1、java中的多线程：

答：CPU的资源是由各个线程来争抢的，这样就形成了多线程同时工作，创建线程的有两种方式，一个是实现Runnable接口并重写run方法，二是继承Thread类，调用start方法，线程的生命周期分为：创建、就绪、运行、阻塞、死亡；

2、start与run的区别：

答：run只是类的普通方法，直接调用run方法程序中还是只有一个主线程，需要等run方法执行完毕之后才会执行下面的方法；start是启动一个新的线程，不必等待其他线程运行完，只要得到cpu就可以运行；执行start线程处于就绪状态；

3、wait与sleep的区别：

答：wait方法属于Object类，sleep方法属于Thread类；调用sleep方法的过程中，线程不会释放对象锁，调用wait方法会释放线程锁；sleep睡眠后不会让出系统资源，wait会让出系统资源让其他线程可以占用cpu资源；sleep方法在任何地方能用；wait方法只有在同步方法或同步块中用

4、什么是线程安全？

答：指在进程中有多个线程在同时运行，所有线程都可能会运行同一段代码，但是每次运行出来的结果和单线程运行的结果是一样的

5、同步锁Lock与synchronized的区别：

答：Lock是一个接口，而synchronized是java中的关键字；发生异常时，synchronized会自动释放线程占有的锁（synchronized是jvn层面上实现的），Lock不会（Lock是通过代码实现的），要保证锁被释放，必须将unLock（）放在finally里面；Lock可以让等待线程的锁相应中断，synchronized不行，等待的线程会一直等待下去，无法被中断；Lock可以知道有没有获取到锁，而synchronized不行；在竞争不激烈的时候，两者效率差不多，当竞争激烈的时候，Lock效率高些

6、ReentrantLock获取锁的三种方式：

lock(),如果获取到了锁立即返回，如果别的线程持有锁，当前线程会一直休眠，直到获取到锁；

tryLock(),如果获取到了返回true，如果别的线程持有锁，返回false

tryLock(long timeout, TimeUnit unit) ,获取到了返回true，如果别的线程持有锁，会等待参数给予的时间，在时间内获取到了返回true，超时返回flase；

lockInterruptibly（可中断锁定）如果获取锁立即返回，没有获取到休眠，直到获取到锁或者线程中断

7、Runnable与Callable的区别：

答：两个都代表需要在不同线程中执行的任务；Callable中的call方法可以有返回值和抛出异常，Runnable中的run方法没有；Callable可以返回装载有计算结果的future对象

8、为什么wait和notify方法要在同步块中用：

答：主要是因为java APi 强制要求这么做的，如果不这么做，代码会抛出异常，还有一个原因是为了避免wait和notify之间产生竞态条件

9、堆和栈有什么区别：

答：每个线程都有自己的栈内存，用于存储本地变量，一个线程存储的变量对其他线程是不可见的，堆是所有线程共享的一片公用内存区域；

10、线程池中的submit和execute方法的区别：

答：execute定义在Execute接口中，submit定义在ExecutorService接口中，execute方法没有返回值，submit可以返回future对象

11、notify和notifyAll有什么区别：

答：notify不能具体的唤醒某个线程，只有一个线程等待的时候才有用武之地，notifyAll会唤醒所有线程并允许它们争夺资源确保至少有一个线程能继续运行；

12、volatile和synchronized的区别：

答：1、volatile不会造成阻塞；synchronized会造成阻塞

2、volatile只能修饰变量；synchronized可以修饰变量、方法、类

3、volatile只能保证变量的修改可见性，不能保证原子性；synchronized都可以保证