# 面向对象的特征

**面向对象的特征为：封装、继承、多态（抽象）**

## 1.1封装

### 定义

封装是面向对象方法的重要原则，就是把对象的属性和操作（或服务）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节。封装是一种信息隐藏技术。

简单来讲：封装就是将属性私有化，提供公有的方法访问私有属性

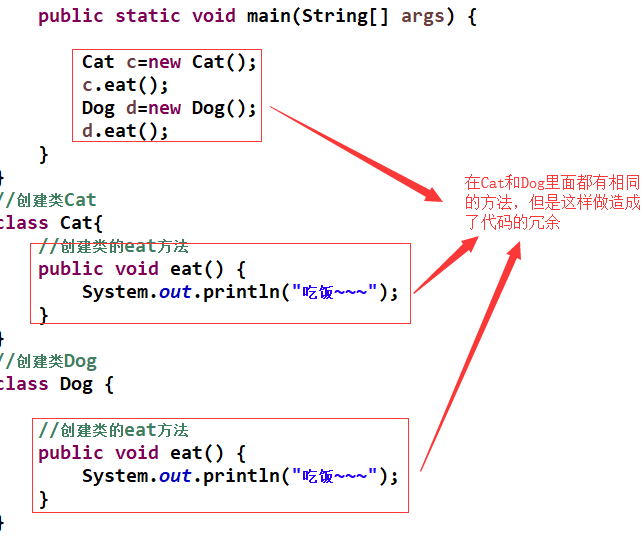
### 体现形式

1. 方法

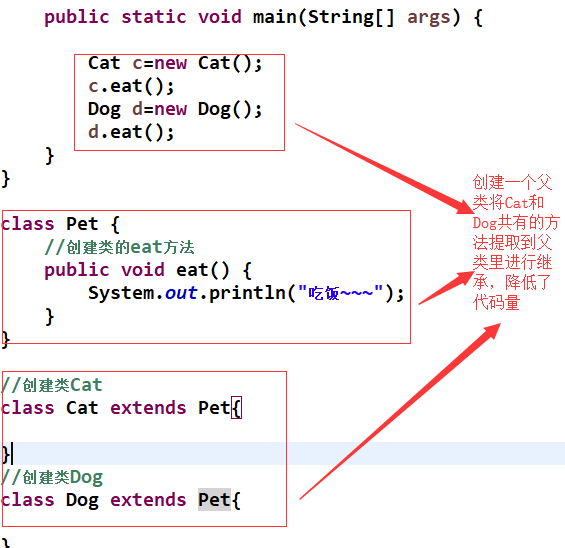
就是在程序里如果需要多次书写同样的属性或者其他相同的代码，为了避免代码的冗余，我们可以将相同的代码内容封装到相同的方法里，在外界直接调用就好。提高了代码的复用性

举例说明：

方法被封装前：



封装后：

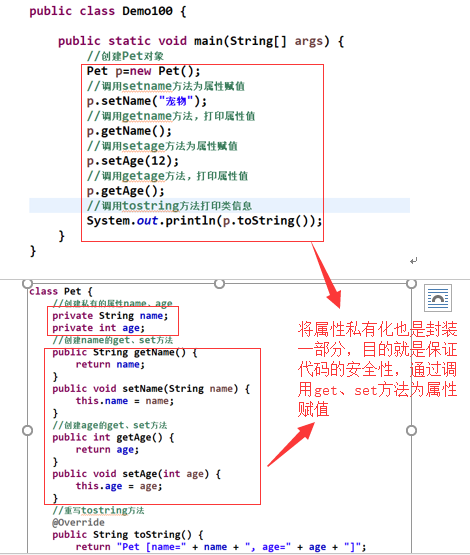


1. 属性的私有化

将属性私有化，为了防止在类外操作属性的时候给属性赋值一些不合常理的值，因此将属性私有化，提供了对类外的访问方法来间接的操作属性。——属性私有化保护了数据的安全性。

