

笔记总链接：<http://bbs.itheima.com/thread-200600-1-1.html>

3. 面向对象

3.4 构造函数

特点：

1. 函数名与类名相同。
2. 不用定义返回值类型。
3. 没有具体的返回值。

P.S.

在构造函数前面加上返回值得只是一般函数了。

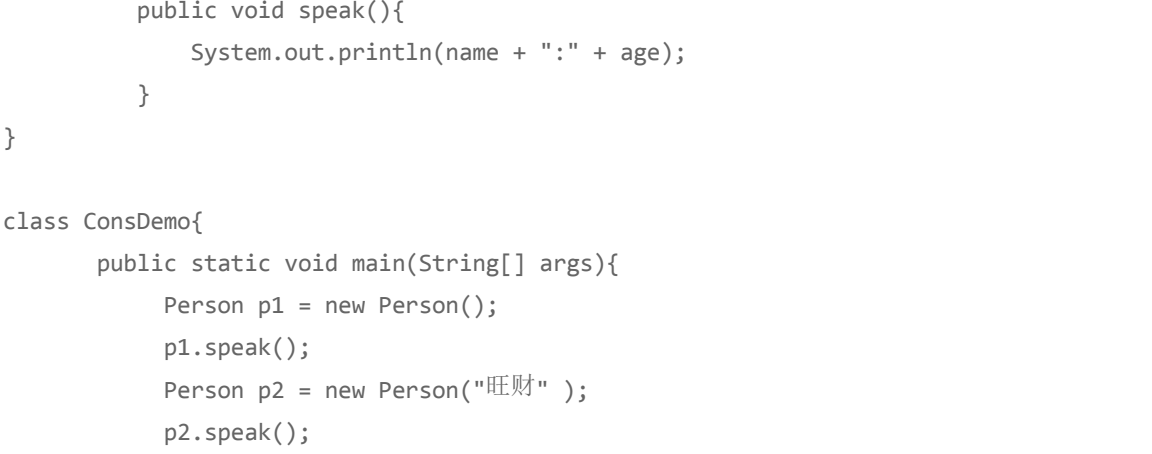
作用：给对象进行初始化。

示例：

```
01. class Person{
02.     private String name ;
03.     private int age ;
04.
05.     //定义一个Person类的构造函数
06.     //构造函数，而且是空参数的
07.     Person(){
08.         System.out.println("person run");
09.     }
10.
11.     public void speak(){
12.         System.out.println(name + ":" + age);
13.     }
14. }
15.
16. class ConsDemo{
17.     public static void main(String[] args){
18.         //构造函数：构建创建对象时调用的函数
19.         //作用：可以给对象进行初始化
20.         Person p = new Person();
21.         p.speak();
22.     }
23. }
```

复制代码

运行结果：



P.S.

