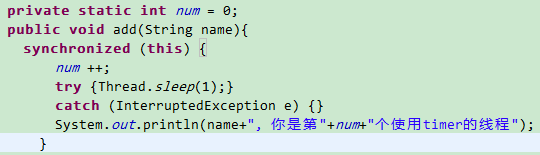
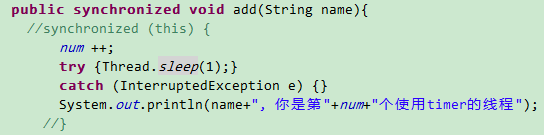
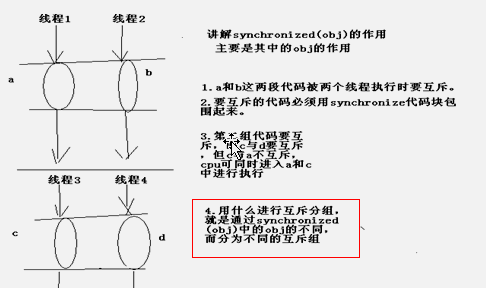
1. 在Java语言中，引入了对象互斥锁的概念，保证共享数据操作的完整性，每个对象都对应于一个可称为“互斥锁”的标记，这个标记保证在任意时刻，只能有一个线程访问该对象的同步方法
2. 关键词synchronized来表示对象的互斥锁联系，当某个对象synchronized修饰时，表明该对象在任意时刻只能由一个线程来访问(锁住的不是数据，而是存取数据的**方法**)。
3. 关键字synchronized(Object obj){}：锁定某一段代码，执行该段代码时候**锁定obj对象(里面的代码被组合成原子单位)**



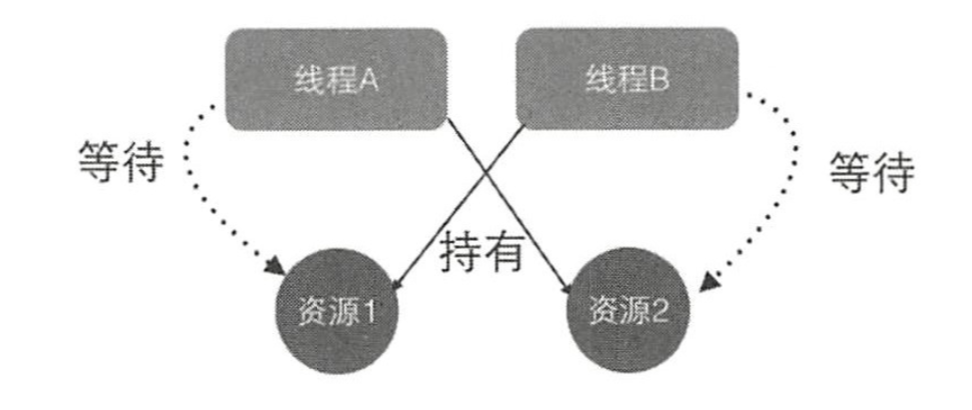
或者采用synchronized关键词(方法内的代码组成了原子单位)修饰整个方法，表示整个方法为同步方法，执行该方法的过程中，**当前对象被锁定，方法内的各种资源被加上了锁，其他同步方法不能访问其中的资源，但是非同步方法可以访问其中的资源。**



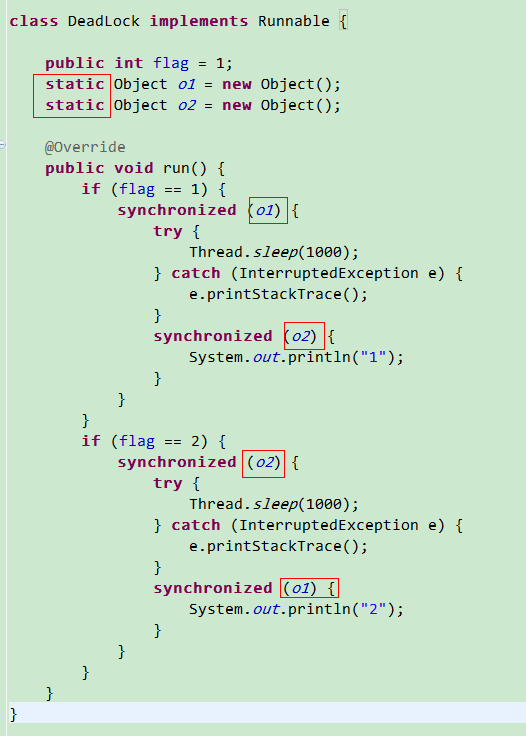


**在方法上采用synchronized关键字修饰，非static方法的同步锁为this，即当前调用对象；static方法的同步锁是当前类的字节码。**

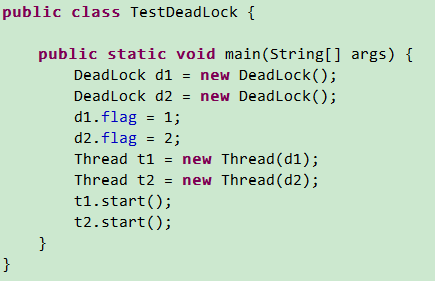
1. 线程死锁，两个或两个以上的线程在执行过程中，因争夺资源而造成的互相等待的现象， 在无外力作用的情况下，这些线程会一直相互等待而无法继续运行下去



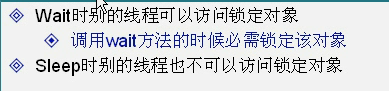
模拟死锁程序



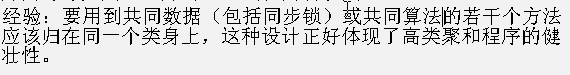
* 1. 线程死锁的必要条件
     1. 互斥条件，即互斥锁，资源只能由一个线程占用
     2. 请求并持有条件，一个线程己经持有了至少一个资源，但又提出了新的资源请求，而新资源己被其他线程占有，所以当前线程会被阻塞， 但阻塞的同时并不释放自己已经获取的资源
     3. 不可剥夺条件，线程获取到的资源在自己使用完之前不能被其他线程抢占，只有在自己使用完毕后才由自己释放该资源
     4. 环路等待条件，在发生死锁时，必然存在一个线程→资源的环形链
  2. 如何避免死锁，破坏形成死锁的任何一个条件即可避免死锁，目前只有『请求并持有』和『环路等待』条件是可以破坏的
     1. 申请锁资源的时候按序申请可以避免死锁



1. 同步化的意义是指定某段工作要在不能分割的状态(**原子状态**)下执行，也就是说单独的操作不重要，重要的是有多个步骤的方法
2. 锁释放与获取的**内存语义**
3. 当线程**释放锁**时，JMM会把该线程对应的本地内存中的*共享变量***刷新到主内存中**
4. 当线程**获取锁**时，JMM会把该线程对应的本地内存**置为无效**，从而使得被监视器保护的临界区代码必须要从主内存中去读取*共享变量*
5. wait、sleep方法的区别



1. 经验



1. 例：子线程循环10次，接着主线程循环100，接着又回到子线程循环10次，接着再回到主线程又循环100，如此循环50次，请写出程序。

