Java更多的库谜题81：烧焦到无法识别 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_138718.htm 下面这个程序看起来是在用一种特殊的方法做一件普通的事。那么，它会打印出什么呢？ public class Greeter{ public static void main(String[] args){ String greeting = "Hello World". for(int i = 0. i System.out.write(greeting.charAt(i)). }} 尽管这个程序有点奇怪，但是我们没有理由怀疑它会产生不正确的行为。它将“ Hello World ”写入了System.out，每次写一个字符。你可能会意识到write方法只会使用其输入参数的低位字节（lower-order byte）。所以当“ Hello World ”含有任何外来字符的时候，可能会造成一些麻烦，但这里不会：因为“ Hello World ”完全是由ASCII字符组成的。无论你是每次打印一个字符，还是一次全部打印，结果都应该是一样的：这个程序应该打印Hello World。然而，如果你运行该程序，就会发现它不会打印任何东西。那句问候语到哪里去了？难道是程序认为它并不令人愉快？ 这里的问题在于System.out是带有缓冲的。Hello World中的字符被写入了System.out的缓冲区，但是缓冲区从来都没有被刷新（flush）。大多数的程序员认为，当有输出产生的时候System.out和System.err会自动地进行刷新，这并不完全正确。这2个流都属于PrintStream类型，在5.0版[Java-API]中，有关这个类型的文档叙述道： 一个PrintStream可以被创建为自动刷新的；这意味着当一个字节数组(byte array)被写入，或者某个println方法被调用，或者一个换行字符或字节(‘ \n ’)被写入之后，PrintStream类型

的flush方法就会被自动地调用。 System.out和System.err所引用的流确实是PrintStream的能够自动刷新的变体，但是上面的文档中并没有提及write(int)方法。有关write(int)方法的文档叙述道：将指定的byte写入流。如果这个byte是一个换行字符，并且流可以自动刷新，那么flush方法将被调用[Java-API]。实际上，write(int)是唯一一个在自动刷新(automatic flushing)功能开启的情况下不刷新PrintStream的输出方法(output method)。 令人好奇的是，如果这个程序改用print(char)去替代write(int)，它就会刷新System.out并打印出Hello World。这种行为与print(char)的文档是矛盾的，因为其文档叙述道[Java-API]：打印一个字符：这个字符将根据平台缺省的字符编码方式被翻译成为一个或多个字节，并且这些字节将完全按照write(int)方法的方式被写出。 类似地，如果程序改用print(String)，它也会对流进行刷新，虽然文档中是禁止这么做的。相应的文档确实应该被修改为描述该方法的实际行为，而修改方法的行为则会破坏稳定性。 修改这个程序最简单的方法就是在循环之后加上一个对System.out.flush方法的调用。经过这样的修改之后，程序就会正常地打印出Hello World。当然，更好的办法是重写这个程序，使用我们更熟悉的System.out.println方法在控制台上产生输出。 这个谜题的教训与谜题23一样：尽可能使用熟悉的惯用法；如果你不得不使用陌生的API，请一定要参考相关的文档。这里有3条教训给API的设计者们：请让你们的方法的行为能够清晰的反映在方法名上；请清楚而详细地给出这些行为的文档；请正确地实现这些行为。 