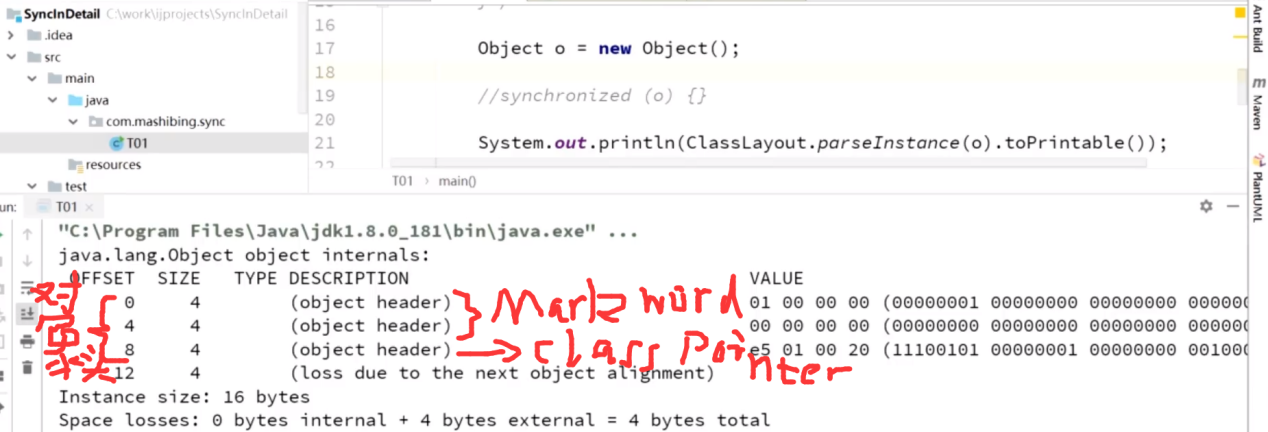
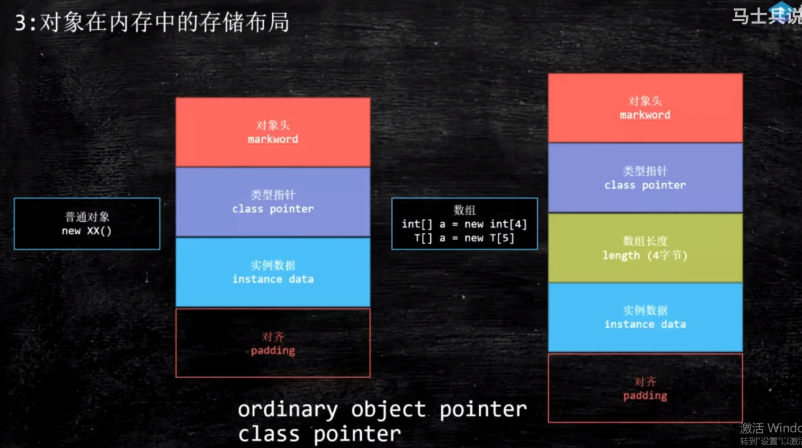
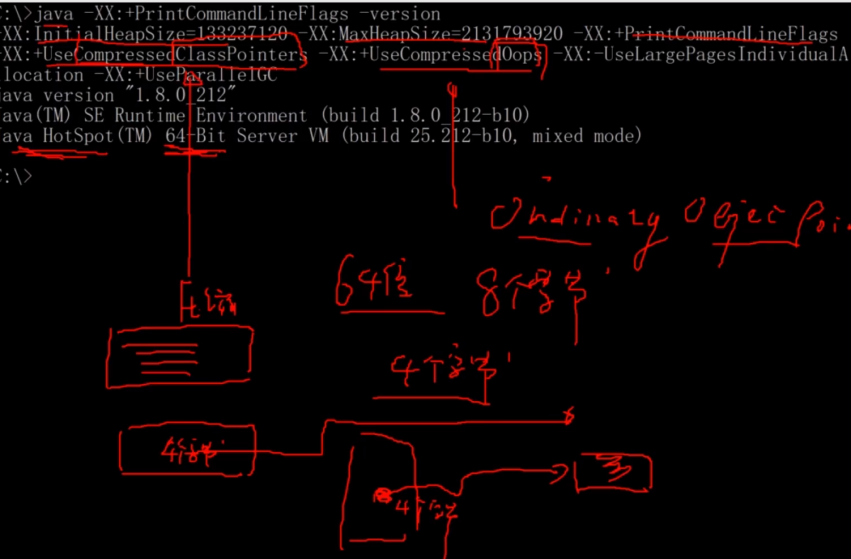
1. 对象在内存中布局

