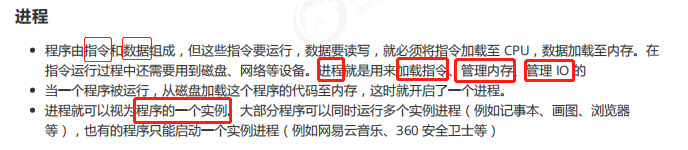
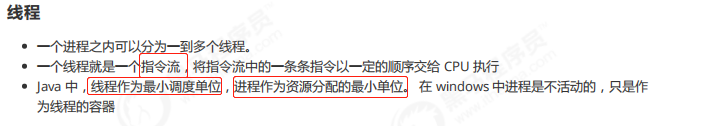
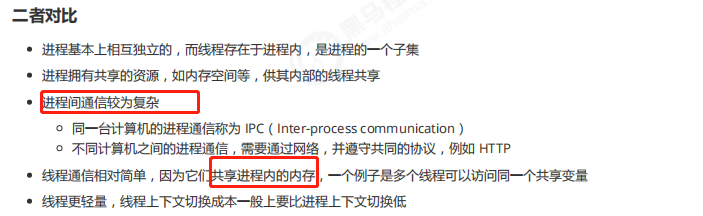
# 第一章、基础知识

# 一、进程与线程、并行与并发

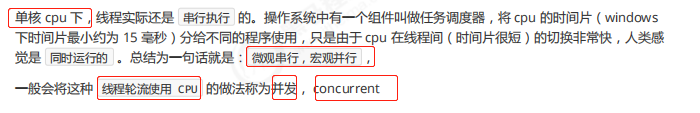
## 1、进程与线程



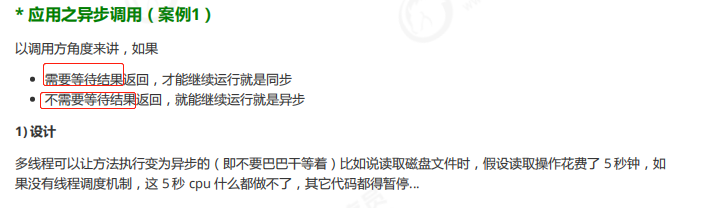


