目录

[01\_JUC-课程介绍 2](#_Toc121074067)

[02\_JUC-JUC概述和进程线程概念1 2](#_Toc121074068)

[线程状态（6种） 2](#_Toc121074069)

[Wait和sleep 3](#_Toc121074070)

[并发和并行 3](#_Toc121074071)

[03\_JUC-JUC概述和进程线程概念2 3](#_Toc121074072)

[管程 3](#_Toc121074073)

[04\_JUC-JUC概述和进程线程概念3 4](#_Toc121074074)

[用户线程和守护线程 4](#_Toc121074075)

[05\_JUC-Synchronized复习和案例分析 4](#_Toc121074076)

[Synchronized复习 4](#_Toc121074077)

[卖票 5](#_Toc121074078)

[06\_JUC-Synchronized实现案例（卖票） 5](#_Toc121074079)

[07\_JUC-Lock（可重复锁）接口概述和实现案例 6](#_Toc121074080)

[Lock与的Synchronized区别 7](#_Toc121074081)

[Lock实现卖票 7](#_Toc121074082)

[08\_JUC-线程间通信-概述和案例分析 8](#_Toc121074083)

[Lock和synchronized有以下几点不同 8](#_Toc121074084)

[线程间通信（按照指定的线程顺序，因为线程start后不一定会立马创建，由操作系统决定） 9](#_Toc121074085)

[09\_JUC-线程间通信-Synchronized实现案例 9](#_Toc121074086)

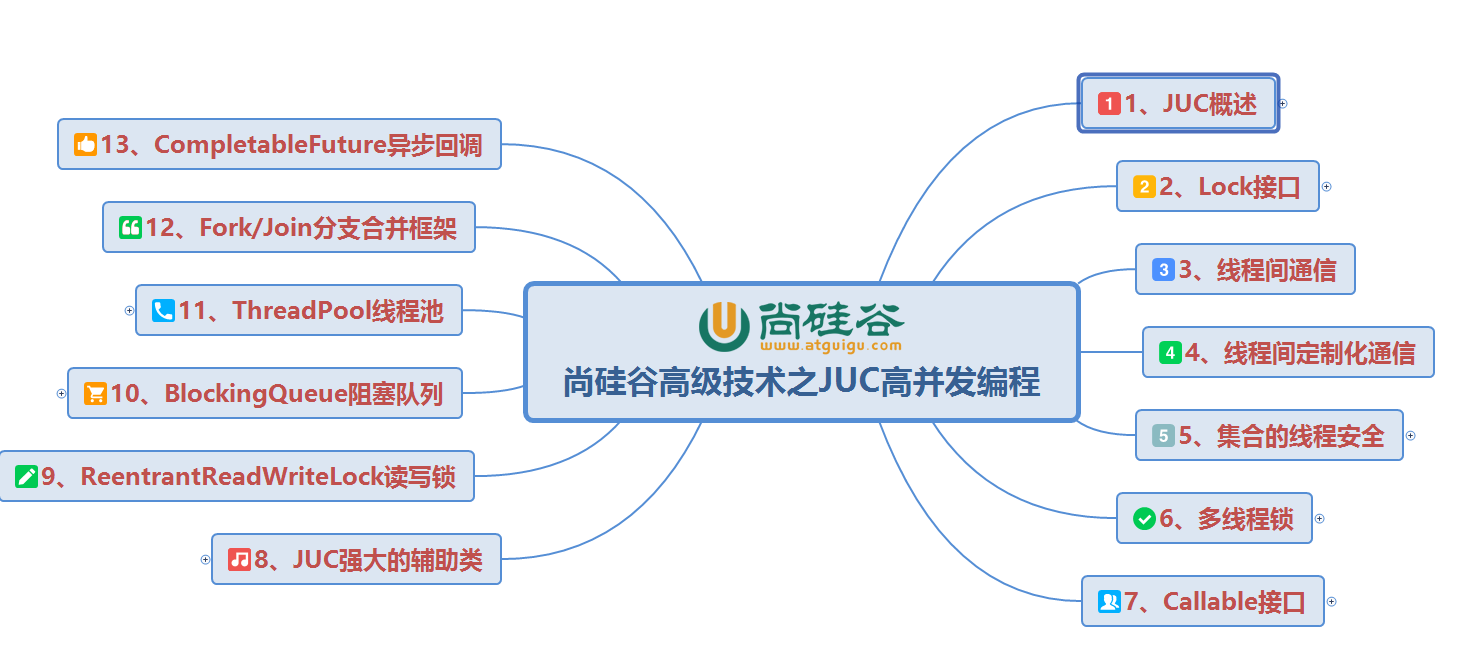
[一个线程加一然后通知另一个线程减一 10](#_Toc121074087)

[10\_JUC-线程间通信-虚假唤醒问题 11](#_Toc121074088)

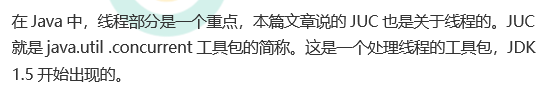
[解决虚假唤醒 12](#_Toc121074089)

[并发编程三部曲 14](#_Toc121074090)

# 01\_JUC-课程介绍



# 02\_JUC-JUC概述和进程线程概念1



进程（Process）是系统进行资源分配和调度的基本单位。

线程（thread） 是操作系统能够进行运算调度的最小单位。

总结来说:

进程：指在系统中正在运行的一个应用程序；程序一旦运行就是进程；进程— —资源分配的最小单位。

线程：系统分配处理器时间资源的基本单元，或者说进程之内独立执行的一个 单元执行流。线程——程序执行的最小单位。

## 线程状态（6种）

New(新建)

Runnable(准备就绪)

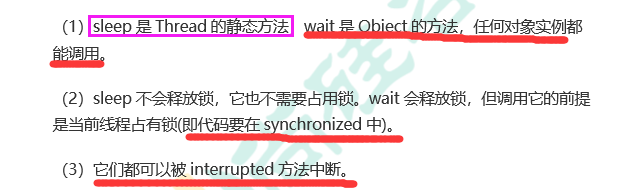
Blocked(阻塞)

Waiting(不见不散)

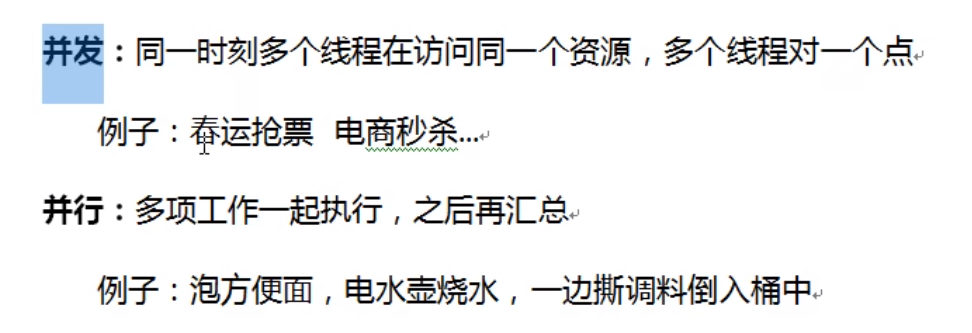
Timed\_waiting(过时不候)

Terminater(终结)

## Wait和sleep



## 并发和并行



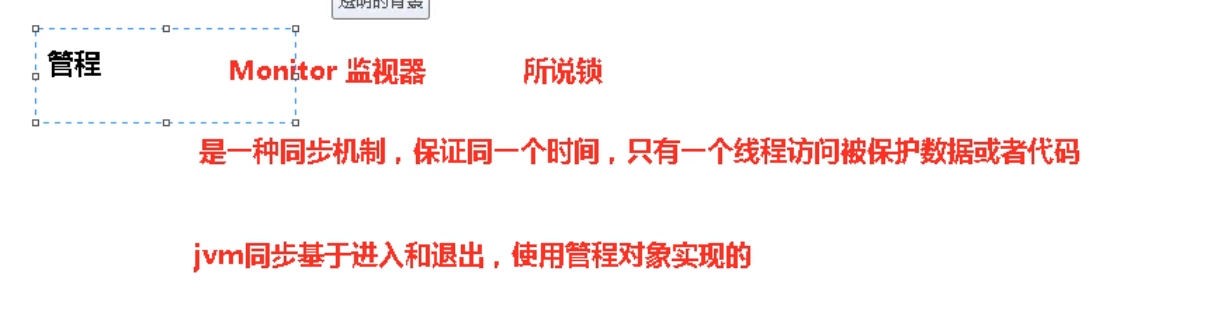
# 03\_JUC-JUC概述和进程线程概念2

## 管程

管程(monitor)是保证了同一时刻只有一个进程在管程内活动,即管程内定义的操作在同 一时刻只被一个进程调用(由编译器实现).但是这样并不能保证进程以设计的顺序执行

