Extend Thread 和 implement runnable 的区别

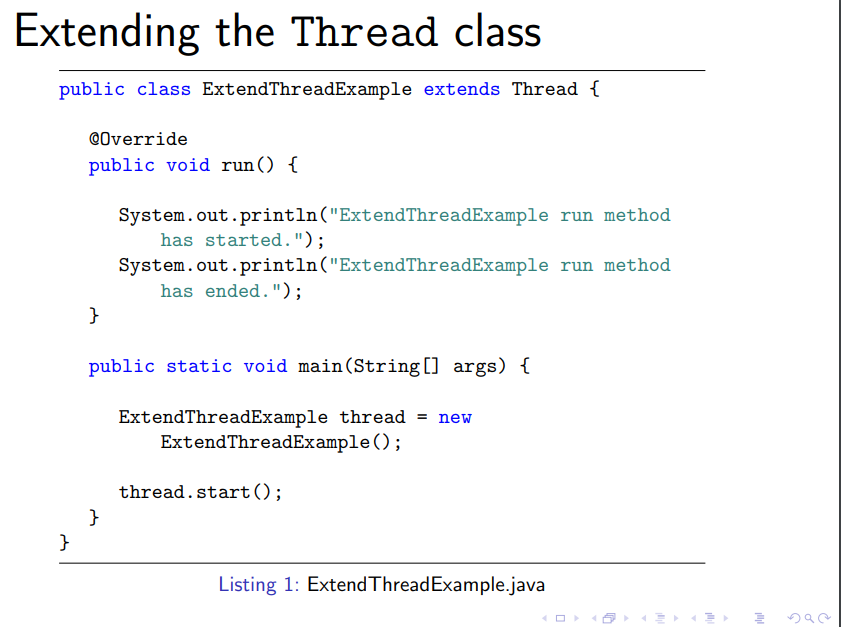
Because runnable is an interface, therefore we are forced to rewrite the function in runnable, and there’s only 1 function in runnable which is run().

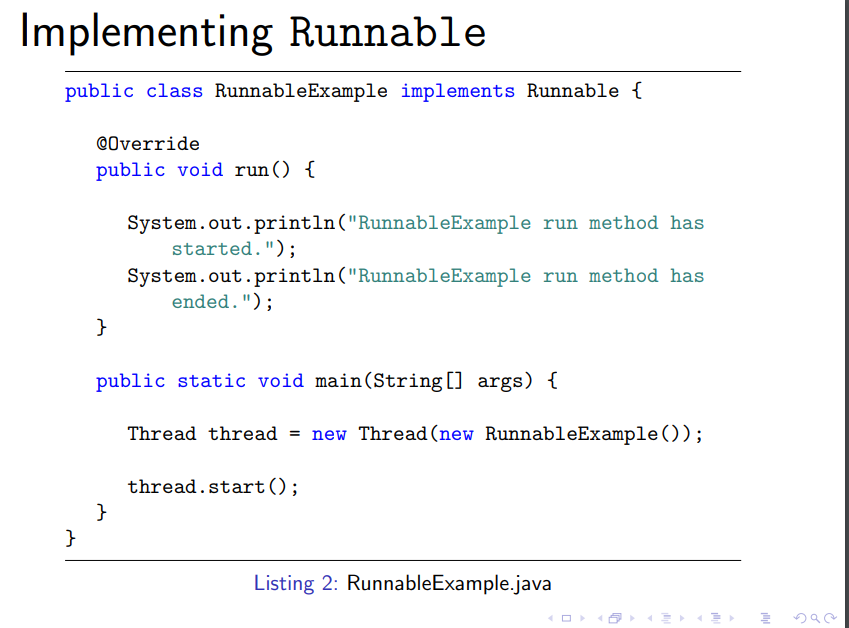
When extending thread class, it’s optional whether to rewrite the run() method or not, but it’s necessary anyway, since if you don’t rewrite it you can’t do anything.

