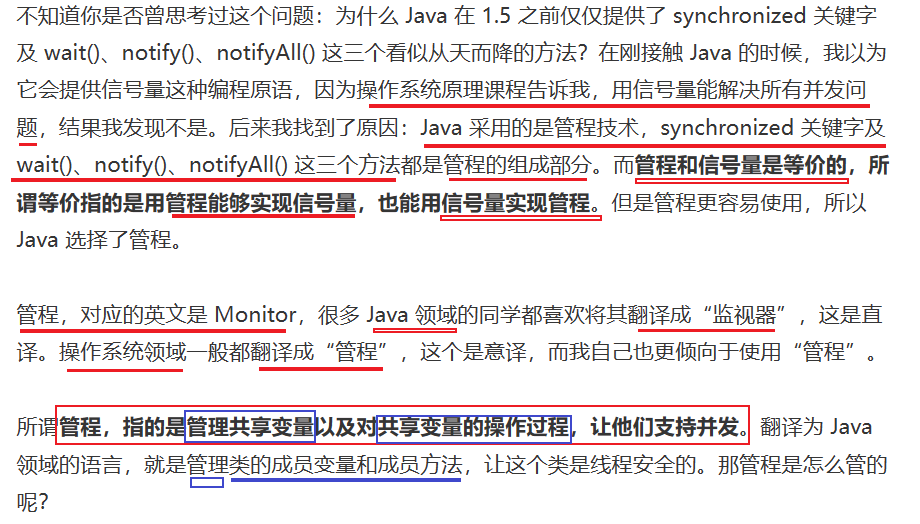
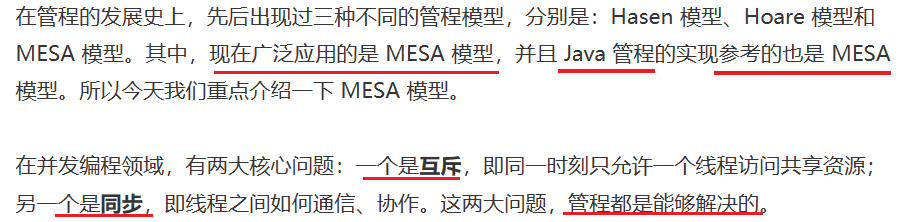
## 08 | 管程：并发编程的万能钥匙

