

非贷款，0元入学，不1W就业不给1分钱学费，我们已干四年了！

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

### 3. 面向对象

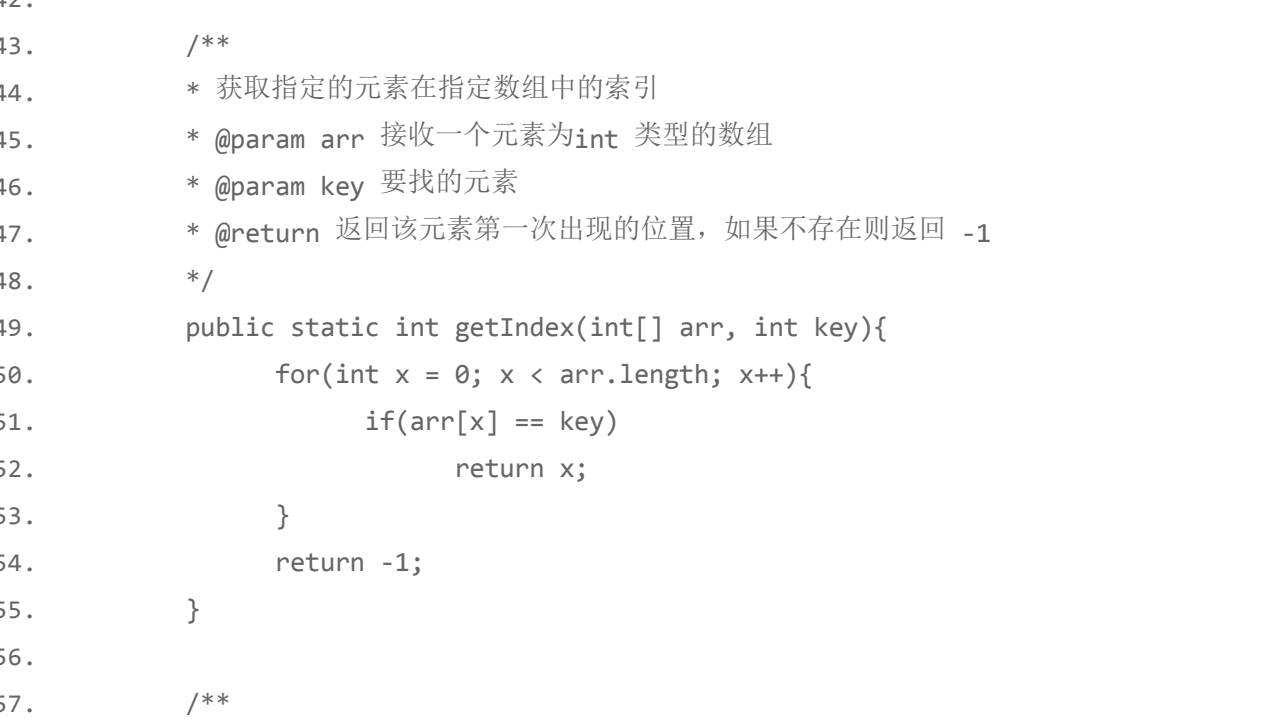
#### 3.7 数组工具类

示例：

```
01. public class ArrayTool{
02.     //该类中的方法都是静态的，所以该类是不需要创建对象的
03.     //为了保证不让人创建该类对象，可以将构造函数私有化
04.     private ArrayTool(){
05.
06.     }
07.     //获取整型数组的最大值
08.     int maxIndex = 0;
09.     for(int x = 1; x < arr.length; x++){
10.         if(arr[x] > arr[maxIndex]){
11.             maxIndex = x;
12.         }
13.     }
14.     return arr[maxIndex];
15. }
16. //对数组进行选择排序
17. public static void selectSort(int[] arr){
18.     for(int x = 0; x < arr.length -1; x++){
19.         for(int y = x + 1; y < arr.length; y++){
20.             if(arr[x] > arr[y]){
21.                 swap(arr,x,y);
22.             }
23.         }
24.     }
25. }
26. //用于给数组进行元素的位置互换。
27. private static void swap(int[] arr, int a,int b){
28.     int temp = arr[a];
29.     arr[a] = arr[b];
30.     arr[b] = temp;
31. }
32.
33. //获取指定的元素在指定数组中的索引
34. public static int getIndex(int[] arr, int key){
35.     for(int x = 0; x < arr.length; x++){
36.         if(arr[x] == key)
37.             return x;
38.     }
39.     return -1;
40. }
41.
42. //将int 数组转换成字符串，格式是：[e1,e2,...]
43. public static String arrayToString(int[] arr){
44.     String str = "[";
45.
46.     for(int x = 0; x < arr.length; x++){
47.         if(x != arr.length - 1)
48.             str = str + arr[x] + ",";
49.         else
50.             str = str + arr[x] + "]";
51.     }
52.     return str;
53. }
54. }
55.
56. class ArrayToolDemo{
57.     //保证程序的独立运行
58.     public static void main(String[] args){
59.         int[] arr = {4,8,2,9,7,2,6};
60.
61.         int max = ArrayTool.getMax(arr);
62.         System.out.println("max = " + max);
63.         int index = ArrayTool.getIndex(arr,10);
64.         System.out.println("index = " + index);
65.     }
66. }
```

