Java 多线程系列课程03

------ Java 中的锁

本文档包含以下内容：

1. 学习目标
2. 临界资源、临界区
3. Volatile
4. synchronized
5. ReentrantLock
6. 读写锁
7. 锁优化
8. 死锁
9. 练习题

**前言：**

 \* lock加锁的过程  
         \* 以非公平性锁为例：  
         \* 1、当前线程通过CAS操作来抢占锁，抢占成功则修改锁状态为1，将线程信息记录到锁当中，返回  
         \* 2、否则抢占不成功  
         \*    2.1、获取当前锁的状态 getState  
         \*    2.2、当前锁状态为0，表示锁空闲，没有线程获取则当前线程通过CAS操作直接获取锁，成功则将锁状态，线程信息记录，返回  
         \*    2.3、当当前线程和获取锁的线程相同时：对锁状态+1操作，判断锁是否到达上限，到达则抛出异常，否则更新锁状态值，返回  
         \*  
         \* 以公平性锁为例：  
         \* 1、获取当前锁状态（int state）  
         \* 2、当前锁状态为0，表示没有线程获取锁，先判断请求锁线程队列（AQS中队列）头的线程是否和当前线程相同，相同则通过CAS操作来更新锁状态，  
         \*    更新成功则将锁线程信息记录到锁中，返回  
         \* 3、当当前线程和获取锁的线程相同时：对锁状态+1操作，判断锁是否到达上限，到达则抛出异常，否则更新锁状态值，返回  
         \*  
         \*  
         \* unlock释放锁的过程  
         \* 1、获取新的锁状态值（获取原来锁的状态值-1）  
         \* 2、判断当前释放锁线程和锁中线程信息是否一致，不一致则抛出异常  
         \* 3、当线程信息一致时  
         \*    3.1、判断锁状态是否是0，即锁不在被占用，将锁中当前线程信息清除掉  
         \*    3.2、当锁状态不为空闲状态，将最新锁状态值更细一下  
         \*/  
  
  
        /\*\*  
         \* condition也是一种通信机制，和wait、notify、notifyAll作用类似，但其操作更加丰富  
         \*  
         \*/  
  
  
课堂练习  
  
1、一个生产者、一个消费者，仓库大小为1，实现生产者、消费者模型（要求：使用ReentrantLock锁及其Condition机制）  
2、一个生产者、一个消费者，仓库大小为6，实现生产者、消费者模型（要求：使用BlockingQueue阻塞队列）  
3、三个生产者、三个消费者，仓库大小为6，实现生产者、消费者模型（要求：生产者、消费者可以并发操作）