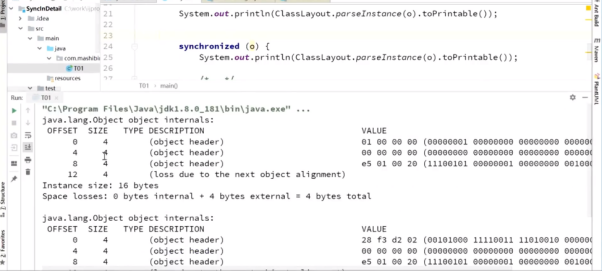




若开启Class对象压缩指针，或者普通对象压缩指针，则其引用都是4个字节（无论是32还是64位）



1. 锁信息存储位置，在MarkWord



1. 锁

无锁态、偏向锁、轻量级锁或者自旋锁或者无锁（CAS）：运行在用户态，效率高

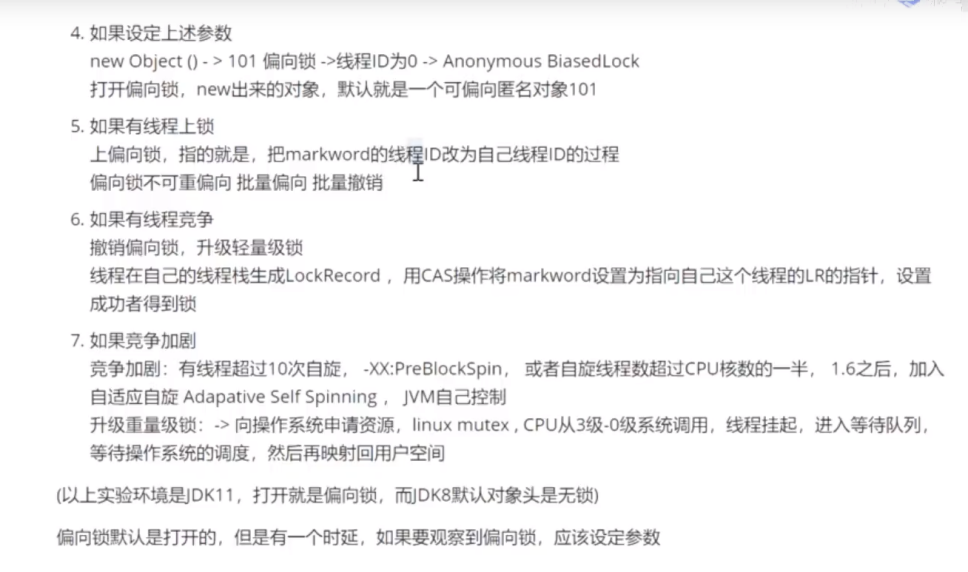
自旋锁：CAS操作，CPU不断的while循环判断，很消耗CPU

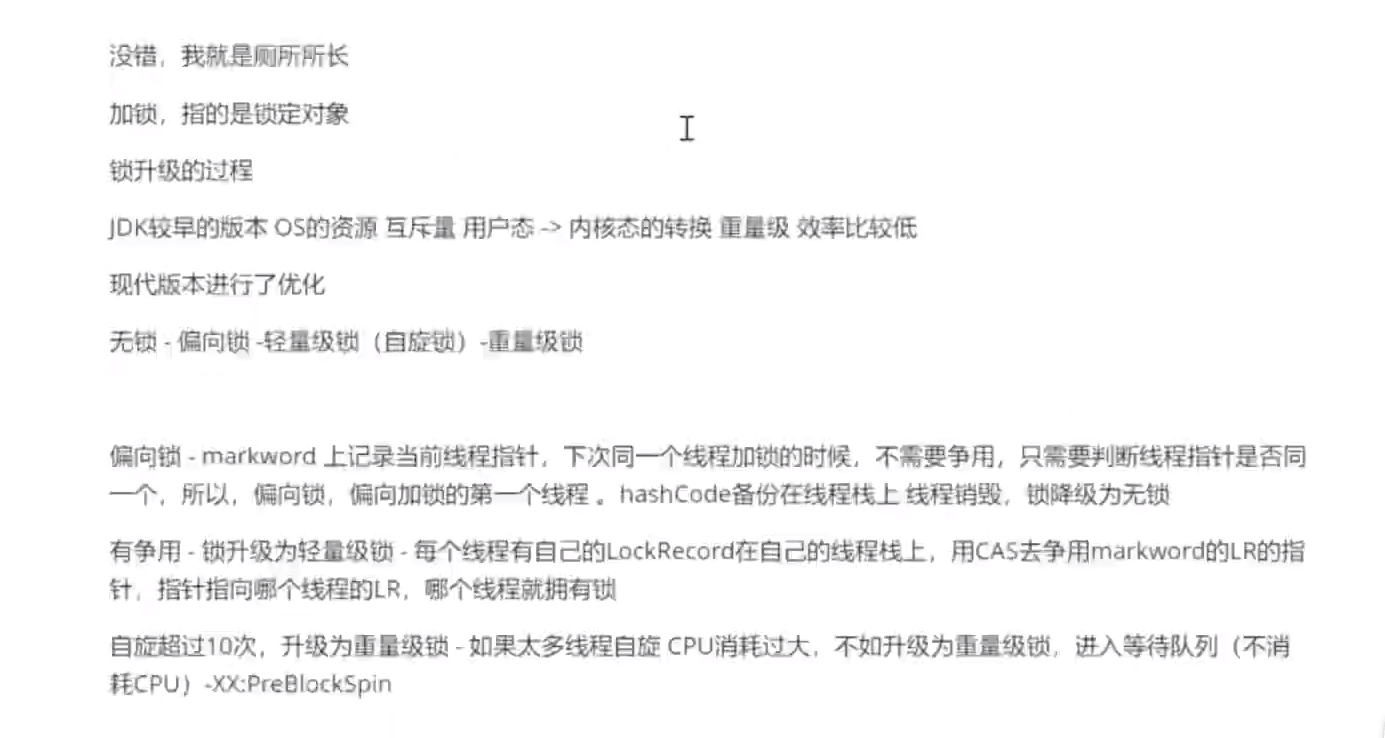
重量级锁：需要向内核申请锁（mutex），然后返回的是锁（互斥量）指针；当申请到锁时，会有个线程队列，只有一个线程在运行，其他线程阻塞或者wait，就不会消耗CPU；但是很耗费资源，因为需要状态转化（用户态<-->内核态），效率低