复制代码

运行结果：



```
D:\code\day08>javac ArrayTool.java
D:\code\day08>java ArrayToolDemo
max = 9
index = -1
```

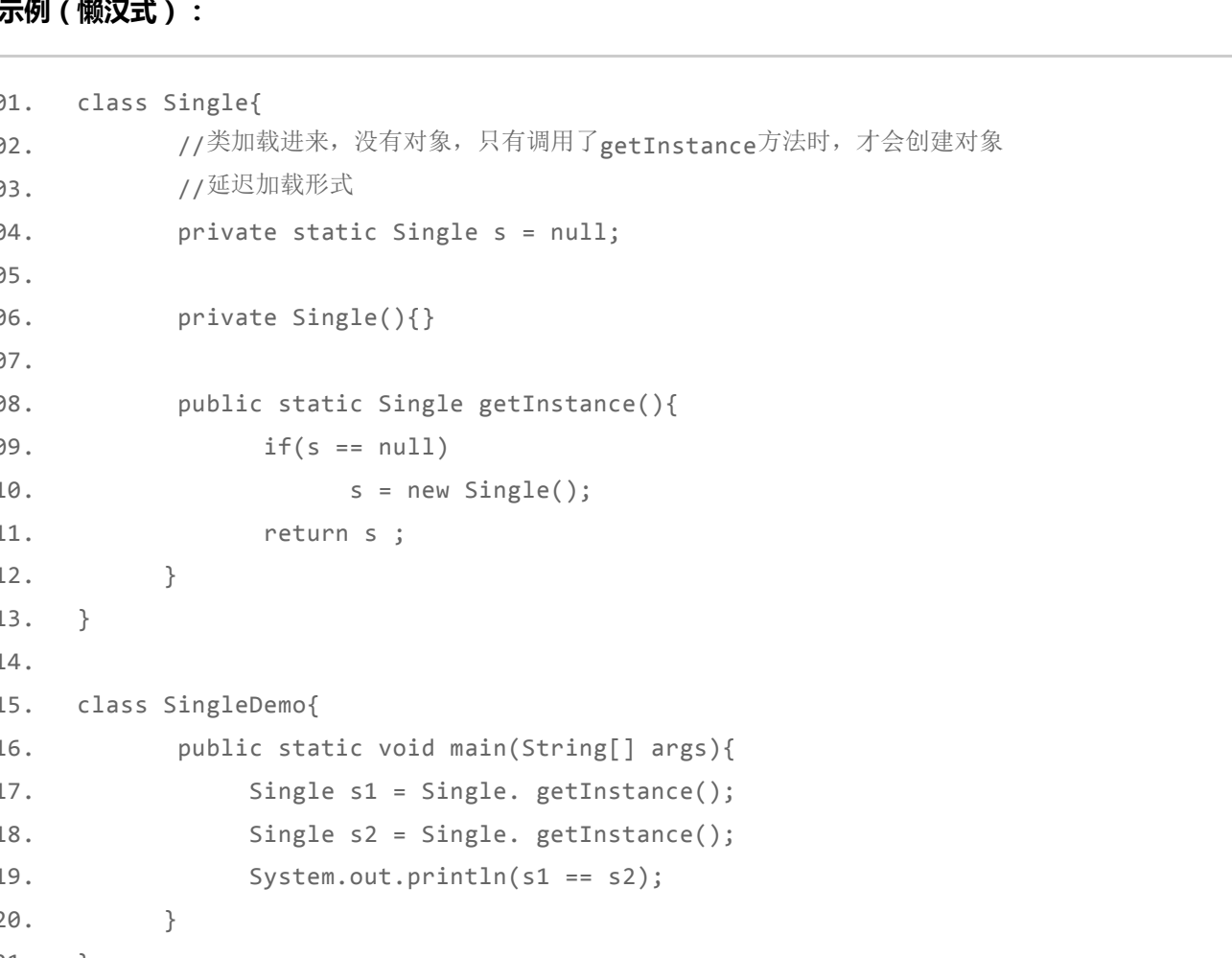
#### 3.10 文档注释

示例：

```
01. /**
02.  * 建立一个用于操作数组的工具类，其中包含着常见的对数组操作的函数，如：最值，排序等。
03.  * @author 张三
04.  * @version v1.0
05.  */
06. public class ArrayTool{
07.     private ArrayTool(){
08.
09.     }
10.     /**
11.     * 获取整型数组的最大值
12.     * @param arr 接收一个元素为int 类型的数组
13.     * @Return 该数组的最大的元素值
14.     */
15.     public static int getMax(int[] arr){
16.         int maxIndex = 0;
17.         for(int x = 1; x < arr.length; x++){
18.             if(arr[x] > arr[maxIndex]){
19.                 maxIndex = x;
20.             }
21.         }
22.         return arr[maxIndex];
23.     }
24.     /**
25.     * 对数组进行选择排序
26.     * @param arr 接收一个元素为int 的数组
27.     */
28.     public static void selectSort(int[] arr){
29.         for(int x = 0; x < arr.length -1; x++){
30.             for(int y = x + 1; y < arr.length; y++){
31.                 if(arr[x] > arr[y]){
32.                     swap(arr,x,y);
33.                 }
34.             }
35.         }
36.     }
37.     //用于给数组进行元素的位置互换。
38.     private static void swap(int[] arr, int a,int b){
39.         int temp = arr[a];
40.         arr[a] = arr[b];
41.         arr[b] = temp;
42.     }
43.     /**
44.     * 获取指定的元素在指定数组中的索引
45.     * @param arr 接收一个元素为int 类型的数组
46.     * @param key 要找回的元素