JVM中同步是基于进入和退出管程(monitor)对象实现的，每个对象都会有一个管程 (monitor)对象，管程(monitor)会随着java对象一同创建和销毁

执行线程首先要持有管程对象，然后才能执行方法，当方法完成之后会释放管程，方 法在执行时候会持有管程，其他线程无法再获取同一个管程



# 04\_JUC-JUC概述和进程线程概念3

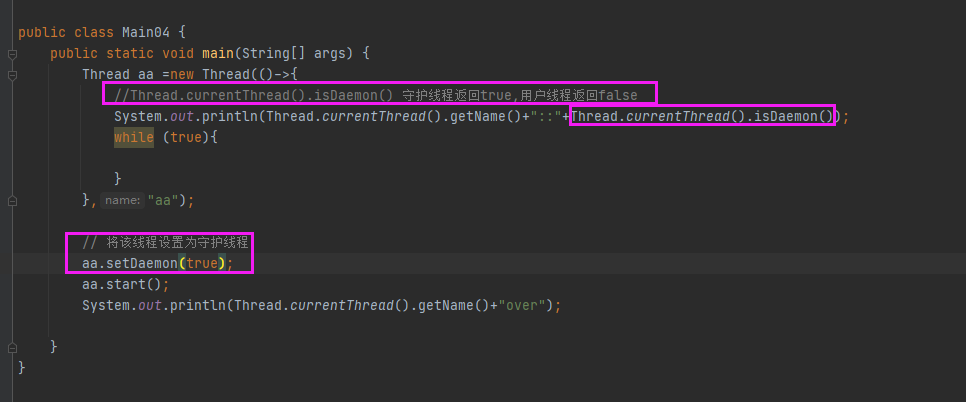
## 用户线程和守护线程

用户线程:平时用到的普通线程,自定义线程

守护线程:运行在后台,是一种特殊的线程,比如垃圾回收

当主线程结束后,用户线程还在运行,JVM存活

如果没有用户线程,都是守护线程,JVM结束



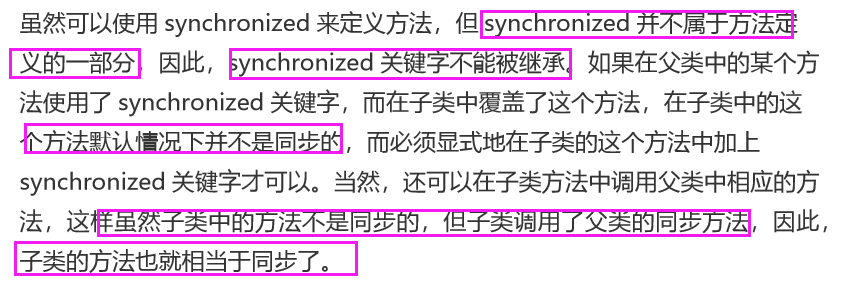
# 05\_JUC-Synchronized复习和案例分析

## Synchronized复习

1. 修饰一个代码块，被修饰的代码块称为同步语句块，其作用的范围是大括号{} 括起来的代码，作用的对象是调用这个代码块的对象；

2. 修饰一个方法，被修饰的方法称为同步方法，其作用的范围是整个方法，作用 的对象是调用这个方法的对象；

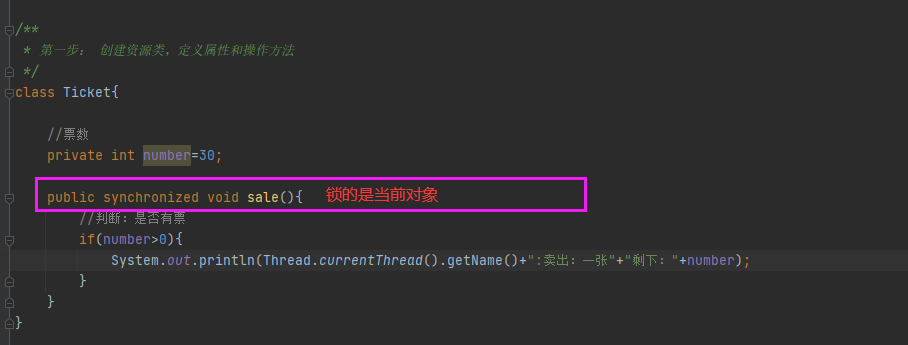
3. 修改一个静态的方法，其作用的范围是整个静态方法，作用的对象是这个类的 所有对象； 4. 修改一个类，其作用的范围是synchronized后面括号括起来的部分，作用主 的对象是这个类的所有对象。