1、一般函数和构造函数什么区别呢？

构造函数：对象创建时，就会调用与之对应的构造函数，对对象进行初始化。

一般函数：对象创建后，需要函数功能时才调用。

构造函数：对象创建时，会调用并且只调用一次。

一般函数：对象创建后，可以被调用多次。

2、创建对象都必须要通过构造函数初始化。

一个类如果没有定义过构造函数，那么该类中会有一个默认的空参数构造函数。

如果在类中定义了指定的构造函数，那么类中的默认构造函数就没有了。

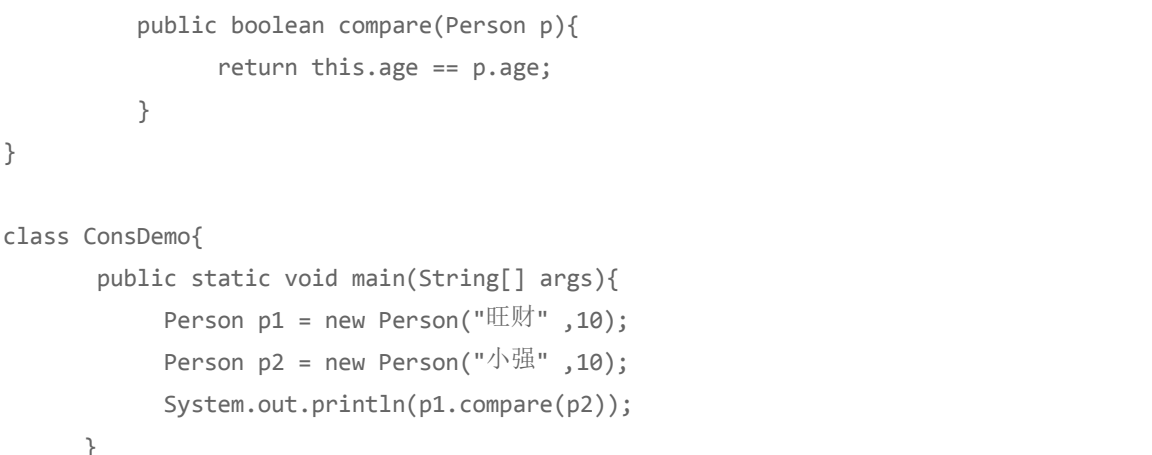
3、多个构造函数是以重载的形式存在的。

示例：

```
01. class Person{
02.     private String name ;
03.     private int age ;
04.
05.     Person(){
06.         name = "baby" ;
07.         age = 1;
08.         System.out.println("person run");
09.     }
10.
11.     //如果有孩子一出生就有名字
12.     Person(String n){
13.         name = n;
14.     }
15.
16.     //如果有孩子一出生就有名字和年龄
17.     Person(String n, int a){
18.         name = n;
19.         age = a;
20.     }
21.
22.     public void speak(){
23.         System.out.println(name + ":" + age);
24.     }
25. }
26.
27. class ConsDemo{
28.     public static void main(String[] args){
29.         Person p1 = new Person();
30.         p1.speak();
31.         Person p2 = new Person("旺财" );
32.         p2.speak();
33.         Person p3 = new Person("小强",10);
34.         p3.speak();
35.     }
36. }
```