47.     * @Return 返回该元素第一次出现的位置，如果不存在则返回 -1
48.     */
49.     public static int getIndex(int[] arr, int key){
50.         for(int x = 0; x < arr.length; x++){
51.             if(arr[x] == key)
52.                 return x;
53.         }
54.         return -1;
55.     }
56.     /**
57.     * 将int数组转换成字符串，格式是：[e1,e2,...]
58.     * @param arr 接收一个元素为int类型的数组
59.     * @return 返回该数组的字符串表现形式
60.     */
61.     public static String arrayToString(int[] arr){
62.         String str = "[";
63.
64.         for(int x = 0; x < arr.length; x++){
65.             if(x != arr.length - 1)
66.                 str = str + arr[x] + ",";
67.             else
68.                 str = str + arr[x] + "]";
69.         }
70.         return str;
71.     }
72. }
73. }
```

复制代码

运行结果：

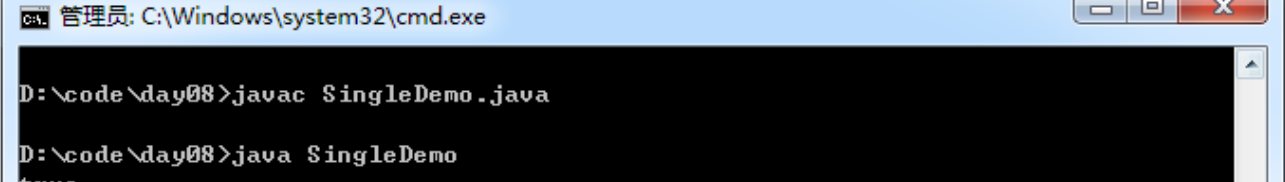


```
D:\code\day08>javacod -d myhelp -author -version ArrayTool.java
正在创建目标目录 "myhelp"
正在从源文件 ArrayTool.java...
正在构建 javadoc 信息...
Doclet 版本 1.6.0_21
正在构建所有软件包和类的树...
正在生成 myhelp\overview-tree.html...
正在生成 myhelp\package-frame.html...
正在生成 myhelp\package-summary.html...
正在生成 myhelp\package-tree.html...
正在生成 myhelp\constant-values.html...
正在构建所有软件包和类的索引...
正在生成 myhelp\overview-tree.html...
正在生成 myhelp\index-all.html...
正在生成 myhelp\deprecated-list.html...
正在生成所有类的索引...
正在生成 myhelp\allclasses-frame.html...
正在生成 myhelp\allclasses-noframe.html...
正在生成 myhelp\index.html...
正在生成 myhelp\help-doc.html...
正在生成 myhelp\stylesheet.css...
```