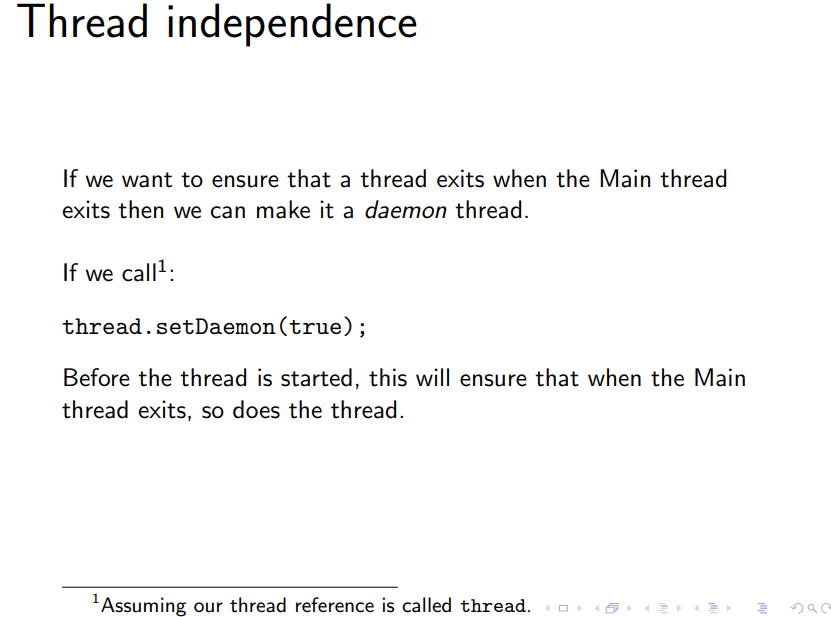
**Advantage of implementing a runnable interface is that we can make a class implement as many interfaces as we like**. **If we extend a class, we can’t extend another one, which may be too much of a restriction.**

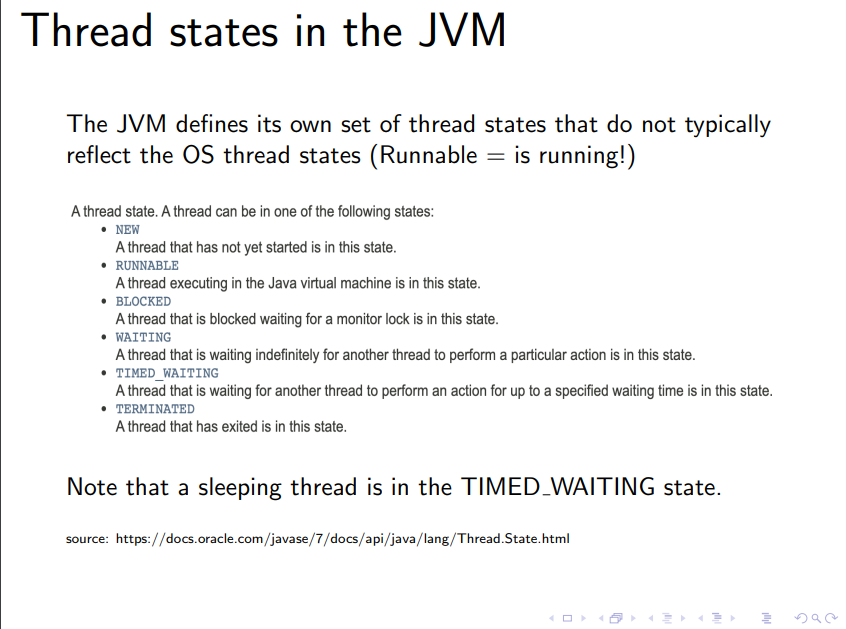
**When extending the Thread class, you are using that class to create a new thread, which means you can’t share the behavior of the different threads.**

**But when you implement runnable, you can create an object of that type and pass the same object to multiple threads This makes data sharing between the threads much easier since they are all accessing the same runnable object.**





Threads are independent, include main thread and the thread it create, so if you want other thread to terminate with main thread you do:  


