第01天 面向对象

今日内容介绍

* 基础知识回顾
* 继承

**今日学习目标**

* 能够解释类名作为参数和返回值类型
* 能够理解匿名对象
* 能够写出类的继承格式
* 能够说出继承的特点
* 能够说出子类调用父类的成员特点

# 1.面向对象-基础内容回顾

## 面向对象思想回顾

学习Java的一个基本核心是理解面向对象思想：万物皆对象。

面向对象是一种解决问题的思路。按照这种思路，我们关心的是有什么东西能帮助我们解决问题，然后找到这个东西，使用这个东西的功能来解决问题。

用面向对象思想写程序时，我们首先要进行需求分析，搞清楚要做什么事情，然后就要找具有能够实现我们需求的功能的类（如果找不到，再自己手写这样的类），最后创建对象，调用函数，实现需求。

举例：从上海到北京。

## 类和对象回顾

### 类的书写

先有类后有对象，Java程序就是由不同的类组成的，所以写好类是写好一个Java程序最基本的要求。那么怎么才能写出一个类呢？

大家都知道，类的书写格式很简单，就是先写一个class关键字，后面跟上一个类名和一个大括号就行了，关键问题是类里面该怎么写。

**类是对象的抽象，对象是类的实例化。**

**一个类中可以书写的成员有以下几种：**

成员变量；

set和get函数；

其他一般函数；

构造函数；

需求：定义类描述一个人，人的属性有姓名和年龄；人的行为有吃饭和说话

//需求：定义类描述一个人，人的属性有姓名和年龄；人的行为有吃饭和说话

**public** **class** Person {

//在类中，属性使用成员变量表示；

//为了数据安全，所以使用private关键字修饰成员变量

**private** String name;//表示人的姓名；因为姓名是字符串，所以使用String类型的变量表示；

**private** **int** age;//表示人的年龄；因为年龄是整数，所以使用int型变量表示

//因为成员变量都已经私有化，为了外界可以访问，所以需要提供公开的get和set函数

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

//在Java的类中，使用方法表示事物的行为；所以接下来要定义两个方法表示吃饭和说话这两个行为

//表示人的吃饭行为

**public** **void** eat(){

System.***out***.println("吃饭");

}

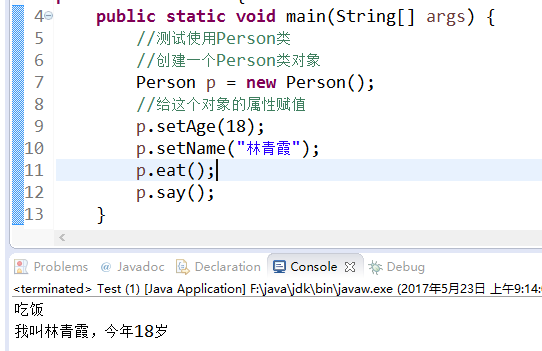
//表示人的说话的行为

**public** **void** say(){

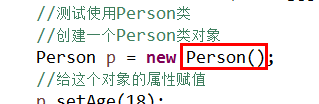
System.***out***.println("我叫"+name+"，今年"+age+"岁");

}

}



使用默认构造函数创建对象



### 构造函数

**概念**：构造函数就是在创建对象时由JVM自动调用的函数，作用是在创建对象时给对象的成员变量赋值的；

**写法**：

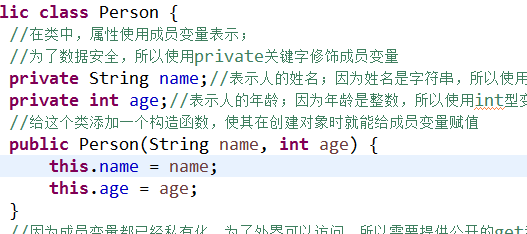
**修饰符**：构造函数只能使用访问权限修饰符修饰；如public、private等；不允许使用static关键字修饰构造函数

