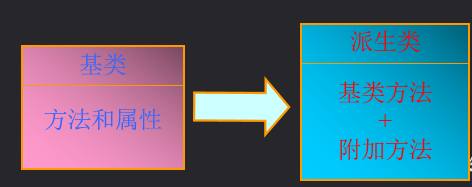
# 继承

## 作用及目的

### 作用

让类与类之间产生关系,子父类关系（描述类和类之间的关系）; 重用现有的类来生成新类的一种特征；

通俗地讲就是从现有的类（即父类或基类）创建新类（子类或派生类）的过程。

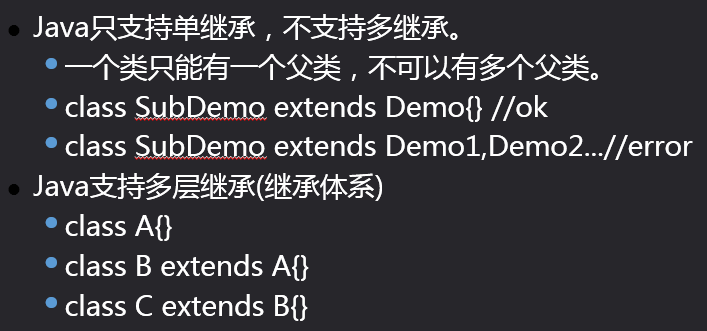


### 目的

面向对象程序设计中的继承，则是代码重用；继承是利用现有的类创建新类的过程，现有的类称作基类（或父类），创建的新类称作派生类（子类）。

## 特点

Java只支持单继承，不支持多继承。(一个儿子只能有一个爹)；Java支持多层继承(继承体系)（儿子，老子，爷爷）；如果想用这个体系的所有功能用最底层的类创建对象；如果想看这个体系的共性功能,看最顶层的类



## 实现

使用extends关键字：class 子类名 extends 父类名 {}

## 好处与弊端

### 好处

继承的好处

a:提高了代码的复用性，多个类相同的成员可以放到同一个类中（降低类和类之间的重复代码）

b:提高了代码的维护性，如果功能的代码需要修改，修改一处即可（降低对象和对象之间的代码重复使用静态变量）

c:让类与类之间产生了关系，是多态的前提，其实这也是继承的一个弊端：类的耦合性很强。

### 弊端

:继承的弊端

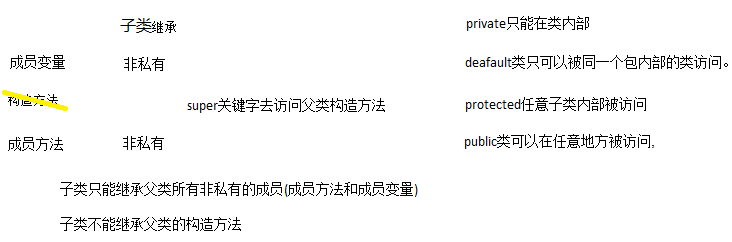
类的耦合性增强了。

## 继承原则

子类只能继承父类所有非私有的成员(成员方法和成员变量)

