并发基础篇（4）： 深入浅出 Java 线程的状态

一、线程的五种状态

线程的生命周期可以大致分为 5 种，但这种说法是比较旧的一种说法，有点过时了，或者更确切的来说，这是操作系统的说法，而不是 java 的说法。但对下面所说的六种状态的理解有所帮助，所以也写出来，作为参考。

1. NEW： 线程的新建状态，是指通过 New 关键字创建了 Thread 类（或其子类）的对象。

2. RUNNABLE： 这种情况指的是 Thread 类的对象调用了 start()方法，这时的线程就等待时间片轮转到自己这，以便获得 CPU；第二种情况是线程在处于 RUNNABLE 状态时并没有运行完自己的 run 方法，时间片用完之后回到 RUNNABLE 状态；还有种情况就是处于 BLOCKED 状态的线程结束了当前的 BLOCKED 状态之后重新回到 RUNNABLE 状态。

3. RUNNING： 这时的线程指的是获得 CPU 的 RUNNABLE 线程，RUNNING 状态是所有线程都希望获得的状态。

4. DEAD： 处于 RUNNING 状态的线程，在执行完 run 方法之后，或者异常退出时，就变成了 DEAD 状态了。

5. BLOCKED： 这种状态指的是处于 RUNNING 状态的线程，出于某种原因，比如调用了 sleep 方法、等待用户输入等而让出当前的 CPU 给其他的线程。

注意： BLOCKED 状态，包括三种类型状态：等待（wait）、睡眠（sleep）、阻塞（申请资源：I\O、对象的锁）；

二、线程的六种状态

在 java 中，线程的状态其实是六种，对应着枚举类型 Thread.State 的六个枚举常量：NEW 、BLOCKED、RUNNABLE、WAITING 、TIMED_WAITING、TERMINATED

1. NEW ： 新建状态，至今尚未启动的线程的状态。

```
static void NEW() {
      Thread t = new Thread ();//也就是可以理解为刚刚创建 thread
      System. out.println(t.getState());
}
```

2. BLOCKED： 阻塞状态，受阻塞并且正在等待监视器锁的某一线程的线程状态。

```
private static void BLOCKED() {
        final Object lock = new Object();
        Runnable run = new Runnable() {
             @Override
```

```
            public void run() {
                for(int i=0; i<Integer.MAX_VALUE; i++){
                     synchronized (lock) {
                        System. out.println(i);
                    }
                }
            }
        };
        Thread t1 = new Thread(run);
        t1.setName( “t1”);
        Thread t2 = new Thread(run);
        t2.setName( “t2”);
        t1.start();
        t2.start();
}
```

3. RUNNABLE： 可运行线程的线程状态。这里其实合并了两种状态（RUNNING、RUNABLE）

```
private static void RUNNABLE() {
        Thread t = new Thread(){
            public void run(){
                for(int i=0; i<Integer.MAX_VALUE; i++){
                    System. out.println(i);
                }
            }
        };
        t.start();
}
```

当线程调用了 start()方法之后，就是运行状态了。

4. WAITING ： 等待状态，表示线程进入状态。进入此状态后，会无限等待，直到其他线程做出一些特定的动作（唤醒通知、中断通知）才会再次运行。

```
private static void WAITING() {
         final Object lock = new Object();
        Thread t1 = new Thread(){
            @Override
            public void run() {
                int i = 0;
                while(true ){
                     synchronized (lock) {
                        try {
                            lock.wait();//调用这个方法，进入等待状态
```

```
                    } catch (InterruptedException e) {
                    }
                    System. out.println(i++);
                }
            }
        }
    };
    Thread t2 = new Thread(){
        @Override
        public void run() {
            while(true ){
                synchronized (lock) {
                    for(int i = 0; i< 10000000; i++){
                        System. out.println(i);
                    }
                    lock.notifyAll();
                }
            }
        }
    };
    t1.setName( “^^t1^^”);
    t2.setName( “^^t2^^”);
    t1.start();
    t2.start();
}
```

5. TIMED_WAITING ：计时等待状态 ,此状态与 WAITING 状态有些类似 ,但它是有时间限制的，即只会等待一段指定的时间 ,当时间到来前 ,没有被唤醒或或中断 ,那么时间到来了 ,就自动"醒来"，进入 RUNNABLE 状态。