**返回值类型**：构造函数没有返回值类型，连void都没有；这个是区分一个函数是否是构造函数的标准

**函数名**：构造函数的函数名必须和类名一致

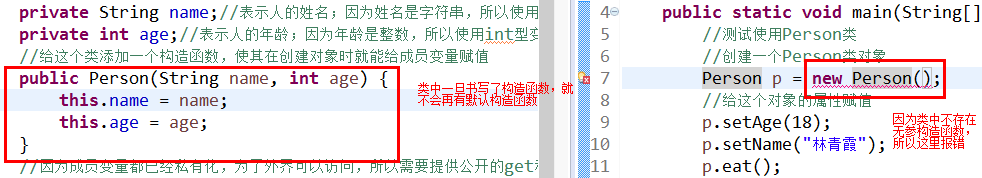
**参数列表**：构造函数可以重载，参数列表可以根据需要定义；

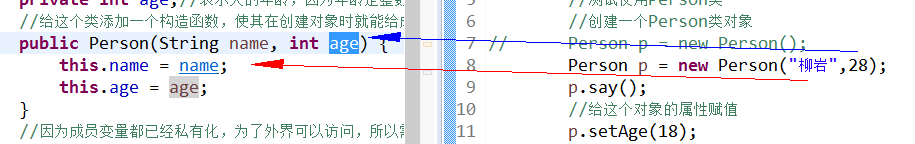
**return语句**：所有Java的函数都需要通过return语句出栈，所以构造函数中也有return语句，只是因为不返回数据，所以一般不写；

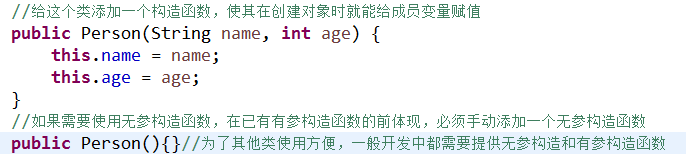


**默认构造函数**：

在Java中，所有类都有构造函数，使用new关键字创建对象时都会调用构造函数；如果类中没有写，则编译器会自动添加一个构造函数，这个构造函数就是默认构造函数。构造函数没有参数；如果类中手动书写了构造函数，就不会有默认构造函数；







### 对象内存图解

对象在内存中的位置：

对象由new关键字创建，如同数组，实体存在于堆内存中

任何事物均可以定义成类，创建对象，属于引用类型

而对象的引用变量是一个普通变量。存储的值是该对象堆内存中的地址。

**基本类型赋值代码解释**

变量名 具体数值

int p = 100;

变量数据类型 赋值运算符

**基本类型赋值对应内存图**

int p = 100

栈内存

堆内存

**引用类型赋值代码解释**

变量名 创建对象固定关键字

(0x2254)

Pesron p = new Person(); 具体数值为内存地址

变量数据类型 赋值运算符 创建对象目前固定写法 具体数值

**引用类型赋值对应内存图**

0x2254

new Person()

Person p = 0x2254

栈内存

堆内存

String name

int age

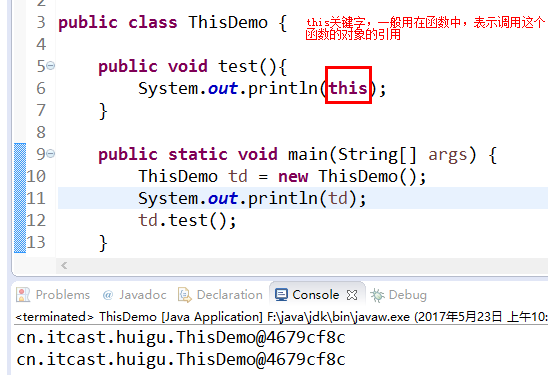
null(张三)

0(34)