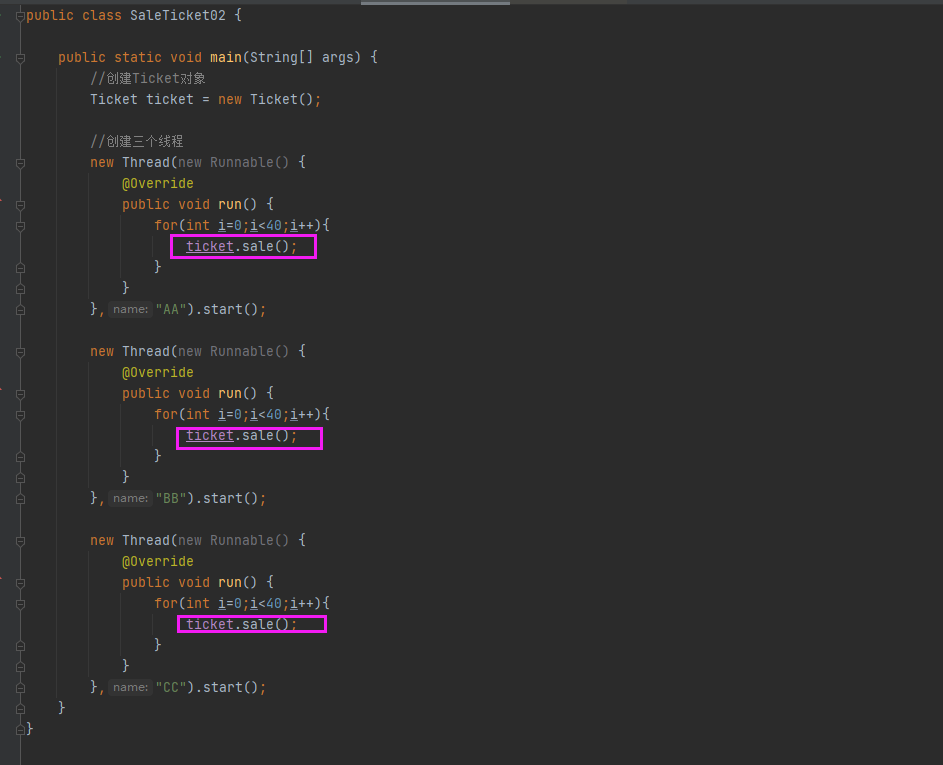


## 卖票



# 06\_JUC-Synchronized实现案例（卖票）



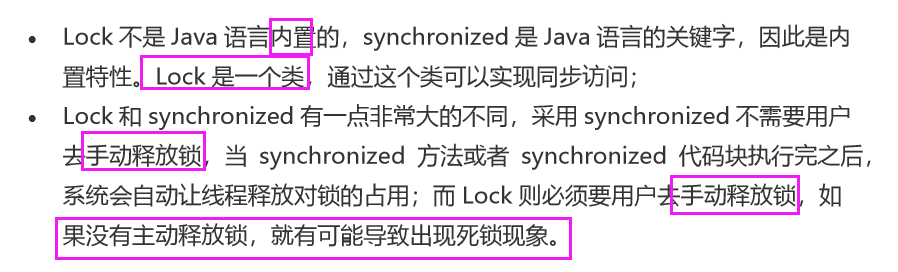




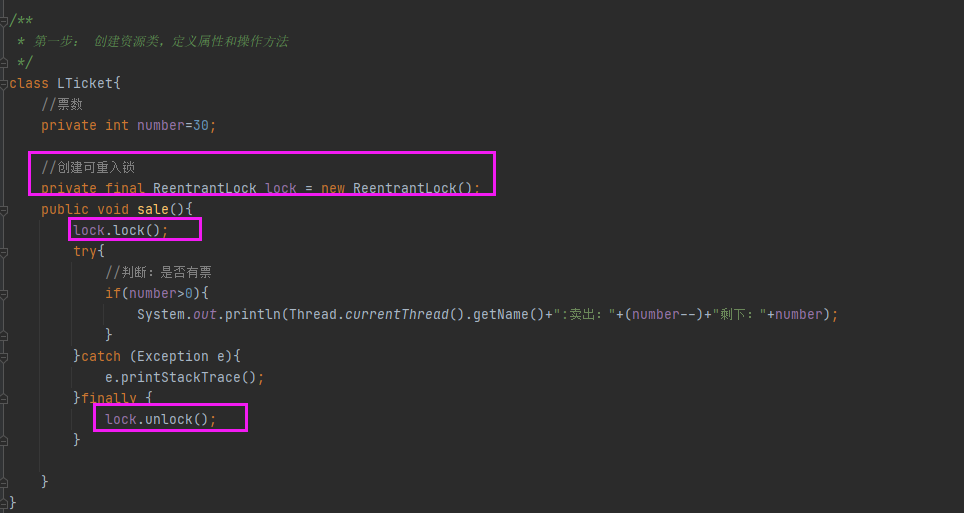
# 07\_JUC-Lock（可重复锁）接口概述和实现案例

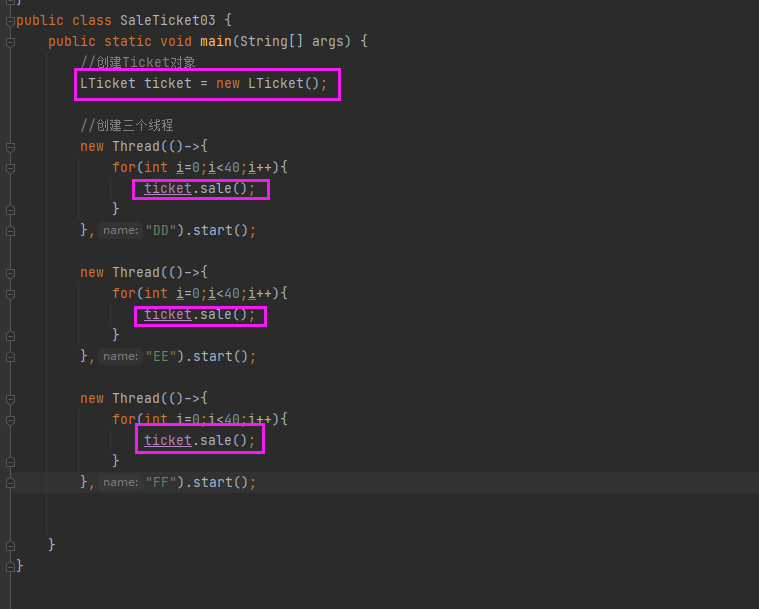
lock()方法是平常使用得最多的一个方法，就是用来获取锁。如果锁已被其他 线程获取，则进行等待。

## Lock与的Synchronized区别



## Lock实现卖票





# 08\_JUC-线程间通信-概述和案例分析

## Lock和synchronized有以下几点不同

1. Lock是一个接口，而synchronized是Java中的关键字，synchronized是内置的语言实现；

2. synchronized在发生异常时，会自动释放线程占有的锁，因此不会导致死锁现 象发生；而Lock在发生异常时，如果没有主动通过unLock()去释放锁，则很 可能造成死锁现象，因此使用Lock时需要在finally块中释放锁；

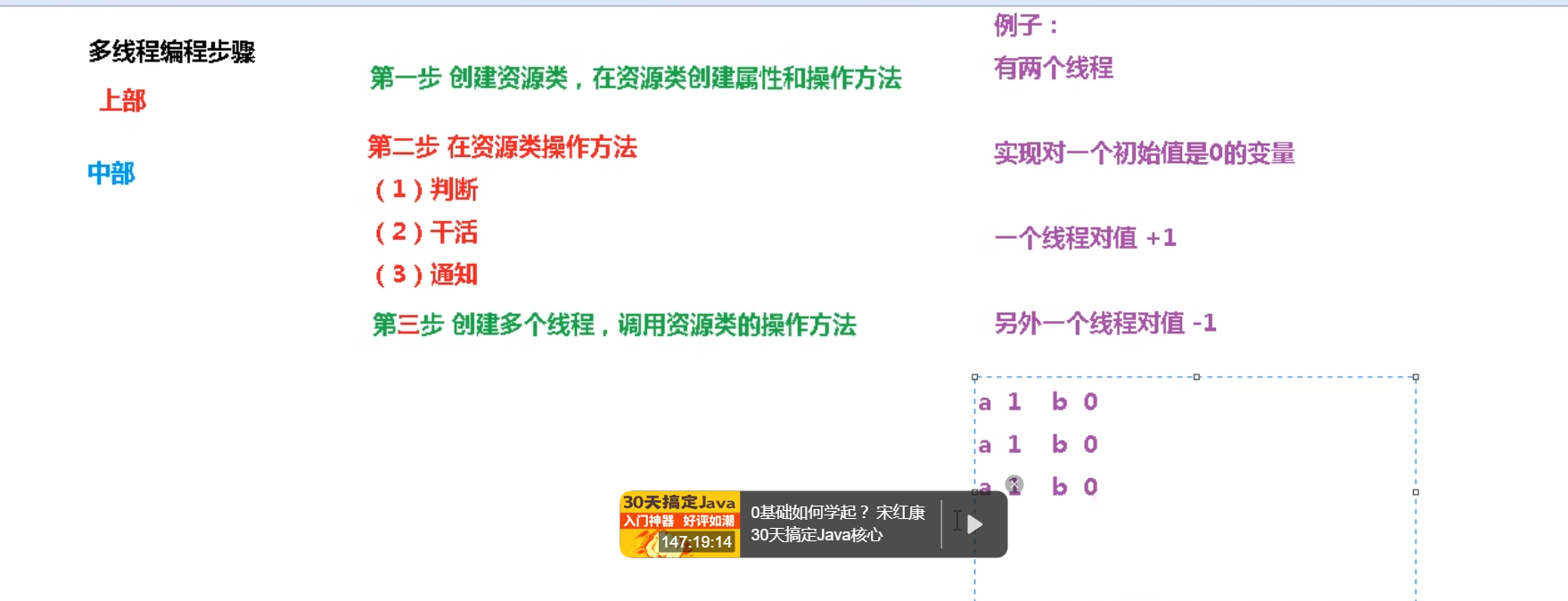
3. Lock可以让等待锁的线程响应中断，而synchronized却不行，使用 synchronized时，等待的线程会一直等待下去，不能够响应中断；