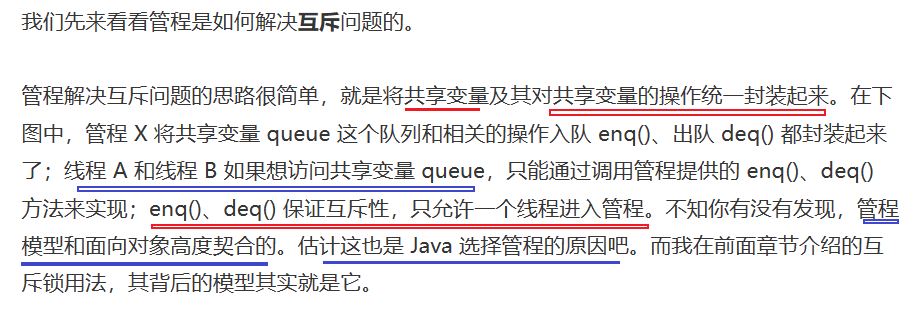
### 什么时管程

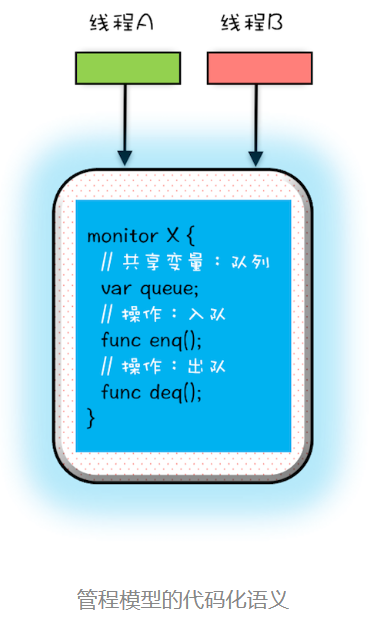


### 管程模型-MESA

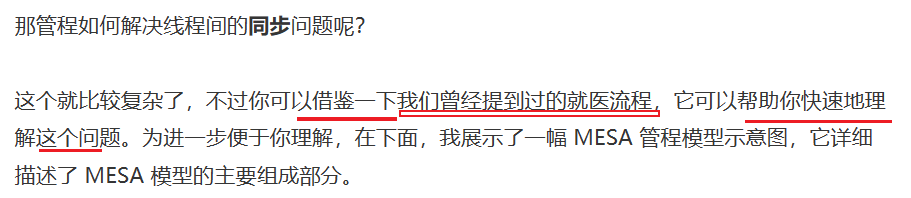


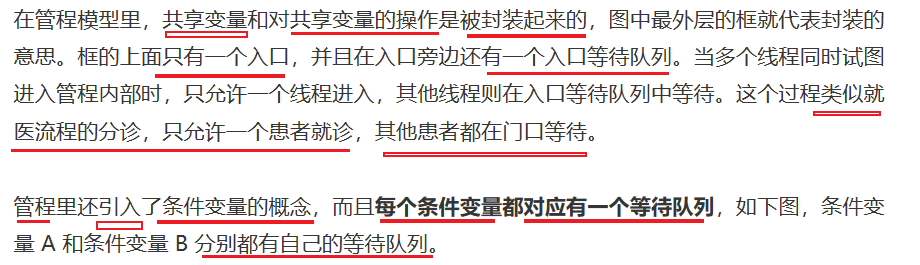
