JAVA异常谜题39:您好，再见 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_138513.htm 下面的程序在寻常的Hello world程序中添加了一段不寻常的曲折操作。那么，它将会打印出什么呢？ public class HelloGoodbye { public static void main(String[] args) { try { System.out.println("Hello world"). System.exit(0). } finally { System.out.println("Goodbye world"). } } } 这个程序包含两个println语句：一个在try语句块中，另一个在相应的finally语句块中。try语句块执行它的println语句，并且通过调用System.exit来提前结束执行。在此时，你可能希望控制权会转交给finally语句块。然而，如果你运行该程序，就会发现它永远不会说再见：它只打印了Hello world。这是否违背了谜题36中所解释的原则呢？ 不论try语句块的执行是正常地还是意外地结束，finally语句块确实都会执行。然而在这个程序中，try语句块根本就没有结束其执行过程。System.exit方法将停止当前线程和所有其他当场死亡的线程。finally子句的出现并不能给予线程继续去执行的特殊权限。 当System.exit被调用时，虚拟机在关闭前要执行两项清理工作。首先，它执行所有的关闭挂钩操作，这些挂钩已经注册到了Runtime.addShutdownHook上。这对于释放VM之外的资源将很有帮助。务必要为那些必须在VM退出之前发生的行为关闭挂钩。下面的程序版本示范了这种技术，它可以如我们所期望地打印出Hello world和Goodbye world： public class HelloGoodbye1 { public static void main(String[] args) { System.out.println("Hello world").

Runtime.getRuntime().addShutdownHook(new Thread() { public void run() { System.out.println("Goodbye world"). } }). System.exit(0). } } VM执行在System.exit被调用时执行的第二个清理任务与终结器有关。如果System.runFinalizerOnExit或它的魔鬼双胞胎Runtime.runFinalizersOnExit被调用了，那么VM将在所有还未终结的对象上面调用终结器。这些方法很久以前就已经过时了，而且其原因也很合理。无论什么原因，永远不要调用System.runFinalizersOnExit
和Runtime.runFinalizersOnExit：它们属于Java类库中最危险的方法之一[ThreadStop]。调用这些方法导致的结果是，终结器会在那些其他线程正在并发操作的对象上面运行，从而导致不确定的行为或导致死锁。 总之，System.exit将立即停止所有的程序线程，它并不会使finally语句块得到调用，但是它在停止VM之前会执行关闭挂钩操作。当VM被关闭时，请使用关闭挂钩来终止外部资源。通过调用System.halt可以在不执行关闭挂钩的情况下停止VM，但是这个方法很少使用。