实战Java多线程编程中不提倡使用的方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/227/2021_2022__E5_AE_9E_E6_88_98Java_c104_227646.htm 不提倡使用的方法是为支持向后兼容性而保留的那些方法，它们在以后的版本中可能出现，也可能不出现。Java 多线程支持在版本 1.1 和版本 1.2 中做了重大修订，stop()、suspend() 和 resume() 函数已不提倡使用。这些函数在 JVM 中可能引入微妙的错误。虽然函数名可能听起来很诱人，但请抵制诱惑不要使用它们。 调试线程化的程序 在线程化的程序中，可能发生的某些常见而讨厌的情况是死锁、活锁、内存损坏和资源耗尽。 死锁 死锁可能是多线程程序最常见的问题。当一个线程需要一个资源而另一个线程持有该资源的锁时，就会发生死锁。这种情况通常很难检测。但是，解决方案却相当好：在所有的线程中按相同的次序获取所有资源锁。例如，如果有四个资源 A、B、C 和 D 并且一个线程可能要获取四个资源中任何一个资源的锁，则请确保在获取对 B 的锁之前首先获取对 A 的锁，依此类推。如果“ 线程 1 ”希望获取对 B 和 C 的锁，而“ 线程 2 ”获取了 A、C 和 D 的锁，则这一技术可能导致阻塞，但它永远不会在这四个锁上造成死锁。 活锁 当一个线程忙于接受新任务以致它永远没有机会完成任何任务时，就会发生活锁。这个线程最终将超出缓冲区并导致程序崩溃。试想一个秘书需要录入一封信，但她一直在忙于接电话，所以这封信永远不会被录入。 内存损坏 如果明智地使用 synchronized 关键字，则完全可以避免内存错误这种气死人的问题。 资源耗尽 某些系统资源是有限的，如文件描述符。多线程程序可能耗尽资源，

因为每个线程都可能希望有一个这样的资源。如果线程数相当大，或者某个资源的侯选线程数远远超过了可用的资源数，则最好使用 资源池。一个最好的示例是数据库连接池。只要线程需要使用一个数据库连接，它就从池中取出一个，使用以后再将它返回池中。资源池也称为 资源库。 调试大量的线程 有时一个程序因为有大量的线程在运行而极难调试。在这种情况下，下面的这个类可能会派上用场： public class Probe extends Thread { public Probe() {} public void run() { while(true) { Thread[] x = new Thread[100]. Thread.enumerate(x). for(int i=0. i 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