Java中的线程的生命周期大体可分为5种状态。

1. **新建(NEW)**：**新建了一个线程对象。**

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(BLOCKED)**：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。

**二.初始状态**

1. 实**现Runnable接口和继承Thread可以得到一个线程类，new一个实例出来**，线程就进入了初始状态

**三.可运行状态**

1. 可运行状态只是说你资格运行，**调度程序没有挑选到你，你就永远是可运行状态**。
2. 调用线程**的start()方法，此线程进入可运行状态**。
3. 当前线程**sleep()方法结束，其他线程join()结束，**等待用户输入完毕，某个线程拿到对象锁，这些线程也将**进入可运行状态**。
4. 当前线程时间片用完了，调用**当前线程的yield()方法，当前线程进入可运行状态**。
5. 锁池里的线程**拿到对象锁后，进入可运行状态**。

**四.运行状态**

1. 线程调度程序从**可运行池中选择一个线程**作为**当前线程时线程所处的状态**。这也是线程进入运行状态的唯一一种方式。

**五.死亡状态**

1. 当线程**的run()方法完成**时，或者主线程**的main()方法完成**时，我们**就认为它死去**。这个线程对象也许是活的，但是，它已经不是一个单独执行的线程。线程一旦死亡，就不能复生。
2. 在一个死去的线程上调用start()方法，会抛出java.lang.IllegalThreadStateException异常。

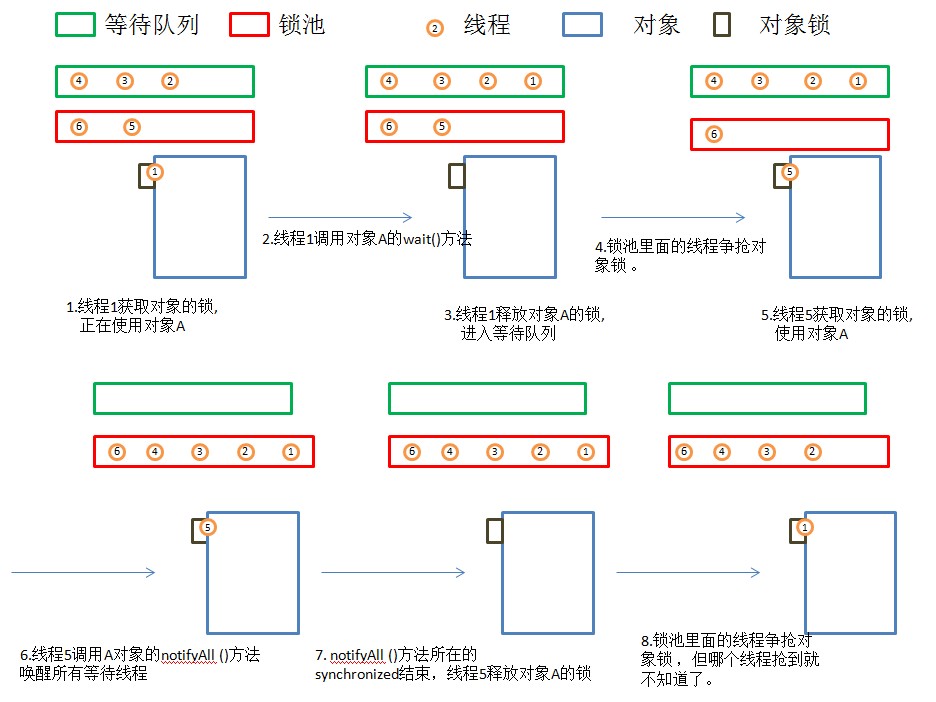
**六.阻塞状态**

1. 当前线程T调用**Thread.sleep()**方法，当前线程进**入阻塞状**态。
2. 运行在当前线程里的其它线程**t2调用join()方**法，当前线程进**入阻塞状态**。
3. **等待用户输入的**时候，当**前线程进入阻塞状**态。

**七.等待队列(本是Object里的方法，但影响了线程)**

1. 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。**
2. 与等待队列相关的步骤和图

* 线程1获取对象A的锁，正在使用对象A。
* 线程1调用对象A的wait()方法。
* 线程1释放对象A的锁，并马上进入等待队列。
* 锁池里面的对象争抢对象A的锁。
* 线程5获得对象A的锁，进入synchronized块，使用对象A。
* 线程5调用对象A的notifyAll()方法，唤醒所有线程，所有线程进入锁池。||||| 线程5调用对象A的notify()方法，唤醒一个线程，不知道会唤醒谁，被唤醒的那个线程进入锁池。
* notifyAll()方法所在synchronized结束，线程5释放对象A的锁。
* 锁池里面的线程争抢对象锁，但线程1什么时候能抢到就不知道了。||||| 原本锁池+第6步被唤醒的线程一起争抢对象锁。

**八.锁池状态**

1. 当前线程想调用对象A的同步方法时，发现对象A的锁被别的线程占有，此时当前线程进入锁池状态。简言之，锁池里面放的都是想争夺对象锁的线程。
2. 当一个线程1被另外一个线程2唤醒时，1线程进入锁池状态，去争夺对象锁。
3. 锁池是在同步的环境下才有的概念，一个对象对应一个锁池。

**九.几个方法的比较**

1. Thread.sleep(long millis)，一定是当前线程调用此方法，当前线程进入阻塞**，但不释放对象锁，**millis后线程自动苏醒进入可运行状态。作用：给**其它线程执行机会的**最佳方式。
2. Thread.yield()，一定是当前线程调用此方法，当**前线程放弃获取的cpu时间片，由运行状态变会可运行状态**，让OS再次选择线程。作用：让相同优先级的线程轮流执行，但并不保证一定会轮流执行。实际中无法保证yield()达到让步目的，因**为让步的线程还有可能被线程调度程序再次选中**。**Thread.yield()不会导致阻塞**。
3. **t.join()/t.join(long millis)**，当前线程里调用其它线程1的join方法，当前线程阻塞，但不释放对象锁，直到线程1执行完毕或者millis时间到，当前线程进入可运行状态。
4. **obj.wait()，**当前线程调用对象的wait()方法，当前线程释放**对象锁，进入等待队列。**依靠**notify()/notifyAll()唤醒或者wait(long timeout)timeout时间到自动唤醒**。
5. obj**.notify()唤醒在**此对象监视器上**等待的单个线程**，选择是任意性的。n**otifyAll()唤醒**在此对象监视器上**等待的所有线程。**