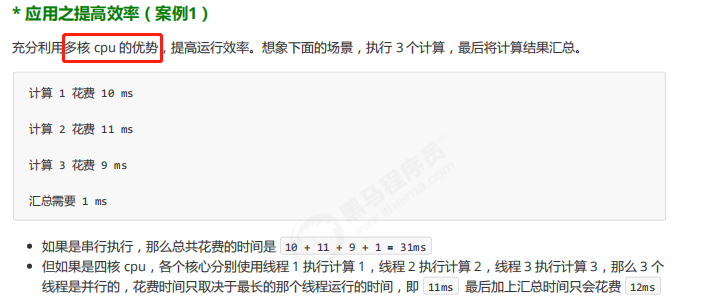


## 2、并行与并发



## 3、应用





# 二、Java线程

## 1、创建线程



## 2、查看线程

- Windows

tasklist | findstr

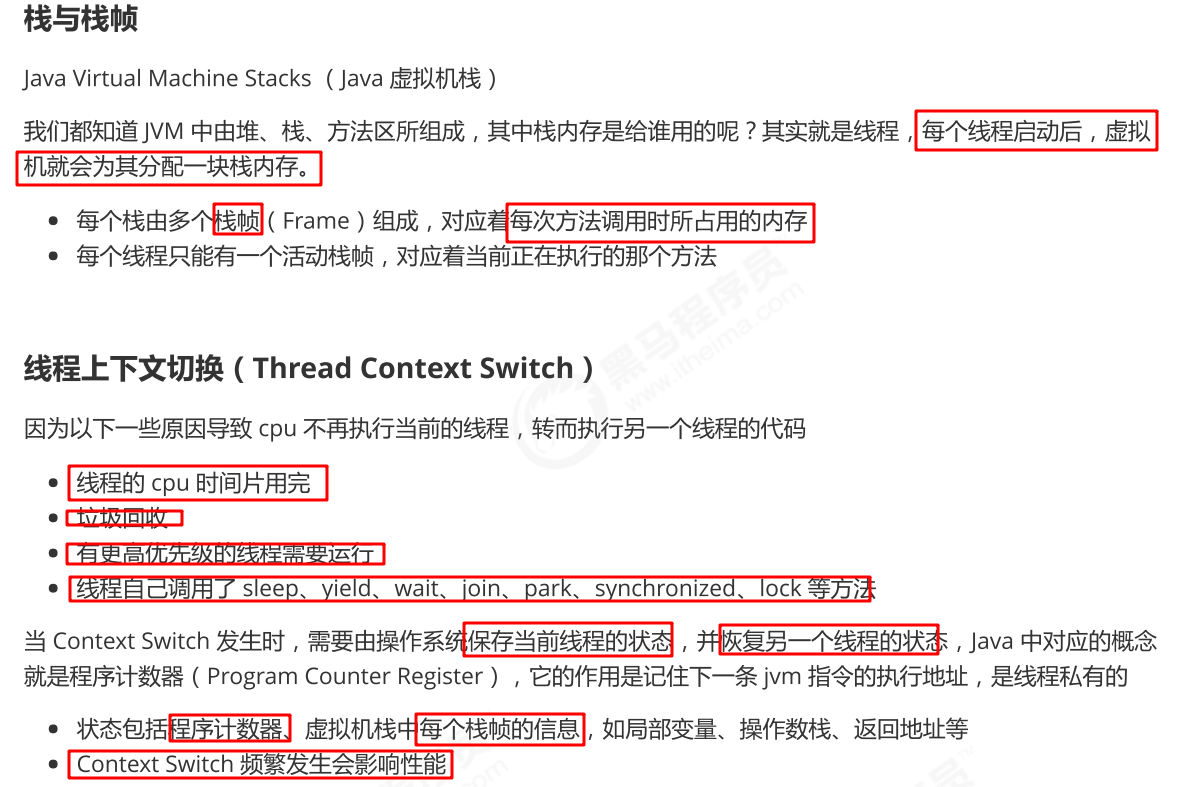
- Linux

ps –ef | grep

- jps 查看Java进程（一个main方法对应一个jvm，是一个进程，相当于Windows下执行一个exe）