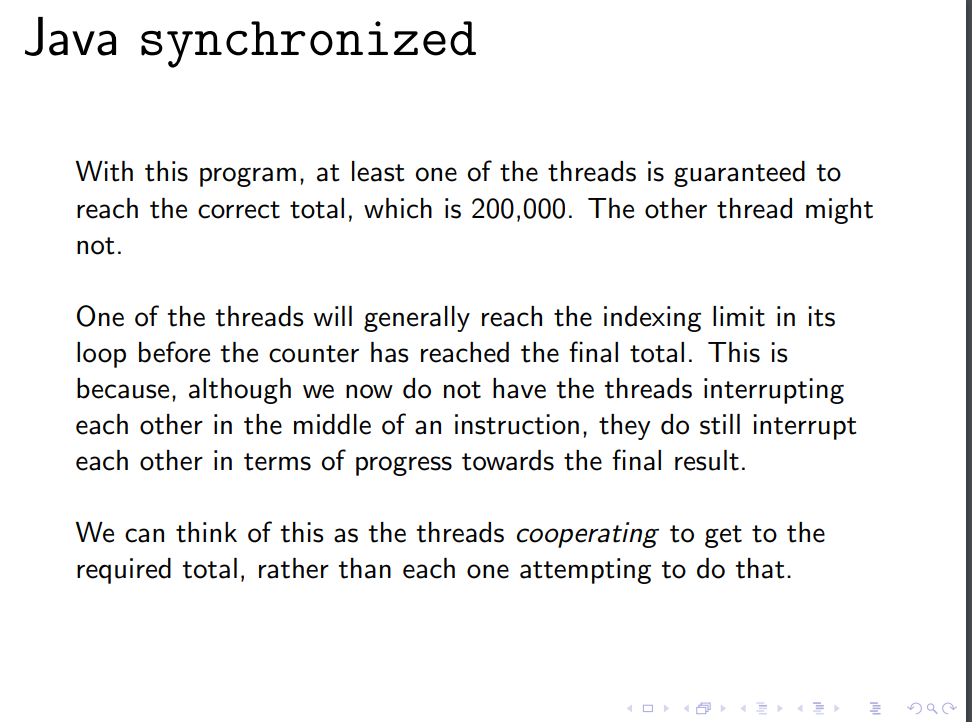
Thread states: NEW, RUNNABLE, BLOCKED, WAITING, TIME\_WAITING, TERMINATED  


Stack and Heap:

Each thread gets its own stack, but they **share the same heap**. primitive type variables (int, boolean ..) as a local variable or method parameter are stored on the stack, also reference of an object will store on the stack as well.

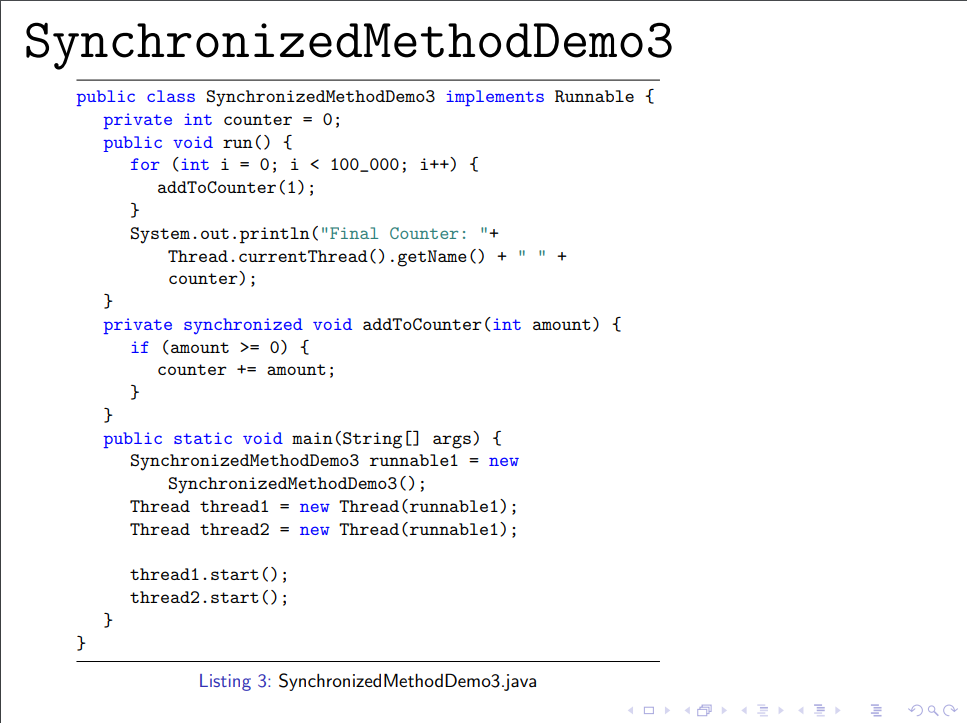
When create an object, you create two things: a reference and the object itself. When create in this manner in a method, the the reference will be stored on the stack but itself will be store on the heap.

