1. **如何保证线程安全？**

参考回答：

通过合理的时间调度，避开共享资源的存取冲突。另外，在并行任务设计上可以通过适当的策略，保证任务与任务之间不存在共享资源，设计一个规则来保证一个客户的计算工作和数据访问只会被一个线程或一台工作机完成，而不是把一个客户的计算工作分配给多个线程去完成。

1. **请你简要说明一下线程的基本状态以及状态之间的关系？**

参考回答：

其中Running表示运行状态，Runnable表示就绪状态（万事俱备，只欠CPU），Blocked表示阻塞状态，阻塞状态又有多种情况，可能是因为调用wait()方法进入等待池，也可能是执行同步方法或同步代码块进入等锁池，或者是调用了sleep()方法或join()方法等待休眠或其他线程结束，或是因为发生了I/O中断。

1. **举例说明同步和异步**

参考回答：

如果系统中存在临界资源（资源数量少于竞争资源的线程数量的资源），例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就必须进行同步存取（数据库操作中的排他锁就是最好的例子）。当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。事实上，所谓的同步就是指阻塞式操作，而异步就是非阻塞式操作。

1. 请介绍一下线程同步和线程调度的相关方法。

参考回答：

- wait()：使一个线程处于等待（阻塞）状态，并且释放所持有的对象的锁；  
- sleep()：使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要处理InterruptedException异常；  
- notify()：唤醒一个处于等待状态的线程，当然在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且与优先级无关；  
- notityAll()：唤醒所有处于等待状态的线程，该方法并不是将对象的锁给所有线程，而是让它们竞争，只有获得锁的线程才能进入就绪状态；  
通过Lock接口提供了显式的锁机制（explicit lock），增强了灵活性以及对线程的协调。Lock接口中定义了加锁（lock()）和解锁（unlock()）的方法，同时还提供了newCondition()方法来产生用于线程之间通信的Condition对象；此外，Java 5还提供了信号量机制（semaphore），信号量可以用来限制对某个共享资源进行访问的线程的数量。在对资源进行访问之前，线程必须得到信号量的许可（调用Semaphore对象的acquire()方法）；在完成对资源的访问后，线程必须向信号量归还许可（调用Semaphore对象的release()方法）。

1. **请问当一个线程进入一个对象的synchronized方法A之后，其它线程是否可进入此对象的synchronized方法B？**

参考回答：

不能。其它线程只能访问该对象的非同步方法，同步方法则不能进入。因为非静态方法上的synchronized修饰符要求执行方法时要获得对象的锁，如果已经进入A方法说明对象锁已经被取走，那么试图进入B方法的线程就只能在等锁池（注意不是等待池哦）中等待对象的锁。

1. **请简述一下线程的sleep()方法和yield()方法有什么区别？**

参考回答：

①sleep()方法给其他线程运行机会时不考虑线程的优先级，因此会给低优先级的线程以运行的机会；yield()方法只会给相同优先级或更高优先级的线程以运行的机会；

② 线程执行sleep()方法后转入阻塞（blocked）状态，而执行yield()方法后转入就绪（ready）状态；  
③ sleep()方法声明抛出InterruptedException，而yield()方法没有声明任何异常；  
④ sleep()方法比yield()方法（跟操作系统CPU调度相关）具有更好的可移植性。

1. **请回答以下几个问题： 第一个 问题：Java中有几种方法可以实现一个线程？ 第二个问题：用什么关键字修饰同步方法?  第三个问题：stop()和suspend()方法为何不推荐使用，请说明原因**？

参考回答：

有两种实现方法，分别是继承Thread类与实现Runnable接口用synchronized关键字修饰同步方法，反对使用stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被”挂起”的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用 wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。

1. 请分别说明一下多线程和同步有几种实现方法,并且这些实现方法具体内容都是什么?

参考回答：

多线程有两种实现方法，分别是继承Thread类与实现Runnable接口同步的实现方面有两种，分别是synchronized,wait与notify。

1. 请说出你所知道的线程同步的方法

参考回答：

wait():使一个线程处于等待状态，并且释放所持有的对象的lock。  
sleep():使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉InterruptedException异常。  
notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且不是按优先级。  
Allnotity():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。

1. **启动一个线程是用run()还是start()?**

参考回答：

启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。