synchronized：原子性、可见性、有序性

- 同步方法（当前实例对象）
- 同步代码块
  - synchronized(this)：当前实例对象
  - synchronized(class)：类对象
  - synchronized(object)：任意实例对象；注意Integer对象等的线程池，null对象报错.
- 同步静态方法（类对象）

锁及锁的升级

synchronized用的锁存在Java对象头Mark Word里,有四种锁状态，从级别低到高为:无锁状态、偏向锁、
轻量级锁、重量级锁，这几个状态随着竞争情况而逐渐升级，但不能降级，这是为了提高获得、释放锁的效率.

锁状态	25bit		4bit	1bit	2bit
	23bit	2bit		是否是偏向锁	锁标志位
轻量级锁	指向栈中所记录的指针				00
重量级锁	指向互斥量（重量级锁）的指针				10
GC标志	空				11
偏向锁	线程ID	Epoch	对象分代年龄	1	01

JavaSE 1.6引入偏向锁和轻量级锁的原因?

```
private synchronized void testLock() {
        //doSomething();
}
```

- 重量级锁场景

  假设**多线程**t1,t2**同时**访问testLock方法，假设t1先拿到锁执行同步块内代码，则t2被阻塞等待t1释放锁;线程的挂起/唤醒需要CPU切换上下文，该过程的开销较大，因此称为重量级锁.

- 轻量级锁场景

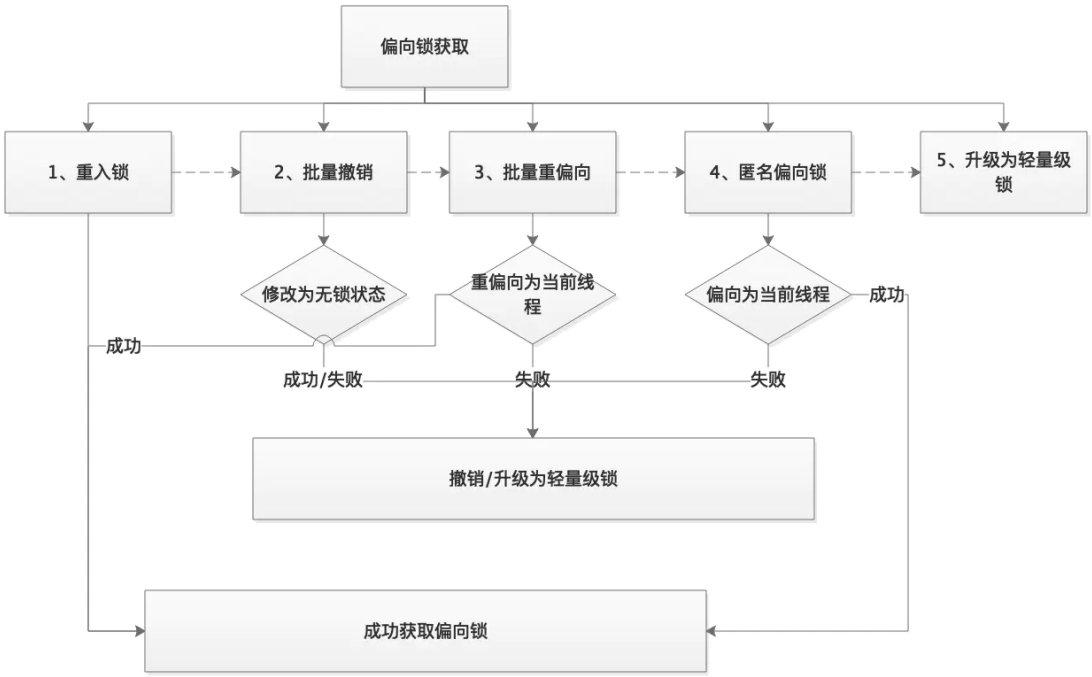
  假设多线程**多线程**t1,t2**交替**访问testLock方法(其它时间可能在访问非同步方法)，假设t1先拿到锁执行同步块内代码，此时t2在访问非同步方法，因此不存在竞争，就没必要加重量级锁将t2阻塞，为此引入了轻量级锁；轻量级锁，只需要使用CAS将对象头中的Mark Word替换为指向锁记录的指针，无需阻塞t2.