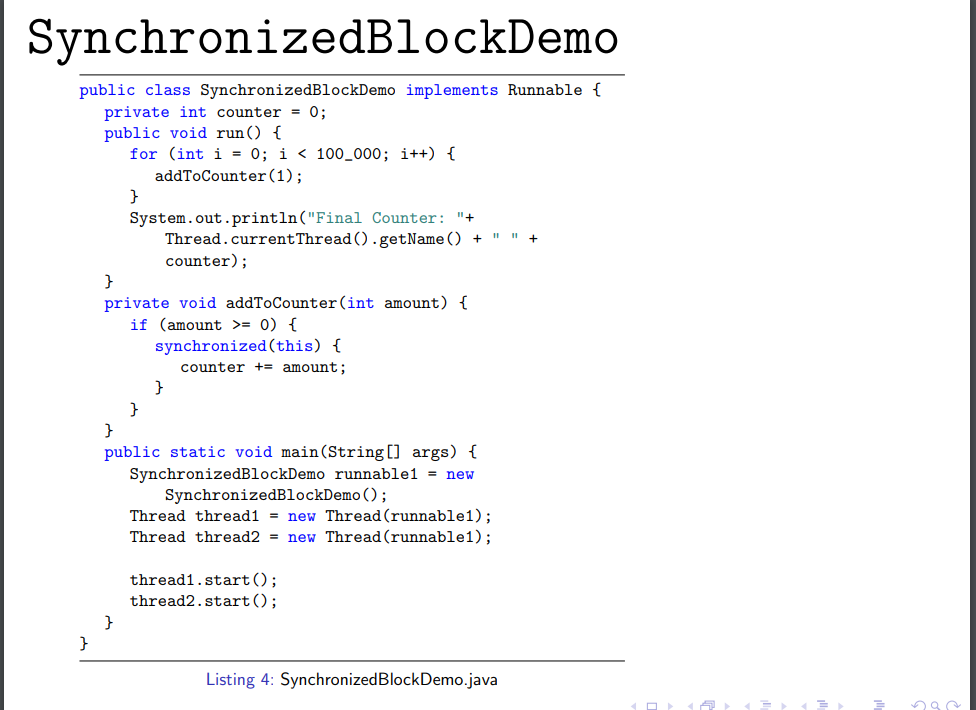
E,g: Operation op = new Operation(2, 5, "\*");



因为thread之间依然不是你一次我一次的进行，可能其中一个比另一个提早很多就完成了，因此这个thread的数值就会比另一个低，但是另一个任然能到达200000

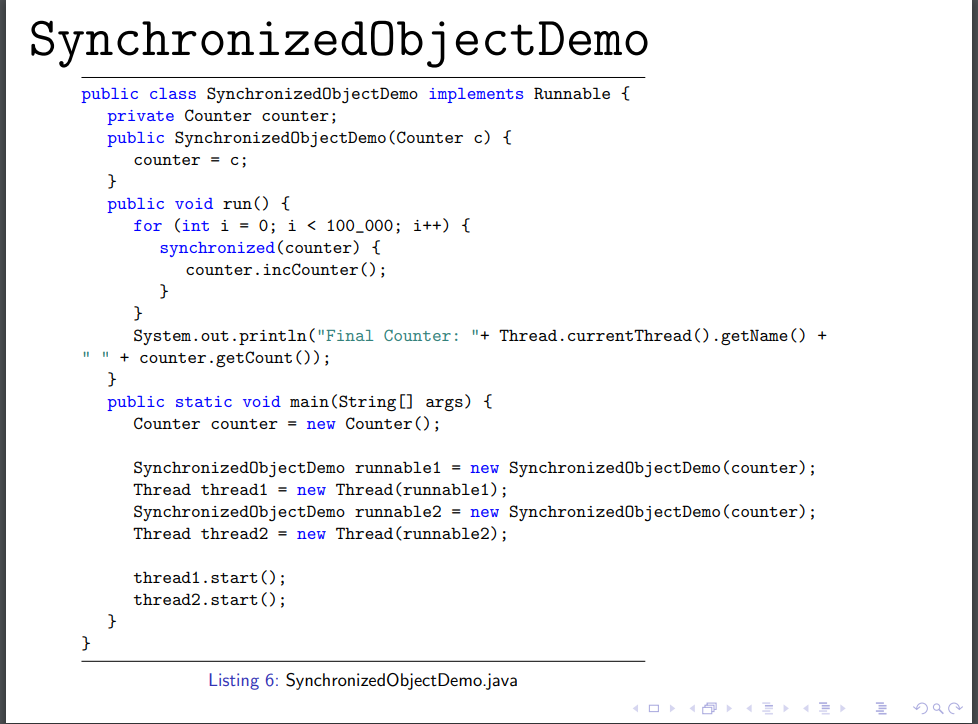
Critical sections就是用synchronized block标记的function