将属性私有化之后，可以通过在类内定义get和set方法来实现在类外通过对象来实现对私有属性的操作。

举例说明：



1. 内部类

内部类也是封装的一种体现形式。（在此不做详细的赘述，后续会详细讲解内部类）

### 优点

提高了复用性、安全性、使代码结构更加紧密。

1. 使属性私有化---隐藏信息，实现细节。
2. 使属性值更符合要求---可以对成员进行更精确的控制。
3. 提高了代码的安全性---类内部的结构可以自由修改
4. 良好的封装能够减少耦合。

## 继承

### 定义

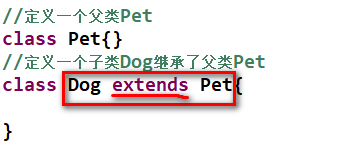
如果一些类中含有相同的属性和方法，那么可以将这些相同的属性和方法提取到一个新的类中，利用extends关键字让原来的类和新的类产生联系，而这个关系称之为继承。新产生的类称之为父类（也可以称之为超类/基类），原来的类称之为子类（也可以称之为派生类）。

子类通过继承父类可以使用父类中的一些方法和属性。

注意：在Java中支持的是单继承——一个子类只能有一个父类，而一个父类可以有多个子类。

### 格式

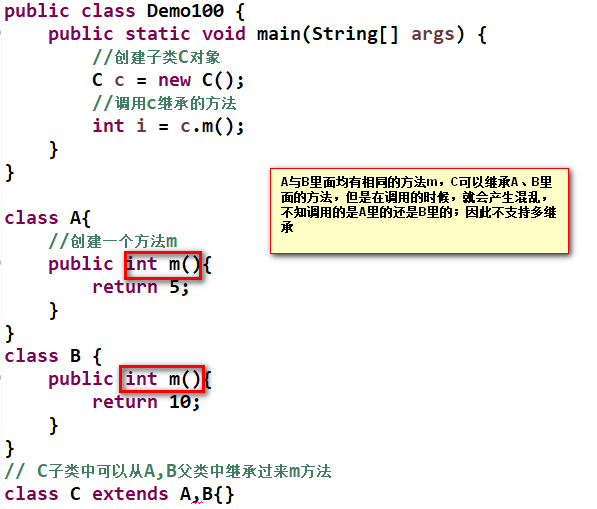
如下图所示：



### 单继承和多继承的优劣性比较

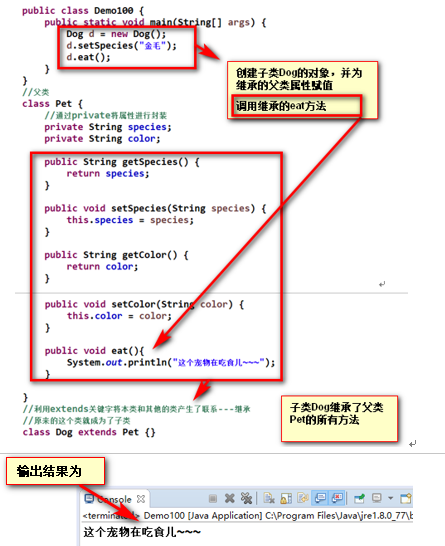
举例说明：如下图所示

1. 多继承在代码的复用性上要优于单继承，但是存在方法调用的混乱。



1. 单继承也可以提高代码的复用性，可以避免方法调用的混乱，提高了方法调用的安全性。

举例说明：



### 特征

1. 继承关系是传递的。
2. 继承简化了人们对事物的认识和描述，能清晰的体现相关类间的层次结构关系。
3. 继承提供了软件复用功能。
4. 继承通过增强一致性来减少模块间的接口和界面，大大增加了程序的易维护性。
5. 提供了多重继承机制。通过使用接口机制来实现多重继承。