- 偏向锁场景

  假设**单线程**t1访问testLock方法，不存在线程竞争同步块，若每次使用CAS进行加锁，是对资源的浪费.为此，引入了偏向锁；偏向锁进行加锁时，只需将对象头中的Mark Word的线程号指向当前偏向线程，在单线程下(除首次，必然指向自己)，只需每次做个简单测试偏向线程是否为自己即可，无需进行CAS.

锁的加锁、释放(撤销)

- 偏向锁



假设有两个线程t1,t2,设定偏向锁开启:

获得

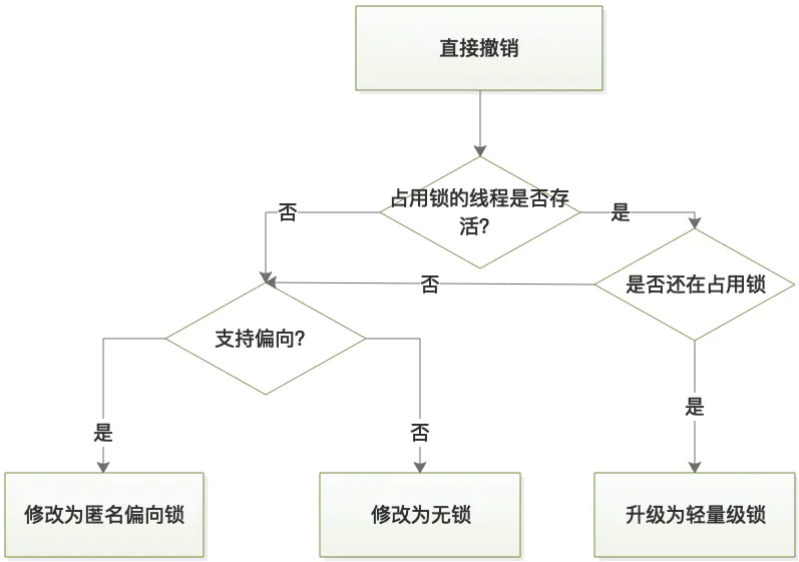
t1先获取偏向锁，此时，对象头中的MW没有偏向线程，即为上图4匿名偏向锁，t1用CAS将MW中偏向线程改为自己，获取偏向锁成功;

t1再次获取偏向锁，此时，对象头中的MW偏向线程为自己，即为上图1重入锁，直接获得锁;

撤销

t2来获取偏向锁，测试MW发现指向线程不是自己(5，t2用轻量级锁获取)，此时进入偏向锁撤销(对对象头而言)，升级为轻量级锁:

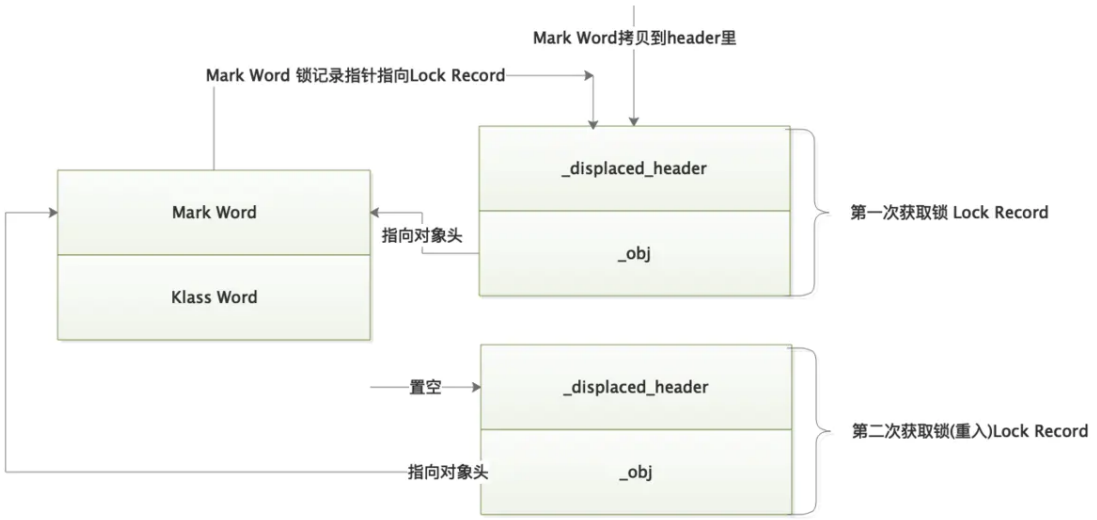
等待全局安全点，暂停偏向对象偏向的线程t1，检查占用锁的线程是否活着且继续占用；若继续占用则升级为轻量级锁(在t1线程中创建Lock Record，放置对象头MW，将对象头锁MW替换为指向LR的指针)，t1继续持有锁，否则根据是否支持偏向将对象头设置为无锁或者匿名偏向锁状态.