文件资源管理器显示 myhelp 文件夹内容：

名称	修改日期	类型
resources	2015/6/4 20:29	文件夹
allclasses-frame.html	2015/6/4 20:29	HTML 文档
allclasses-noframe.html	2015/6/4 20:29	HTML 文档
ArrayTool.html	2015/6/4 20:29	HTML 文档
constant-values.html	2015/6/4 20:29	HTML 文档
deprecated-list.html	2015/6/4 20:29	HTML 文档
help-doc.html	2015/6/4 20:29	HTML 文档
index.html	2015/6/4 20:29	HTML 文档
index-all.html	2015/6/4 20:29	HTML 文档
overview-tree.html	2015/6/4 20:29	HTML 文档



浏览器显示 ArrayTool 类文档：

```
java.lang.Object
└─ ArrayTool

public class ArrayTool
    extends java.lang.Object

建立一个用于操作数组的工具类，其中包含着常见的对数组操作的函数，如：最值，排序等

版本：
    .. v1.0
```

P.S.

1. 如果想把一个类进行文档化，该类必须是public的。
2. 私有的方法在文档中不会体现，例如ArrayTool类中的swap方法。

#### 3.11 单例设计模式

设计模式：对问题行之有效的解决方式，其实，它是一种思想。

单例设计模式解决的问题：就是可以保证一个类在内存中的对象唯一性。

比如多个程序使用同一个配置信息对象时，就需要保证该对象的唯一性。

如何保证对象唯一性呢？

- 1、不允许其他程序用new创建该类对象。
- 2、在该类创建一个本类实例。
- 3、对外提供一个方法让其他程序可以获取该对象。

步骤：

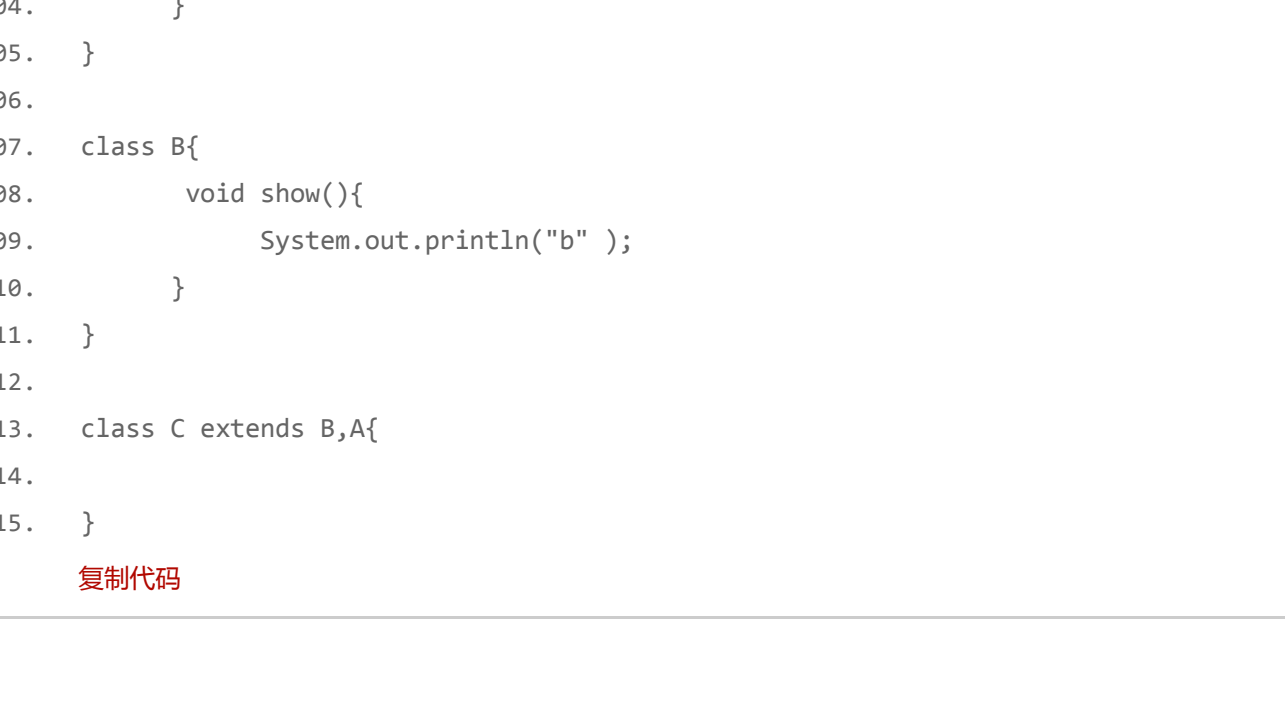
- 1、私有化该类构造函数。
- 2、通过new在本类中创建一个本类对象。
- 3、定义一个公有的方法，将创建的对象返回。