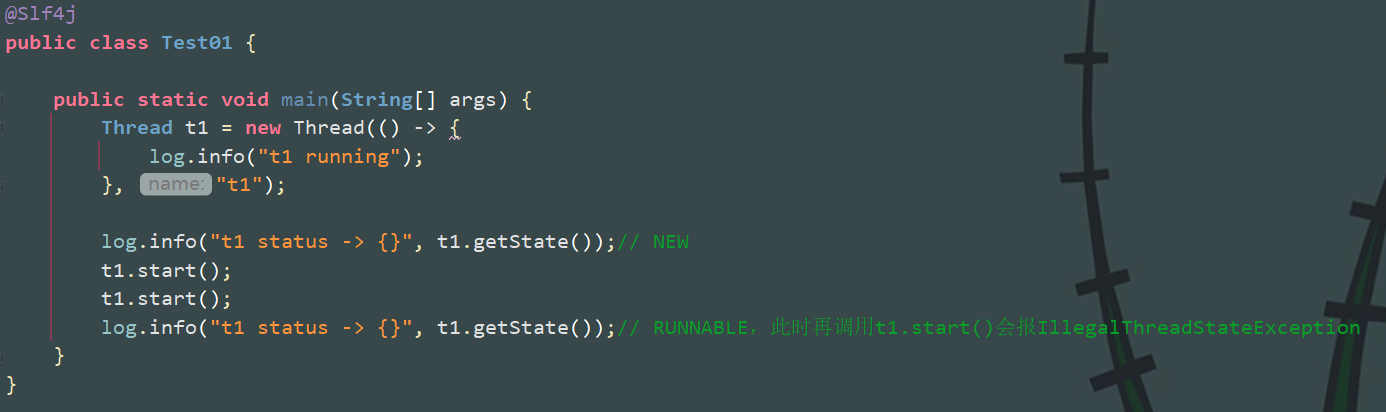
- jconsole

## 3、JVM知识

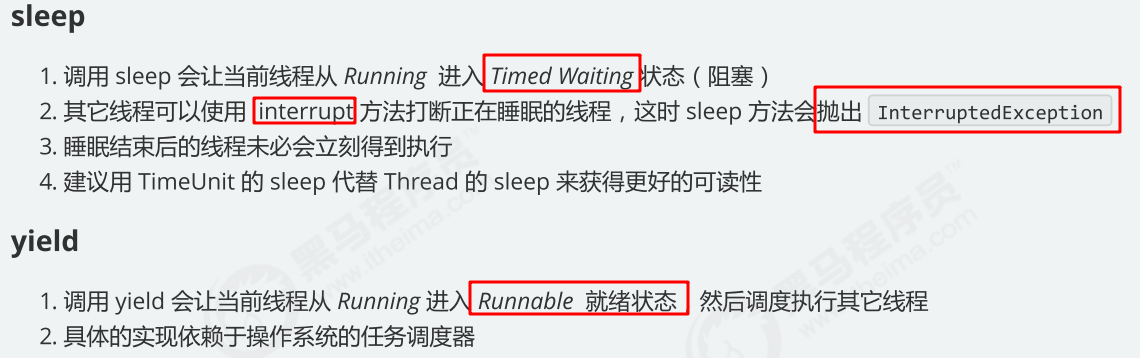


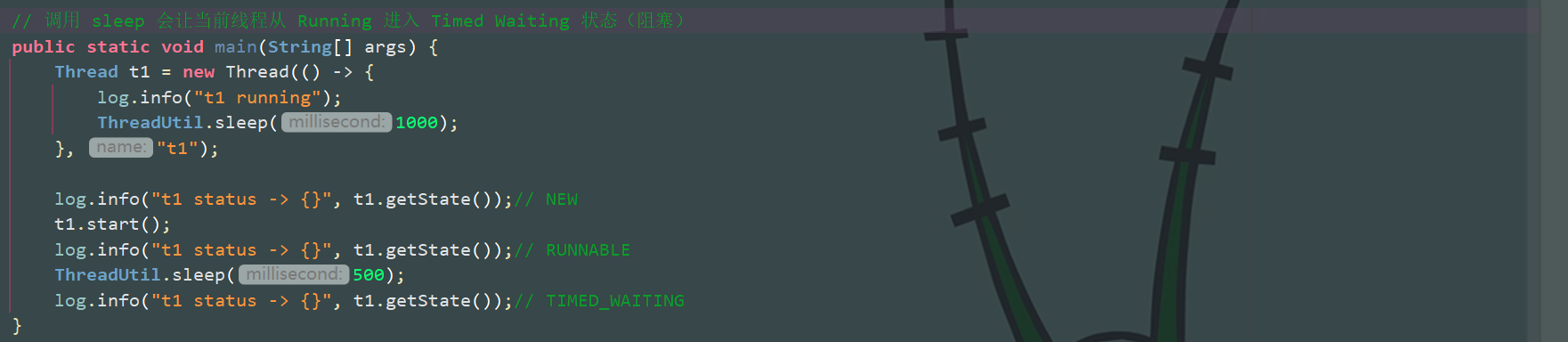
## 4、Thread类方法

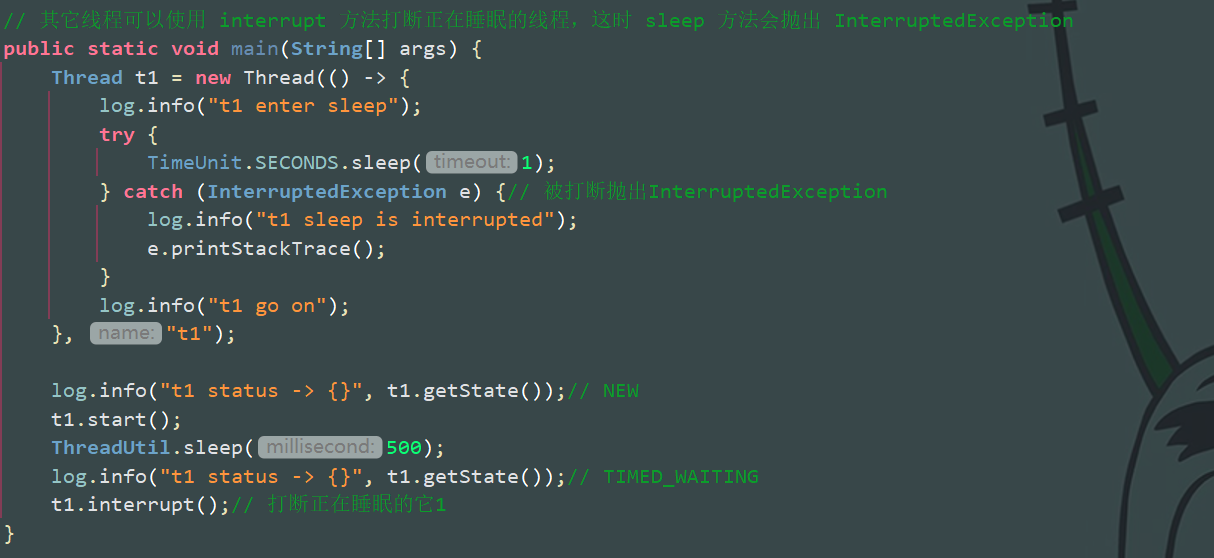
### 1）start()与run()



