

Java多线程设计模式：wait_notify机制 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/461/2021_2022_Java_E5_A4_9A_E7_BA_BF_c104_461636.htm 通常，多线程之间需要协调工作。例如，浏览器的一个显示图片的线程displayThread想要执行显示图片的任务，必须等待下载线程downloadThread将该图片下载完毕。如果图片还没有下载完，displayThread可以暂停，当downloadThread完成了任务后，再通知displayThread“图片准备完毕，可以显示了”，这时，displayThread继续执行。以上逻辑简单的说就是：如果条件不满足，则等待。当条件满足时，等待该条件的线程将被唤醒。在Java中，这个机制的实现依赖于wait/notify.等待机制与锁机制是密切关联的。例如：`synchronized(obj) {while(!condition)`

`{obj.wait().}obj.doSomething().}` 当线程A获得了obj锁后，发现条件condition不满足，无法继续下一处理，于是线程A就wait（）。在另一线程B中，如果B更改了某些条件，使得线程A的condition条件满足了，就可以唤醒线程A

：`synchronized(obj) {condition = true.obj.notify().}` 需要注意的概念是：调用obj的wait（），notify（）方法前，必须获得obj锁，也就是必须写在synchronized（obj）{.....}代码段内。调用obj.wait（）后，线程A就释放了obj的锁，否则线程B无法获得obj锁，也就无法在synchronized（obj）{.....}代码段内唤醒A。当obj.wait（）方法返回后，线程A需要再次获得obj锁，才能继续执行。如果A1，A2，A3都在obj.wait（），则B调用obj.notify（）只能唤醒A1，A2，A3中的一个（具体哪一个由JVM决定）。obj.notifyAll（）则能全部唤

醒A1，A2，A3，但是要继续执行obj.wait（）的下一条语句，必须获得obj锁，因此，A1，A2，A3只有一个有机会获得锁继续执行，例如A1，其余的需要等待A1释放obj锁之后才能继续执行。当B调用obj.notify/notifyAll的时候，B正持有obj锁，因此，A1，A2，A3虽被唤醒，但是仍无法获得obj锁。直到B退出synchronized块，释放obj锁后，A1，A2，A3中的一个才有机会获得锁继续执行。 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