#### 互斥问题

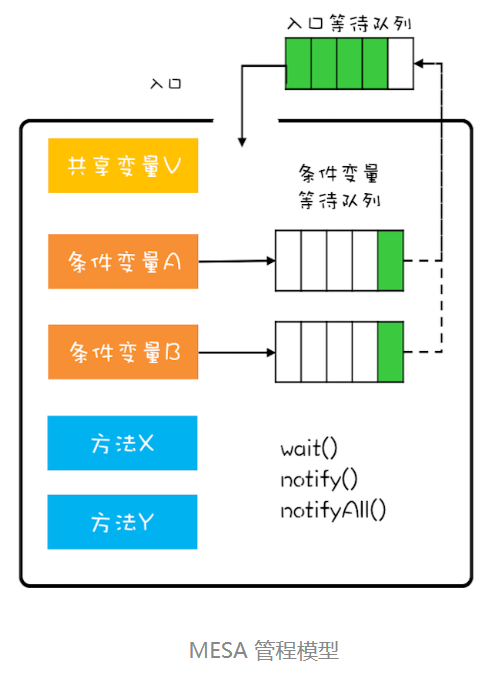




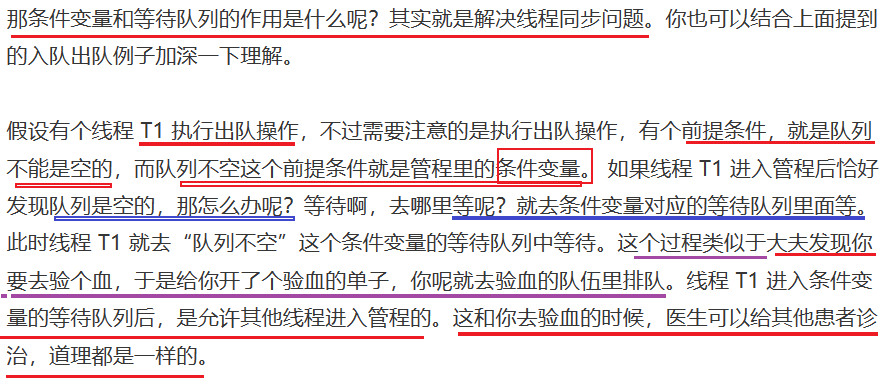
#### 同步问题

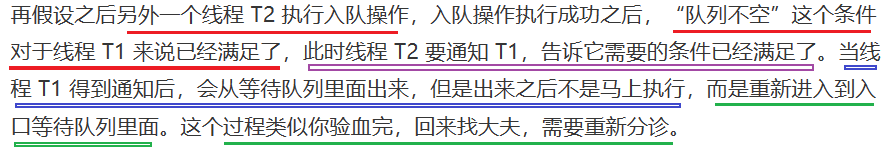