### 优点

提高复用性、提高安全性、统一结构。

### 使用条件

1. 子类拥有父类非private的属性和方法。
2. 子类可以拥有自己的属性和方法，即子类可以对父类进行扩展。
3. 子类可以用自己的方法实现父类的方法。

### 缺陷

1.父类变，子类就必须变。

2.继承破坏了封装，对于父类而言，它的实现细节对于子类来说都是透明的。

1. 继承是一种强耦合关系。

## 权限修饰符

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 本类 | 子类 | 同包类 | 其他类 |
| public | 可以 | 可以 | 可以 | 可以 |
| protected | 可以 | 可以 | 可以 | 不可以 |
| 默认 | 可以 | 同包子类可以 | 可以 | 不可以 |
| private | 可以 | 不可以 | 不可以 | 不可以 |

注意：权限修饰符用于限定变量或者方法的使用范围。其中public范围最大，private范围最小。

## 方法的重写

### 定义

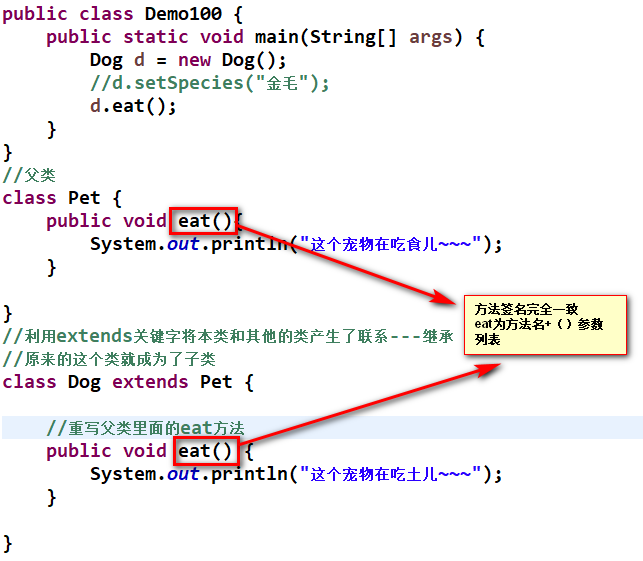
子类可继承父类中的方法，而不需要重新编写相同的方法。但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就[需要](https://baike.baidu.com/item/%E9%9C%80%E8%A6%81/82595)采用方法的重写。方法重写又称[方法覆盖](https://baike.baidu.com/item/%E6%96%B9%E6%B3%95%E8%A6%86%E7%9B%96)。

简单来讲就是指在父子类里存在了方法签名（方法名+参数列表）完全一致的的方法，就构成了方法的重写（覆盖）。

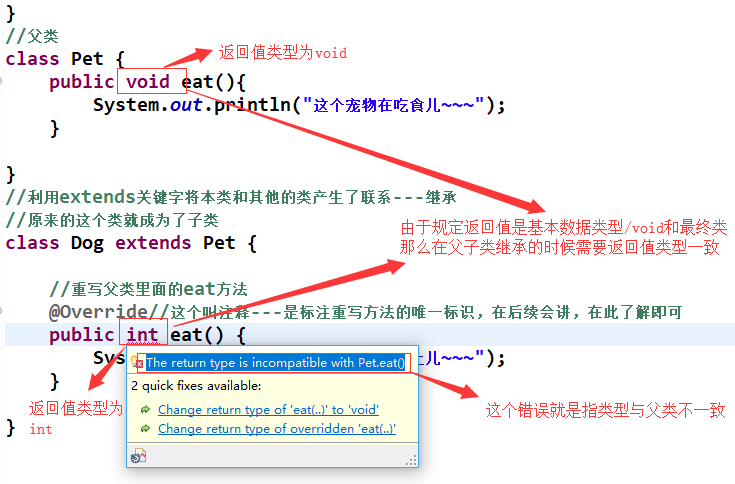
### 遵循规则

方法的重写需要遵循“两等两小一大”

