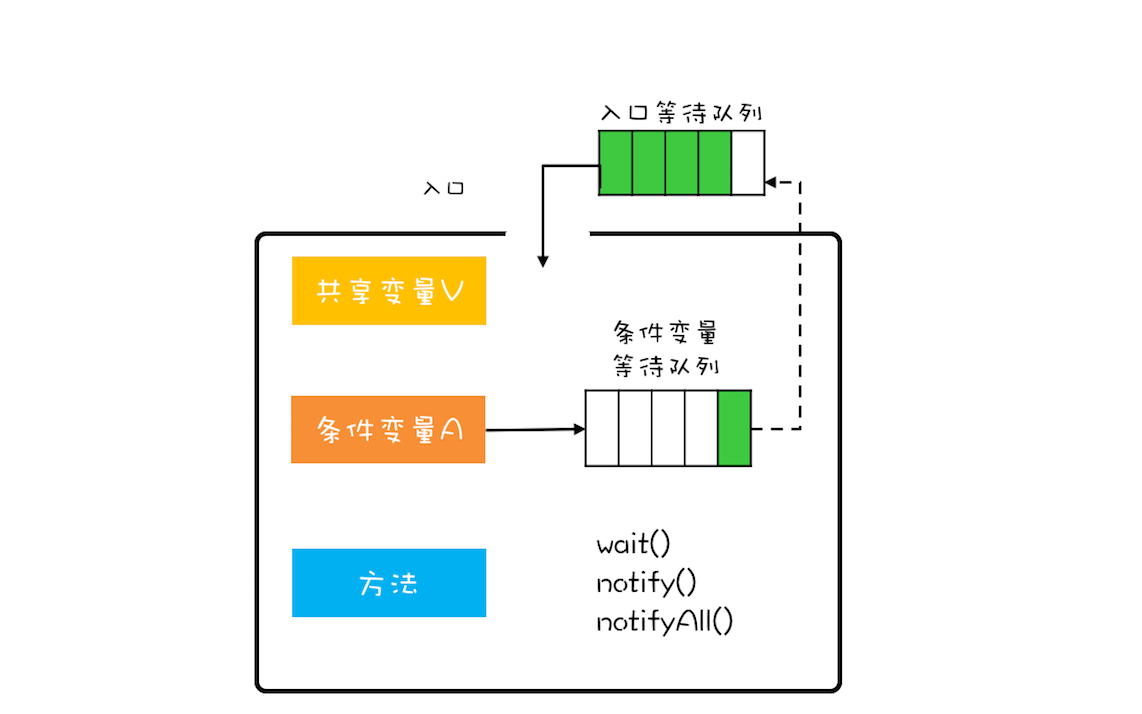
1. synchronized 又被称作管程，是Java常用的锁机制，jdk8后进行了优化，可以放心使用。

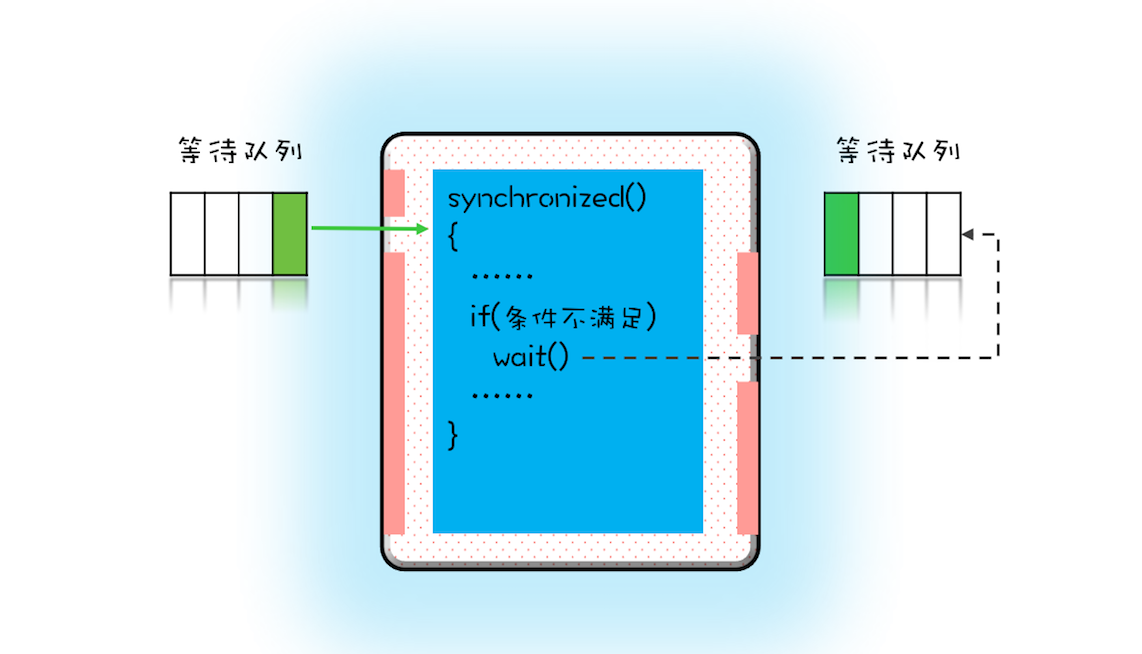
所谓管程，指的是**管理共享变量以及对共享变量的操作过程，让他们支持并发**。翻译为 Java 领域的语言，就是管理类的成员变量和成员方法，让这个类是线程安全的。