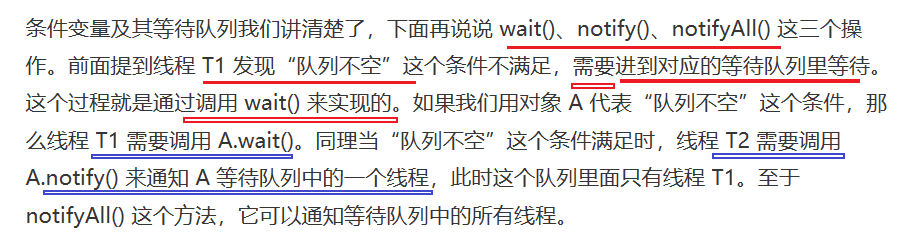




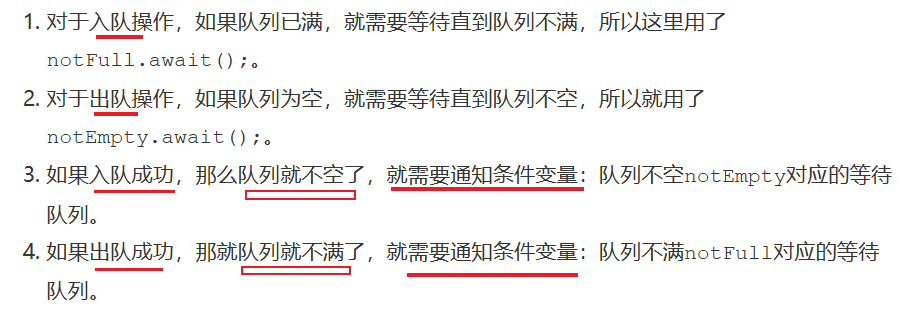
#### 那条件变量和等待队列的作用是什么呢

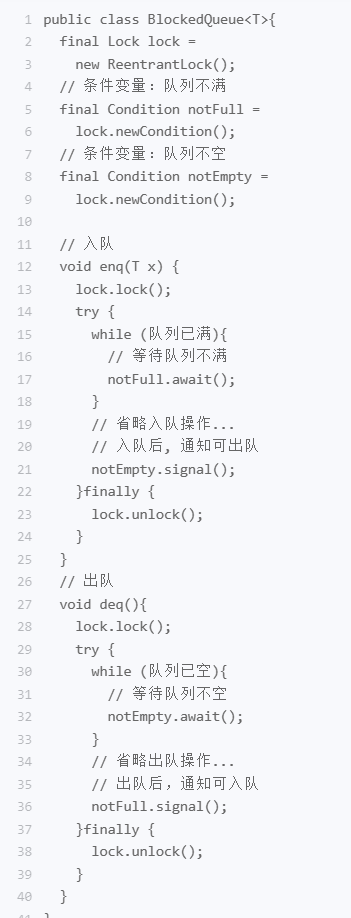
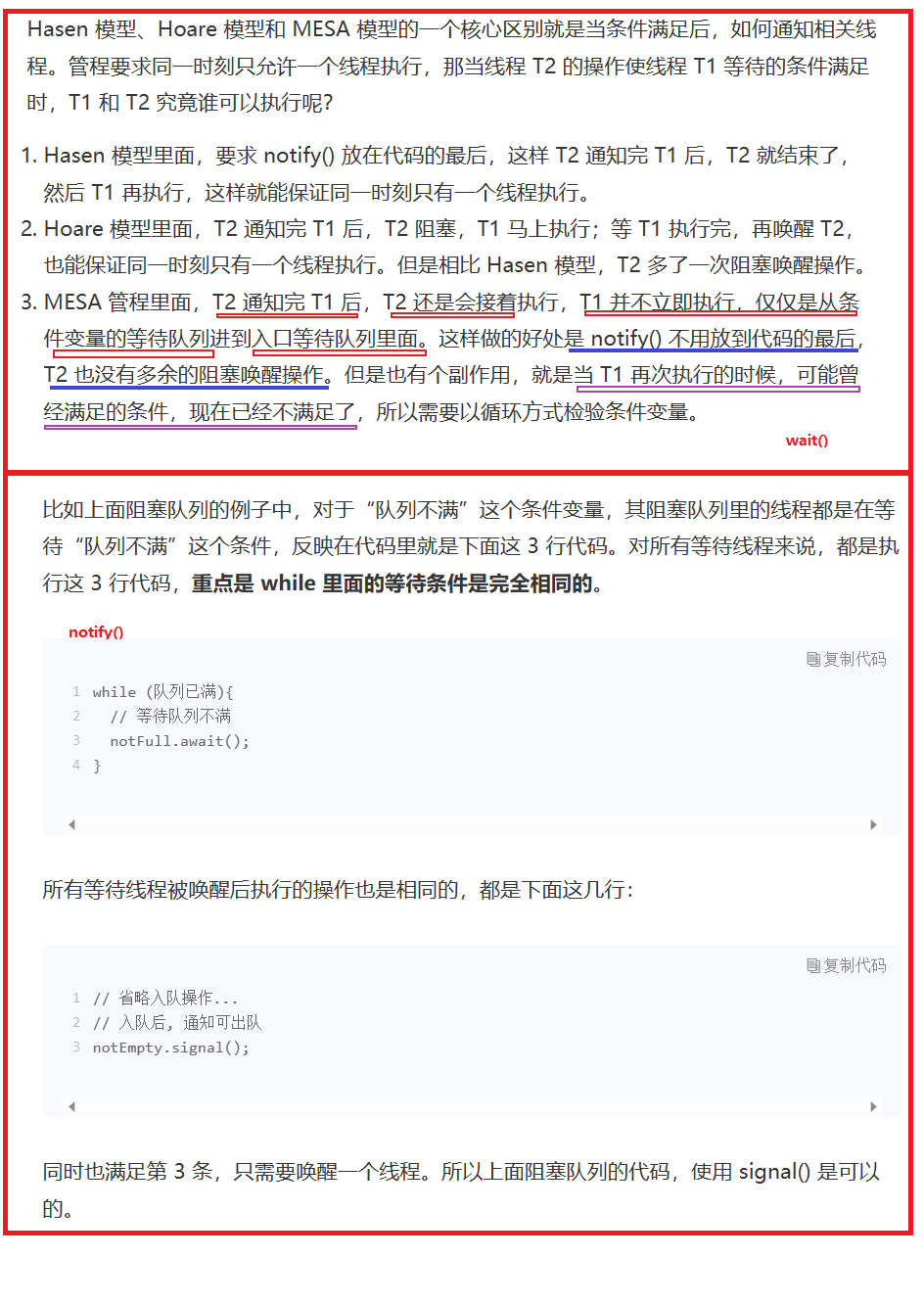




