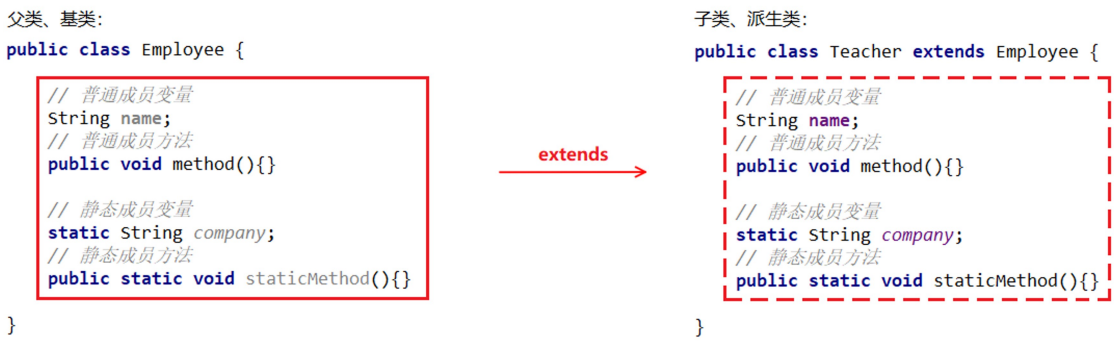
day09_继承、super、this、抽象类

一.继承

面向对象的三大特征：封装、继承、多态。继承就是子类继承父类的属性、行为，子类相当于拥有了父类的所有成员变量、成员方法。继承可以实现共性抽取、代码复用，使类与类之间产生了关系，是多态的前提。



(1).子类从父类继承过来的成员变量、成员方法，使用起来和子类本身的成员变量、成员方法无异，完全符合"成员变量、成员方法的使用规则"。

(2).子类也可以继承父类的私有属性成员。但是在子类中，不能直接访问继承过来的private私有属性，只能通过继承过来的get、set方法来间接访问继承过来的私有属性。

1.成员变量、成员方法重名时的访问规则

之前我们说，在父、子类成员没有重名的情况下。子类从父类继承过来的成员变量、成员方法，使用起来和子类本身的成员变量、成员方法无异，完全符合"成员变量、成员方法的使用规则"。当父、子类成员有重名的时候怎么办呢？重名的成员变量、成员方法的使用规则如下：

(1).重名的普通成员变量、方法

(2).重名的静态成员变量、方法

2.构造方法的访问规则

在子类的所有构造方法中，都有一个默认隐含的 "super();" 调用，这就是在调用父类的无参构造方法。所以当我们用构造方法new一个子类对象时，一定是先调用了父类的构造方法，后执行的子类构造方法。(调用父类构造方法的目的，是为了初始化从父类继承过来的成员属性)

- 当我们把 "super()" 调用语句显式补写出来时，原有隐含的 "super()" 调用就会失效。
- 可以使用super关键字来调用父类重载的有参构造方法，来初始化从父类继承过来的成员属性，此时默认隐含的 "super()" 调用就会失效。
- 调用父类构造方法的 "super()" 语句，必须是子类构造方法的第一个语句，并且只能有唯一一个 "super()" 调用语句。

3.方法的覆盖重写

在继承关系中，当子类的成员方法和父类的成员方法名称一样，参数列表也一样时，称为子类对父类成员方法的覆盖重写（Override）。由此可见，方法的覆盖重写是"父、子类成员方法重名" 的一种特殊情况，即：父、子类重名的成员方法不一定能构成方法的覆盖重写，但是能构成覆盖重写的一定是父、子类重名的成员方法。因此覆盖重写的方法也符合 "1.成员变量、成员方法重名时的访问规则"。

(1).构成方法覆盖重写的条件：

a.必须保证父、子类之间方法的名称相同，参数列表也相同。

b.子类方法的返回值必须小于等于父类方法的返回值范围。
Object > String等其他类：java.lang.Object类是所有类的公共最高父类，java.lang.String类就是Object的子类。

c.子类方法的权限修饰符必须大于等于父类方法的权限修饰符。权限修饰符的权限大小关系如下：
public > protected > (default) > private　　(default)不是关键字default，而是什么都不写，留空