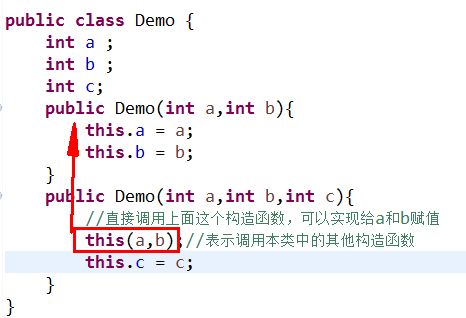
## this关键字回顾

this关键字，一般出现在不带static关键字修饰的函数里面，表示对调用这个函数的对象的引用，也就是哪个对象调用这个函数，函数中的this就指向那个对象；

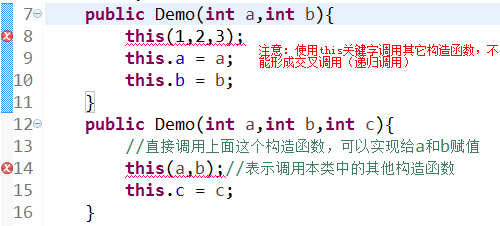
一般用来区分成员变量和局部变量；



this还有一个作用，可以在**一个构造函数中调用本类的其它构造函数**；



注意：当使用this关键字调用其它构造函数时，必须放在构造函数的第一行；



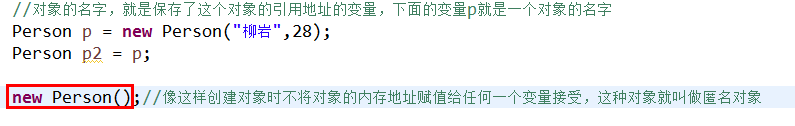
## 匿名对象

### 匿名对象的概念

匿名对象就是没有名字的对象；

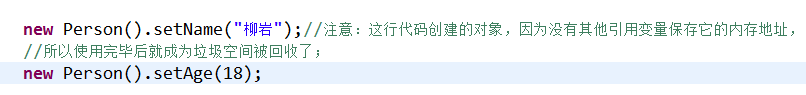
对象的名字就是保存了这个对象的引用地址的变量；

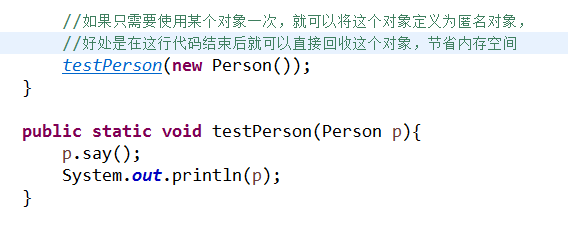
所以匿名对象，就是指**创建对象时，只有创建对象的语句，却没有把对象地址值赋值给某个变量。**



### 匿名对象特点





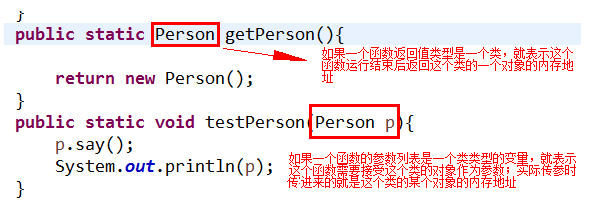


1. 匿名对象必须创建的时候就直接使用；
2. 匿名对象只能使用一次；
3. 匿名对象可以作为方法的返回值或者给方法传参；

## 类名作为参数类型和返回值类型

函数的参数列表，里面其实都是变量；如果参数列表的数据类型是一个类名，表示接受的参数是这个类的对象；

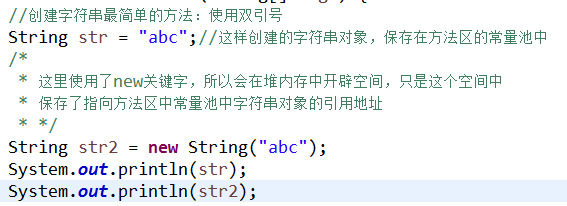
函数的返回值类型，表示这个函数要返回的数据的类型；如果一个函数的返回值类型是一个类名，表示这个函数运行结束后要返回这个类的对象；