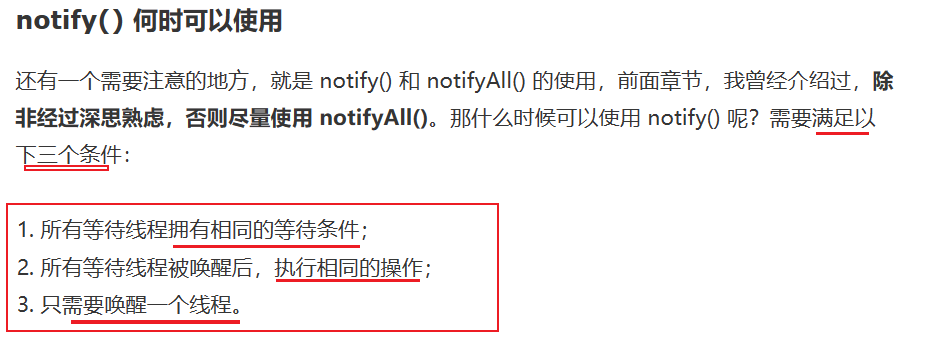


### 等待-通知机制

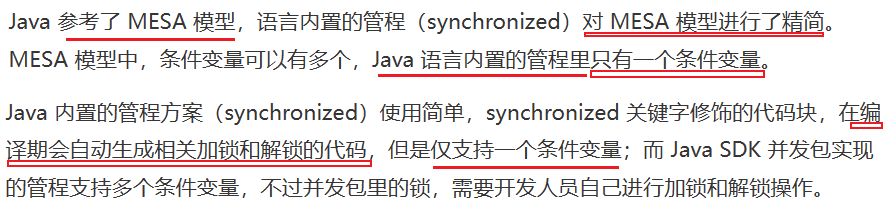


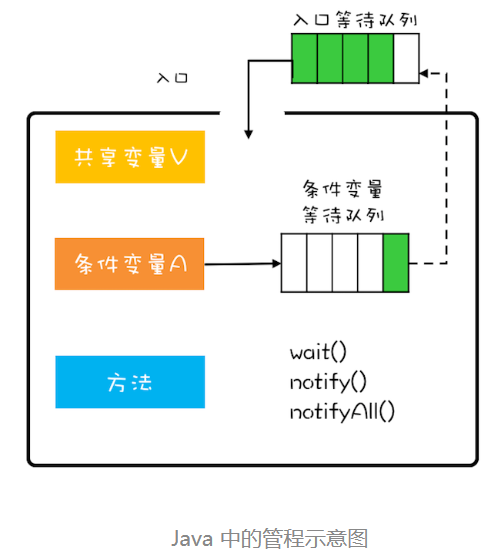
 





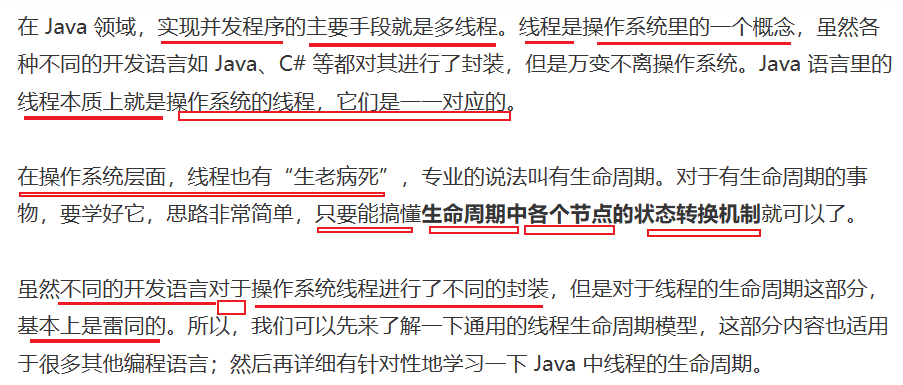
### Java内置管程



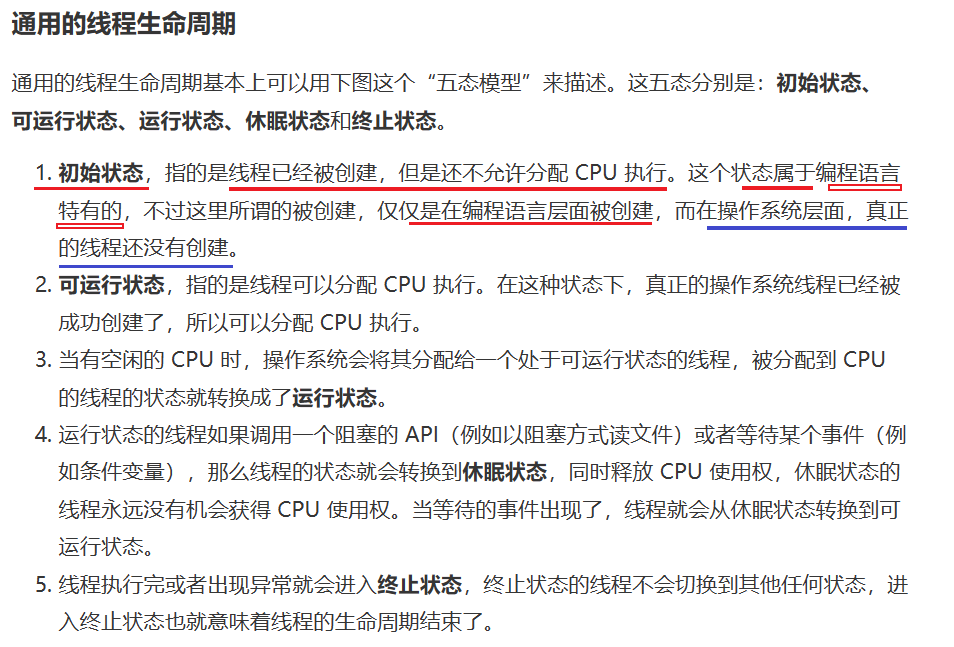


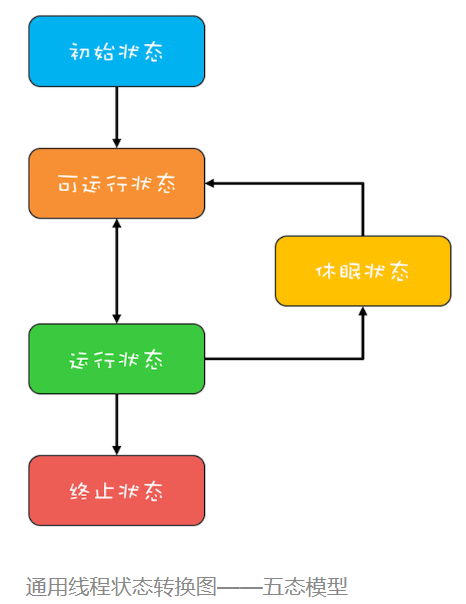
## 09|Java线程（上）：Java线程的生命周期

### 线程的由来

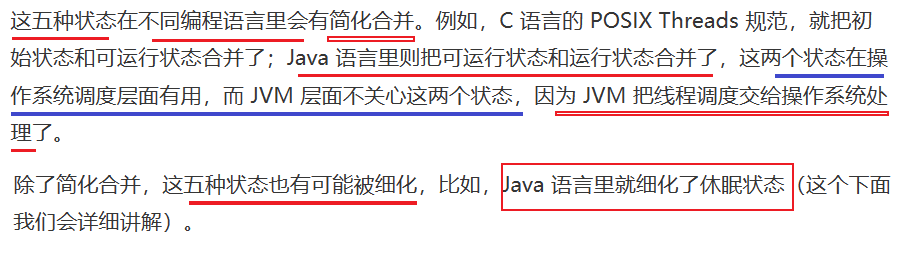


