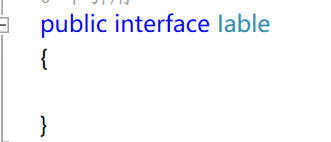
**接口：interface**

**接口是一种规范，是一种能力，按照接口的规范继承了这个接口，那么就拥有了接口的能力**

比如说电脑的usb口就是一种接口，在生产的时候规范了只能插usb、usb口大小等等，当我们按照这种规范生产了这种规模的设备，插上设备后能让设备使用，这就是他的能力

**接口的使用注意：**

1. 接口的命名为I..able



1. 为了多态，接口不允许创建自身的对象（实例化自身）

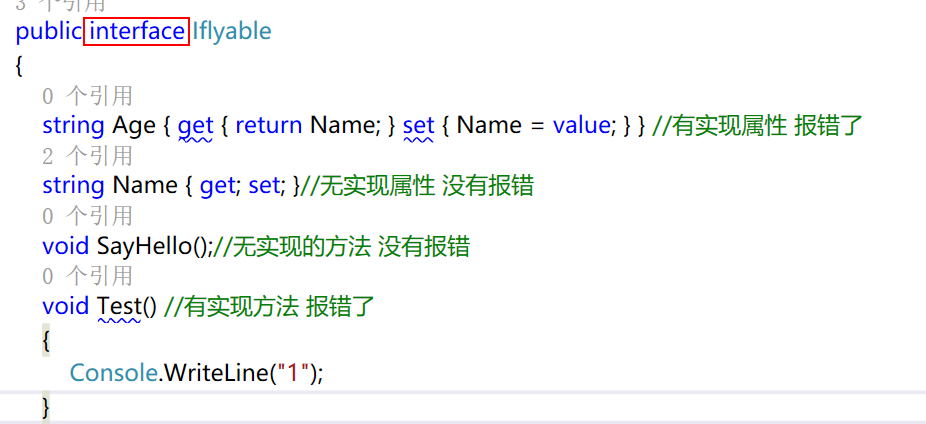


3. **接口中只允许存在方法、自动属性、索引器、事件，这些本质上都是方法**

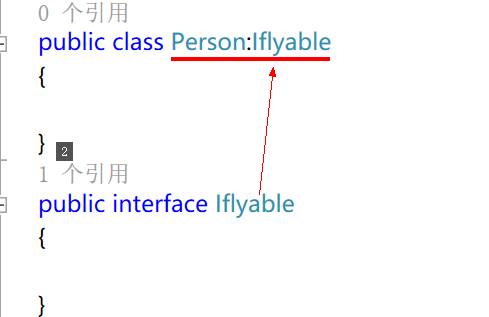
4. 接口中的成员不能加“访问修饰符”，接口中的成员“默认访问修饰符”为：Public，并且不能修改



