什么场景下可以使用volatile替换synchronized

volatile的应用场景一句话概括下来就是，对于共享资源，要做到读不加锁，写加锁，那么就必须使用volatile，这里的写加锁可以使用互斥锁，也可以使用CAS的乐观锁机制

如何让一段程序并发的执行，并最终汇总结果？  
1. 可以使用Callable+FutureTask+Executors+Future操作，让收集结果的线程阻塞等待与get()方法等待结果  
2. 可以使用Thread的join方法来阻塞等待结果  
3. 可以使用CountDownLatch来让收集结果的线程等待  
4. 可以使用CyclicBarrier来收让收集结果的线程等待，  
CountDownLatch是CyclicBarrier的一种，所以CountDownLatch能解决的问题，CyclicBarrier一定可以

1. 如何让一段程序并发的执行，并最终汇总结果？  
   1. 可以使用Callable+FutureTask+Executors+Future操作，让收集结果的线程阻塞等待与get()方法等待结果  
   2. 可以使用Thread的join方法来阻塞等待结果  
   3. 可以使用CountDownLatch来让收集结果的线程等待  
   4. 可以使用CyclicBarrier来收让收集结果的线程等待，  
   CountDownLatch是CyclicBarrier的一种，所以CountDownLatch能解决的问题，CyclicBarrier一定可以
2. http://2.gravatar.com/avatar/?s=35&d=mm&r=g
   * Anonymous
   * 12/29. 2013 11:23am
   * [Log in to Reply](http://ifeve.com/wp-login.php?redirect_to=http%3A%2F%2Fifeve.com%2Fjavaconcurrency-interview-questions-combat%2F)

* + [QUOTE](javascript:void(0);)

如何使用阻塞队列实现一个生产者和消费者模型？请写代码

使用基于数组的阻塞队列，有限次取水果和放水果

package com.tom.jdk5.concurrent.collections;

import java.util.concurrent.BlockingQueue;  
import java.util.concurrent.LinkedBlockingQueue;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* 使用阻塞队列实现生产者消费者问题  
\*  
\* BlockingQueue的offer/poll操作不能满足阻塞等待的效果  
\*  
\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Plate {  
// 一个盘子，可以放10个水果  
private BlockingQueue fruits = new LinkedBlockingQueue(10);

// 如果有水果，则取得，否则取不走  
public String get() {  
try {  
return fruits.take();  
} catch (InterruptedException e) {  
Thread.currentThread().interrupt();  
return null;  
}  
}

public void put(String fruit) {  
try {  
fruits.put(fruit);  
} catch (InterruptedException e) {  
Thread.currentThread().interrupt();  
}  
}  
}

class Producer implements Runnable {

private Plate plate;

public Producer(Plate p) {  
this.plate = p;  
}

@Override  
public void run() {  
try {  
for (int i = 0; i < 100; i++) {  
this.plate.put("" + i);  
System.out.println("第" + i + "个水果放入盘子");  
Thread.sleep((long) (200 \* Math.random()));  
}  
} catch (InterruptedException e) {  
// TODO Auto-generated catch block  
e.printStackTrace();  
}  
}

}

class Consumer implements Runnable {

private Plate plate;

public Consumer(Plate p) {  
this.plate = p;  
}

@Override  
public void run() {  
try {  
for (int i = 0; i < 100; i++) {  
String j = this.plate.get();  
System.out.println("第" + j + "个水果取出盘子");  
Thread.sleep((long) (400 \* Math.random()));  
}  
} catch (InterruptedException e) {  
// TODO Auto-generated catch block  
e.printStackTrace();  
}  
}

}

public class ProducerConsumerTest {  
public static void main(String[] args) {  
Plate p = new Plate();  
Producer producer = new Producer(p);  
Consumer consumer = new Consumer(p);  
new Thread(producer).start();  
new Thread(consumer).start();

}  
}

如何实现乐观锁（CAS）？如何避免ABA问题？

CAS原语有三个值，一个是内存值，一个是期望值，一个是写入值。 在不加锁的情况下写入时，每次读取内存值，然后跟预期值比对，如果比对失败，反复的读和比对，直到成功。在CAS原语是一个原子操作，如果写入时，内存值发生改变，则写入值失败

ABA问题可以使用AtomicStampedReference在做CAS操作时，一方面比较内存中的操作数与预期值是否一样，同时比较内存中的操作数的时间戳（或者修改次数）是否与预期值一样。如果本次修改操作成功，一定要修改操作数的时间戳，可以通过每次加1的方式。

读写锁可以用于什么应用场景？  
读写锁适用于读操作>>写操作，同时允许多个线程同时读，读读允许，读写互斥，写写互斥。这样的场景比较常见，比如，一个文件，我们可以要求同时读，但是不能读写或者写写同时发生。缓存Cache也可以实现同时读，互斥写。对于lucene这样的文本检索系统，也可以使用读写锁实现更新索引文件上写锁，读取索引加读锁。

什么时候应该使用可重入锁？

1.公平锁  
2.多路条件，比如我要实现5个线程的同步，A->B->C->D->E->A,可以使用一个Lock，5个Condition来控制5个线程的步调  
3.实现轮询，中断的功能，只听说过，没用过。

如何实现一个流控程序，用于控制请求的调用次数？

应该可以考虑使用Semaphore来控制， Semaphore本身是用来限制有限资源的争用。比如我要限制数据数据库连接池的连接数不超过20个，我可以在连接池的配置里面设置最大值，也可以在连接池的访问层来控制，这样可以减轻连接池的负担，也可以控制得不到连接的线程的行为，比如等待而不是扔个SQLException出来