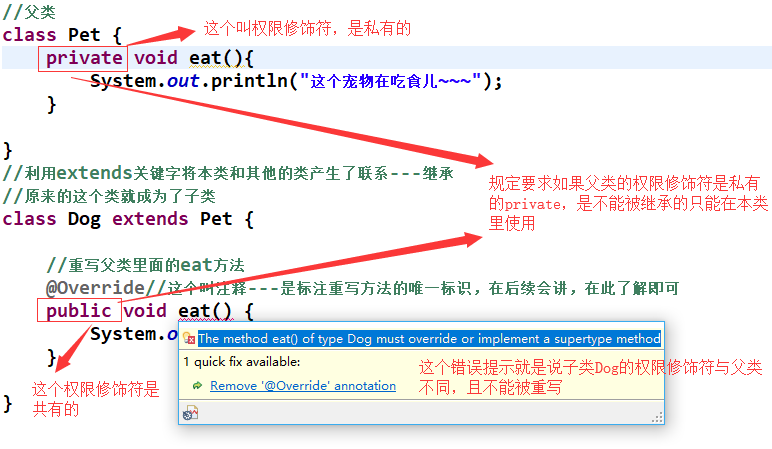
1. 方法签名要求完全一致



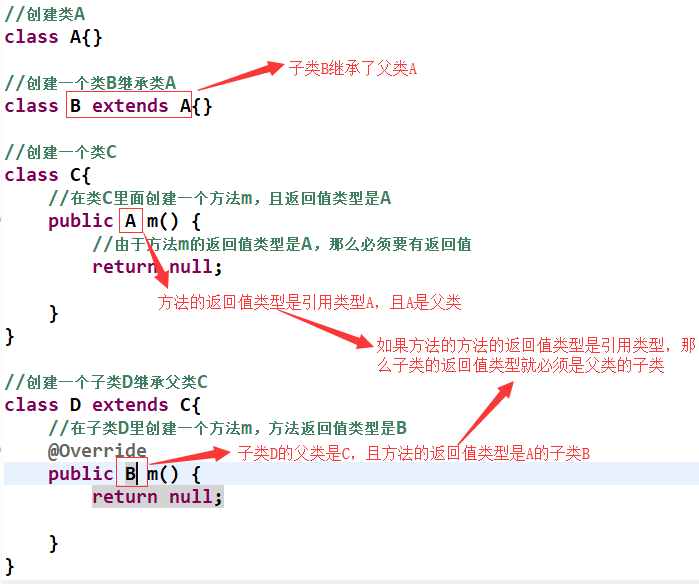
1. 如果父类里的返回值是基本数据类型/void/最终类，那么子类里重写的返回值类型必须一致（如果父类的方法是private的话，子类是无法被继承的，即使子类重新定义这个方法，也不算重写）