@Override注解：写在方法前面，用来检测是不是有效的正确的覆盖重写。（不写也可以，不影响方法的覆盖重写，只是单纯起检测作用）

(2).对方法覆盖重写的理解：
在子类中，对继承过来的父类方法进行覆盖重写时，无论是在子类当中，还是其它类中，都优先使用子类的方法。此时就相当于是："从父类继承过来且被覆盖重写的方法"，被子类方法覆盖、遮掩住了，然后子类再来添加更多自己的内容。
（可看成是子类中已经不存在"从父类继承过来且被覆盖重写的方法"了，被子类方法替换掉了）

4.super、this关键字的用法总结

(1).super关键字的用法有两种：

a.在子类的普通成员方法中，访问父类重名的成员变量、成员方法。

b.在子类的构造方法中，访问父类的构造方法。

(2).super关键字用来访问父类内容，而this关键字用来访问本类内容。用法也有两种：

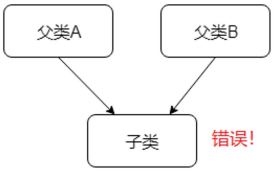
a.在本类的普通成员方法中，访问本类的成员变量、成员方法。

b.在本类的构造方法中，访问本类的另一个构造方法。

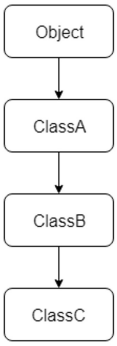
- 可以使用this关键字来调用本类中重载的有参构造方法，但是构造方法之间不能形成递归调用！
- 与"super()"调用不同，编译器不会默认赠送一个隐含的"this()"调用。"this()"调用必须是构造方法的第一个语句，而且是唯一一个。
- 由于super()、this()两种构造调用不能同时使用，所以一旦使用了"this()"调用，那么原本隐含的"super()"调用将会失效。

5.Java继承的特点

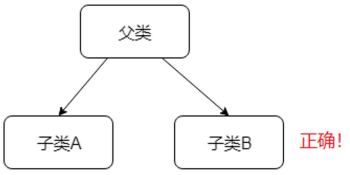
(1).Java只支持单继承，不支持多继承。即一个类的直接父类只能有唯一一个。



(2).Java支持多层继承。（所有类都继承自Object类，它是最高公共父类）



(3).一个子类的直接父类是唯一的，但一个父类可以拥有多个子类。



二.抽象类

抽象类：是没有具体对象的类，例如：动物类、图形类等，没有具体的对象，是一个抽象的概念，所以称为抽象类。定义抽象类的语法格式如下：

```
public abstract class 类名 { // 在class之前写上abstract关键字即可

}
```

抽象方法：只有方法声明，没有方法体的方法称为抽象方法。定义抽象方法的语法格式如下：

```
public abstract 返回值类型 方法名(); // 加上abstract关键字，去掉方法体大括号，直接分号结束
```

如果一个类包含抽象方法，那么该类必须是抽象类，但是抽象类中不一定会有抽象方法。未包含抽象方法的抽象类，目的就是不想让调用者创建该类对象，通常用于某些特殊的类结构设计。

(1).抽象类不能直接通过new创建对象，必须通过一个子类来继承抽象父类，通过创建其非抽象子类的对象，来使用抽象父类的内容。

(2).由于子类继承了抽象父类的抽象方法，但是子类并不是抽象类，却含有继承过来的抽象方法，那么该怎么办呢？

所以子类必须覆盖重写继承自抽象父类中的所有抽象方法，除非该子类也是抽象类。此时相当于是子类将继承过来的父类抽象方法，给覆盖重写掉了。可看成是子类当中已经不存在这些"从父类中继承过来的抽象方法了"，被子类方法替换掉了。

此时方法的覆盖重写是子类对父类抽象方法方法体的完成实现，因此也叫做"方法实现"。（去掉抽象方法的关键字abstract，然后补上方法体大括号和方法体内

容）

(3).虽然抽象类抽象类中可以有构造方法，是供子类创建对象时，初始化父类成员使用的。因为在子类的构造方法中，有默认的super()调用，此时就会调用抽象父类的构造方法，对继承自抽象父类的成员进行初始化。

三.继承的综合案例——发红包