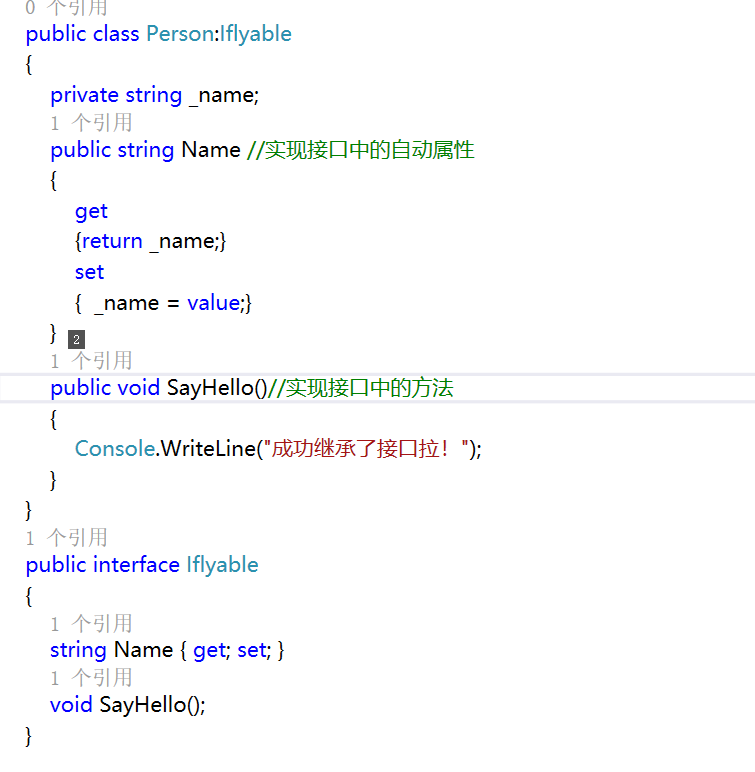
5. **接口中的成员必须都没有实现，不允许存在有实现的成员！**



1. 接口使用的前提，需要一个类去继承

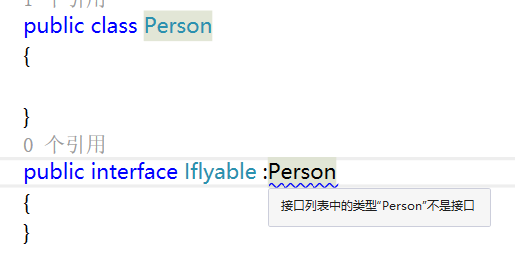
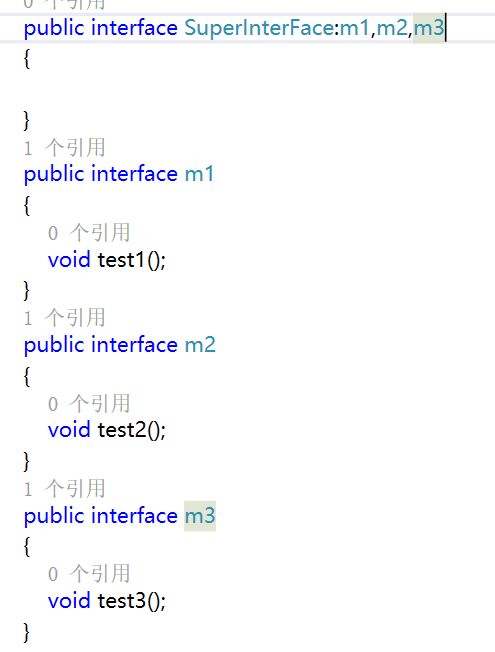


