Java更多的库谜题84：被粗暴地中断 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_E6_9B_B4_E5_A4_9A_c97_138709.htm 在下面这个程序中，一个线程试图中断自己，然后检查中断是否成功。它会打印什么呢？ public class SelfInterruption { public static void main(String[ ] args) { Thread.currentThread().interrupt(). if(Thread.interrupted()) { System.out.println("Interrupted: " Thread.interrupted()). } else{ System.out.println("Not interrupted: " Thread.interrupted()). } } } 虽然一个线程中断自己不是很常见，但这也不是没有听说过的。当一个方法捕捉到了一个InterruptedException异常，而且没有做好处理这个异常的准备时，那么这个方法通常会将该异常重新抛出（rethrow）。但是由于这是一个“ 被检查的异常 ”，所以只有在方法声明允许的情况下该方法才能够将异常重新抛出。如果不能重新抛出，该方法可以通过中断当前线程对异常“ 再构建 ”(reraise)。这种方式工作得很好，所以这个程序中的线程中断自己应该是没有任何问题的。所以，该程序应该进入if语句的第一个分支，打印出 Interrupted: true 。如果你运行该程序，你会发现并不是这样。但是它也没有打印 Not interrupted: false，它打印的是 Interrupted: false。 看起来该程序好像不能确定线程是否被中断了。当然，这种看法是毫无意义的。实际上发生的事情是，Thread.interrupted方法第一次被调用的时候返回了true，并且清除了线程的中断状态，所以在if-then-else语句的分支中第2次调用该方法的时候，返回的就是false。调用Thread.interrupted方法总是会清除当前线程的中断状态。方法的名称没有为这种行为提供任何线索

，而对于5.0版本，在相应的文档中有一句话概要地也同样具有误导性地叙述道：“ 测试当前的线程是否中断 ”[Java-API]。所以，可以理解为什么很多程序员都没有意识到Thread.interrupted方法会对线程的中断状态造成影响。Thread类有2个方法可以查询一个线程的中断状态。另外一个方法是一个名为isInterrupted的实例方法，而它不会清除线程的中断状态。如果使用这个方法重写程序，它就会打印出我们想要的结果true： public class SelfInterruption { public static void main(String[ ] args) { Thread.currentThread().interrupt(). if(Thread.currentThread().isInterrupted()) { System.out.println("Interrupted: " Thread.currentThread().isInterrupted()). }else{ System.out.println("Not interrupted: " Thread.currentThread().isInterrupted()). } } } 这个谜题的教训是：不要使用Thread.interrupted方法，除非你想要清除当前线程的中断状态。如果你只是想查询中断状态，请使用isInterrupted方法。这里给API设计者们的教训是方法的名称应该用来描述它们主要功能。根据Thread.interrupted方法的行为，它的名称应该是 clearInterruptStatus，因为相对于它对中断状态的改变，它的返回值是次要的。特别是当一个方法的名称并不完美的时候，文档是否能清楚地描述它的行为就显得非常重要了。