## String回顾

### String类的作用

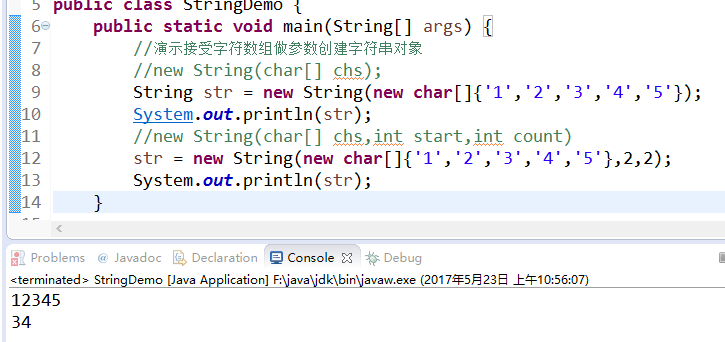
String类是专门用来描述字符串这种数据的类，在String类中，定义了大量操作字符串的函数；



### String类的常用方法回顾

构造方法：

接受字符数组做参数；



判断方法：

1. equals 判断两个字符串是否全等  
         2. equalsIgnoreCase 判断两个字符串是否全等,忽略大小写  
         3. startsWith 判断字符串开头  
         4. endsWith   判断字符串结尾

5. contains   是否包含否字段

获取方法：

         1. charAt   传递索引,获取字符 返回 char  
         2. indexOf  传递字符,获取索引 返回int(若无返回-1)  
         3. substring 获取字符串一部分 返回String  
         4. length   字符串长度 返回int转换方法：

其它方法：

 1. toCharArray() 字符串变成字符数组,返回char[]  
          2. toUpperCase() 转成大写 返回String  
          3. toLowerCase() 转成小写 返回String  
          4. split() 切割字符串 返回String[]  
          5. trim() 去掉字符串两端空格 返回String  
          6. replace 替换已有字符

## 集合技术回顾

### 什么是集合

集合是一种容器，可以保存不同类型的数据，而且集合的容量不固定；

### 为什么使用集合

开发中总会遇到数据需要保存的情况；如果是单个数据，可以使用变量保存；

如果是多个数据，还使用变量就很不方便，所以就需要使用容器保存；

我们基础班学习过的容器有三种：

**数组容器**：

1. 可以保存基本类型数据和引用类型数据；
2. 数组一旦创建，容量就不能改变；
3. 一个数组只能保存一种类型的数据；

**字符串缓冲区**：

1. 只能保存字符数据；不管添加什么数据进去，都会转换为字符数据保存；
2. 容量不固定，可以随着使用自动扩大
3. 主要用在需要对字符串频繁修改的场合

**集合**：

1. 只能保存引用类型的数据
2. 可以保存不同类型的数据
3. 容量不固定，可以随着使用自动扩大

所以当我们需要保存多个对象，并且对象的数量不确定，或者数据类型不一致时，就需要使用集合容器；

### 集合容器使用演示

需求：使用集合容器保存下面的书名，并遍历输出所有小说：

“神雕侠侣”，“小李飞刀”，“天龙八部”，“白发魔女”，“笑傲江湖”，“寻秦记”；

/\*

\* 需求：使用集合容器保存下面的书名，并遍历输出所有小说

\* “神雕侠侣”，“小李飞刀”，“天龙八部”，“白发魔女”，“笑傲江湖”，“寻秦记”；

\*/

**public** **static** **void** main(String[] args) {

//创建一个集合容器对象

ArrayList<String> list = **new** ArrayList<>();

//添加数据

list.add("神雕侠侣");

list.add("小李飞刀");

list.add("天龙八部");

list.add("白发魔女");

list.add("笑傲江湖");

list.add("寻秦记");

System.***out***.println(list);

//遍历集合

**for** (**int** i = 0; i < list.size(); i++) {

//获取i索引的数据,并且输出

String str = list.get(i);

System.***out***.println(str);

}

}

## IO回顾

### IO的概念及作用

IO技术，I指的是Input，输入的意思，表示将数据从计算机内存之外读进计算机内存；

O指的是Output，输出的意思，表示将数据从计算机内存写出到其他地方；

IO技术，就是用来实现数据在计算机内存里面和计算机内存之外进行转移的；

### IO使用演示

需求：读取一个txt文件的内容，并写出到一个新的txt文件中；

**public** **class** IODemo {

//需求：读取一个txt文件的内容，并写出到一个新的txt文件中；

**public** **static** **void** main(String[] args) **throws** IOException {

//读文件中的数据，需要使用FileReader这个类的功能；

//1、创建FileReader类的对象，关联到要读的文件

FileReader fr = **new** FileReader("1.txt");