7. 一个类继承（实现）了接口，如果接口中定义了成员，那么这个类就必须**实现接口中的所有成员**

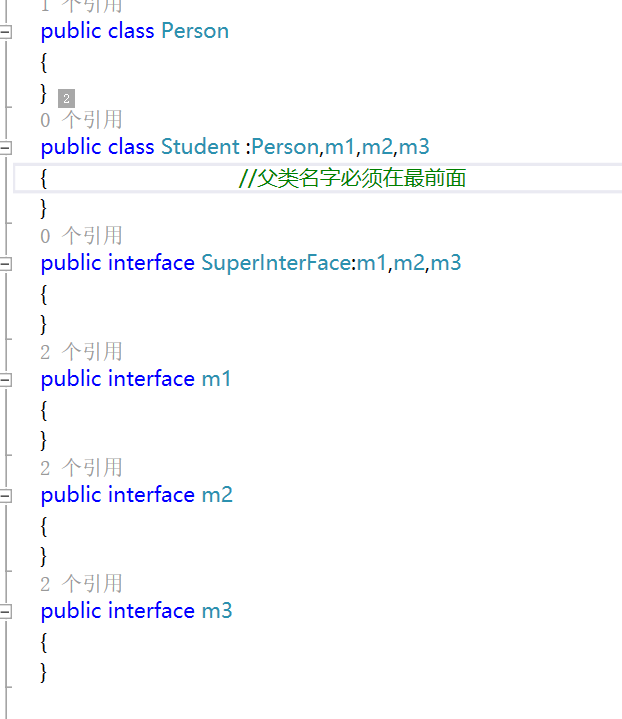


8. 接口不能继承类，**可以继承接口，并且可以多继承**

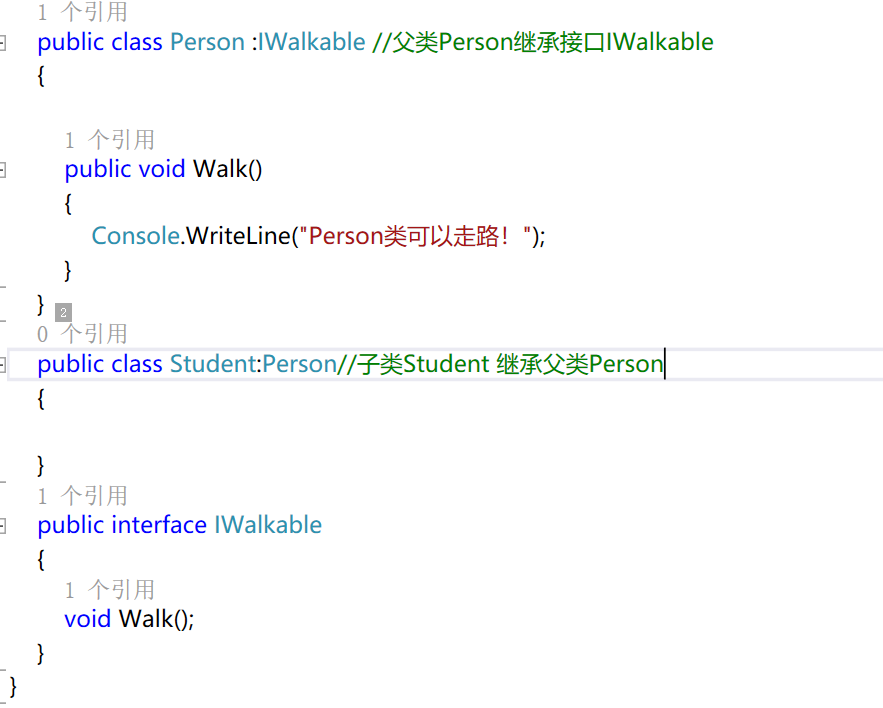
接口不能继承类： 接口能继承接口，并且能多继承

9. 类可以继承类的同时继承单个或多个接口，在同时继承类和接口的时候，**父类的名字必须在最前面**



1. **一般都是接口去指向子类，要避免指向父类的情况发生！这样做没有意义，因为接口代表的是一种能力，谁有这种能力就让谁实现这个接口，如果父类继承并实现了这种接口，那么父类的子类也就都间接的继承并实现了这个接口，拥有这种能力，这样就没有了能力间的差异性，因为并不是每一个子类都拥有这种能力，并且，这样做还多此一举，如果我们想让子类都拥有某个父类的成员，单单的继承不就能实现了吗？还用接口？**



1. 类中的原有的方法名和参数 与 接口中的方法名和参数一摸一样的时候就需要显示实现接口

例如：我原本在一个Person类中定义了一个方法，并且继承了接口，这时候接口中需要实现的方法名和参数与我原本在Person类中定义的方法名和参数一摸一样，系统就会把我们原本写给Person类的方法当成是实现接口中的方法的了，这样就不好分辨，如下图：



这时候我们就需要显示的实现接口，来达成区分的目的，显示的实现接口，与普通的实现效果是一模一样的，作用只是用来解决：**方法中的重名的问题，区分类中原本定义的方法与实现接口的方法**

**（注意：红色方块中并没有加入“访问修饰符”，因为这里是调用并实现接口中的方法了，而接口中的方法不能写“访问修饰符”，且默认为：Public，所以这里的“访问修饰符”也为：Public）**

1. 当一个抽象类继承了接口，且实现接口后，需子类才能调用

当在一个抽象类中实现了接口，那么在没有子类的情况下只能通过两种方法来调用实现接口的方法，第一种：单纯的调用自己的方法，创建自身的对象，通过类的对象名.方法名来调用，第二种：通过多态，实例化接口，接口对象指向抽象类，通过接口的对象名.方法名来调用，但是这两种方法都不行，因为抽象类不能实例化自己，也就是不能new 自身（指向自己类中的对象），如下图：



当我们写一个子类来继承Person，这时候子类会继承Person类中的所有方法、字段、属性等等，**也包括了实现接口的那个方法**，所以这时候我们实例化接口，指向子类的对象，使用多态，就能够调用了Person类中实现接口的方法了，如下图：