示例（懒汉式）：

```
01. class Single{
02.     //类已加载，对象就已经存在了
03.     private static Single s = new Single();
04.
05.     private Single(){
06.
07.     }
08.     public static Single getInstance(){
09.         return s ;
10.     }
11. }
12.
13. class SingleDemo{
14.     public static void main(String[] args){
15.         Single s1 = Single.getInstance();
16.         Single s2 = Single.getInstance();
17.         System.out.println(s1 == s2);
18.     }
19. }
```

复制代码

运行结果：



```
D:\code\day08>javac SingleDemo.java
D:\code\day08>java SingleDemo
true
```

P.S.

之所以不用Single.s的方式获取Single对象，而采用getInstance获取是因为在getInstance方法中我们可以做一些判断来决定是否返回Single的对象，也就是实现了对单例对象的可控。所以，给Single的构造方法加上了private限制，禁止使用者直接采用Single.s的方式获取。

示例（饿汉式）：

```
01. class Single{
02.     //类加载进来，没有对象，只有调用了getInstance方法时，才会创建对象
03.     //延迟加载形式
04.     private static Single s = null;
05.
06.     private Single(){
07.
08.     }
09.     public static Single getInstance(){
10.         if(s == null)
11.             s = new Single();
12.         return s ;
13.     }
14. }
15.
16. class SingleDemo{
17.     public static void main(String[] args){
18.         Single s1 = Single.getInstance();
19.         Single s2 = Single.getInstance();
20.         System.out.println(s1 == s2);
21.     }
22. }
```

复制代码

运行结果：



```
D:\code\day08>javac ExtendDemo.java
D:\code\day08>java ExtendDemo
student study...20
worker work...30
```

好处：

继承的出现提高了代码的复用性。

继承的出现让类与类之间产生了关系，提供了多态的前提。

#### 4.2 继承的特点

Java只支持单继承，不支持多继承。

一个类只能有一个父类，不可以有多个父类。

原因：

因为多继承容易出现問題，两个父类中有相同的方法，子类到底要执行哪一个是不确定的。

示例：

```
01. class A{
02.     void show(){
03.         System.out.println("a") ;
04.     }
05. }
06.
07. class B{
08.     void show(){
09.         System.out.println("b" );
10.     }
11. }
12.
13. class C extends B,A{
14.
15. }
```

复制代码

那么创建类C的对象，调用show方法就不知道调用类A中的show方法还是类B中的show方法。所以java不支持多继承，但将这种机制换成了另一个安全的方式来体现，也就是多实现（后面会详细说明）。