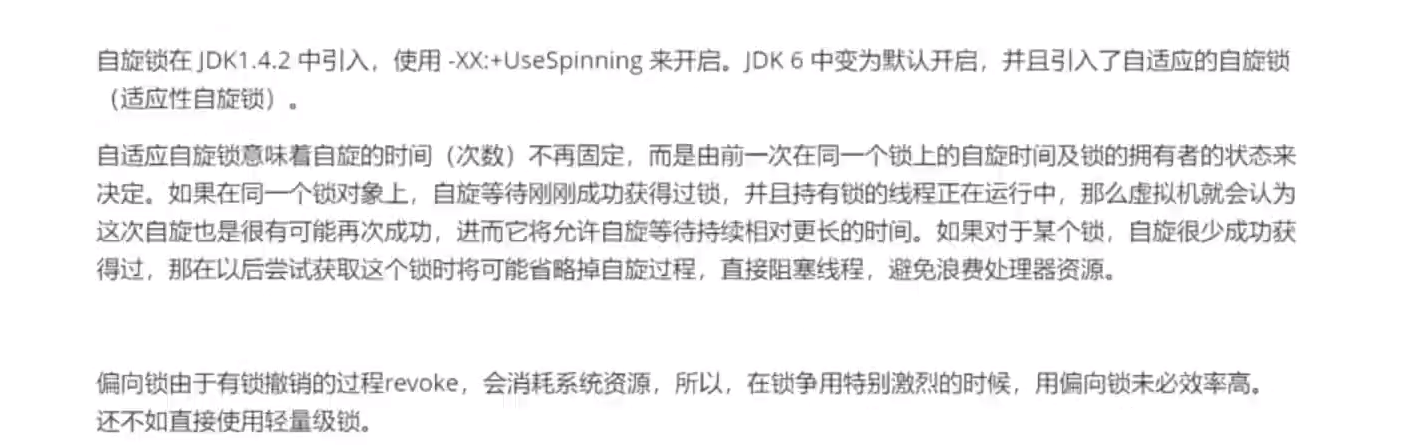


1. 锁的升级







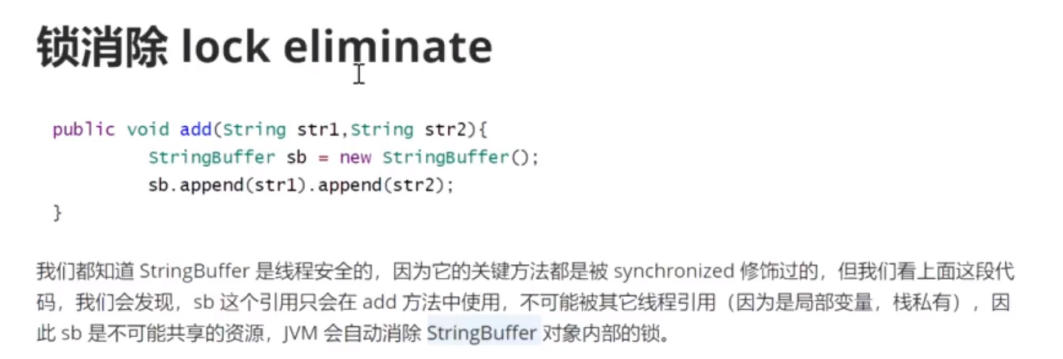


1. 锁的降级

发生在GC时候，因为没有使用当前对象了，锁降级也没有意义

1. 锁消除

对象为栈私有，即线程私有，不可能线程共享，因此可以把锁去掉



1. 锁粗化

若不在外面加锁，则CPU会不断加锁解锁，很耗费资源，直接在外面加锁

y