JAVA循环谜题29:循环者的新娘 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_BE_AA_E7_8E_AF_c97_138534.htm 请提供一个对i的声明，将下面的循环转变为一个无限循环： while (i != i) { } 这个循环可能比前一个还要使人感到困惑。不管在它前面作何种声明，它看起来确实应该立即终止。一个数字总是等于它自己，对吗？ 对，但是IEEE 754浮点算术保留了一个特殊的值用来表示一个不是数字的数量[IEEE 754]。这个值就是NaN（“ 不是一个数字（Not a Number）”的缩写），对于所有没有良好的数字定义的浮点计算，例如0.0/0.0，其值都是它。规范中描述道，NaN不等于任何浮点数值，包括它自身在内[JLS 15.21.1]。因此，如果i在循环开始之前被初始化为NaN，那么终止条件测试(i != i)的计算结果就是true，循环就永远不会终止。很奇怪但却是事实。 你可以用任何计算结果为NaN的浮点算术表达式来初始化i，例如： double i = 0.0 / 0.0. 同样，为了表达清晰，你可以使用标准类库提供的常量： double i = Double.NaN. NaN还有其他的惊人之处。任何浮点操作，只要它的一个或多个操作数为NaN，那么其结果为NaN。这条规则是非常合理的，但是它却具有奇怪的结果。例如，下面的程序将打印false： class Test { public static void main(String[] args) { double i = 0.0 / 0.0. System.out.println(i - i == 0). } } 这条计算NaN的规则所基于的原理是：一旦一个计算产生了NaN，它就被损坏了，没有任何更进一步的计算可以修复这样的损坏。NaN值意图使受损的计算继续执行下去，直到方便处理这种情况的地方为止。 总之，float和double类型都有一个特殊

的NaN值，用来表示不是数字的数量。对于涉及NaN值的计算，其规则很简单也很明智，但是这些规则的结果可能是违背直觉的。