子类不能继承父类的构造方法，但是可以通过super关键字去访问父类构造方法；

不要为了部分功能而去继承



对于class的权限修饰只可以用public和deafault。

private只能在类内部，

deafault类只可以被同一个包内部的类访问。

protected任意子类内部被访问，

public类可以在任意地方被访问,



# 继承中的关系—隐藏,重写

## 概述

子类只能继承父类所有非私有的成员(成员方法和成员变量)，其实这也体现了继承的另一个弊端：打破了封装性

子类不能继承父类的构造方法，但是可以通过super(后面讲)关键字去访问父类构造方法。不要为了部分功能而去继承

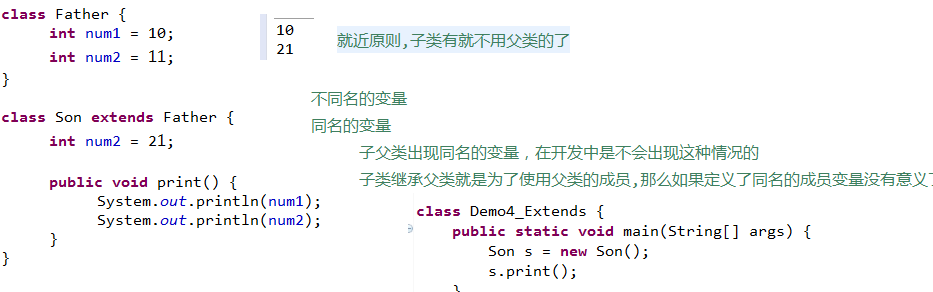
我们到底在什么时候使用继承呢?

继承中类之间体现的是：”is a”的关系。

## 成员变量的关系

父类成员变量被隐藏

对于子类可以从父类继承的成员变量，只要子类中声明的成员变量和父类中的成员变量同名时，子类就隐藏了继承的成员变量。



## 构造方法的关系

### 子类中构造方法默认都会访问父类中构造方法（空参）

子类中所有的构造方法默认都会访问父类中空参数的构造方法

为什么呢?

1.因为子类会继承父类中的数据，可能还会使用父类的数据。所以，子类初始化之前，一定要先完成父类数据的初始化。

2.每一个构造方法的第一条语句默认都是：super()；可省略，重写被覆盖



### 父类没有无参构造方法,子类怎么办?

父类没有无参构造方法,子类怎么办?（了解）

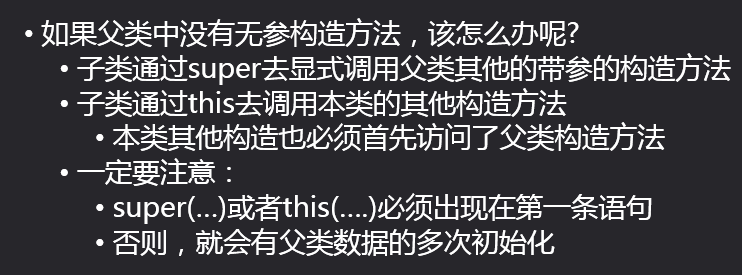
super解决

this解决



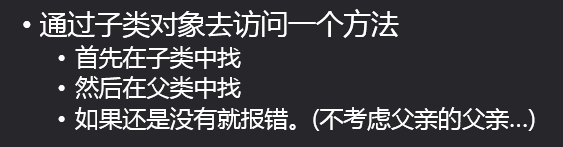
注意事项

\* super(…)或者this(….)必须出现在构造方法的第一条语句上



## 实例方法的关系—方法重写

### 逐级访问



### 什么是方法重写

子类中出现了和父类中一模一样的方法声明，也被称为方法覆盖，方法复写。

使用特点：

如果方法名不同，就调用对应的方法

如果方法名相同，最终使用的是子类自己的

### 方法重写的应用：

当子类需要父类的功能，而功能主体子类有自己特有内容时，可以重写父类中的方法。这样，即沿袭了父类的功能，又定义了子类特有的内容。

间接隐藏了父类的方法（super.方法可调用）



### 注意事项

#### 父类中私有方法不能被重写

a:父类中私有方法不能被重写

因为父类私有方法子类根本就无法继承

#### 子类重写父类方法时，访问权限不能更低

b:子类重写父类方法时，访问权限不能更低

最好就一致

#### 父类静态方法，子类也必须通过静态方法进行重写

c:父类静态方法，子类也必须通过静态方法进行重写

其实这个算不上方法重写，但是现象确实如此，至于为什么算不上方法重写，静态只能覆盖静态

#### 子类重写父类方法的时候，最好声明一模一样

d: 子类重写父类方法的时候，最好声明一模一样。

## 静态方法—没有

静态方法和类相关，算不上重写