释放

当获得偏向锁的t1，自然(没有上面的升级、撤销情况)执行完同步区代码后，会对偏向锁进行释放，即
将Lock Record中_obj对象头指针置为null，但是注意**对象头中的MW依然偏向t1**(下次t1获取无需CAS).

- 轻量级锁



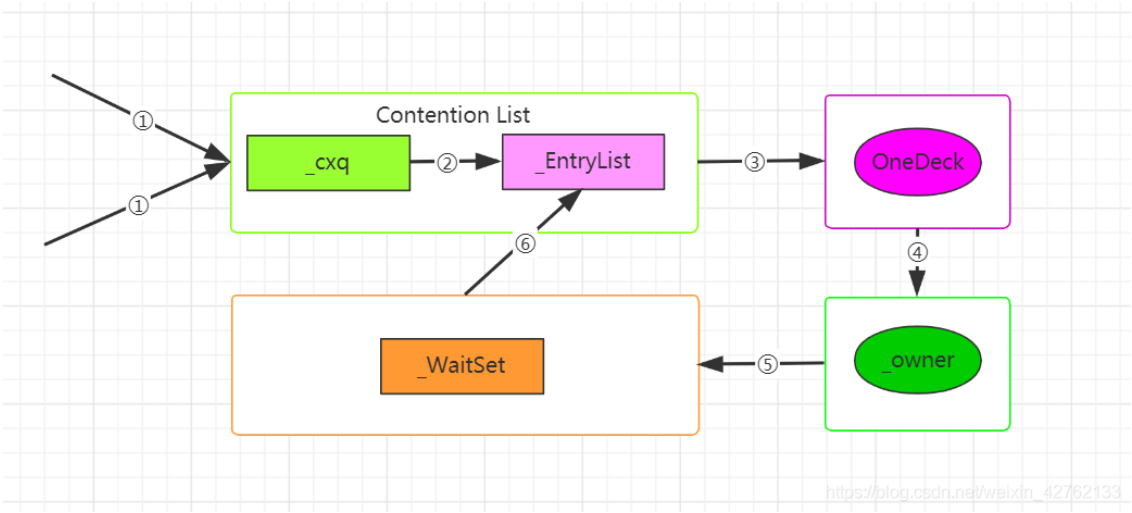
升级

接着上面t2获取偏向锁失败，锁升级为轻量级锁，t1继续持有的情况，t2会尝试用CAS获取轻量级锁，以自旋的方式(重复一定次数)，如果在自旋的过程中获得了轻量锁，则执行到结束，释放为无锁；否则，t2在自旋过程中未获得锁，则膨胀为重量级锁，对象头标志改为10，创建objectMonitor对象，对象头MW指向监视器对象，t1持有重量级锁，t2阻塞.

释放

持有轻量级锁的线程，正常执行完同步块，将锁释放为无锁.

- 重量级锁



获得

线程通过CAS将monitor对象中的_owner字段指向自己

释放

释放，同时唤醒阻塞线程

- tips(锁升级全过程)
  - t1 CAS 获取匿名偏向锁.
  - t2尝试获取偏向锁，发现偏向锁指向t1，获取失败.
  - t2失败后开始偏向锁撤销，如果t1还存活将轻量级锁指向它，它继续运行；t2尝试获取锁，开始自旋等待t1释放轻量级锁.
  - 如果在自旋过程中t1释放了锁，那么t2获取轻量级锁成功.
  - 如果在自旋结束后，t2未能获取轻量锁，那么锁升级为重量级锁，使t1持有objectmonitor对象，将t2加入EntryList，t2开始阻塞，等待t1释放监视器.
- **参考资料**
  - <<Java并发编程的艺术>>
  - Synchronized关键字深析（小白慎入，深入jvm源码，两万字长文）_Java新生代-CSDN博客
  - Java Synchronized 偏向锁/轻量级锁/重量级锁的演变过程 - 简书 (jianshu.com)