Java支持多层继承(继承体系)：

C继承B，B继承A，就会出现继承体系。

多层继承出现的继承体系中，通常看父类中的功能，了解该体系的基本功能，建立了子类对象，即可使用该体系功能。

定义继承需要注意：

不要仅为了获取其他类中某个功能而去继承，类与类之间要有所属("is a")关系。

#### 4.3 super关键字&函数覆盖

在子类中，成员的特点体现：

- 1.成员变量
- this和super的用法很相似。
- this代表本类对象的引用。
- super代表父类的内存空间的标识。

当本类的成员和局部变量同名用this区分。

当子类中的成员变量同名用super区分父类。

示例：

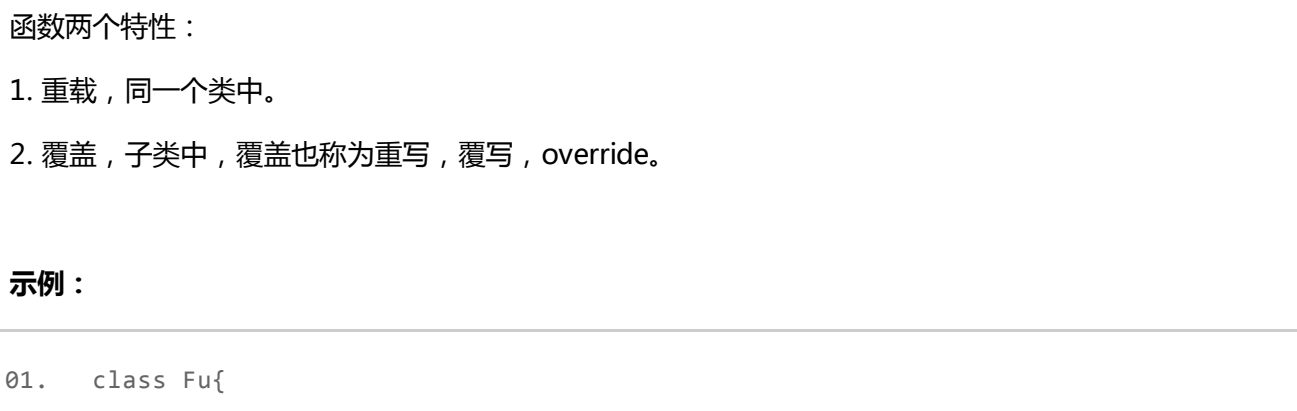
```
01. class Fu{
02.     private int num = 4;
03.
04.     public int getNum(){
05.         return num ;
06.     }
07. }
08.
09. class Zi extends Fu{
10.     private int num = 5;
11.
12.     void show(){
13.         System.out.println(this.num + "..." + super.getNum());
14.     }
15. }
```



```
14.     }
15. }
16.
17. class ExtendDemo{
18.     public static void main(String[] args){
19.         Zi z = new Zi();
20.         z.show();
21.     }
22. }
```

复制代码

运行结果：



## 2. 成员函数

当子类中出现成员函数一模一样的情况，会运行子类的函数。

这种现象，称为覆盖操作，这是函数在子类中的特性。

在子类覆盖方法中，继续使用被覆盖的方法可以通过super.函数名获取。

函数两个特性：

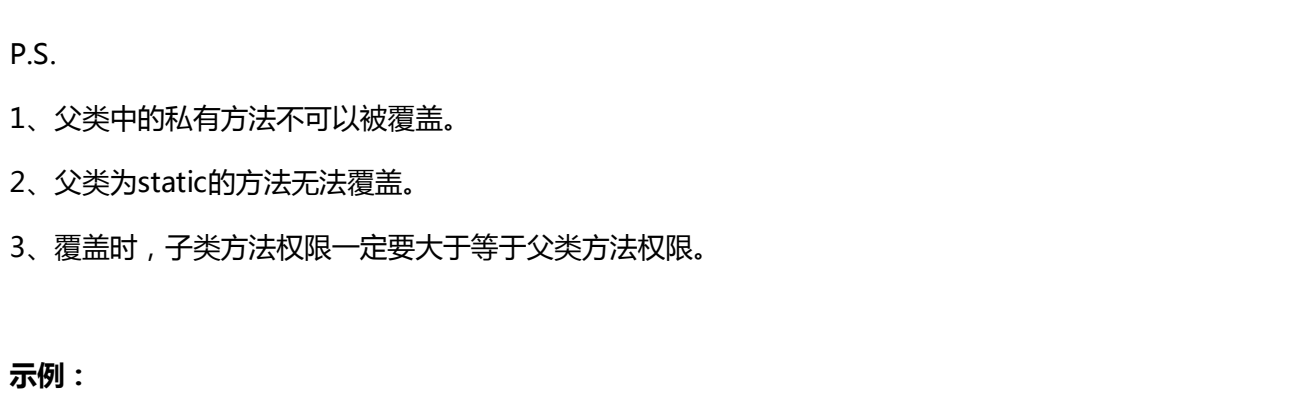
1. 重载，同一个类中。
2. 覆盖，子类中，覆盖也称为重写，覆盖， override。

示例：

```
01. class Fu{
02.     public void show(){
03.         System.out.println("fu show run" );
04.     }
05. }
06.
07. class Zi extends Fu{
08.     public void show(){
09.         System.out.println("zi show run" );
10.     }
11. }
12.
13. class ExtendDemo{
14.     public static void main(String[] args){
15.         Zi z = new Zi();
16.         z.show();
17.     }
18. }
```