### 2）sleep()与yied()



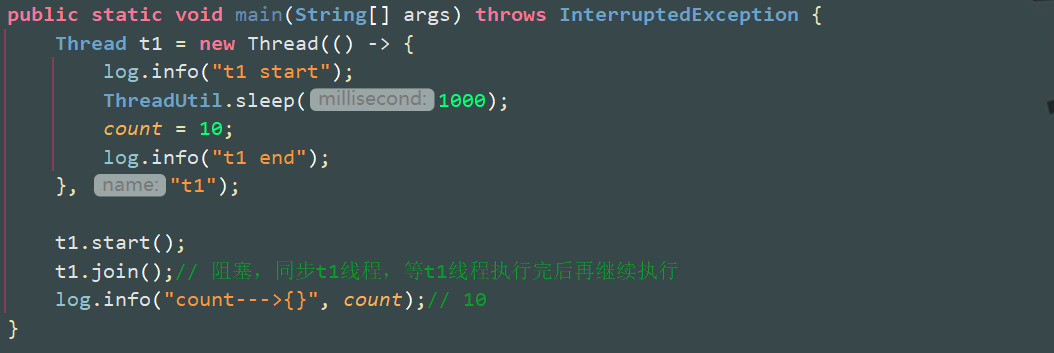




### 3）setPriority()与getPriority()

### 4）join()

用于线程同步，等待特定线程执行完后当前现场再继续执行



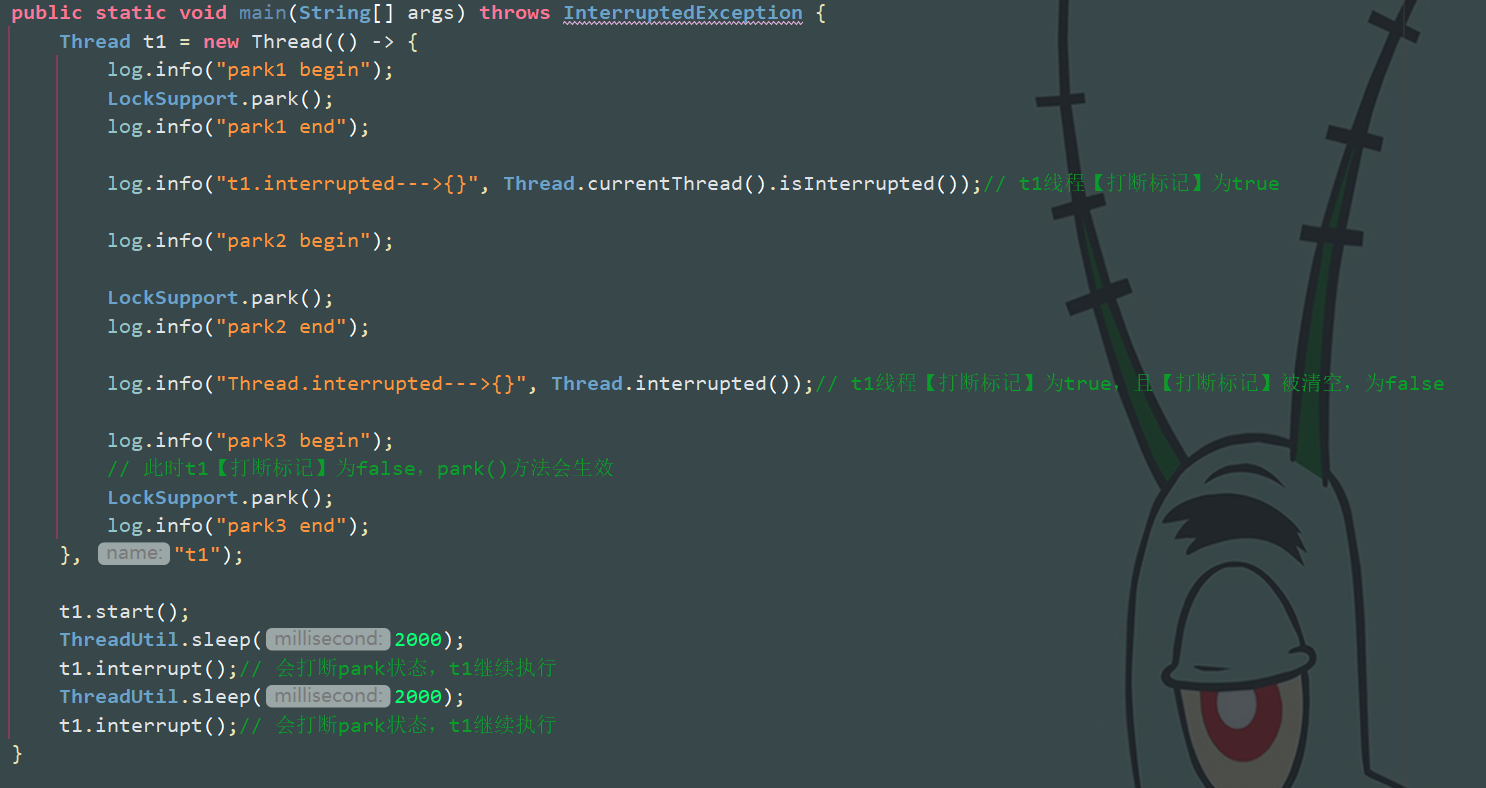
### 5）interrupt()与isInterrupted()与interrupted()

1> interrupt()方法能够打断线程

2> interrupt()方法打断【运行状态】的线程，被打断线程会被打上一个【被打断标记】，即调用isInterrupted()方法返回true；如果打断的是【阻塞状态】的现场，则被打断线程的【被打断标记】会被清空，即调用调用isInterrupted()方法返回false（间两阶段终止模型）

