[**java提高篇（十一）-----代码块**](http://blog.csdn.net/chenssy/article/details/14486833)

2013-11-07 21:21 2608人阅读 [评论](http://blog.csdn.net/chenssy/article/details/14486833#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/chenssy/article/details/14486833#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

【JAVA开发】-----Java提高篇（36） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[-]](http://blog.csdn.net/chenssy/article/details/14486833)

1. [一 普通代码块](http://blog.csdn.net/chenssy/article/details/14486833#t0)
2. [二 静态代码块](http://blog.csdn.net/chenssy/article/details/14486833#t1)
3. [三 同步代码块](http://blog.csdn.net/chenssy/article/details/14486833#t2)
4. [四 构造代码块](http://blog.csdn.net/chenssy/article/details/14486833#t3)
5. [五 静态代码块构造代码块构造函数执行顺序](http://blog.csdn.net/chenssy/article/details/14486833#t4)

       在编程过程中我们可能会遇到如下这种形式的程序：

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. **public** **class** Test {
2. {
3. ////
4. }
5. }

       这种形式的程序段我们将其称之为代码块，所谓代码块就是用大括号({})将多行代码封装在一起，形成一个独立的数据体，用于实现特定的**[算法](http://lib.csdn.net/base/31" \o "算法与数据结构知识库" \t "_blank)**。一般来说代码块是不能单独运行的，它必须要有运行主体。在[**Java**](http://lib.csdn.net/base/17)中代码块主要分为四种：

**一、 普通代码块**

       普通代码块是我们用得最多的也是最普遍的，它就是在方法名后面用{}括起来的代码段。普通代码块是不能够单独存在的，它必须要紧跟在方法名后面。同时也必须要使用方法名调用它。

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. **public** **class** Test {
2. **public** **void** test(){
3. System.out.println("普通代码块");
4. }
5. }

**二 、 静态代码块**

       想到静态我们就会想到static，静态代码块就是用static修饰的用{}括起来的代码段，它的主要目的就是对静态属性进行初始化。

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. **public** **class** Test {
2. **static**{
3. System.out.println("静态代码块");
4. }
5. }

**三、 同步代码块**

       使用 synchronized 关键字修饰，并使用“{}”括起来的代码片段，它表示同一时间只能有一个线程进入到该方法块中，是一种多线程保护机制。

**四、 构造代码块**

       在类中直接定义没有任何修饰符、前缀、后缀的代码块即为构造代码块。我们明白一个类必须至少有一个构造函数，构造函数在生成对象时被调用。构造代码块和构造函数一样同样是在生成一个对象时被调用，那么构造代码在什么时候被调用？如何调用的呢？看如下代码：

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. **public** **class** Test {
2. /\*\*
3. \* 构造代码
4. \*/
5. {
6. System.out.println("执行构造代码块...");
7. }
9. /\*\*
10. \* 无参构造函数
11. \*/
12. **public** Test(){
13. System.out.println("执行无参构造函数...");
14. }
16. /\*\*
17. \* 有参构造函数
18. \* @param id  id
19. \*/
20. **public** Test(String id){
21. System.out.println("执行有参构造函数...");
22. }
23. }

       上面定义了一个非常简单的类，该类包含无参构造函数、有参构造函数以及构造代码块，同时在上面也提过代码块是没有独立运行的能力，他必须要有一个可以承载的载体，那么编译器会如何来处理构造代码块呢？编译器会将代码块按照他们的顺序(假如有多个代码块)插入到所有的构造函数的最前端，这样就能保证不管调用哪个构造函数都会执行所有的构造代码块。上面代码等同于如下形式：

**[java]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. **public** **class** Test {
2. /\*\*
3. \* 无参构造函数
4. \*/
5. **public** Test(){
6. System.out.println("执行构造代码块...");
7. System.out.println("执行无参构造函数...");
8. }
10. /\*\*
11. \* 有参构造函数
12. \* @param id  id
13. \*/
14. **public** Test(String id){
15. System.out.println("执行构造代码块...");
16. System.out.println("执行有参构造函数...");
17. }
19. }
21. 运行结果
23. **public** **static** **void** main(String[] args) {
24. **new** Test();
25. System.out.println("----------------");
26. **new** Test("1");
27. }
28. ------------
29. Output:
30. 执行构造代码块...
31. 执行无参构造函数...
32. ----------------
33. 执行构造代码块...
34. 执行有参构造函数...