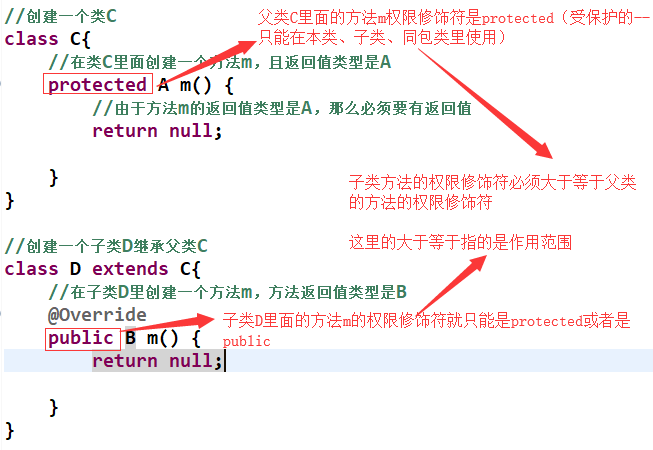
权限修饰符为private



1. 如果父类方法的返回值类型是一个引用类型，子类方法的返回值类型是父类方法的返回值类型的子类



1. 子类方法的权限修饰符的范围要大于等于父类方法权限修饰符的范围



注意：对于8种基本数据类型之间是没有继承关系的，他们是同级关系

## Super关键字

### 定义

在创建子类的时候，会先创建一个父类对象，而这个父类对象在子类里一super关键的形式存在

super关键字就是父类对象的引用，通过super来调用父类里的方法或者属性，可以将super看成一个虚拟对象。

### super语句

书写格式：super（）；

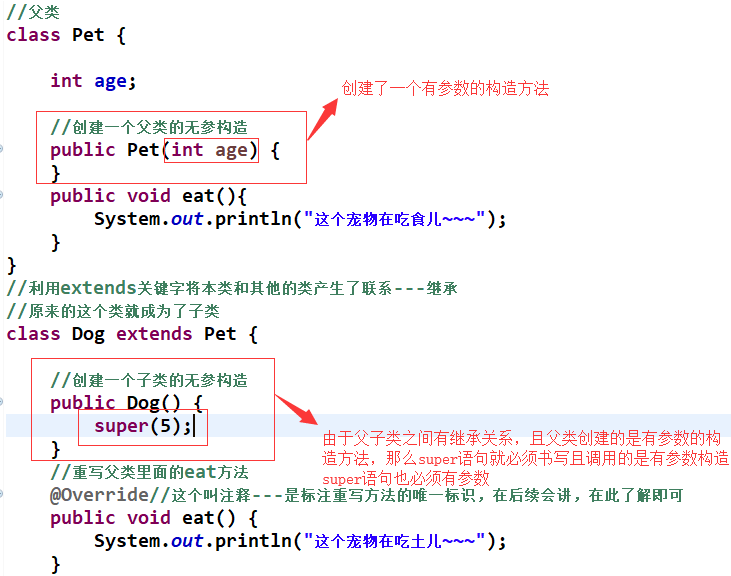
在子类构造函数种通过调用父类对应形式的构造函数。

注意：如果在子类构造函数里没有手动添加，那么JVM在编译的时候就会自动添加一个无参数的super语句。

1.创建无参构造 如下图所示



1. 创建有参构造 如下图所示

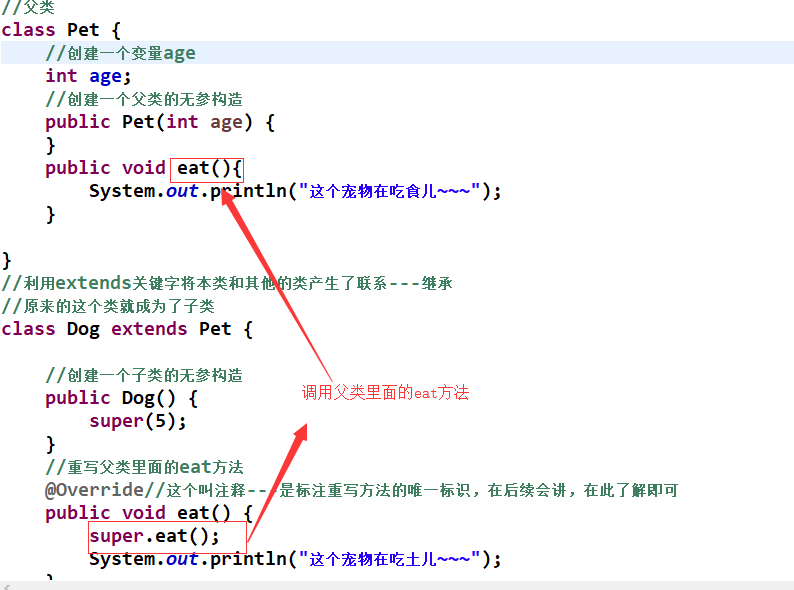


### super用法

1. super.成员变量/super.成员方法；

用来在子类中调用父类的同名成员变量或者方法。

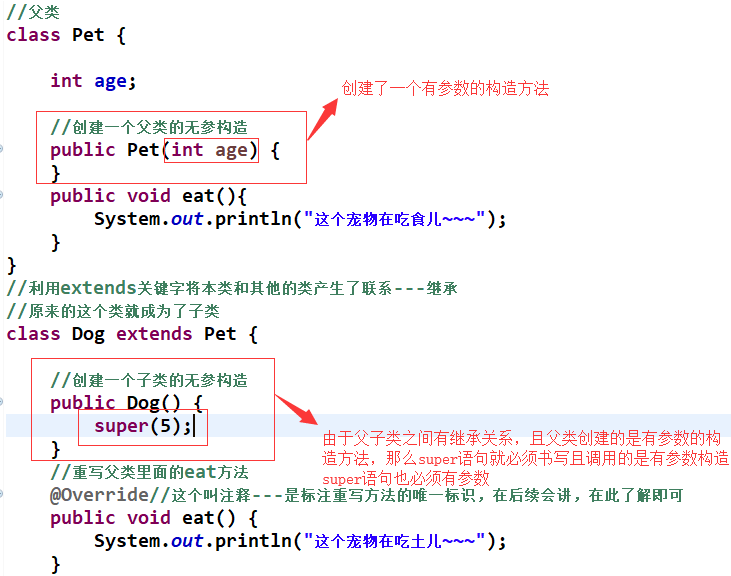
如下图所示：



1. super(parameter1,parameter2……)；