4. 通过Lock可以知道有没有成功获取锁，而synchronized却无法办到。

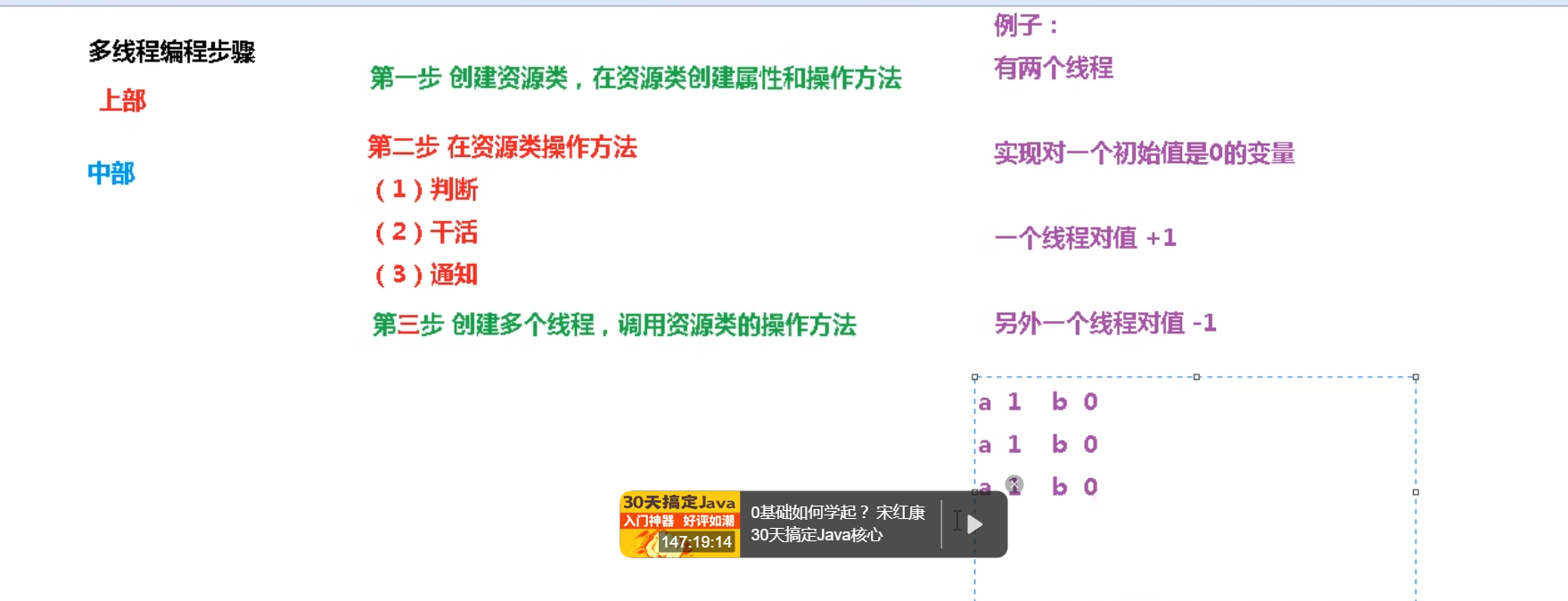
5. Lock可以提高多个线程进行读操作的效率。

在性能上来说，如果竞争资源不激烈，两者的性能是差不多的，而当竞争资源 非常激烈时（即有大量线程同时竞争），此时Lock的性能要远远优于 synchronized。

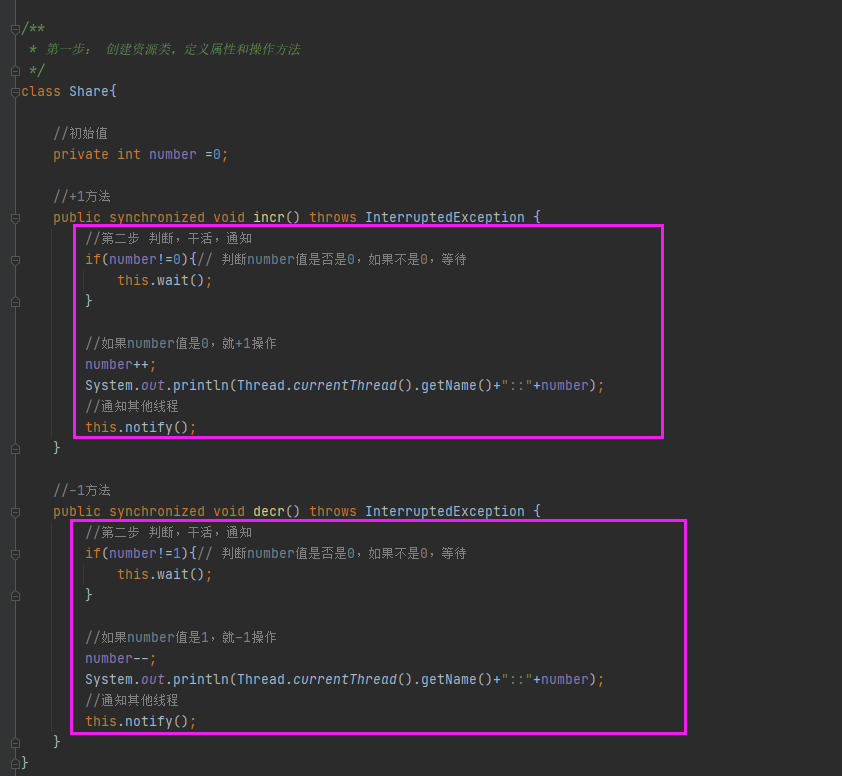
## 线程间通信（按照指定的线程顺序，因为线程start后不一定会立马创建，由操作系统决定）