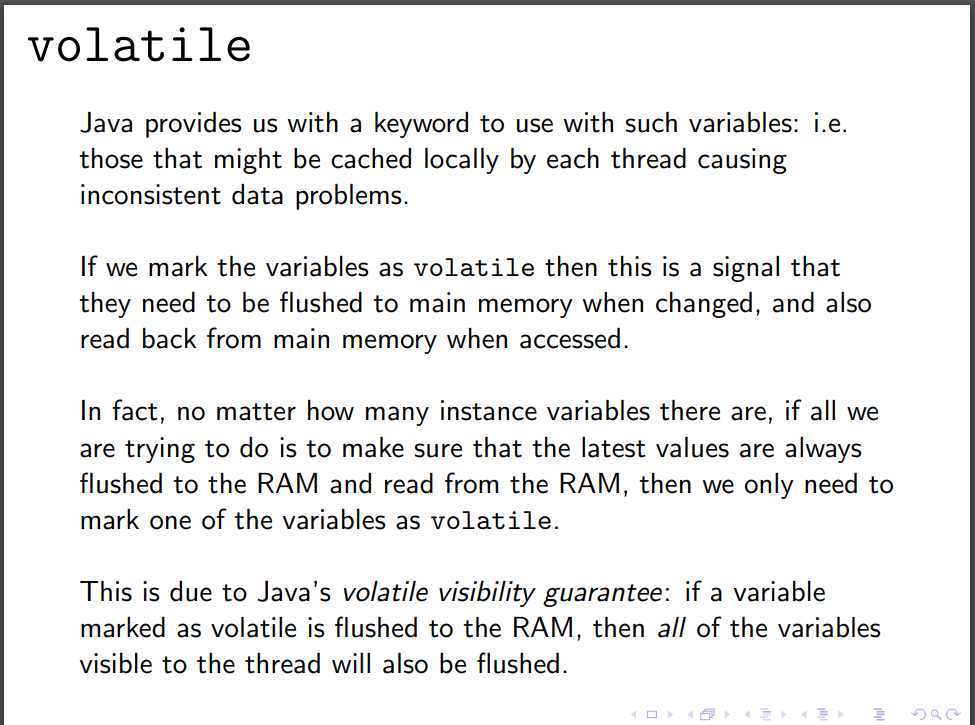
  
在这种情况下，amount其实是一个local variable，所以我们并不需要lock他，所以我们可以用一个单独的sychronized block来保护她



