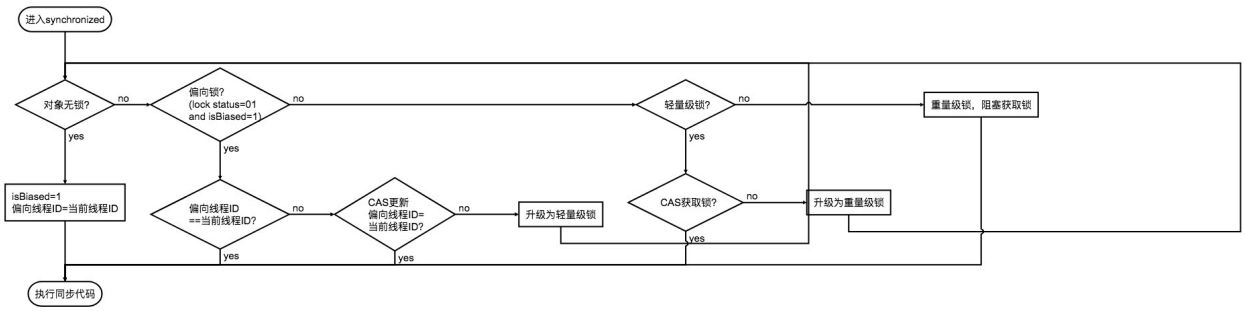
sychronized的原理

synchronized是jdk原生提供的锁，底层由偏向锁、轻量级和重量级锁来回切换实现。偏向锁并不算锁，它在对象头中直接记录偏向线程ID，认为不会发生锁竞争。轻量级是通过CAS来获取锁，重量级锁则需要通过阻塞等待来获取锁。
当线程执行到synchronized代码块时，会判断加锁对象头Mark Word上锁状态标识(lock status)，以此来判断对象当前被加锁的级别。
锁状态标识(lock status)和是否偏向共同表示5个状态

锁	lock status	是否偏向标识(isBiased)
无锁	01	0
偏向锁	01	1
轻量锁	00	
重量锁	10	
GC标识	11	

锁膨胀流程



锁优化

减少锁占用时间、减小锁粒度、锁分离、锁粗化、锁擦除
减少锁占用时间：减小加锁代码
减小锁粒度：将大对象改为小对象进行加锁，减少锁竞争。例如ConcurrentHashMap
锁分离：例如读写锁，读读不加锁，读写、写写才加锁
锁粗化：将多个锁合并成一个锁，减少锁获取与释放的开销，需要考虑成本
锁擦除：由JVM决定，例如StringBuffer虽然都带锁，但是在局部使用不产生并发，会将锁擦除