### 通用生命周期



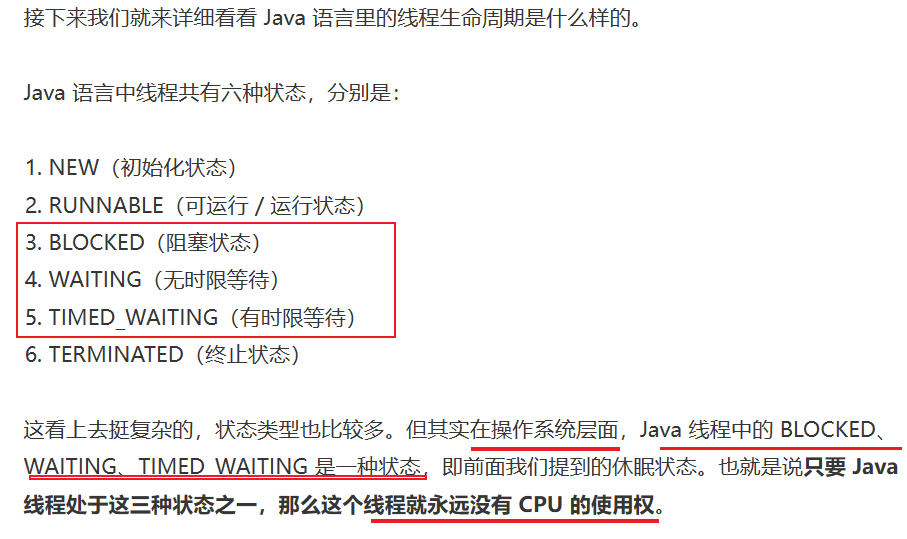


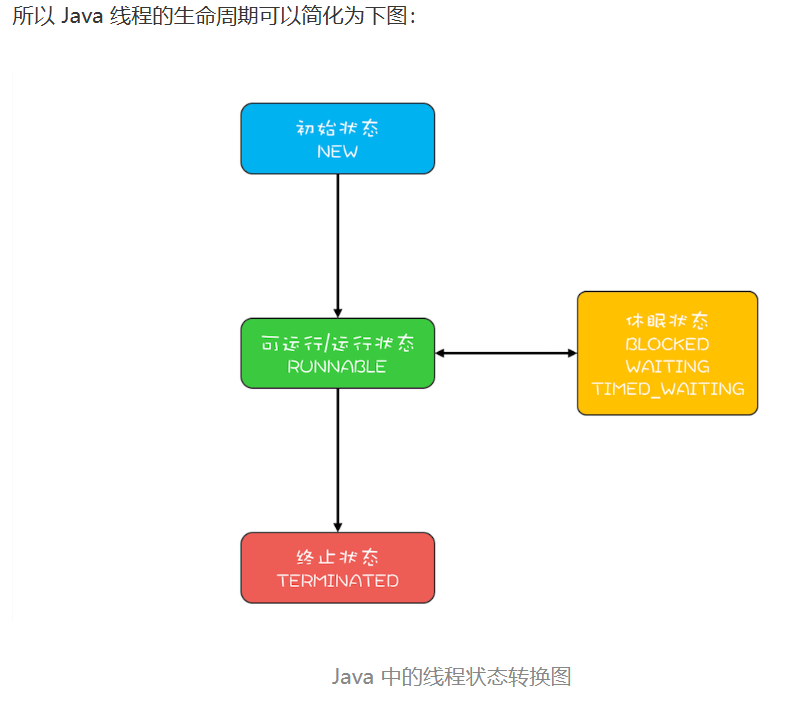
#### 简化生命周期

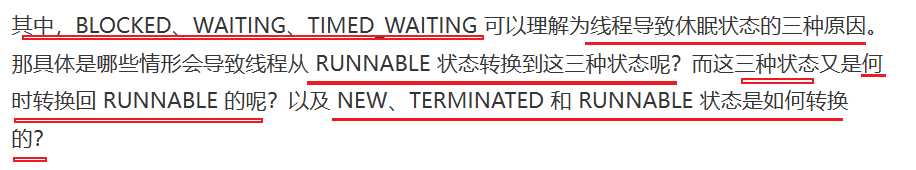


### Java生命周期

#### 1.简述







#### 2.转换到睡眠状态

**Blocked [阻塞] waiting[等待] timed-waiting[定时等待]**

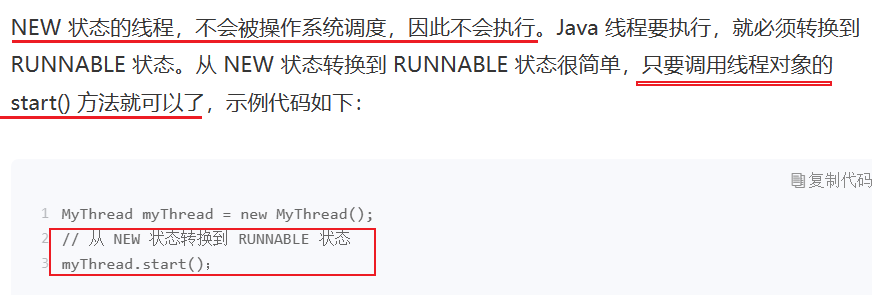


