Java库谜题63：更多同样的问题 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022_Java_E5_BA_93_E8_B0_9C_c97_138634.htm 下面的程序除了是面向对象的这一点之外，与前一个非常相似。因为从前一个程序中已经吸取了教训，这个程序使用了一个通用目的的Map实现，即一个HashMap，来替代前一个程序的IdentityHashMap。那么，这个程序会打印出什么呢？ import java.util.*. public class MoreNames { private Map m = new HashMap(). public void MoreNames() { m.put("Mickey", "Mouse"). m.put("Mickey", "Mantle"). } public int size() { return m.size(). } public static void main(String args[ ]) { MoreNames moreNames = new MoreNames(). System.out.println(moreNames.size()). } } 这个程序看起来很直观，其main方法通过调用无参数的构造器创建了一个MoreNames实例。这个MoreNames实例包含一个私有的Map域（m），它被初始化成一个空的HashMap。该无参数的构造器似乎将两个映射关系放置到了映射表m中，这两个映射关系都具有相同的键（Mickey）。我们从前一个谜题已知，棒球手（Mickey Mantle）应该覆盖啮齿明星（Mickey Mouse），从而只留下一个映射关系。main方法之后在MoreNames实例上调用了size方法，它会调用映射表m上的size方法，并返回结果，我们假设其为1。这种分析还剩下一个问题：该程序打印的是0而不是1。这种分析出了什么错呢？ 问题在于MoreNames没有任何程序员声明的构造器。它拥有的只是一个返回值为void的实例方法，即MoreNames，作者可能是想让它作为构造器的。遗憾的是，返回类型（void

）的出现将想要的构造器声明变成了一个方法声明，而且该方法永远都不会被调用。因为MoreNames没有任何程序员声明的构造器，所以编译器会帮助（真的是在帮忙吗？）生成一个公共的无参数构造器，它除了初始化它所创建的域实例之外，不做任何事情。就像前面提到的，m被初始化成了一个空的HashMap。当在这个HashMap上调用size方法时，它将返回0，这正是该程序打印出来的内容。 订正该程序很简单，只需将void返回类型从MoreNames声明中移除即可，这将使它从一个实例方法声明变成一个构造器声明。通过这种修改，该程序就可以打印出我们所期望的1。 本谜题的教训是：不要因为偶然地添加了一个返回类型，而将一个构造器声明变成了一个方法声明。尽管一个方法的名字与声明它的类的名字相同是合法的，但是你千万不要这么做。更一般地讲，要遵守标准的命名习惯，它强制要求方法名必须以小写字母开头，而类名应该以大写字母开头。 对语言设计者来说，在没有任何程序员声明的构造器的情况下，自动生成一个缺省的构造器这种做法并非是一个很好的主意。如果确实生成了这样的构造器，也许应该让它们是私有的。有好几种其他的方法可以消除这个陷阱。一种方法是禁止方法名与类名相同，就像C#所作的那样，另一种是彻底消灭所有的构造器，就像Smalltalk所作的那样。