读书笔记：多线程程序设计23个要点 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/269/2021_2022__E8_AF_BB_E4_B9_A6_E7_AC_94_E8_c67_269944.htm 1.多线程中有主内存和工作内存之分， 在JVM中，有一个主内存，专门负责所有线程共享数据；而每个线程都有他自己私有的工作内存， 主内存和工作内存分贝在JVM的stack区和heap区。 2.线程的状态有Ready, Running, Sleeping, Blocked, 和 Waiting几个状态，Ready 表示线程正在等待CPU分配允许运行的时间。 3.线程运行次序并不是按照我们创建他们时的顺序来运行的，CPU处理线程的顺序是不确定的，如果需要确定，那么必须手工介入，使用setPriority()方法设置优先级。 4.我们无从知道一个线程什么时候运行，两个或多个线程在访问同一个资源时，需要synchronized 5. 每个线程会注册自己，实际某处存在着对它的引用，因此，垃圾回收机制对它就“ 束手无策 ”了。 6. Daemon线程区别一般线程之处是：主程序一旦结束，Daemon线程就会结束。 7. 一个对象中的所有synchronized方法都共享一把锁，这把锁能够防止多个方法对通用内存同时进行的写操作。synchronized static方法可在一个类范围内被相互间锁定起来。 8. 对于访问某个关键共享资源的所有方法，都必须把它们设为synchronized，否则就不能正常工作。 9. 假设已知一个方法不会造成冲突，最明智的方法是不要使用synchronized，能提高些性能。 10. 如果一个\"同步"方法修改了一个变量，而我们的方法要用到这个变量(可能是只读)，最好将自己的这个方法也设为 synchronized。 11. synchronized不能继承， 父类的方法是synchronized，那么其子类重载方法

中就不会继承“ 同步 ”。 12. 线程堵塞Blocked有几个原因造成： (1)线程在等候一些IO操作 (2)线程试图调用另外一个对象的“ 同步 ”方法，但那个对象处于锁定状态，暂时无法使用。 13.原子型操作(atomic), 对原始型变量(primitive)的操作是原子型的atomic. 意味着这些操作是线程安全的， 但是大部分情况下，我们并不能正确使用，来看看 i = i 1 , i是int型，属于原始型变量： (1)从主内存中读取i值到本地内存. (2)将值从本地内存装载到线程工作拷贝中. (3)装载变量1. (4)将i 加 1. (5)将结果给变量i. (6)将i保存到线程本地工作拷贝中. (7)写回主内存. 注意原子型操作只限于第1步到第2步的读取以及第6到第7步的写, i的值还是可能被同时执行i=i 1的多线程中断打扰(在第4步)。 double 和long 变量是非原子型的(non-atomic)。数组是object 非原子型。 14. 由于13条的原因，我们解决办法是： class xxx extends Thread{ //i会被经常修改 private int i. public synchronized int read(){ return i.} public synchronized void 0update(){ i = i 1.} ..... } 15. Volatile变量， volatile变量表示保证它必须是与主内存保持一致，它实际是"变量的同步", 也就是说对于volatile变量的操作是原子型的，如用在long 或 double 变量前。 16. 使用yield()会自动放弃CPU，有时比sleep更能提升性能。 17. sleep()和wait()的区别是：wait()方法被调用时会解除锁定，但是我们能使用它的地方只是在一个同步的方法或代码块内。 18. 通过制造缩小同步范围，尽可能的实现代码块同步，wait(毫秒数)可在指定的毫秒数可退出wait；对于wait()需要被notisfy()或notifyAll()踢醒。 19. 构造两个线程之间实时通信的方法分几步： (1). 创建一个PipedWriter和一个PipedReader和它们之间的管道. PipedReader in = new

PipedReader(new PipedWriter()) (2). 在需要发送信息的线程开始之前，将外部的PipedWriter导向给其内部的Writer实例out (3). 在需要接受信息的线程开始之前，将外部的PipedReader导向给其内部的Reader实例in (4). 这样放入out的所有东西度可从in中提取出来。 20. synchronized带来的问题除性能有所下降外，最大的缺点是会带来死锁DeadLock，只有通过谨慎设计来防止死锁，其他毫无办法，这也是线程难以驯服的一个原因。不要再使用stop() suspend() resume()和destory()方法 21. 在大量线程被堵塞时，最高优先级的线程先运行。但是不表示低级别线程不会运行，运行概率小而已。 22. 线程组的主要优点是：使用单个命令可完成对整个线程组的操作。很少需要用到线程组。 23. 从以下几个方面提升多线程的性能： 检查所有可能Block的地方，尽可能的多的使用sleep或yield()以及wait(). 尽可能延长sleep(毫秒数)的时间. 运行的线程不用超过100个，不能太多；不同平台linux或windows以及不同JVM运行性能差别很大。