复制代码

运行结果：



3.5 this关键字

this代表其所在函数所属对象的引用。换言之，this代本类对象的引用。

当成员变量和局部变量重名，可以用关键字this来区分，this就是所在函数所属对象的引用。

简单说，哪个对象调用了this所在的函数，this就代表那个对象。一般方法调用默认加this。

什么时候使用this关键字呢？

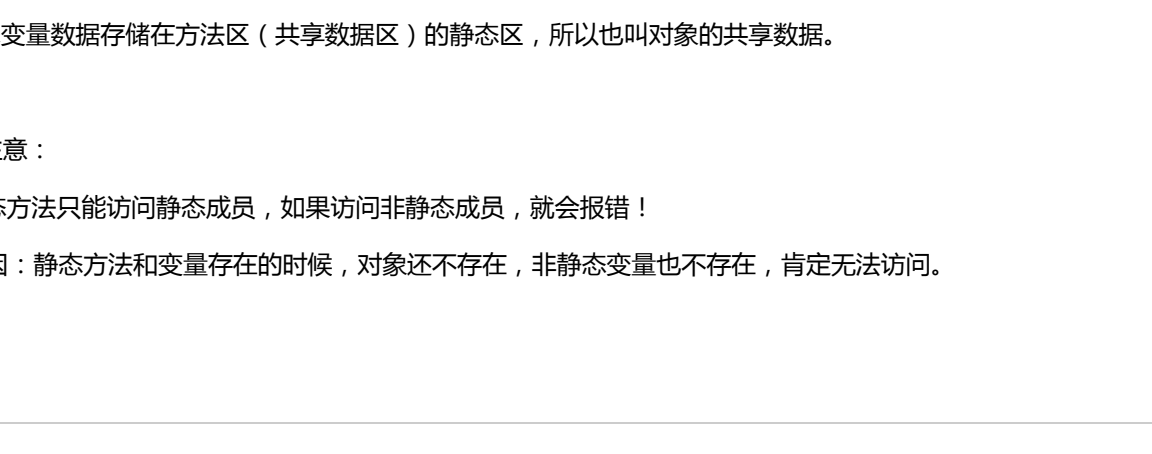
当在函数内部需要用到调用该函数的对象时，就用this。

示例：

```
01. class Person{
02.     private String name;
03.     private int age;
04.
05.     Person(String name){
06.         //通过这区分成员变量和局部变量
07.         this.name = name;
08.     }
09.
10.     Person(String name, int age){
11.         //this也可以用于在构造函数中调用其他构造函数
12.         this.name = name;
13.         this.age = age;
14.     }
15.
16.     public void speak(){
17.         System.out.println(name + ":" + age);
18.     }
19. }
20.
21. class ConsDemo{
22.     public static void main(String[] args){
23.         Person p1 = new Person("旺财" );
24.         p1.speak();
25.         Person p2 = new Person("小强",10);
26.         p2.speak();
27.     }
28. }
```

复制代码

运行结果：



P.S.

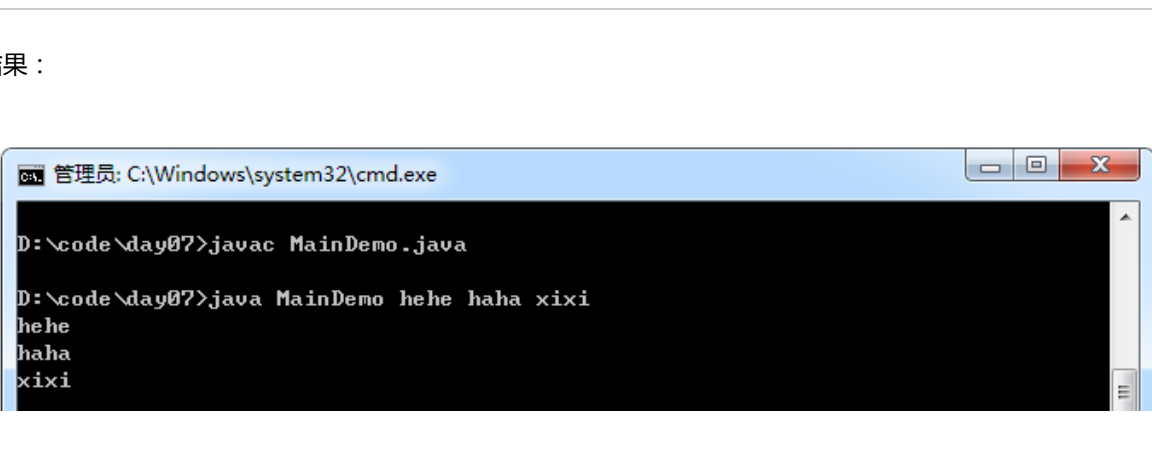
通过this在构造函数中调用其他构造函数的时候，只能定义在构造函数的第一行，因为初始化动作要先执行，否则就会报错。

示例：

```
01. class Person{
02.     private String name;
03.     private int age;
04.
05.     Person(String name, int age){
06.         this.name = name;
07.         this.age = age;
08.     }
09.
10.     /*
11.     判断是否是同龄人
12.     */
13.     public boolean compare(Person p){
14.         return this.age == p.age;
15.     }
16. }
17.
18. class ConsDemo{
19.     public static void main(String[] args){
20.         Person p1 = new Person("旺财",10);
21.         Person p2 = new Person("小强",10);
22.         System.out.println(p1.compare(p2));
23.     }
24. }
```

