2021219113 2021213595 沈尉林

Java的同步模型主要依赖于监视器（Monitors），这是控制对对象方法或块访问的构造。当一个方法或块被声明为`synchronized`时，其他试图进入同一对象的任何其他同步块的线程将被锁定。下面是关于Java同步的关键点：

- Java使用监视器，也称为监视器锁或内在锁，来提供同步。一个同步块在给定对象上一次只能由一个线程执行。
- 监视器带有条件变量，线程可以用`wait()`在其上等待，用`notify()`唤醒，或者用`notifyAll()`通知所有等待的线程。当调用`wait()`时，线程释放监视器的锁并进入等待状态。当调用`notify()`或`notifyAll()`时，它会通知在该对象的条件变量上等待的线程醒来，尽管锁不会立即释放，直到通知线程退出同步块或方法。
- `wait()`、`notify()`和`notifyAll()`方法必须在Java的同步上下文中调用。调用`wait()`的线程会这样做，直到另一个线程在同一个对象上调用`notify()`或`notifyAll()`。
- 这些方法的实现表明，Java同步是基于Mesa模型而不是Hoare模型。在Mesa风格的监视器中，`wait()`返回时的唯一保证是某个时刻断言为真，用户在`wait()`返回后负责重新检查断言。相比之下，Hoare风格的监视器保证在`wait()`返回时断言成立，但它们更复杂，实际中使用较少。

伪代码表示：

```
class Semaphore {
  int count = 1 // 初始信号量计数

  void wait() {
    while (true) {
      while (test_and_set(lock)) {} // 获取锁
      if (count > 0) {
        count -= 1 // 减少计数
        lock = false // 释放锁
        break // 如果成功则退出循环
      }
      lock = false // 释放锁
    }
  }

  void signal() {
    while (test_and_set(lock)) {} // 获取锁
    count += 1 // 增加计数
    lock = false // 释放锁
  }
```

```
}

boolean test_and_set(boolean lock) {
  boolean old = lock
  lock = true
  return old
}
```