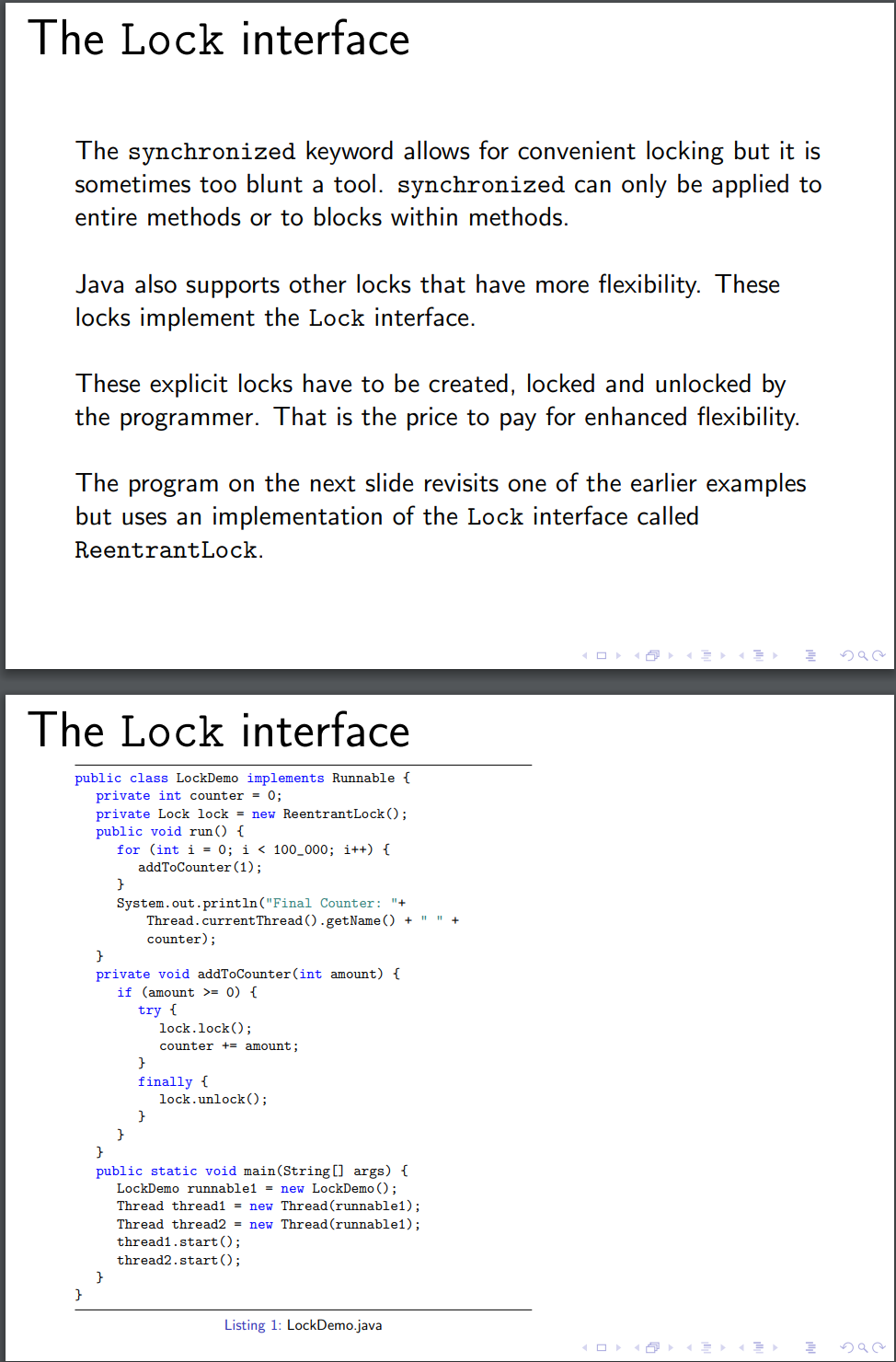
This 自动代表了当前保护的是这个block内部call的object

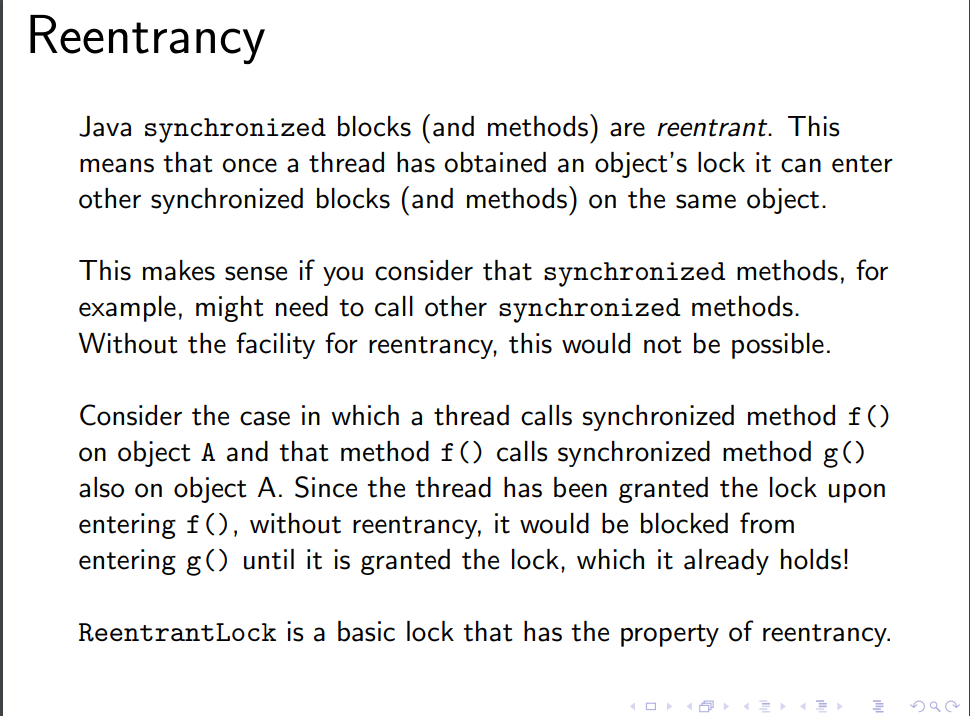