复制代码

运行结果：



3.6 static关键字

static关键字：用于修饰成员（成员变量和成员函数）。

被修饰后的成员具备以下特点：

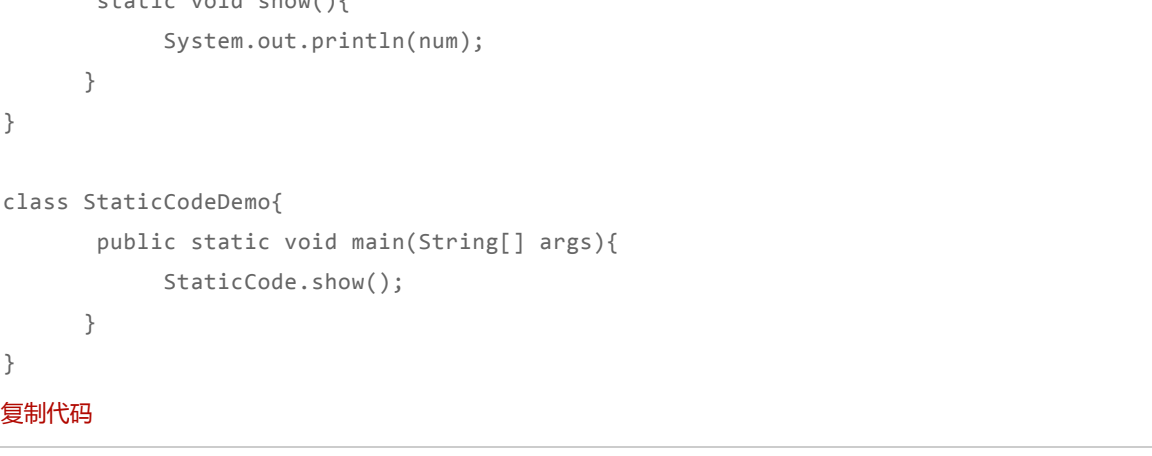
- 1、随着类的加载而加载。
- 2、优先于对象存在。
- 3、被所有对象所共享。
- 4、可以直接被类名调用。

示例：

```
01. class Person{
02.     //成员变量，实例变量
03.     String name;
04.     //静态变量，类变量
05.     //所有对象共享的属性用static修饰
06.     static String country = "CN";
07.     public void show(){
08.         System.out.println(country + ":" + name);
09.         //等效语句：System.out.println(Person.country + ":" + this.name);
10.     }
11. }
12.
13. class StaticDemo{
14.     public static void main(String[] args){
15.         Person p = new Person();
16.         System.out.println(p.country);
17.         //可以用类名直接调用
18.         System.out.println(Person.country);
19.     }
20. }
```

复制代码

运行结果：



成员变量和静态变量的区别？

1. 两个变量的生命周期不同

成员变量随着对象的创建而存在，随着对象被回收而释放。

静态变量随着类的加载而存在，随着类的消失而消失。

2. 调用方式不同

成员变量只能被对象调用。

静态变量可以被对象调用，还可以被类名调用。

3. 别名不同

成员变量也称为实例变量。

静态变量也称为类变量。

4. 数据存储位置不同

成员变量存储在堆内存的对象中，所以也叫对象的特有数据。

静态变量数据存储在方法区（共享数据区）的静态区，所以也叫对象的共享数据。

使用注意：

1. 静态方法只能访问静态成员，如果访问非静态成员，就会报错！

原因：静态方法和变量存在的时候，对象还不存在，非静态变量也不存在，肯定无法访问。

示例：

```
01. class Person{
02.     String name;
03.     static String country = "CN";
04.     //静态方法
05.     public static void show(){
06.         System.out.println(country + ":" + name);
07.     }
08. }
09.
10. class ConsDemo{
11.     public static void main(String[] args){
12.         Person p1 = new Person();
13.         Person p2 = new Person("小强",10);
14.         System.out.println(p1.country);
15.     }
16. }
```

复制代码

运行结果：



静态代码块

随着类的加载而执行，而且只执行一次。

作用：用于给类进行初始化。

示例：

```
01. class Person{
02.     private String name;
03.
04.     //构造代码块，可以给所有对象进行初始化的
05.     {
06.         System.out.println("person run");
07.     }
08.
09.     //是给对应的对象进行针对性的初始化
10.     Person(){
11.         name = "baby";
12.     }
13.
14.     Person(String name){
15.         this.name = name;
16.     }
17.
18.     public void speak(){
19.         System.out.println("name:" + name);
20.     }
21. }
22.
23. class StaticCodeDemo{
24.     public static void main(String[] args){
25.         Person p1 = new Person();
26.         p1.speak();
27.         Person p2 = new Person("旺财");
28.         p2.speak();
29.     }
30. }
```

复制代码

运行结果：

~END~

~爱上海，爱黑马~