3> 调用isInterrupted()方法返回【被打断标记】

4> 调用interrupted()方法返回线程【被打断标记】，并将【被打断标记】清空



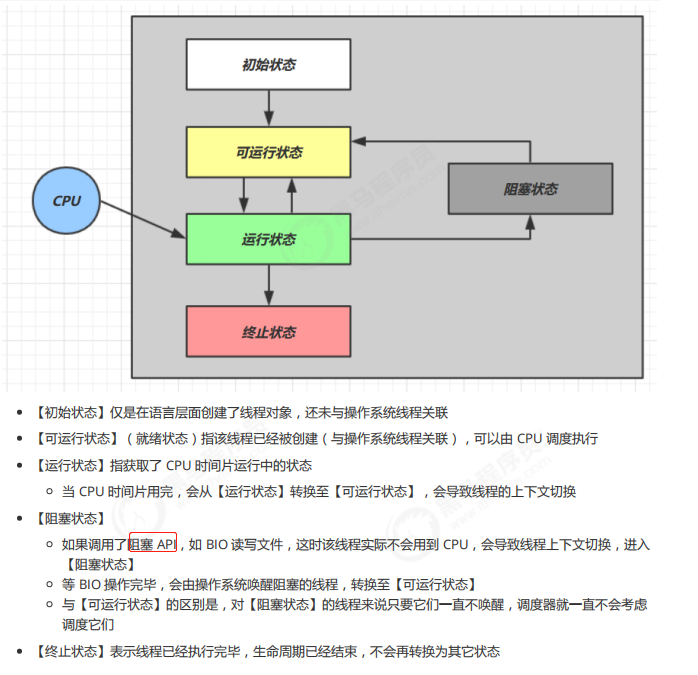
## 5、守护线程



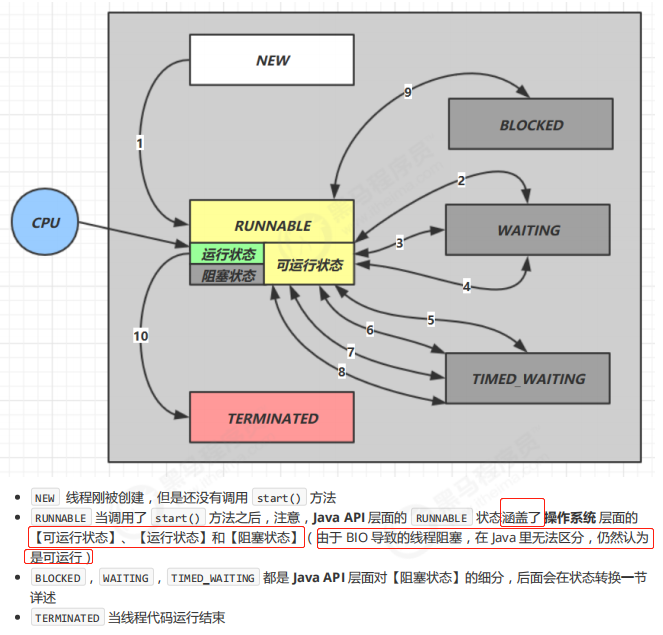


## 6、线程的状态

### 1）操作系统层面——五种



### 2）java层面——六种（Thread.State）



# 第二章、并发之共享模型

# 一、管程

## 1、共享问题

### 1）临界区

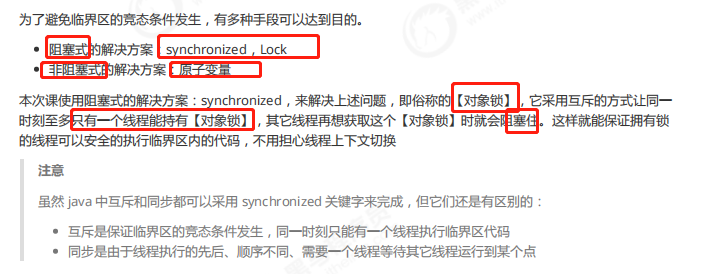
对共享数据读写的代码块

### 2）竞态条件

多个线程访问同一个临界区时，会发生指令执行顺序的不可预测性，及发生竞态条件