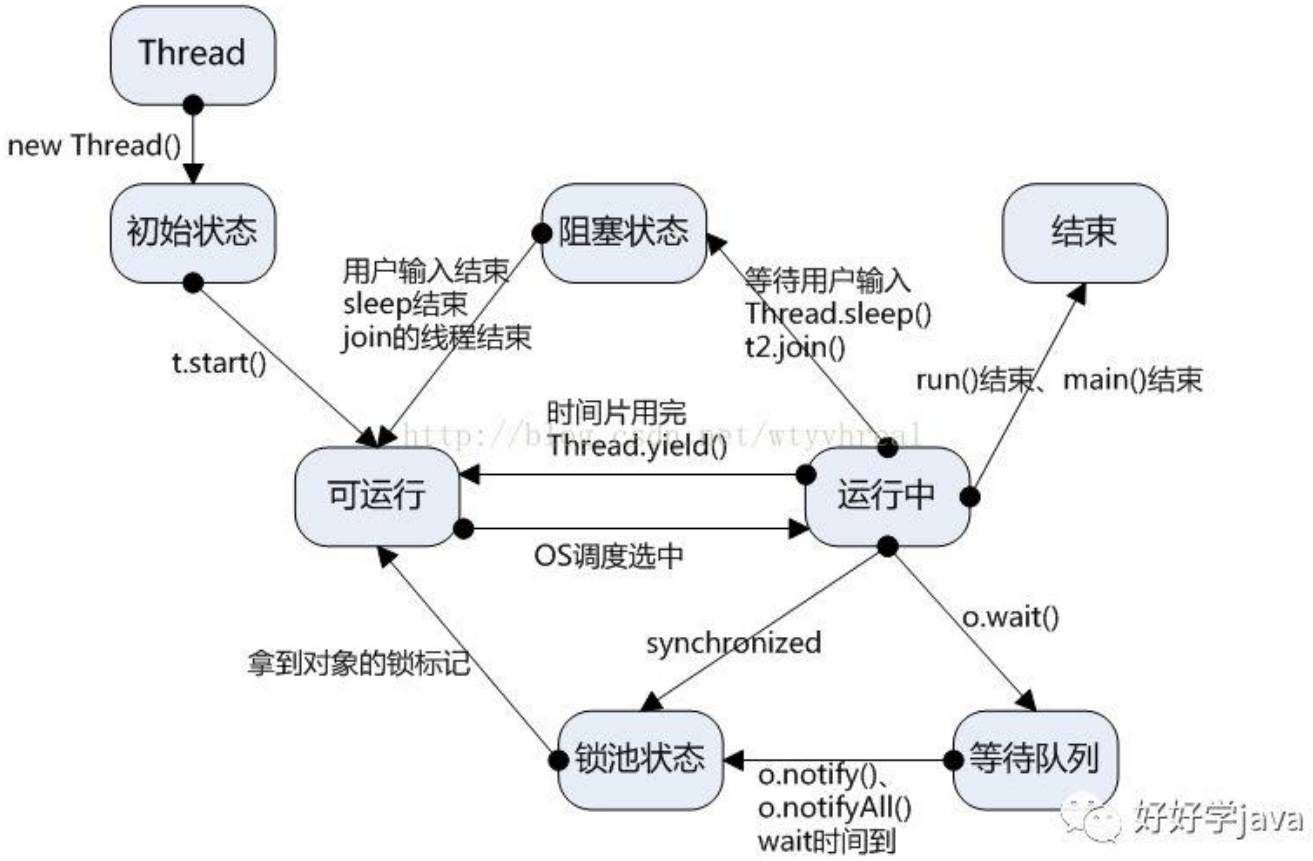
```
synchronized (lock) {
   try {
      lock.wait(60 * 1000L);//只会等待一段指定的时间，当时间到来前，没有被唤醒或或中断
   } catch (InterruptedException e) {
   }
   System. out .println(i++);
}
```

6. TERMINATED ： 终止状态，已终止线程的线程状态。

```
private static void TERMINATED() {
    Thread t1 = new Thread();
    t1.start();
```

```
        System. out.println(t1.getState());
         try {
            Thread. sleep(1000L);
        } catch (InterruptedException e) {
        }
        System. out.println(t1.getState());
}
```

三、线程的状态转换



- 当一个线程执行了 start 方法后，不代表这个线程就会立即被执行，只代表这个线程处于可运行的状态，最终由 OS 的线程调度来决定哪个可运行状态下的线程被执行。
- 一个线程一次被选中执行是有时间限制的，这个时间段叫做 CPU 的时间片，当时间片用完但线程还没有结束时，这个线程又会变为可运行状态，等待 OS 的再次调度；在运行的线程里执行 Thread.yeild()方法同样可以使当前线程变为可运行状态。
- 在一个运行中的线程等待用户输入、调用 Thread.sleep()、调用了其他线程的 join()方法，则当前线程变为阻塞状态。
- 阻塞状态的线程用户输入完毕、sleep 时间到、join 的线程结束，则当前线程由阻塞状态变为可运行状态。
- 运行中的线程调用 wait 方法，此线程进入等待队列。
- 运行中的线程遇到 synchronized 同时没有拿到对象的锁标记、等待队列的线程 wait 时间到、

等待队列的线程被 notify 方法唤醒、有其他线程调用 notifyAll 方法 ,则线程变成锁池状态。

- 锁池状态的线程获得对象锁标记，则线程变成可运行状态。
- 运行中的线程 run 方法执行完毕或 main 线程结束，则线程运行结束。