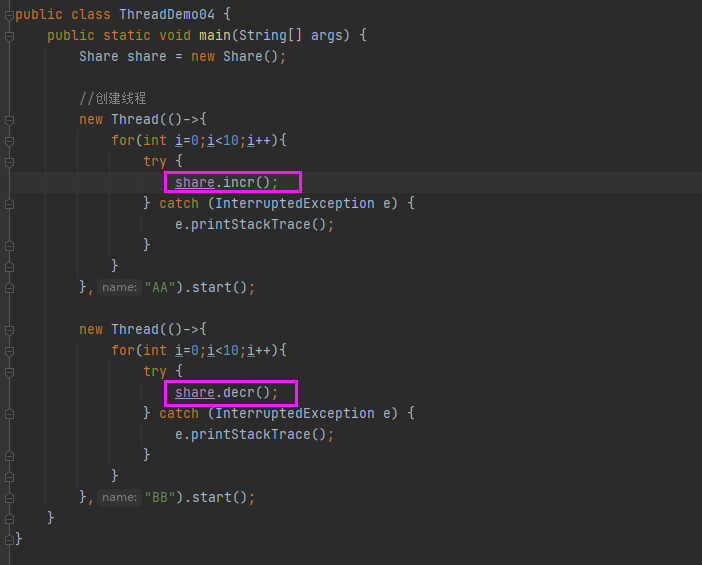


# 09\_JUC-线程间通信-Synchronized实现案例

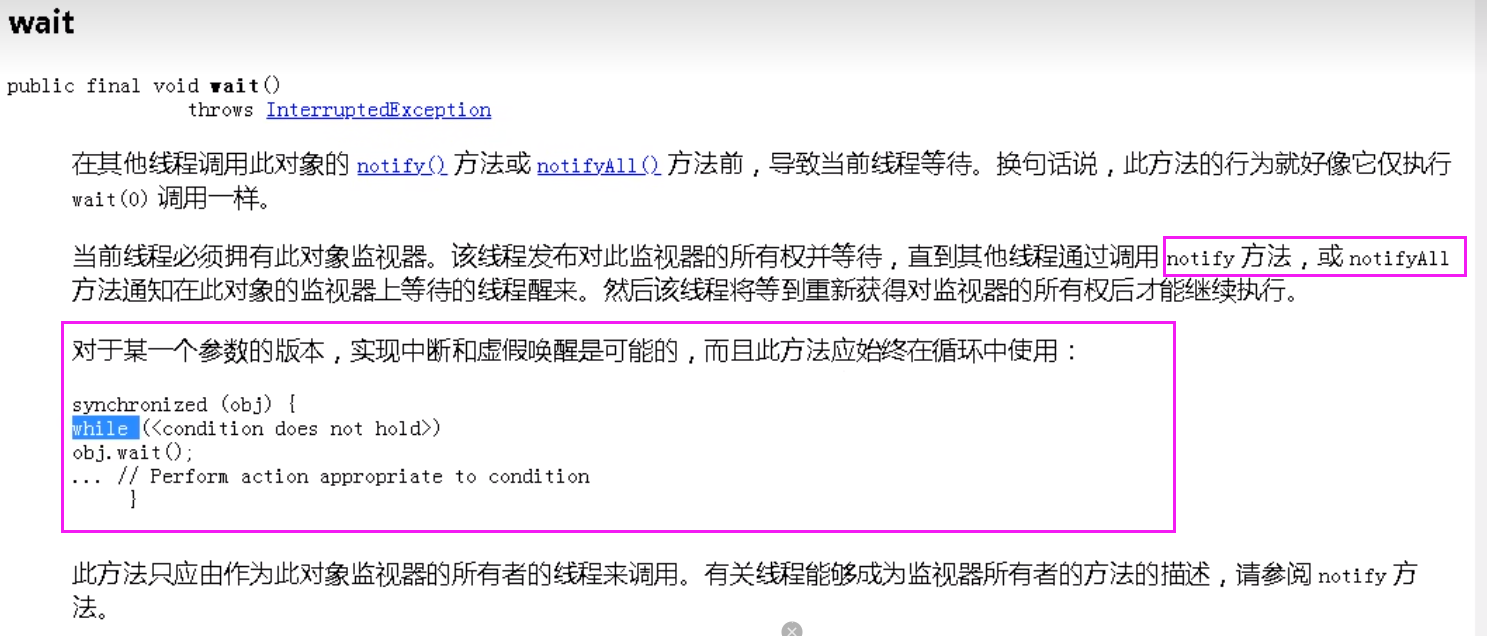


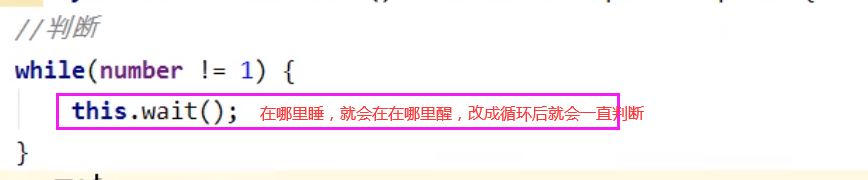
## 一个线程加一然后通知另一个线程减一



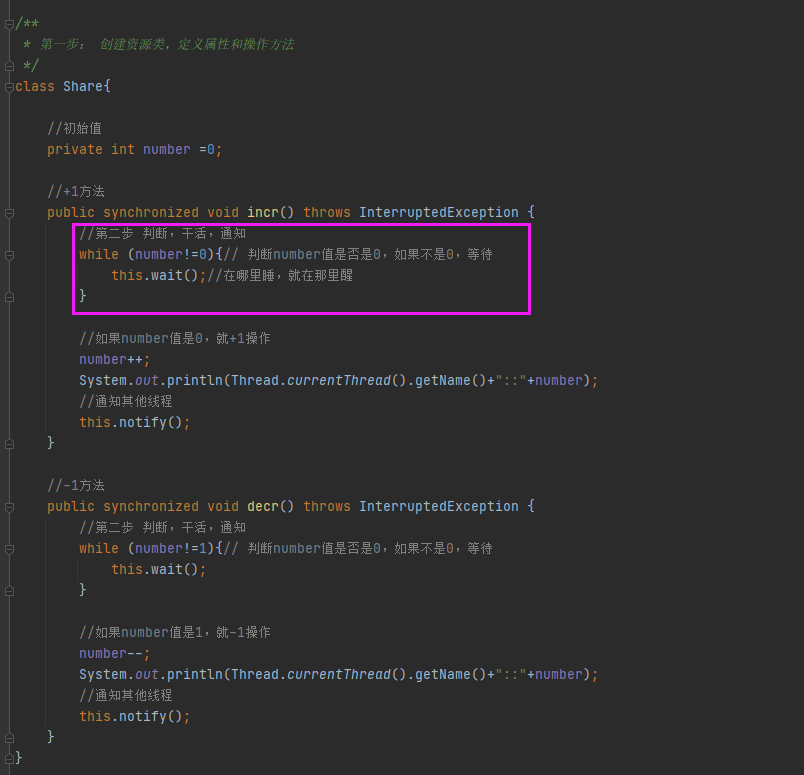


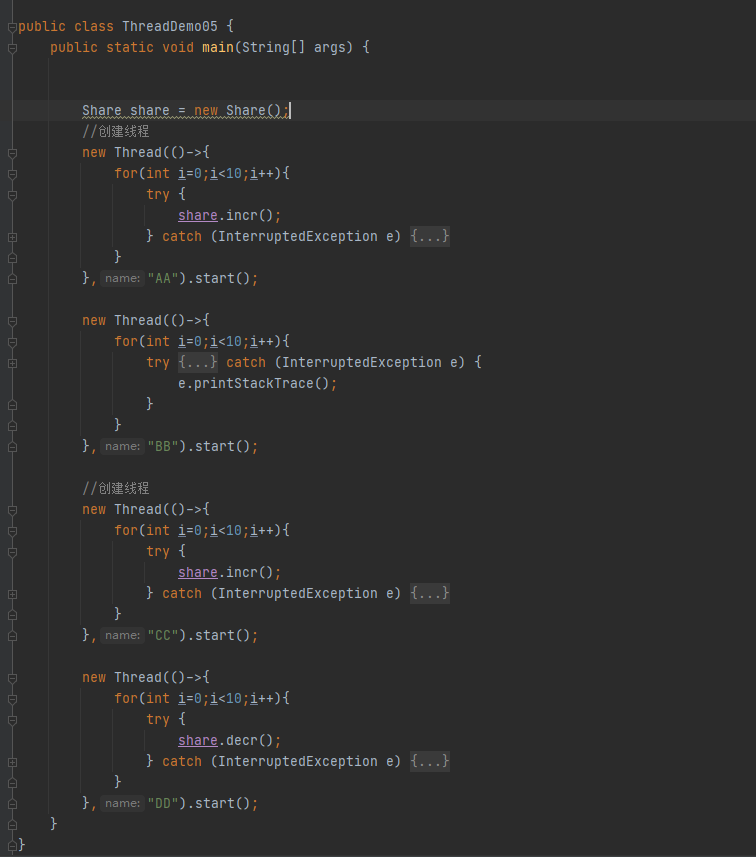
# 10\_JUC-线程间通信-虚假唤醒问题





## 解决虚假唤醒

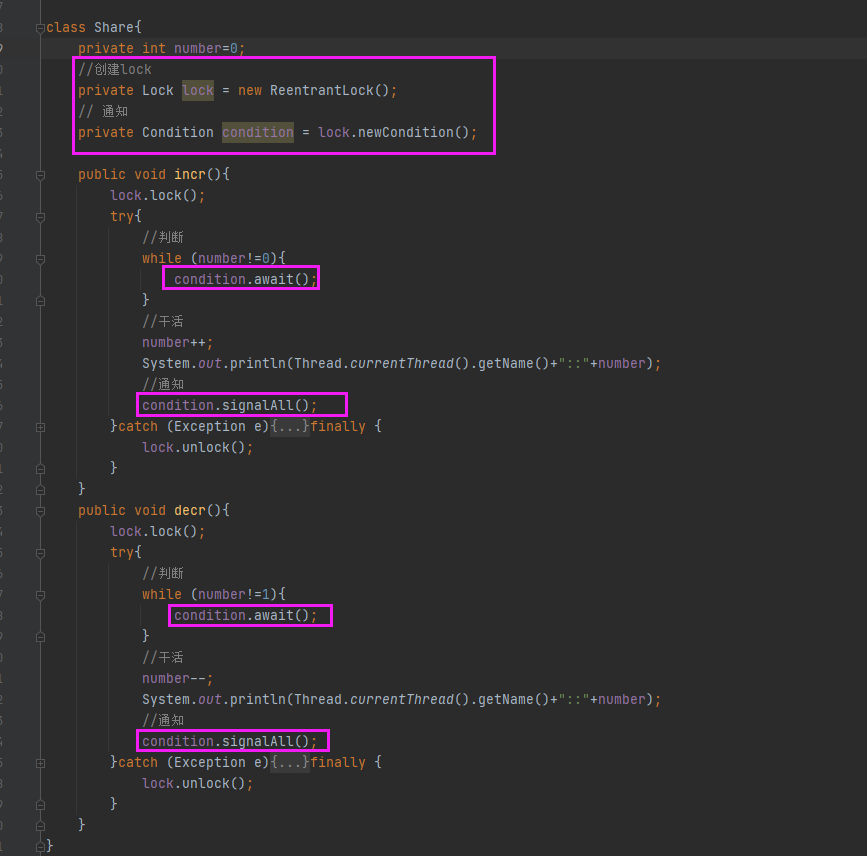


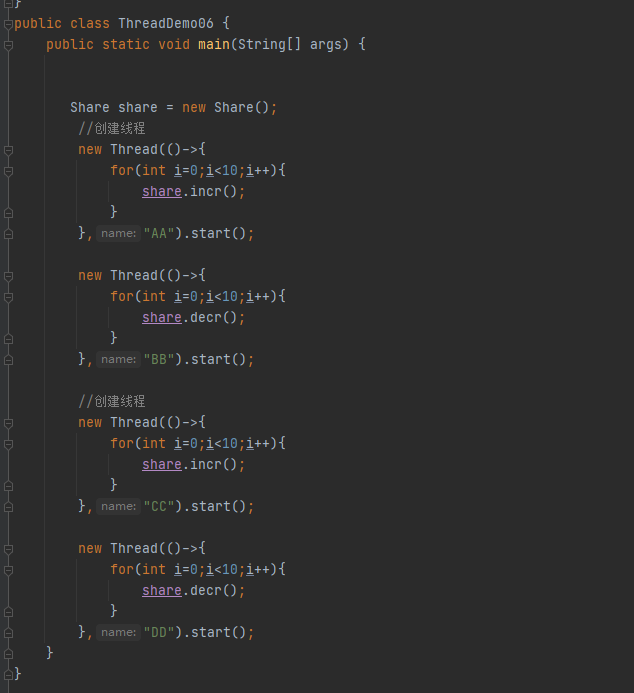