复制代码

运行结果：



什么时候使用覆盖操作？

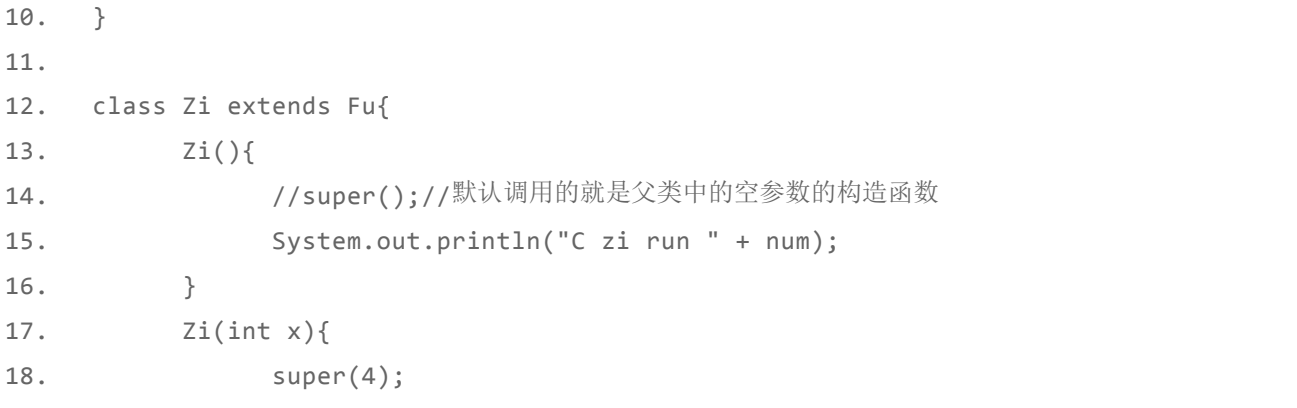
当子类需要父类的功能，而功能主体子类有自己特有内容时，可以复写父类中的方法，这样，即沿袭了父类的功能，又定义了子类特有的内容。

示例：

```
01. class Phone{
02.     void call(){}
03.     void show(){
04.         System.out.println("number" );
05.     }
06. }
07.
08. class NewPhone extends Phone{
09.     void show(){
10.         System.out.println("name" );
11.         System.out.println("pic" );
12.         super.show();
13.     }
14. }
15.
16. class ExtendDemo{
17.     public static void main(String[] args){
18.         NewPhone p = new NewPhone();
19.         p.show();
20.     }
21. }
```

复制代码

运行结果：



P.S.

- 1、父类中的私有方法不可以被覆盖。
- 2、父类为static的方法无法覆盖。
- 3、覆盖时，子类方法权限一定要大于等于父类方法权限。

示例：

```
01. class Fu{
02.     public void show(){
03.         System.out.println("fu show run" );
04.     }
05. }
06.
07. class Zi extends Fu{
08.     private void show(){
09.         System.out.println("zi show run" );
10.     }
11. }
12.
13. class ExtendDemo{
14.     public static void main(String[] args){
15.         Zi z = new Zi();
16.         z.show();
17.     }
18. }
```

复制代码

运行结果：



## 3. 构造函数

子类中构造函数的特点：

在子类构造函数执行时，发现父类构造函数也运行了。

原因：在子类的构造函数中，第一行有一个默认的隐式语句：super()。

注意：如果使用super(4);语句调用父类的其他构造函数，那么默认的父亲构造函数将不会再被调用。

示例：

```
01. class Fu{
02.     int num ;
03.     Fu(){
04.         num = 10;
05.         System.out.println("A fu run" );
06.     }
07.     Fu(int x){
08.         System.out.println("B fu run..." + x);
09.     }
10. }
11.
12. class Zi extends Fu{
13.     Zi(){
14.         //super();//默认调用的就是父类中的空参数的构造函数
15.         System.out.println("C zi run " + num);
16.     }
17.     Zi(int x){
18.         super(4);
19.         System.out.println("D zi run " + x);
20.     }
21. }
22.
23. class ExtendDemo{
24.     public static void main(String[] args){
25.         new Zi();
26.         System.out.println("-----" );
27.         new Zi(6);
28.     }
29. }
```

复制代码

运行结果：



~END~



~爱上海，爱黑马~

