Java库谜题62：名字游戏 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_138633.htm 下面的程序将两个映射关系放置到了一个映射表中，然后打印它们的尺寸。那么，它会打印出什么呢？ import java.util.*. public class NameGame { public static void main(String args[ ]) { Map m = new IdentityHashMap(). m.put("Mickey", "Mouse"). m.put("Mickey", "Mantle"). System.out.println(m.size()). } } 对该程序的一种幼稚的分析认为，它应该打印1。该程序虽然将两个映射关系放置到了映射表中，但是它们具有相同的键（Mickey）。这是一个映射表，不是一个多重映射表，所以棒球传奇人物（Mickey Mantle）应该覆盖了啮齿类动画明星（Mickey Mouse），从而只留下一个映射关系在映射表中。 更透彻一些的分析会对这个预测产生质疑。IdentityHashMap的文档中叙述道：“ 这个类用一个散列表实现了Map接口，它在比较键时，使用的是引用等价性而不是值等价性 ” [Java-API]。换句话说，如果第二次出现的字符串字面常量“ Mickey ”被计算出来是与第一次出现的“ Mickey ”字符串不同的String实例的话，那么该程序应该打印2而不是1。如此说来，该程序到底是打印1，还是打印2，抑或是其行为会根据不同的实现而有所变化？ 如果你试着运行该程序，你就会发现，尽管我们那个幼稚的分析是有缺陷的，但是该程序正如这种分析所指出的一样，打印出来的是1。这是为什么呢？语言规范保证了字符串是内存限定的，换句话说，相等的字符串常量同时也是相同的[JLS 15.28]。这可以确保在我们的程序中第二次出现的字符串字面常量

“ Mickey ”引用到了与第一次相同的String实例上，因此尽管我们使用了一个IdentityHashMap来代替诸如HashMap这样的通用目的的Map实现，但是对程序的行为却不会产生任何影响。我们那个幼稚的分析忽略了两个细节，但是这些细节造成的影响却彼此有效地抵消了。 本谜题的一个重要教训是：不要使用IdentityHashMap，除非你需要其基于标识的语义；它不是一个通用目的的Map实现。这些语义对于实现保持拓扑结构的对象图转换（topology-preserving object graph transformations）非常有用，例如序列化和深层复制。我们得到的次要教训是字符串常量是内存限定的。正如在谜题13中所述，在任何时候，程序都应该尽量不依赖于这种行为去保证它们的操作正确。