用在子类的构造器中显示调用父类的构造器

说白了就是调用父类的构造方法，如下图所示：



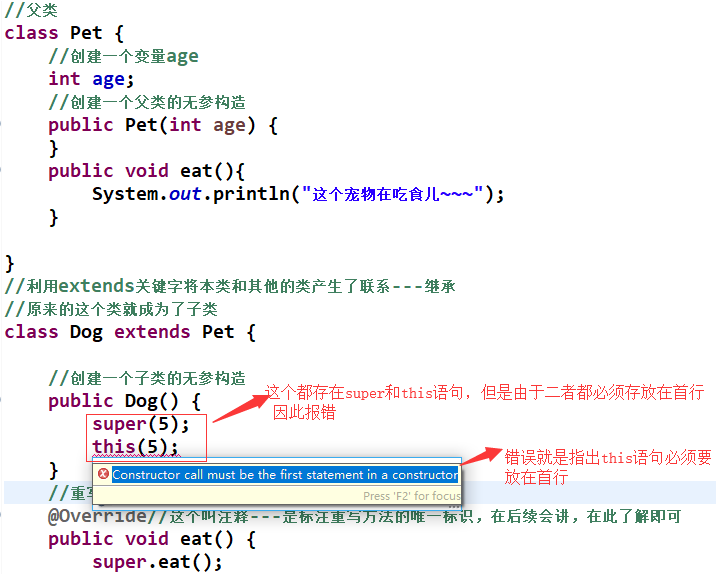
### this和super的比较

this语句和super语句不能同时使用，如果子类的构造函数中已经存在来this语句，它会调用本身的其他含有super语句的的构造函数，来创建父类。

子类的构造函数中必然直接或者间接的有一个super语句，在创建子类对象的时候必然先创建一个父类对象。

由于this和super都必须放在语句的首行，那么而者不可同时出现

如下图所示：



## 多态

### 定义

多态是指允许不同类的对象对同一消息做出响应。即同一消息可以根据发送对象的不同而采用多种不同的行为方式。多态主要针对对象的行为即方法，而不是对象的属性。

简单来讲java中多态的代码体现在一个子类对象(实现类对象)既可以给这个子类(实现类对象)引用变量赋值，又可以给这个子类(实现类对象)的父类(接口)变量赋值。

### 动态绑定

实现多态的技术称为动态绑定，是指在执行期间判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。

### 作用

降低模块与模块之间的依赖性就是解耦；多态的产生使为了解耦，消除类型之间的耦合关系。

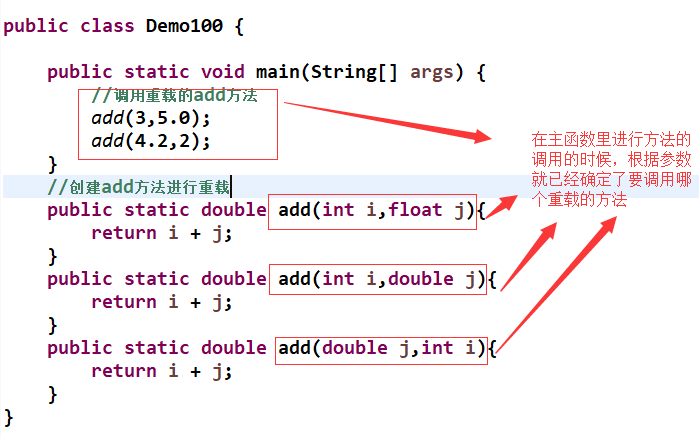
### 必要条件

要有继承、要有重写、父类引用指向子类对象。

