JAVA循环谜题28:循环者 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_138537.htm 下面的谜题以及随后的五个谜题对你来说是扭转了局面，它们不是向你展示某些代码，然后询问你这些代码将做些什么，它们要让你去写代码，但是数量会很少。这些谜题被称为“ 循环者（looper） ”。你眼前会展示出一个循环，它看起来应该很快就终止的，而你的任务就是写一个变量声明，在将它作用于该循环之上时，使得该循环无限循环下去。例如，考虑下面的for循环： for (int i = start. i 看起来它好像应该只迭代两次，但是通过利用在谜题26中所展示的溢出行为，可以使它无限循环下去。下面的的声明就采用了这项技巧： int start = Integer.MAX_VALUE - 1. 现在该轮到你了。什么样的声明能够让下面的循环变成一个无限循环？ While (i == i 1) {} 仔细查看这个while循环，它真的好像应该立即终止。一个数字永远不会等于它自己加1，对吗？嗯，如果这个数字是无穷大的，又会怎样呢？Java强制要求使用IEEE 754浮点数算术运算[IEEE 754]，它可以让你用一个double或float来表示无穷大。正如我们在学校里面学到的，无穷大加1还是无穷大。如果i在循环开始之前被初始化为无穷大，那么终止条件测试(i == i 1)就会被计算为true，从而使循环永远都不会终止。 你可以用任何被计算为无穷大的浮点算术表达式来初始化i，例如： double i = 1.0 / 0.0. 不过，你最好是能够利用标准类库为你提供的常量： double i = Double.POSITIVE_INFINITY. 事实上，你不必将i初始化为无穷大以确保循环永远执行。任何足够大的浮点数都可以实现

这一目的，例如： double i = 1.0e40. 这样做之所以可以起作用，是因为一个浮点数值越大，它和其后继数值之间的间隔就越大。浮点数的这种分布是用固定数量的有效位来表示它们的必然结果。对一个足够大的浮点数加1不会改变它的值，因为1是不足以“ 填补它与其后继者之间的空隙 ”。 浮点数操作返回的是最接近其精确的数学结果的浮点数值。一旦毗邻的浮点数值之间的距离大于2，那么对其中的一个浮点数值加1将不会产生任何效果，因为其结果没有达到两个数值之间的一半。对于float类型，加1不会产生任何效果的最小级数是225，即33,554,432；而对于double类型，最小级数是254，大约是1.8 × 1016。 毗邻的浮点数值之间的距离被称为一个ulp，它是“ 最小单位（unit in the last place） ”的首字母缩写词。在5.0版中，引入了Math.ulp方法来计算float或double数值的ulp。 总之，用一个double或一个float数值来表示无穷大是可以的。大多数人在第一次听到这句话时，多少都会有一点吃惊，可能是因为我们无法用任何整数类型来表示无穷大的原因。第二点，将一个很小的浮点数加到一个很大的浮点数上时，将不会改变大的浮点数的值。这过于违背直觉了，因为对实际的数字来说这是不成立的。我们应该记住二进制浮点算术只是对实际算术的一种近似。 