       从上面的运行结果可以看出在new一个对象的时候总是先执行构造代码，再执行构造函数，但是有一点需要注意构造代码不是在构造函数之前运行的，它是依托构造函数执行的。正是由于构造代码块有这几个特性，所以它常用于如下场景：

**1、 初始化实例变量**

       如果一个类中存在若干个构造函数，这些构造函数都需要对实例变量进行初始化，如果我们直接在构造函数中实例化，必定会产生很多重复代码，繁琐和可读性差。这里我们可以充分利用构造代码块来实现。这是利用编译器会将构造代码块添加到每个构造函数中的特性。

**2、 初始化实例环境**

       一个对象必须在适当的场景下才能存在，如果没有适当的场景，则就需要在创建对象时创建此场景。我们可以利用构造代码块来创建此场景，尤其是该场景的创建过程较为复杂。构造代码会在构造函数之前执行。

       上面两个常用场景都充分利用构造代码块的特性，能够很好的解决在实例化对象时构造函数比较难解决的问题，利用构造代码不仅可以减少代码量，同时也是程序的可读性增强了。特别是当一个对象的创建过程比较复杂，需要实现一些复杂逻辑，这个时候如果在构造函数中实现逻辑，这是不推荐的，因为我们提倡构造函数要尽可能的简单易懂，所以我们可以使用构造代码封装这些逻辑实现部分。

**五、 静态代码块、构造代码块、构造函数执行顺序**

       从词面上我们就可以看出他们的区别。静态代码块，静态，其作用级别为类，构造代码块、构造函数，构造，其作用级别为对象。

       1、 静态代码块，它是随着类的加载而被执行，只要类被加载了就会执行，而且只会加载一次，主要用于给类进行初始化。

       2、 构造代码块，每创建一个对象时就会执行一次，且优先于构造函数，主要用于初始化不同对象共性的初始化内容和初始化实例环境。

       3、 构造函数，每创建一个对象时就会执行一次。同时构造函数是给特定对象进行初始化，而构造代码是给所有对象进行初始化，作用区域不同。

       通过上面的分析，他们三者的执行顺序应该为：静态代码块 > 构造代码块 > 构造函数。

**[html]** [view plain](http://blog.csdn.net/chenssy/article/details/14486833) [copy](http://blog.csdn.net/chenssy/article/details/14486833)

[print?](http://blog.csdn.net/chenssy/article/details/14486833)[在CODE上查看代码片](https://code.csdn.net/snippets/129624)

1. public class Test {
2. /\*\*
3. \* 静态代码块
4. \*/
5. static{
6. System.out.println("执行静态代码块...");
7. }
9. /\*\*
10. \* 构造代码块
11. \*/
12. {
13. System.out.println("执行构造代码块...");
14. }
16. /\*\*
17. \* 无参构造函数
18. \*/
19. public Test(){
20. System.out.println("执行无参构造函数...");
21. }
23. /\*\*
24. \* 有参构造函数
25. \* @param id
26. \*/
27. public Test(String id){
28. System.out.println("执行有参构造函数...");
29. }
31. public static void main(String[] args) {
32. System.out.println("----------------------");
33. new Test();
34. System.out.println("----------------------");
35. new Test("1");
36. }
37. }
38. -----------
39. Output:
40. 执行静态代码块...
41. ----------------------
42. 执行构造代码块...
43. 执行无参构造函数...
44. ----------------------
45. 执行构造代码块...
46. 执行有参构造函数...

参看文献：《编写高质量代码  改善java程序的151个建议》