//2、调用方法读数据

**char**[] cbuf = **new** **char**[1024];//创建一个字符数组，用来装从文件中读到的字符数据

**int** len = fr.read(cbuf);//调用对戏那个的读方法读数据，返回的结果是这一次读到的数据的个数

String str = **new** String(cbuf,0,len);

System.***out***.println(str);

//3、关流，释放资源

fr.close();

//创建一个FileWriter类的对象，用来向新的文件中写出数据

FileWriter fw = **new** FileWriter("2.txt");

//调用函数写数据

// fw.write(str);

fw.write(cbuf,0,len);

//关流，释放资源

fw.close();

}

}

# 继承

## 继承的引入

**案例 : 写一个学生类, 教师类, 校长类.**

****

1. "学生类:"
2. 属性:
3. 1. 姓名
4. 2. 年龄
5. 3. 性别
6. 4. 学号
7. "教师类:"
8. 属性:
9. 1. 姓名
10. 2. 年龄
11. 3. 性别
12. 4. 教学领域
13. "校长类:"
14. 属性:
15. 1. 姓名
16. 2. 年龄
17. 3. 性别
18. 4. 工作年限

**代码实现 :**

1. class ExtendsDemo
2. {
3. public static void main(String[] args)
4. {
5. System.out.println("Hello World!");
6. }
7. }
8. class Student
9. {
10. String name; // 姓名
11. int age; // 年龄
12. char gender; // 性别
13. int stuNumber; // 学号
14. }
15. class Teacher
16. {
17. String name; // 姓名
18. int age; // 年龄
19. char gender; // 性别
20. String field; // 领域
21. }
22. class SchoolMaster
23. {
24. String name; // 姓名
25. int age; // 年龄
26. char gender; // 性别
27. int workingYears; // 工作年限
28. }

**发现了问题 : 有很多重复的代码.  如何解决该问题 ???**

## 继承的好处与限制

**继承 : 这是一个 "拼爹" 的时代** 儿子拥有父亲的财产. (举例 : 王健林 -> 王思聪)

**特点 :** 父亲的财产是父亲辛苦奋斗而来的.儿子不费吹灰之力,生下来就有了.就因为他们之间是父子关系.

    在现实生活中,继承一般是指的子女继承父辈的财产.在程序中,继承描述的是事物之间的所属关系.通过继承可以使多种事物之间形成一种关系体系.

    在 Java 中,类的继承是指在一个现有类的基础上去构建一个新的类.构建出来的新类被称作子类,现有类被称作父类.子类 `无条件` 拥有父类全部可继承的属性和方法.继承通过 "**extends** **关键字**" 实现.

    在 Java 中.类只支持单继承,不允许多重继承.也就是说一个类只能有一个直接父类.

**继承的语法 :**

1. class 儿子类名 extends 父亲类名
2. {
3. 属性列表;
4. 方法列表;
5. }
6. class A extends B
7. {
8. 属性列表;
9. 方法列表;
10. }

