**问题一：**你负责的系统中有哪些可能造成资损风险点或其他严重的风险点，你是如何解决的？

回答：

重复退款，修改交易状态前检查当前交易状态

业务校验，

业务流水号重复，拒绝交易

答案：开放性问题

**问题二：**你在系统中是如何使用线程池的？JDK原生线程池有哪些参数，他们分别代表什么含义？

回答：

参数：初始值、最大最小、队列，队列包括fix、block。

抛弃策略：最老、最新、很久未使用、阻塞

初始有coreSize个，当任务数大于coreSize创建线程

答案：

JDK原生线程池有哪些参数&含义：

corePoolSize：线程池核心线程数，线程空闲也保留，空闲也不销毁，不受keepAliveTime控制，除非将allowCoreThreadTimeout设置为true。

maxPoolSize：线程池最大线程数，当队列已满时会再次创建线程，不超过最大线程数，当配置非空队列时，此参数无效。超过corePoolSize的线程会标记为可回收，当空闲时间超过keepAliveTime时会被回收。

keepAliveTimeout：非核心线程的空闲超时时间，当allowCoreThreadTimeout为true时，对核心线程也生效。

unit：keepAliveTimeout时间单位

workQueue：缓冲队列，当前运行任务数超过corePoolSize后，后续的任务都会入队。一般配置三种队列：

SynchronousQueue：不会缓存任务，要求线程池汇总必须有空闲线程可以提交任务，不超过max时可以创建线程，否则拒绝任务。使用此队列时应保证max足够大，或接受拒绝任务。例如CachedThreadPool使用了无界线程池（max为Integer.MAX\_VALUE）。

有界队列&无界队列：可以使用ArrayBlockingQueue、LinkedBlockingQueue或PriorityQueue，无界队列应与FixedThreadPool配合使用。

threadFactroy：线程工厂，配置线程参数。

RejectedExecutionHandler：当队列已满且等于maxPoolSize时的拒绝策略。JDK有4种拒绝策略：直接抛出异常并丢弃任务；由当前线程执行任务；抛弃最久任务；静默丢弃。注意抛弃最久任务和优先级队列不能同时使用。

线程池添加任务规则：

1. 当前活跃线程数<core，创建线程执行任务

2. 当前活跃线程数>=core，且队列未满时，入队

3. 当前活跃线程数>=core，且队列已满，创建线程执行任务

4. 当前活跃线程数=max，且队列已满，拒绝任务

问题三：你处理过哪些线程安全问题，你是如何解决的？解释一下什么是线程安全？如何更好的保证线程安全？

回答：

同一个数据，同一时间只有一个线程操作

1. 线程安全数据结构：ConcurrentHashMap、、CopyOnWriteArrayList、CopyOnWriteArraySet

2. 信号量，操作：获取、释放

3. 锁

答案：

什么是线程安全：

1. 一段代码可以在多线程中调用，且不会出现错误的交互，始终表现正确性。

2. 对于调用者来说无需采取同步操作。

如何保证线程安全：

1. 线程不安全主要体现在对全局、静态变量的并发读写上。这些对象就是有状态的对象。应尽可能使用局部变量，减少对象可变状态能更大程度保证线程安全。

2. 注意程序中由于多个步骤的原子性，例如“读-修改-写入”和“先检查后执行”。

3. 使用并发容器，例如ConcurrentHashMap；使用原子计数器，例如AtomicInteger。

4. 使用同步代码块synchronized，或加重入锁、读写锁保护可变状态。