垃圾回收器的使用方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/278/2021_2022__E5_9E_83_E5_9C_BE_E5_9B_9E_E6_c104_278558.htm 看了一些 Thinking in java 这本书后，让我对JAVA的垃圾回收器开始研究起来。 经过摸索，发现java回收器(GC，一下用GC代替回收器)并不是象一般人想像的那样定期的回收垃圾，从而让你完全不用当心内存的问题。事实是，JAVA还是存在内存溢出的时刻，只所以一般的系统和开发人员没有这么认识到，或者没有遇到，只是因为java jvm帮我们默默无闻的做了一些力所能及的处理！！其实，GC的工作原理是非常的复杂，以至于很多人没法说清楚，在这里，我就个人的理解， 总结一下： 1.GC并不是定期来回收你的垃圾内存，即是根据需要来回收。 2.GC的回收是因为：它认为你的系统已经开始内存紧张(这个就是jvm的神奇) 3.即使GC开始准备清理你的垃圾内存，但是如果该内存的引用还存在(不等于null)， 这个时候GC仍然无能为力！ 看看下面的两个例子就知道了。 例子一： public class finalizeTest{ public boolean checkout = false. public void checkIn(){ this.checkout = true. } public void finalize(){ if (checkout) { System.out.print("the erroe"). } } public static void main(String[] args) { finalizeTest test = new finalizeTest(). test.checkIn(). System.gc(). System.out.println("Hello World!"). } } 本来希望通过System.gc()命令来强制执行finalize()来处理清理事务，但是事与愿违，它没有执行。 只要main方法没执行完，永远都不会被回收。原因很简单，在main方法内 finalizeTest test = new finalizeTest(). 后的“ System.gc(). ”是要求系统去回收垃圾。系

统线程此时显然有闲暇时间，经过判断，发现test仍然引用着finalizeTest对象。所以，不会去回收test。 例子二： public class TestGC{ public void finalize(){ System.out.print("the erroe"). } public static void main(String[] args) { TestGC test = new TestGC(). test = new TestGC(). test = null. System.gc(). System.out.println("Hello World!"). } } test = null，让test失去了对new TestGC()的引用。new TestGC()对象没有任何人在引用。在你申请GC的时候，就被回收了。因为此时系统只运行你一个线程（还有其他后台辅助的）当然有空闲时间，于是立刻回收了你的。程序输出“ the erroeHello World! ”。 100Test下载频道开通，各类考试题目直接下载。详细请访问www.100test.com