* **术语 :**

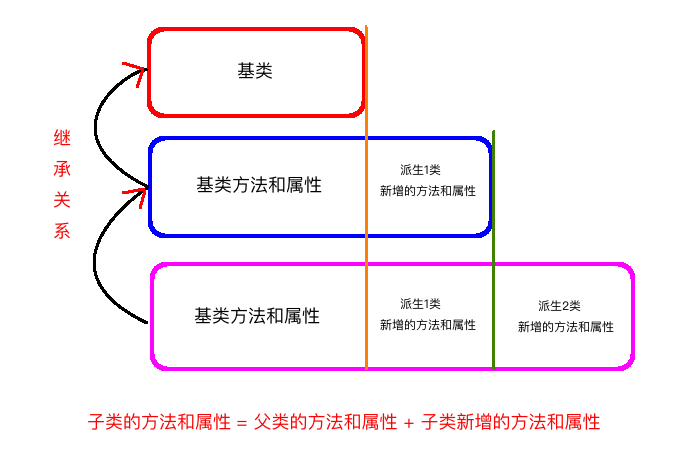
A 类是 B 类的子类, 也称为派生类。 Derived class

B 类是 A 类的父类, 也称为基类、超类。 Super class

**继承的效果** : 子类从父类继承,就代表子类 `无条件` 拥有了父类 `可继承` 的全部成员属性和方法.

**总结 :** **子类 "无条件" 拥有父类全部可继承的属性和方法. 继承是类在继承,而不是对象在继承.**

**父类和子类(派生类)之间的关系**



* 注意：
* 1.Java中的继承是单继承，也就是说一个类只能继承一个父类，不能继承多个父类.
* 2.继承的合理性: 引用 <<大话西游>> 里的一句话来描述继承 "人，是人他妈生的，妖 是妖他妈生的"

## 继承在代码中的体现 (找爸爸)

* **父类 (公共类) :** Person 类. 属性: 姓名, 年龄, 性别
* **子类 :** 学生 (学号)   老师 (领域)   校长 (工作年限)

**解决： `类中` 代码重复的问题.**

* 找出他们之间的一个共性类型.然后将需要提供复用的代码进行抽取.将这些复用代码定义到这个共性的类中. 最后是用 extends 关键字实现类的继承.

1. class ExtendsDemo
2. {
3. public static void main(String[] args)
4. {
5. Student stu = new Student();
6. stu.name = "学生";
7. stu.age = 18;
8. stu.gender = '女';
9. stu.introduce();
10. Teacher tea = new Teacher();
11. tea.name = "老师";
12. tea.age = 30;
13. tea.gender = '男';
14. tea.introduce();
15. SchoolMaster sMaster = new SchoolMaster();
16. sMaster.name = "校长";
17. sMaster.age = 40;
18. sMaster.gender = '女';
19. sMaster.introduce();
20. }
21. }
22. class Person
23. {
24. String name; // 姓名
25. int age; // 年龄
26. char gender; // 性别
27. public void introduce()
28. {
29. System.out.println("大家好,我是"+name+","+age+","+gender);
30. }
31. }
32. class Student extends Person
33. {
34. int studentNumber; // 学号
35. }
36. class Teacher extends Person
37. {
38. String field; // 领域
39. }
40. class SchoolMaster extends Person
41. {
42. int workingYears; // 工作年限
43. }

## 继承的注意点 :

**1> 继承是类在继承,而不是对象在继承.** 每个对象之间是毫无关系的.对象的属性值相互之间不会影响.

**继承的意思:** 子类从父类继承.相当于子类拥有了父类的所有成员.这个时候创建一个子类对象,子类中不仅有自己的特有成员也有父类中的共有成员.

**2> 什么时候需要使用继承 ? (is a 关系)**

当发现多个类具有相同成员的时候,那么就可以使用继承.将共同的成员定义在父类中,让子类去继承.

**3> 不要为了仅仅只是得到某个类的成员而去继承.不要为了继承而继承.**

如: Worker : Dog (因为 Dog 类中有姓名和年龄) 语法上其实任意的类之间都是可以继承的.情理,逻辑上并不是任意的类之间都可以继承的.只有满足A类是一个B类的时候,才可以拥有继承关系.让A类去继承B类.同样可以理解为只有子类是一个父类的时候,才可以继承.

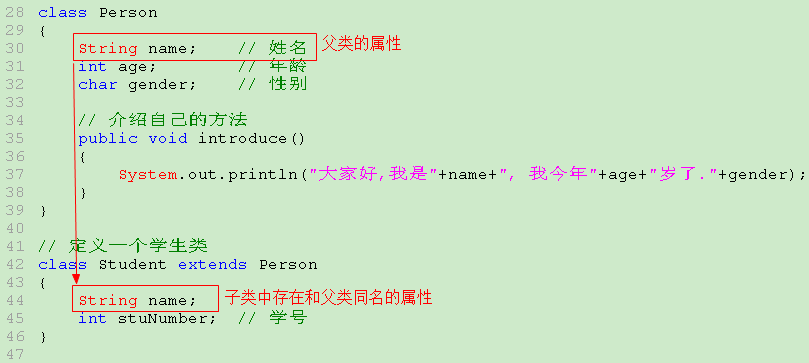
1. 学生     "人"
2. 学生     老师
3. 空调     "家用电器"
4. 车       "交通工具"
5. 车       自行车
6. .......

