Java构造时成员初始化的陷阱计算机等级考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/638/2021_2022_Java_E6_9E_84_E9_80_A0_c97_638808.htm 让我们先来看两个类：Base和Derived类。注意其中的whenAmISet成员变量，和方法preProcess() 1. public class Base 2. { 3. Base() { 4. preProcess(). 5. } 6. 7. void preProcess() {} 8. } 01. public class Derived extends Base 02. { 03. public String whenAmISet = "set when declared" . 04. 05. @Override void preProcess() 06. { 07. whenAmISet = "set in preProcess()" . 08. } 09. } 如果我们构造一个子类实例，那么，whenAmISet 的值会是什么呢？ 1. public class Main 2. { 3. public static void main(String[] args) 4. { 5. Derived d = new Derived(). 6. System.out.println(d.whenAmISet). 7. } 8. } 再续继往下阅读之前，请先给自己一些时间想一下上面的这段程序的输出是什么？是的，这看起来的确相当简单，甚至不需要编译和运行上面的代码，我们也应该知道其答案，那么，你觉得你知道答案吗？你确定你的答案正确吗？ 很多人都会觉得那段程序的输出应该是“ set in preProcess() ”，这是因为当子类Derived 的构造函数被调用时，其会隐晦地调用其基类Base的构造函数（通过super()函数），于是基类Base的构造函数会调用preProcess() 函数，因为这个类的实例是Derived的，而且在子类Derived中对这个函数使用了override关键字，所以，实际上调用到的是：Derived.preProcess()，而这个方法设置了whenAmISet 成员变量的值为：“ set in preProcess() ”。 当然，上面的结论是错误的。如果你编译并运行这个程序，你会发现，程序实际输出的是“ set when declared ”。怎么为这

样呢？难道是基类Base 的preProcess() 方法被调用啦？也不是！你可以在基类的preProcess中输出点什么看看，全国计算机等级考试网，加入收藏你会发现程序运行时，Base.preProcess()并没有被调用到（不然这对于Java所有的应用程序将会是一个极具灾难性的Bug）。 虽然上面的结论是错误的，但推导过程是合理的，只是不完整，下面是整个运行的流程： 进入Derived 构造函数。 Derived 成员变量的内存被分配。 Base 构造函数被隐含调用。 Base 构造函数调用preProcess()。 Derived 的preProcess 设置whenAmISet 值为“ set in preProcess() ”。 Derived 的成员变量初始化被调用。执行Derived 构造函数体。 等一等，这怎么可能？在第6步，Derived 成员的初始化居然在 preProcess() 调用之后？是的，正是这样，我们不能让成员变量的声明和初始化变成一个原子操作，虽然在Java中我们可以把其写在一起，让其看上去像是声明和初始化一体。但这只是假象，我们的错误就在于我们把Java中的声明和初始化看成了一体。在C 的世界中，C 并不支持成员变量在声明的时候进行初始化，其需要你在构造函数中显式的初始化其成员变量的值，看起来很土，但其实C 用心良苦。 在面向对象的世界中，因为程序以对象的形式出现，导致了我们对程序执行的顺序雾里看花。所以，在面向对象的世界中，程序执行的顺序相当的重要 。 下面是对上面各个步骤的逐条解释。 进入构造函数。 为成员变量分配内存。 除非你显式地调用super()，否则Java 会在子类的构造函数最前面偷偷地插入super() 。 调用父类构造函数。 调用preProcess，因为被子类override，所以调用的是子类的。于是，初始化发生在了preProcess()之后。这是因为，Java需要

保证父类的初始化早于子类的成员初始化，否则，在子类中使用父类的成员变量就会出现问题。 正式执行子类的构造函数（当然这是一个空函数，居然我们没有声明）。 你可以查看《Java语言的规格说明书》中的 相关章节 来了解更多的Java创建对象时的细节。 最后，需要向大家推荐一本书，Joshua Bloch 和 Neal Gafter 写的 Java Puzzlers: Traps, Pitfalls, and Corner Cases，中文版《JAVA解惑》。 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