Synchronized获取锁的时候时 blocked 状态，获得锁的时候为 runnable状态。

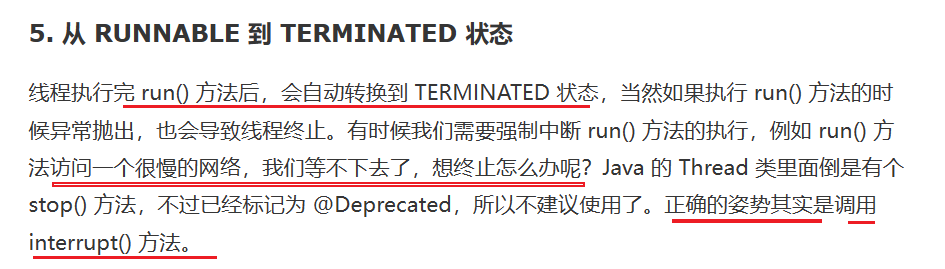


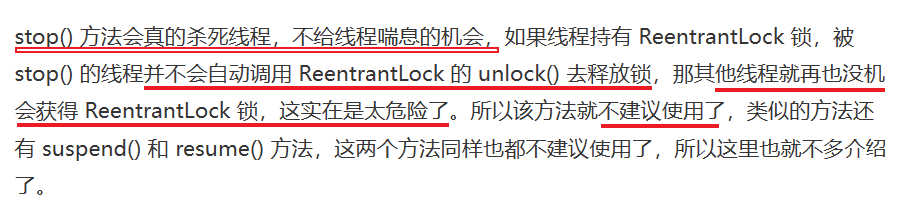


#### 3.New到runnable

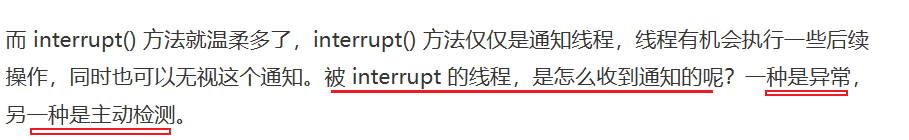


#### 4.Runnable到terminated





#### Interrupt机制



##### 中断方法区别

**对象方法**

interrupt() ：中断线程，当遇到 sleep,wait, join方式时，会抛中断异常， 会清除中断状态