* **继承强调的是: 同一个 "物种".**

**4> 只要有一个成员或方法不是所有子类都拥有的,那么这个成员属性和方法就不应该定义在父类之中.** 如: 鸟类中的"fly"方法. 因为并不是所有的鸟类都会飞.只要有一个成员不是所有的子类都拥有的,那么这个成员就不应该定义在父类之中.

**5> 子类不能直接访问父类中私有的成员变量（不可继承，但是子类对象的堆内存中有）**

**6> 子类一定不要拥有和父类同名的成员变量. 因为子类可以继承父类的成员变量.**



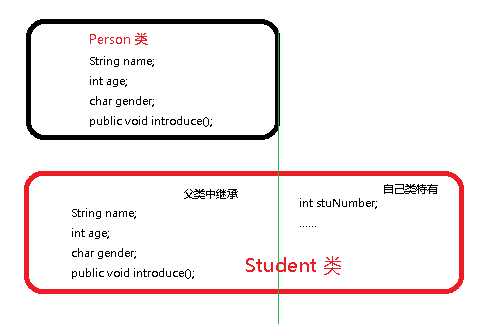


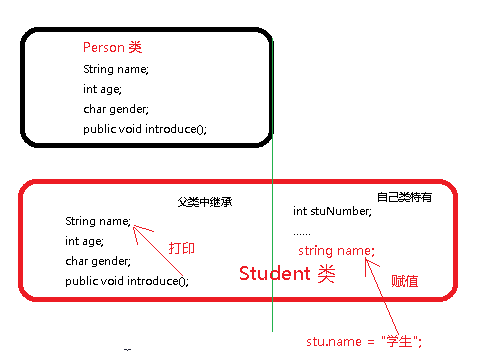
**特殊情况 :**

子父类中定义了一模一样的成员变量.

**解释 :**

* 1. 两个变量都存在于子类中.如果子类调用父类继承过来的方法访问成员变量,那么系统首先会在父类继承的成员中查找.如果找到,则直接使用.如果没有找到,才会在子类中寻找.
  2. 如果子类直接调用自己类中的方法,那么系统首先在子类自己的属性中查找.如果找到则直接拿来使用,如果没有找到.那么才会在父类中进行寻找.
  3. 在子类中若要访问父类中的成员变量，可以使用关键字super来完成。super用来表示当前对象中包含的父类对象空间的引用。super今天不做具体讲解.





**注意 : 这种情况开发见不到.因为父类一旦描述完了属性,子类直接使用就可以了.不需进行重名定义.**

## 继承的练习

**案例 : 继承的代码实现 (掌握)**

* 说明 : Animal 是父类, Dog 和 Cat 都是 Animal 的子类. ArmyDog(军犬) 是 Dog 的子类.



