JAVA循环谜题35:一分钟又一分钟 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_138518.htm 下面的程序在模仿一个简单的时钟。它的循环变量表示一个毫秒计数器，其计数值从0开始直至一小时中包含的毫秒数。循环体以定期的时间间隔对一个分钟计数器执行增量操作。最后，该程序将打印分钟计数器。那么它会打印出什么呢？ public class Clock { public static void main(String[] args) { int minutes = 0. for (int ms = 0. ms if (ms % 60*1000 == 0) minutes . System.out.println(minutes). } } 在这个程序中的循环是一个标准的惯用for循环。它步进毫秒计数器（ms），从0到一小时中的毫秒数，即3,600,000，包括前者但是不包括后者。循环体看起来是在每当毫秒计数器的计数值是60,000（一分钟内所包含毫秒数）的倍数时，对分钟计数器（minutes）执行增量操作。这在循环的生命周期内总共发生了3,600,000/60,000次，即60次，因此你可能期望程序打印出60，毕竟，这就是一小时所包含的分钟数。但是，该程序的运行却会告诉你另外一番景象：它打印的是60000。为什么它会如此频繁地对minutes执行了增量操作呢？ 问题在于那个布尔表达式(ms % 60*1000 == 0)。你可能会认为这个表达式等价于(ms % 60000 == 0)，但是它们并不等价。取余和乘法操作符具有相同的优先级[JLS 15.17]，因此表达式ms % 60*1000 等价于(ms % 60)*1000。如果(ms % 60)等于0的话，这个表达式就等于0，因此循环每60次迭代就对minutes执行增量操作。这使得最终的结果相差1000倍。 订正该程序的最简单的方式就是在布尔表达式中插入一对括号，以强制规定计算的正确顺序

：if (ms % (60 * 1000) == 0) minutes . 然而，有一个更好的方法可以订正该程序。用被恰当命名的常量来替代所有的魔幻数字： public class Clock { private static final int MS_PER_HOUR = 60 * 60 * 1000. private static final int MS_PER_MINUTE = 60 * 1000. public static void main(String[] args) { int minutes = 0. for (int ms = 0. ms if (ms % MS_PER_MINUTE == 0) minutes . System.out.println(minutes). } } 之所以要在最初的程序中展现表达式 ms % 60*1000，是为了诱使你去认为乘法比取余有更高的优先级。然而，编译器是忽略空格的，所以千万不要使用空格来表示分组，要使用括号。空格是靠不住的，而括号是从来不说谎的。