## 并发编程三部曲



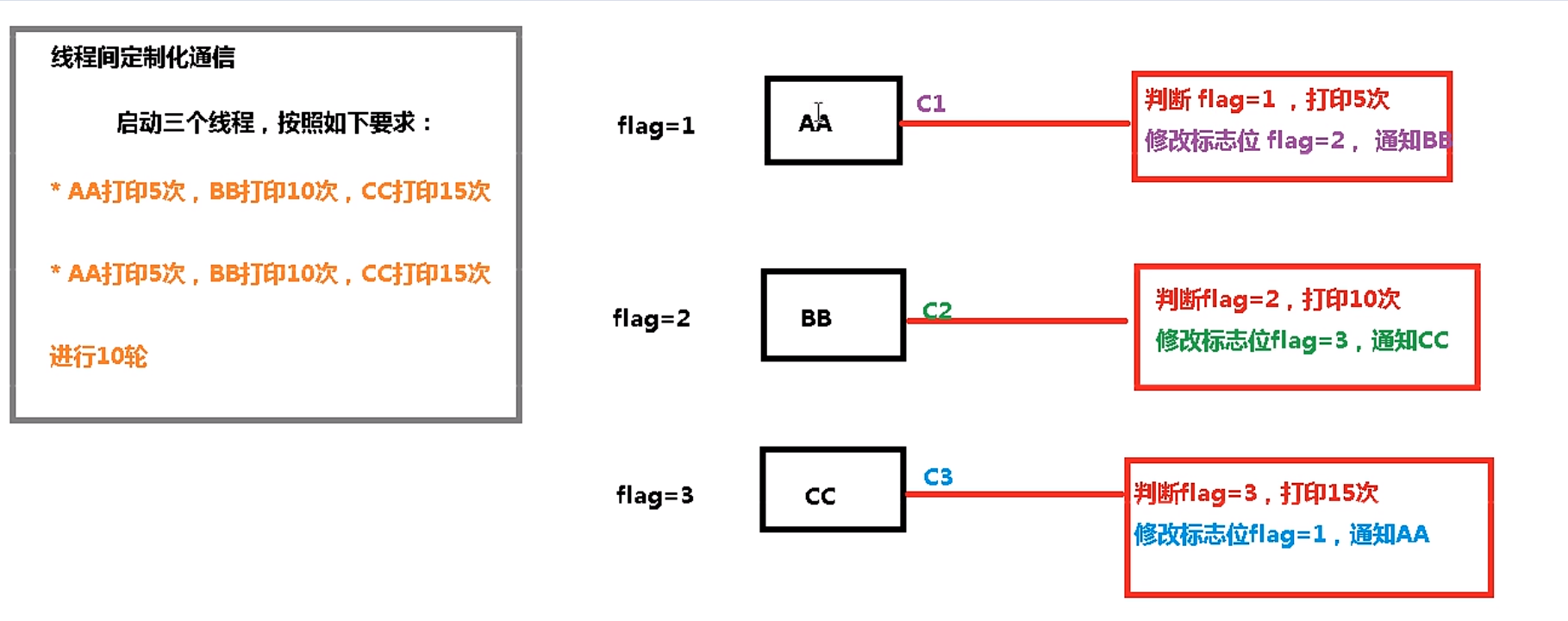
# 11\_JUC-线程间通信-Lock实现案例



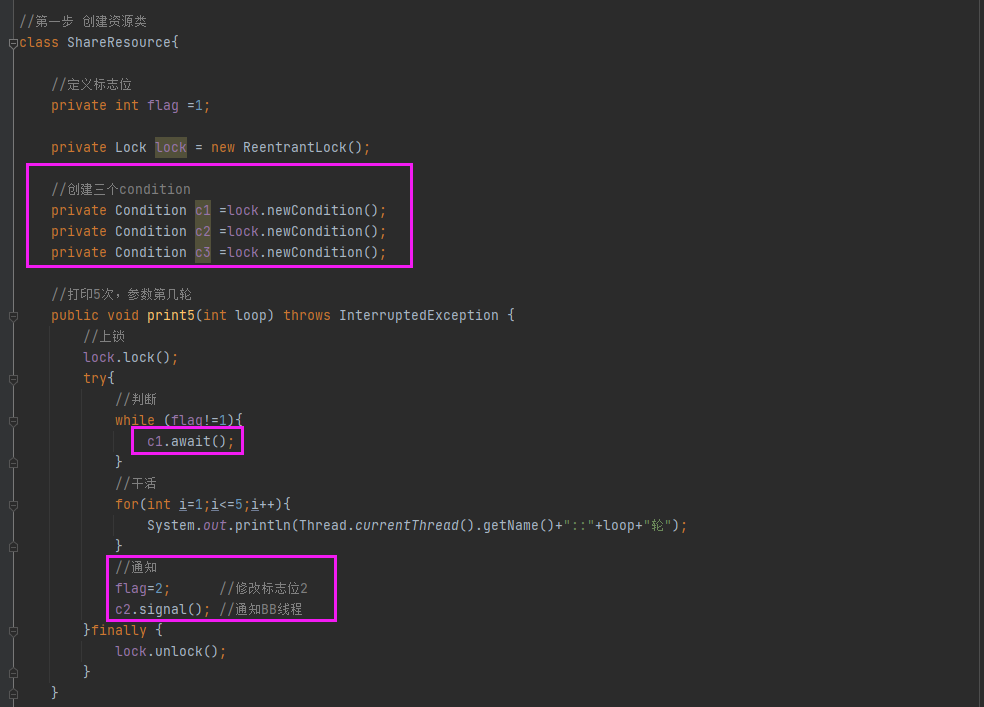


# 12\_JUC-线程间定制化通信-案例分析

==问题: A线程打印5次A，B线程打印10次B，C线程打印15次C,按照 此顺序循环10轮==



# 13\_JUC-线程间定制化通信-案例实现







# 14\_JUC-集合线程安全-异常演示





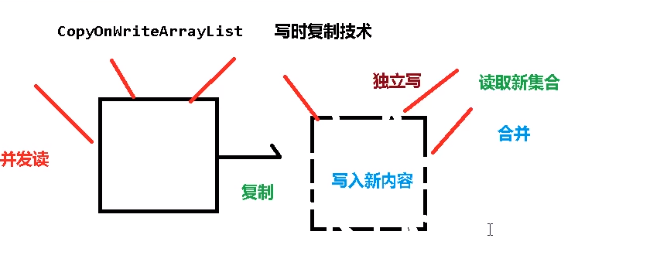
# 15\_JUC-集合线程安全-ArrayList线程 不安全和解决方案一





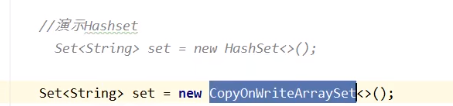
# 16\_JUC-集合线程安全-ArrayList线程 不安全和解决方案二（签到）