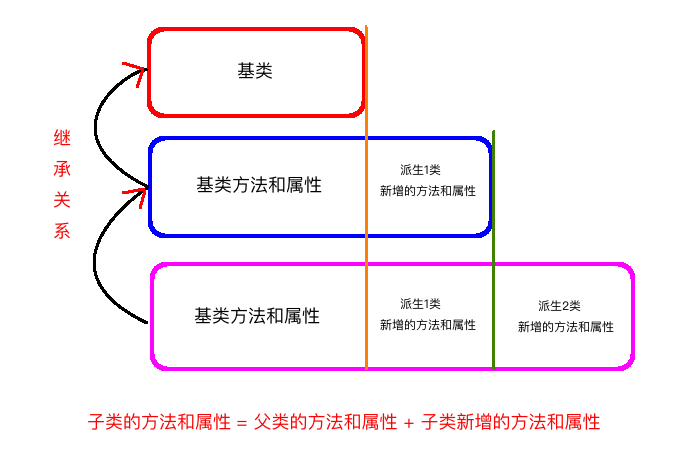


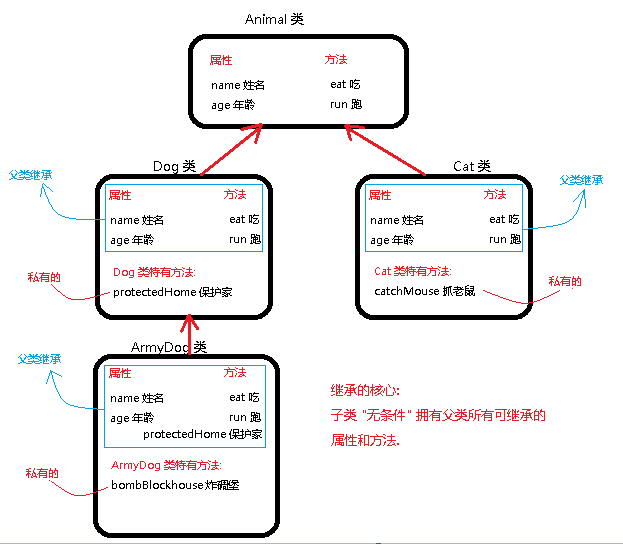


1. Animal 类
2. // 属性
3. 1. name
4. 2. age
5. // 方法
6. 1. eat
7. 2. run
8. Dog 类继承 Animal 类
9. // 特有方法
10. 1. protectedHome
11. Cat 类继承 Animal 类
12. // 特有方法
13. 1. catchMouse
14. ArmyDog 类继承 Dog 类
15. // 特有方法
16. 1. bombBlockhouse

**代码实现 :**

1. class ExtendsDemo02
2. {
3. public static void main(String[] args)
4. {
5. // 1. 创建一条狗对象
6. Dog d = new Dog();
7. d.name = "旺财";
8. d.age = 2;
9. d.protectedHome();
10. // 2. 创建一只猫对象
11. Cat c = new Cat();
12. c.name = "波斯猫";
13. c.age = 5;
14. c.catchMouse();
15. // 3. 创建一条军犬对象
16. ArmyDog armyD = new ArmyDog();
17. armyD.name = "军犬";
18. armyD.age = 10;
19. armyD.bombBlockhouse();
20. armyD.protectedHome();
21. }
22. }
23. // 定义一个 Animal 类
24. class Animal
25. {
26. // 属性
27. String name;
28. int age;
29. // 行为
30. public void eat()
31. {
32. System.out.println("动物正在吃东西 ...");
33. }
34. public void run()
35. {
36. System.out.println("动物正在跑 ...");
37. }
38. }
39. // 定义一个 Dog 类
40. class Dog extends Animal
41. {
42. // Dog 类特有的方法
43. public void protectedHome()
44. {
45. System.out.println(age+"岁的"+name+"正在保护家...");
46. }
47. }
48. // 定义一个 Cat 类
49. class Cat extends Animal
50. {
51. // Cat 类特有的方法
52. public void catchMouse()
53. {
54. System.out.println(age+"岁的"+name+"正在抓老鼠...");
55. }
56. }
57. // 定义一个 ArmyDog 类
58. class ArmyDog extends Dog
59. {
60. // ArmyDog 类特有的方法
61. public void bombBlockhouse()
62. {
63. System.out.println(age+"岁的"+name+"正在炸碉堡...");
64. }
65. }



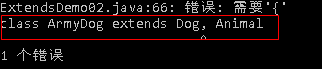


## 单一继承与多重继承

**单一继承 :**

**一个类,只能继承一个父类,也叫单一继承.**





* **Java 不支持多继承,但是支持 "多重继承".**

**多重继承 :**

1. class A Animal
2. {
3. }
4. class B extends A Dog
5. {
6. }
7. class C extends B ArmyDog
8. {
9. }

**多重继承形成了Java的继承体系!**