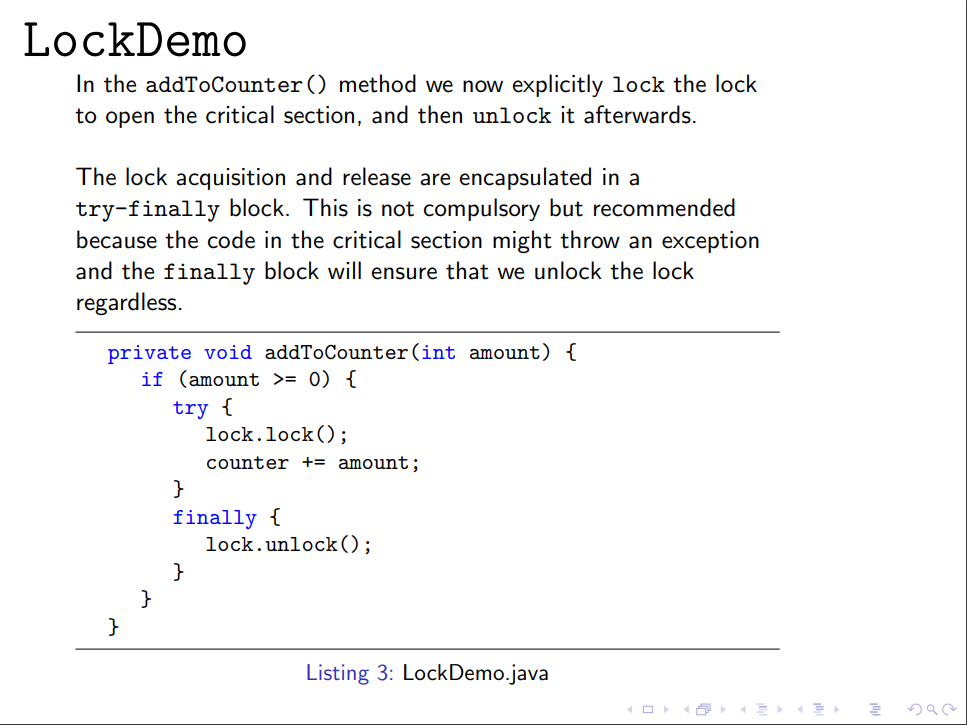
如果需要保护的object来自于一个share的class，你并不确定这个object的class本身有没有做保护，同时你又不想一次syn太多东西，则你可以在block中specify具体保护哪个object

