# java sleep和wait的区别的疑惑

1、首先，要记住这个差别，“sleep是Thread类的方法,wait是Object类中定义的方法”。尽管这两个方法都会影响线程的执行行为，但是本质上是有区别的。

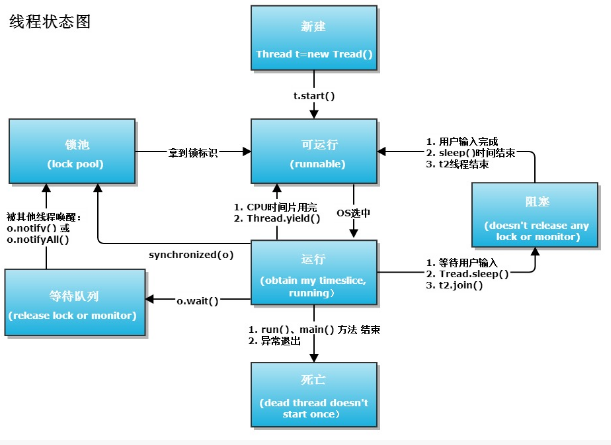
2、Thread.sleep不会导致锁行为的改变，如果当前线程是拥有锁的，那么Thread.sleep不会让线程释放锁。如果能够帮助你记忆的话，可以简单认为和锁相关的方法都定义在Object类中，因此调用Thread.sleep是不会影响锁的相关行为。

3、Thread.sleep和Object.wait都会暂停当前的线程，对于CPU资源来说，不管是哪种方式暂停的线程，都表示它暂时不再需要CPU的执行时间。OS会将执行时间分配给其它线程。区别是，调用wait后，需要别的线程执行notify/notifyAll才能够重新获得CPU执行时间。

4、线程的状态参考 Thread.State的定义。新创建的但是没有执行（还没有调用start())的线程处于“就绪”，或者说Thread.State.NEW状态。

5、Thread.State.BLOCKED（阻塞）表示线程正在获取锁时，因为锁不能获取到而被迫暂停执行下面的指令，一直等到这个锁被别的线程释放。BLOCKED状态下线程，OS调度机制需要决定下一个能够获取锁的线程是哪个，这种情况下，就是产生锁的争用，无论如何这都是很耗时的操作。

**线程间的状态转换：**



sleep 让线程从 【running】 -> 【阻塞态】 时间结束/interrupt -> 【runnable】

wait 让线程从 【running】 -> 【等待队列】notify -> 【锁池】 -> 【runnable】

**1. 新建(new)**：新创建了一个线程对象。

**2. 可运行(runnable)**：线程对象创建后，其他线程(比如main线程）调用了该对象的start()方法。该状态的线程位于可运行线程池中，等待被线程调度选中，获取cpu 的使用权 。

**3. 运行(running)**：可运行状态(runnable)的线程获得了cpu 时间片（timeslice） ，执行程序代码。

**4. 阻塞(block)**：阻塞状态是指线程因为某种原因放弃了cpu 使用权，也即让出了cpu timeslice，暂时停止运行。直到线程进入可运行(runnable)状态，才有机会再次获得cpu timeslice 转到运行(running)状态。阻塞的情况分三种：

(一). 等待阻塞：运行(running)的线程执行o.wait()方法，JVM会把该线程放入等待队列(waitting queue)中。

(二). 同步阻塞：运行(running)的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池(lock pool)中。

(三). 其他阻塞：运行(running)的线程执行Thread.sleep(long ms)或t.join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入可运行(runnable)状态。

**5. 死亡(dead)**：线程run()、main() 方法执行结束，或者因异常退出了run()方法，则该线程结束生命周期。死亡的线程不可再次复生。

