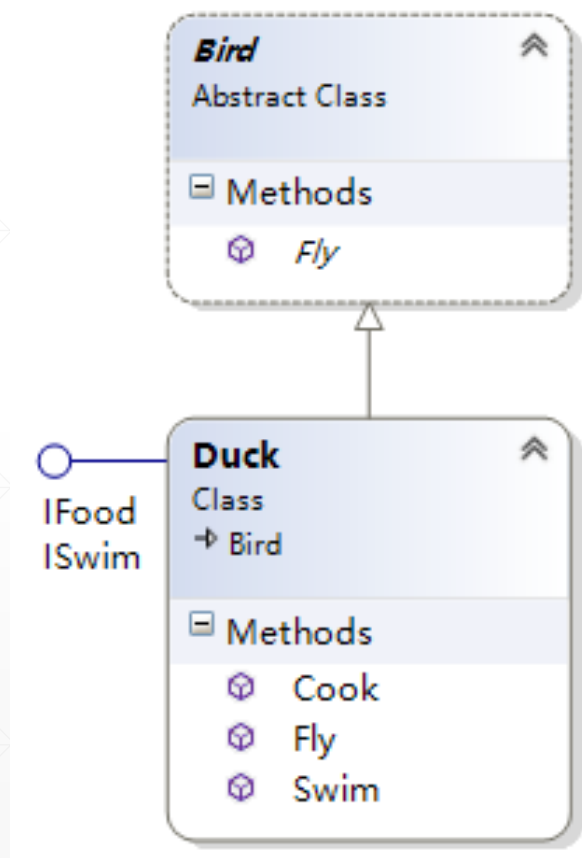
鸭子是一种鸟，会游泳，北京烤鸭很好吃



食物应该实现IFood接口

水鸟则应该实现ISwim接口

一个类可以“实现（ implements ）”多个接口



```
//定义一个抽象类
2 references
public abstract class Bird
{
    3 references
    public abstract void Fly();
}

//继承自一个抽象类，实现两个接口
2 references
public class Duck : Bird, IFood, ISwim
{
    //实现ISwim接口
    3 references
    public void Swim()
    {
        Console.WriteLine("是鸭子就会游泳");
    }

    //实现IFood接口
    3 references
    public void Cook()
    {
        Console.WriteLine("鸭子经常被烧烤，北京烤鸭就很有名");
    }

    //实现抽象类Bird中的抽象方法
    3 references
    public override void Fly()
    {
        Console.WriteLine("只有野鸭才会飞");
    }
}
```

“接口”小结

1

与抽象基类相比，接口不包含任何的实现代码。

2

接口实际上可以看成一种约定，对于所有实现了接口的类，可以说“它们看上去都是这样的……”，但到底类是如何“遵守”与实现这种规定，完全由类自己来定。

3

接口在面向对象开发实践中用得极广。