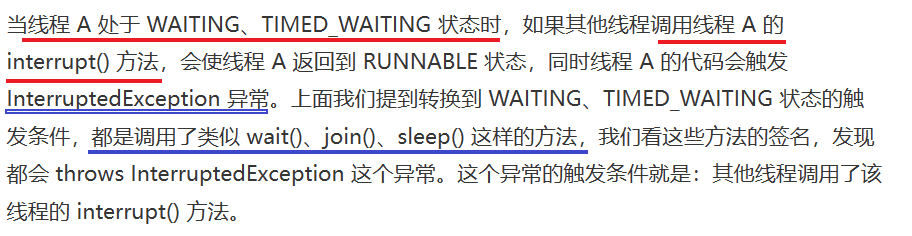
isInterrupted() ：返回线程是否中断，不会清除中断状态。

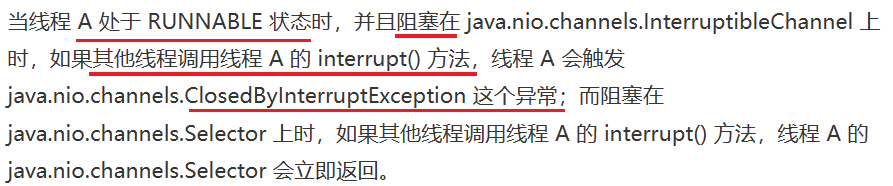
注意：interrupt()、isInterrupted()是成对使用的

**类方法（不需要new对象直接使用）**

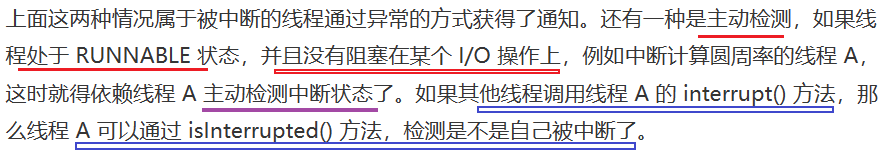
interrupted() ：返回线程是否中断，会清除中断状态。

##### 异常





##### 主动检测



## **10|Java线程（中）：创建多少线程才是合适的**？

## 11|Java线程（下）：为什么局部变量是线程安全的？