Java 采用的是管程技术，synchronized 关键字及 wait()、notify()、notifyAll() 这三个方法都是管程的组成部分。而管程和信号量是等价的，所谓等价指的是**用管程能够实现信号量，也能用信号量实现管程**。但是管程更容易使用，所以 Java 选择了管程。管程示意图：

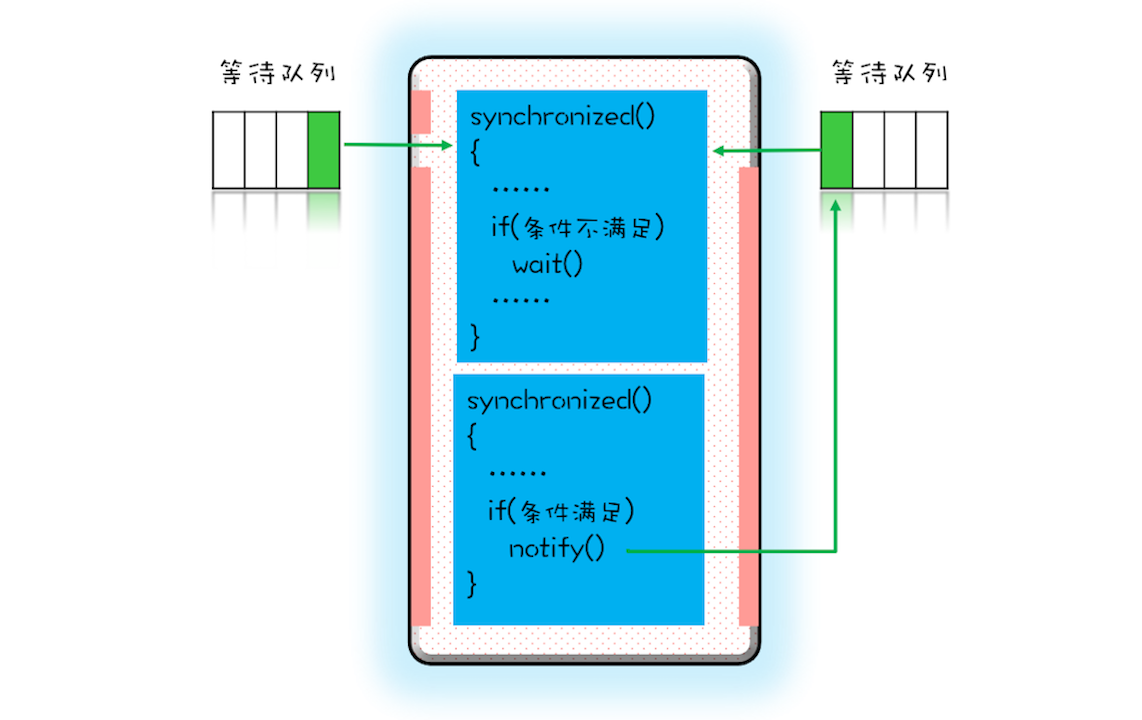


在管程模型里，共享变量和对共享变量的操作是被封装起来的，图中最外层的框就代表封装的意思。框的上面只有一个入口，并且在入口旁边还有一个入口等待队列。**当多个线程同时试图进入管程内部时，只允许一个线程进入，其他线程则在入口等待队列中等待**。这个过程类似就医流程的分诊，只允许一个患者就诊，其他患者都在门口等待。

1. Java 语言内置的 synchronized 配合 wait()、notify()、notifyAll() 这三个方法就能轻松实现等待-通知机制。



调用wait方法之后，**当前线程被阻塞，进入等待队列，同时会释放持有的互斥锁**，让其他线程有机会获得锁，进入临界区。



当条件满足时调用 notify()或notifyAll()，会通知等待队列（互斥锁的等待队列）中的线程，告诉它**条件曾经满足过。**为什么是曾经满足过呢？

因为notify() 只能保证在通知时间点，条件是满足的。而**被通知线程的执行时间点和通知的时间点基本上不会重合**，所以当线程执行的时候，很可能条件已经不满足了（保不齐有其他线程插队）。这一点你需要格外注意。还有一个需要注意的点，**被通知的线程要想重新执行，仍然需要获取到互斥锁**（因为曾经获取的锁在调用 wait() 时已经释放了)。

notify() 是会随机地通知等待队列中的一个线程，**而 notifyAll() 会通知等待队列中的所有线程**。从感觉上来讲，应该是 notify() 更好一些，因为即便通知所有线程，也只有一个线程能够进入临界区。但那所谓的感觉往往都蕴藏着风险，实际上使用 **notify() 也很有风险，它的风险在于可能导致某些线程永远不会被通知到**。所以除非经过深思熟虑，否则尽量使用 notifyAll()。

那什么时候可以使用 notify() 呢？需要满足以下三个条件：

1）所有等待线程拥有相同的等待条件；

2）所有等待线程被唤醒后，执行相同的操作；

3）只需要唤醒一个线程。

1. wait与sleep的区别：  
   1） wait会释放所有锁而sleep不会释放锁资源.  
   2） wait只能在同步方法和同步块中使用，而sleep任何地方都可以.

3）wait需要notify()或者notifyAll唤醒，而sleep等待休眠时间结束就继续执行。  
两者相同点：都会让渡CPU执行时间，等待再次调度。