Java类谜题46：令人混淆的构造器案例 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_E7_B1_BB_E8_B0_9C_c97_138491.htm 本谜题呈现给你了两个容易令人混淆的构造器。main方法调用了一个构造器，但是它调用的到底是哪一个呢？该程序的输出取决于这个问题的答案。那么它到底会打印出什么呢？甚至它是否是合法的呢？ public class Confusing { private Confusing(Object o) { System.out.println("Object"). } private Confusing(double[] dArray) { System.out.println("double array"). } public static void main(String[] args) { new Confusing(null). } } 传递给构造器的参数是一个空的对象引用，因此，初看起来，该程序好像应该调用参数类型为Object的重载版本，并且将打印出Object。另一方面，数组也是引用类型，因此null也可以应用于类型为double[ ]的重载版本。你由此可能会得出结论：这个调用是模棱两可的，该程序应该不能编译。如果你试着去运行该程序，就会发现这些直观感觉都是不对的：该程序打印的是double array。这种行为可能显得有悖常理，但是有一个很好的理由可以解释它。 Java的重载解析过程是以两阶段运行的。第一阶段选取所有可获得并且可应用的方法或构造器。第二阶段在第一阶段选取的方法或构造器中选取最精确的一个。如果一个方法或构造器可以接受传递给另一个方法或构造器的任何参数，那么我们就说第一个方法比第二个方法缺乏精确性[JLS 15.12.2.5]。 在我们的程序中，两个构造器都是可获得并且可应用的。构造器Confusing(Object)可以接受任何传递给Confusing(double[ ])的参数，因此Confusing(Object)相

对缺乏精确性。（每一个double数组都是一个Object，但是每一个Object并不一定是一个double数组。）因此，最精确的构造器就是Confusing(double[ ])，这也就解释了为什么程序会产生这样的输出。