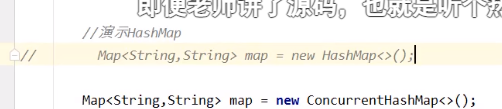




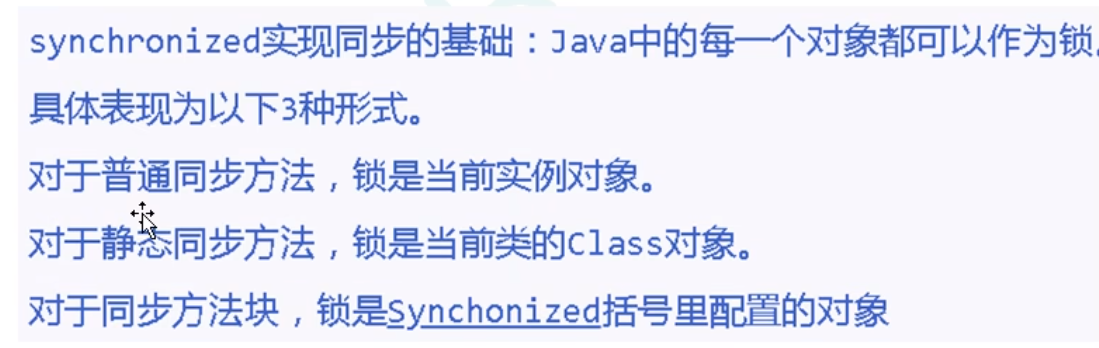


# 17\_JUC-集合线程安全-HashSet和HashMap线程不安全

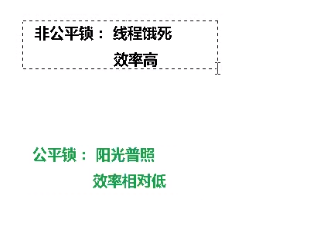


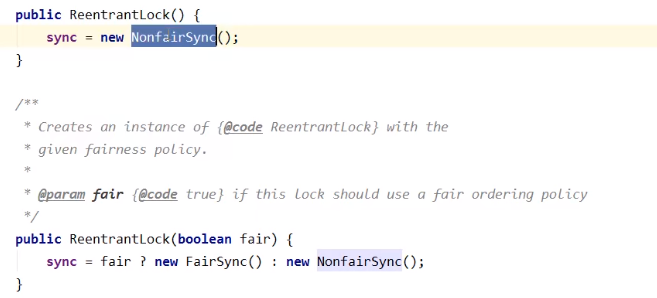


# 18\_JUC-多线程锁-Synchronized锁的8中情况

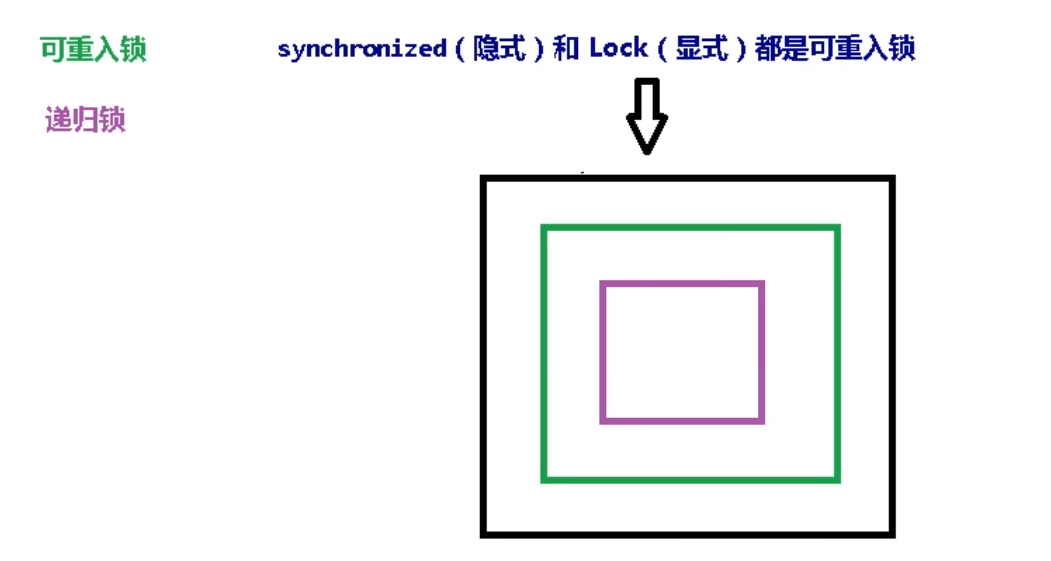