## 2、共享问题解决——Syncronized

### 1）概述





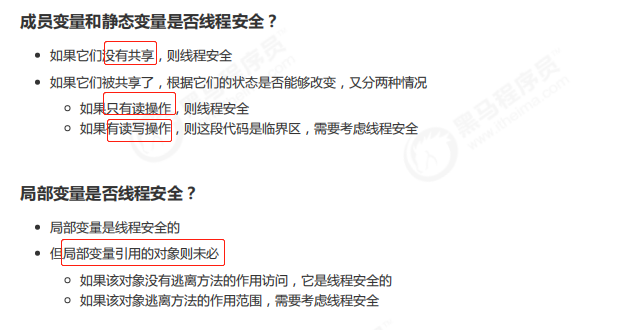
### 2）使用

1> 加在代码块上

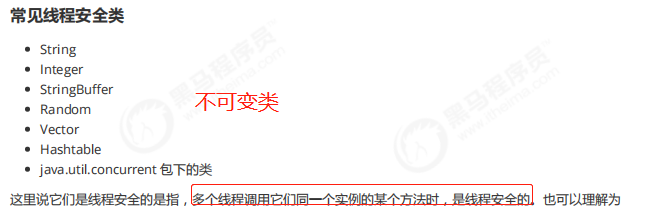
2> 加在成员方法上

3> 加在静态方法上

## 3、变量的线程安全分析



### 1）线程安全类



### 2）线程安全类方法的组合不一定是线程安全的



# 第三章、并发之非共享模型

设计模式

# 一、两阶段终止（interrupt() + isInterrupted()）

1、优雅的停止线程，让线程自己决定是否停止线程，给线程一个执行停止前操作的时机

2、不建议使用stop()方法去停止一个线程，可能该线程正占用同步锁，强制停止该线程会使得同步锁未被释放