### 体现形式

#### 编译时多态

在编译时期就要进行动态绑定的运行过程； 就是方法的重载



#### 运行时多态

在运行时期才能确定绑定的运行过程。就是向上造型、方法的重写。

继承是运行时多态的前提。

##### 向上造型

###### 定义

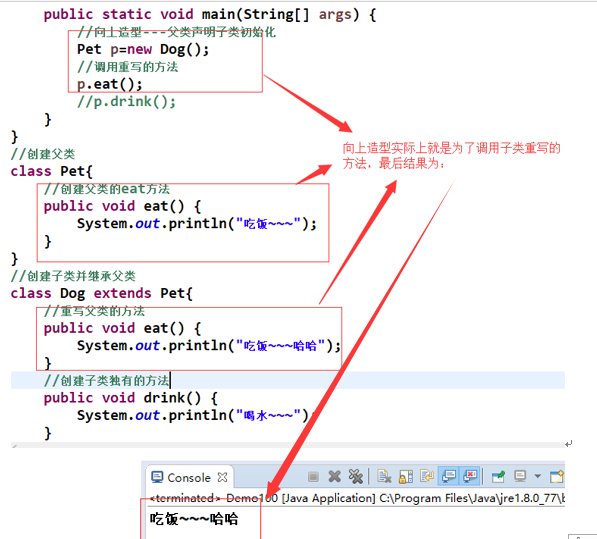
用父类声明的，用子类来创建实际对象叫向上造型。

子类的对象可以向上造型为父类的类型。即父类引用子类对象，这种方式被称为向上造型。

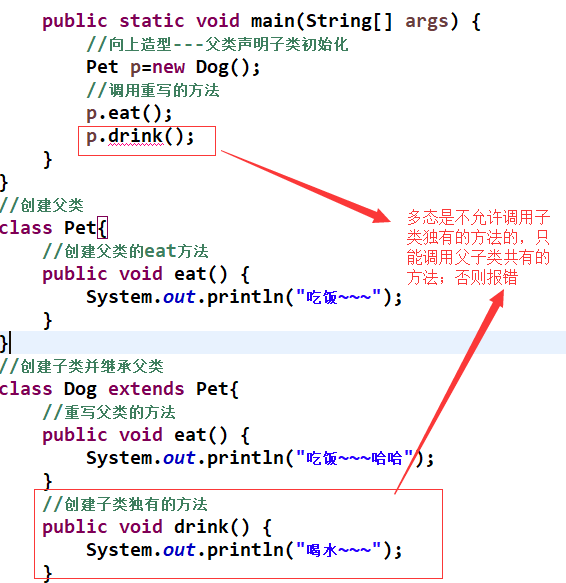
注意: 向上造型是用父类声明用子类初始化——由于这个对象是父类声明的，所以对象能够干什么需要看父类；由于是用子类初始化的，具体怎么干要看子类。

##### 方法的重写

当调用方法的时候就是调用的充血之后的方法，只能调用子类里与父类相同的方法。



如果调用子类独有的方法：



### 优点

1. 可替换性（substitutability）。多态对已经存在代码具有可替换性。
2. 可扩充性（extensibility）。多态对代码具有可扩充性。
3. 接口性（interface-ability）。
4. 灵活性（flexibility）。在应用中体现了灵活多向的操作，提高了使用效率。配合反射实现解耦。
5. 简化性（simplictiy）。多态简化对应用软件的代码编写和修改过程，尤其在处理大量对象的运算和操作时，这个特点尤为突出和重要。