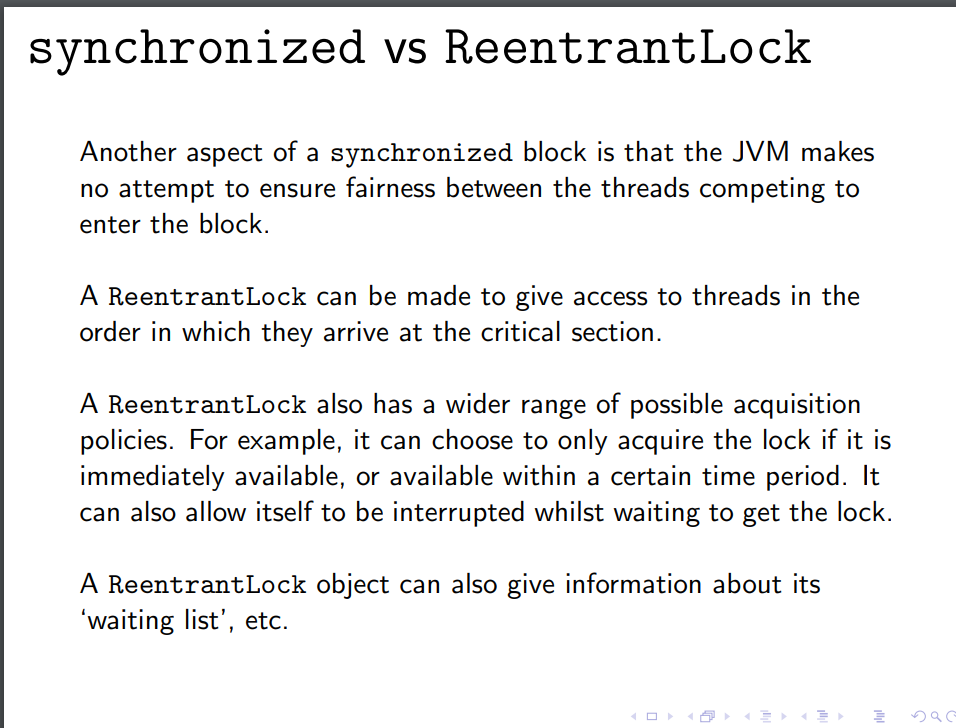


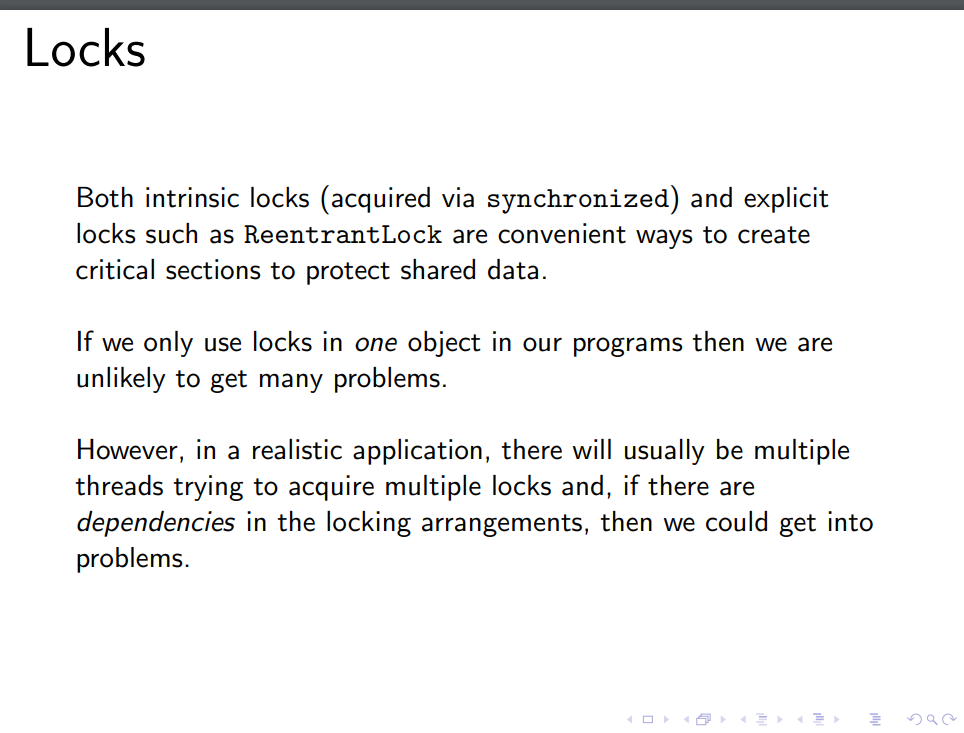


Volatile 保证了每次variable变化都会flush一遍memory然后再从memory读取数据，这样数据就是最新的，同时如果有很多instance variables的话，只需要标记其中一个为volatile，其他的会自动flush









ReentrantLock is an explicit locking mechanism in Java, which means that a programmer must explicitly acquire and release the lock in their code. This is in contrast to intrinsic (or implicit) locks, which are automatically managed by the Java language when you use the synchronized keyword.

