JAVA基础：构造方法的初始化顺序（1） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022_JAVA_E5_9F_BA_E7_A1_80_c104_145391.htm 想像一下你正在用java写程序，并且用下面的代码初始化类 A 和 B 的对象： class A { int a = f(). int f() { return 1. } } class B extends A { int b = a. int f() { return 2. } } public class CtorDemo1 { public static void main(String args[]) { B bobj = new B(). System.out.println(bobj.b). } } 现在，好像很明显的当初始化完成后，bobj.b的值将是1。毕竟，类B中的b 的值是用类A中的a的值初始化的，而a 是用f 的值初始化的，而它的值为1，对吗？ 实际上， bobj.b 的值是2，要知道为什么需要知道对象初始化的问题。 当一个对象被创建时，初始化是以下面的顺序完成的： 1. 设置成员的值为缺省的初始值 (0, false, null) 2. 调用对象的构造方法 (但是还没有执行构造方法体) 3. 调用父类的构造方法 4. 使用初始化程序和初始块初始化成员 5. 执行构造方法体 看看在实际中是如何一步一步完成的，看看下面的例子： class A { A() { System.out.println("A.A called"). } } class B extends A { int i = f(). int j. { j = 37. System.out.println("initialization block executed"). } B() { System.out.println("B.B called"). } int f() { System.out.println("B.f called"). return 47. } } public class CtorDemo2 { public static void main(String args[]) { B bobj = new B(). } } 程序的输出是： A.A called B.f called initialization block executed B.B called B 的构造方法被调用，但是最先做的事情是隐含的调用父类的构造方法。父类必须自己负责初始化它自己的状态而不是让子类来做。 然后B对象的成员被初始化，这包含一个对B.f 的调用和包

围在{}中的初始块的执行。最后B的构造方法体被执行。 你可能会问“ 什么是对父类的构造方法的隐含调用 ”。这意味着如果你的构造方法的第一行不是下面内容之一： super(). super(args). this(). this(args). 则有下面的调用： super(). 提供给构造方法的第一行。 如果类没有构造方法呢？在这种情况下，一个缺省的构造方法(也叫"无参构造方法")由java编译器自动生成。缺省构造方法只有在类没有任何其它的构造方法时才产生。 更深入的明白这个，假设在文件A.java中有这样的代码： public class A { public static void main(String args[]) { A aref = new A(). } } 如果你想编译然后列出A.class 中的字节码，输入下面的内容： $ javac A.java $ javap -c -classpath . A 输出： Compiled from A.java public class A extends java.lang.Object { public A(). public static void main(java.lang.String[]). } Method A() 0 aload_0 1 invokespecial #1 4 return Method void main(java.lang.String[]) 0 new #2 3 dup 4 invokespecial #3 7 astore_1 8 return 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com