Java类谜题50：不是你的类型 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_138487.htm 本谜题要测试你对Java的两个最经典的操作符：instanceof和转型的理解程度。下面的三个程序每一个都会做些什么呢？ public class Type1 { public static void main(String[] args) { String s = null. System.out.println(s instanceof String). } } public class Type2 { public static void main(String[] args) { System.out.println(new Type2() instanceof String). } } public class Type3 { public static void main(String args[]) { Type3 t3 = (Type3) new Object(). } } 第一个程序，Type1，展示了instanceof操作符应用于一个空对象引用时的行为。尽管null对于每一个引用类型来说都是其子类型，但是instanceof操作符被定义为在其左操作数为null时返回false。因此，Type1将打印false。这被证明是实践中非常有用的行为。如果instanceof告诉你一个对象引用是某个特定类型的实例，那么你就可以将其转型为该类型，并调用该类型的方法，而不用担心会抛出ClassCastException或NullPointerException异常。 第二个程序，Type2，展示了instanceof操作符在测试一个类的实例，以查看它是否是某个不相关的类的实例时所表现出来的行为。你可能会期望该程序打印出false。毕竟，Type2的实例不是String的实例，因此该测试应该失败，对吗？不，instanceof测试在编译时刻就失败了，我们只能得到下面这样的出错消息： Type2.java:3: inconvertible types found : Type2, required: java.lang.String System.out.println(new Type2() instanceof String). ^ 该程序编译失败是因为instanceof操作符有这样的要求：如

果两个操作数的类型都是类，其中一个必须是另一个的子类型[JLS 15.20.2, 15.16, 5.5]。Type2和String彼此都不是对方的子类型，所以instanceof测试将导致编译期错误。这个错误有助于让你警惕instanceof测试，它们可能并没有去做你希望它们做的事情。 第三个程序，Type3，展示了当要被转型的表达式的静态类型是转型类型的超类时，转型操作符的行为。与instanceof操作相同，如果在一个转型操作中的两种类型都是类，那么其中一个必须是另一个的子类型。尽管对我们来说，这个转型很显然会失败，但是类型系统还没有强大到能够洞悉表达式new Object()的运行期类型不可能是Type3的一个子类型。因此，该程序将在运行期抛出ClassCastException异常。这有一点违背直觉：第二个程序完全具有实际意义，但是却不能编译；而这个程序没有任何实际意义，但是却可以编译。 总之，第一个程序展示了instanceof运行期行为的一个很有用的冷僻案例。第二个程序展示了其编译期行为的一个很有用的冷僻案例。第三个程序展示了转型操作符的行为的一个冷僻案例，在此案例中，编译器并不能将你从你所做荒唐的事中搭救出来，只能靠VM在运行期来帮你绷紧这根弦。