# 19\_JUC-多线程锁-公平锁和非公平锁





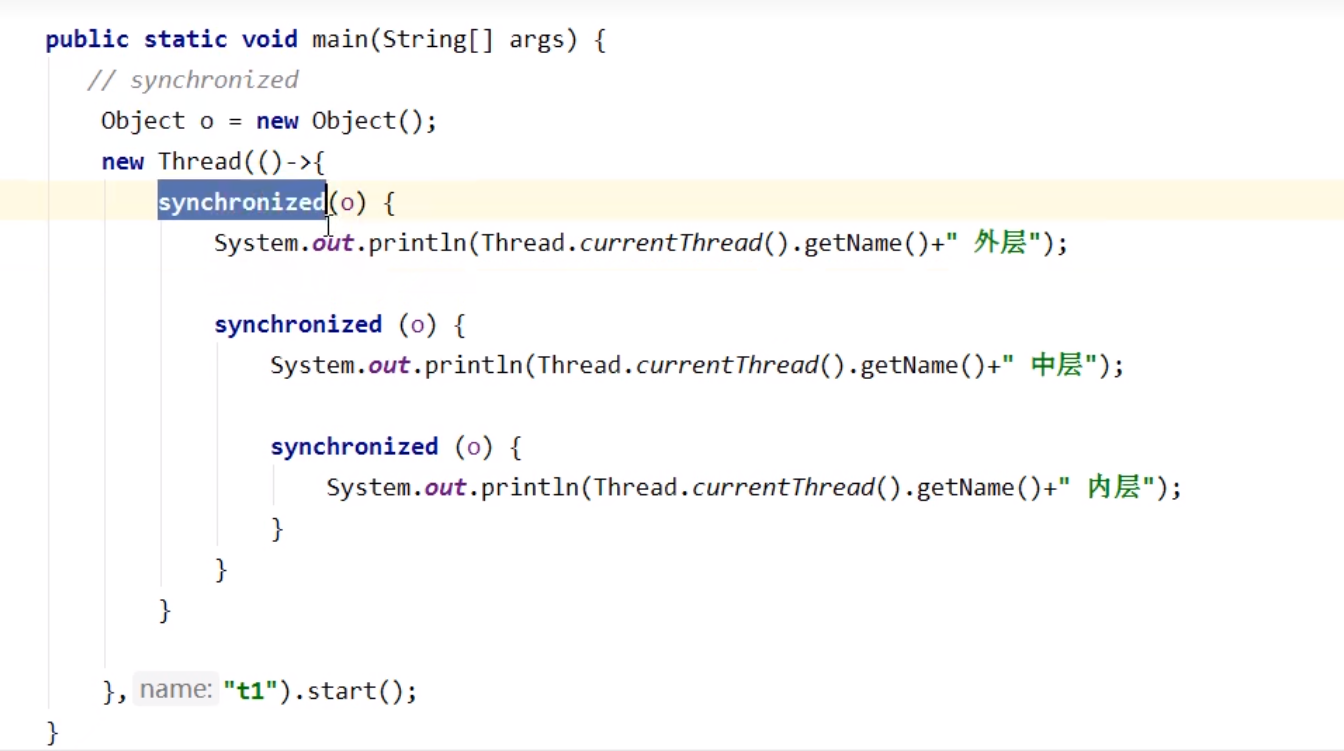
# 20\_JUC-多线程锁-可重入锁1



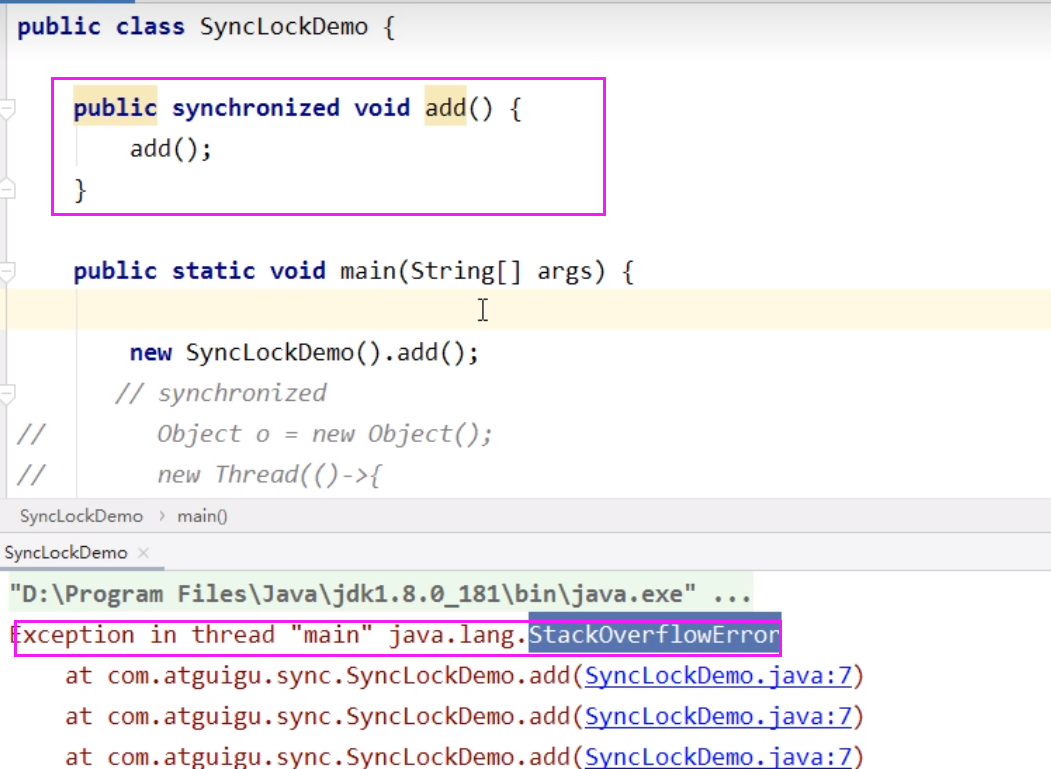
隐士就是自动申请和释放

显示就是手动申请和释放

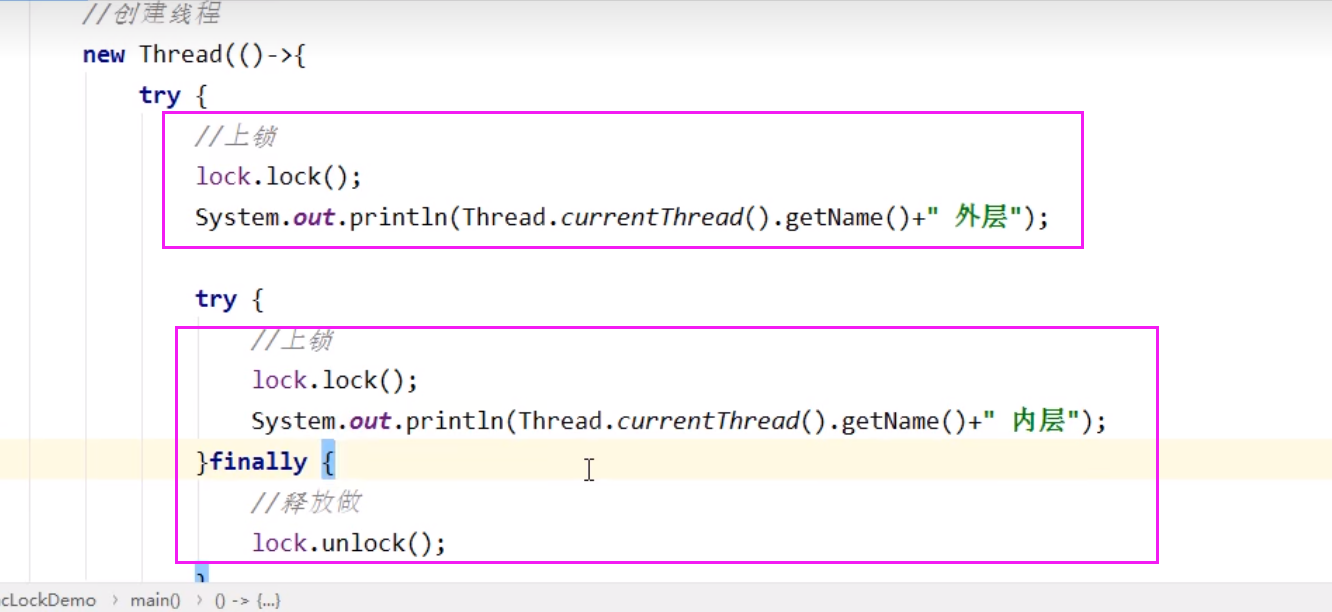
## 例子1·

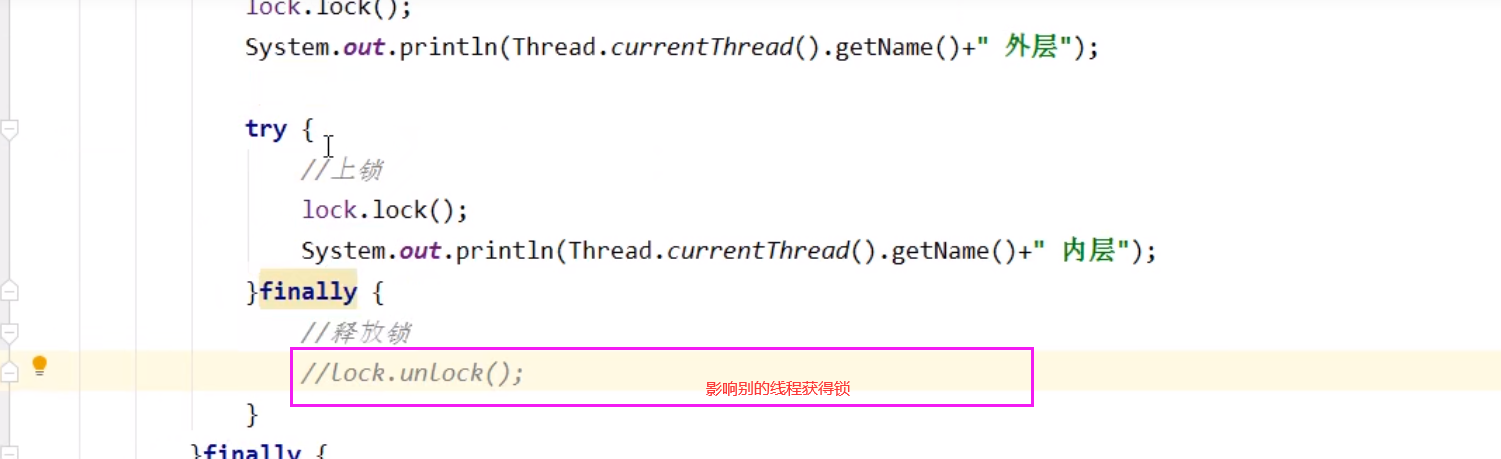


## 例子2



# 21\_JUC-多线程锁-可重入锁2





# 22\_JUC-多线程锁-死锁

