Java异常谜题36:优柔寡断 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_BC_82_E5_B8_B8_c97_138521.htm 下面这个可怜的小程序并不能很好地做出其自己的决定。它的decision方法将返回true，但是它还返回了false。那么，它到底打印的是什么呢？甚至，它是合法的吗？ public class Indecisive { public static void main(String[] args) { System.out.println(decision()). } static boolean decision() { try { return true. } finally { return false. } } } 你可能会认为这个程序是不合法的。毕竟，decision方法不能同时返回true和false。如果你尝试一下，就会发现它编译时没有任何错误，并且它所打印的是false。为什么呢？ 原因就是在一个try-finally语句中，finally语句块总是在控制权离开try语句块时执行的[JLS 14.20.2]。无论try语句块是正常结束的，还是意外结束的，情况都是如此。一条语句或一个语句块在它抛出了一个异常，或者对某个封闭型语句执行了一个break或continue，或是象这个程序一样在方法中执行了一个return时，将发生意外结束。它们之所以被称为意外结束，是因为它们阻止程序去按顺序执行下面的语句。 当try语句块和finally语句块都意外结束时，在try语句块中引发意外结束的原因将被丢弃，而整个try-finally语句意外结束的原因将于finally语句块意外结束的原因相同。在这个程序中，在try语句块中的return语句所引发的意外结束将被丢弃，而try-finally语句意外结束是由finally语句块中的return造成的。简单地讲，程序尝试着（try）返回（return）true，但是它最终（finally）返回（return）的是false。 丢弃意外结束的原因几乎永远都不是你

想要的行为，因为意外结束的最初原因可能对程序的行为来说会显得更重要。对于那些在try语句块中执行break、continue或return语句，只是为了使其行为被finally语句块所否决掉的程序，要理解其行为是特别困难的。 总之，每一个finally语句块都应该正常结束，除非抛出的是不受检查的异常。千万不要用一个return、break、continue或throw来退出一个finally语句块，并且千万不要允许将一个受检查的异常传播到一个finally语句块之外去。 对于语言设计者，也许应该要求finally语句块在未出现不受检查的异常时必须正常结束。朝着这个目标，try-finally结构将要求finally语句块可以正常结束[JLS 14.21]。return、break或continue语句把控制权传递到finally语句块之外应该是被禁止的，任何可以引发将被检查异常传播到finally语句块之外的语句也同样应该是被禁止的。