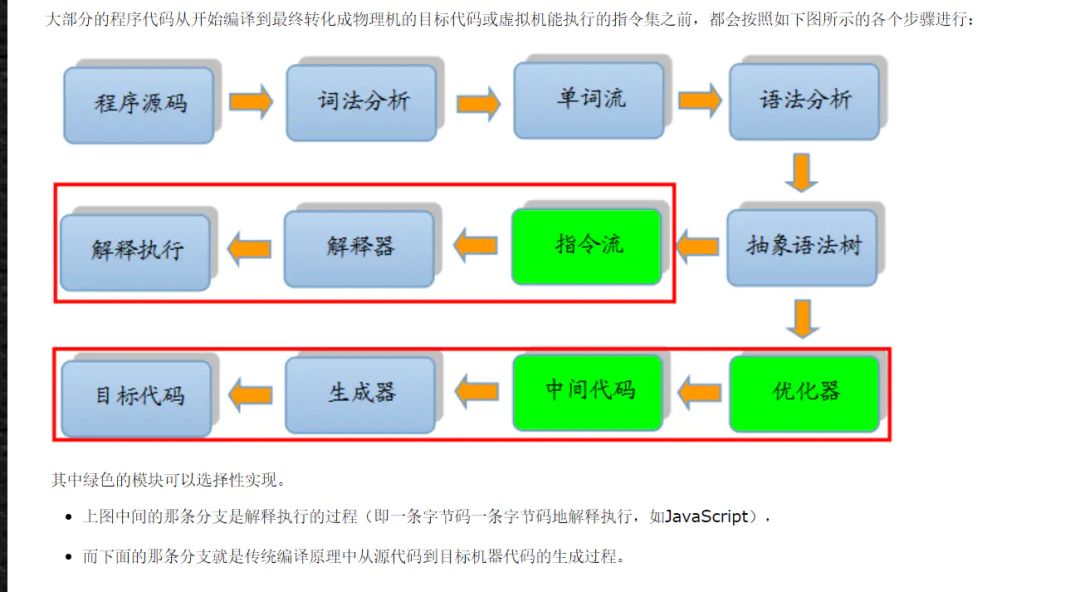
**一面**  
1. 自我介绍下自己，不超过3分钟（实际上我的自我介绍不到一分钟）  
2. 你感觉比本科阶段自己进步了多少，有哪些进步  
3. 研究生期间最大的进步是什么  
4. 你觉得你适合从事哪个方向的开发  
5. synchronized与lock的区别，使用场景。看过synchronized的源码没  
6. JVM自动内存管理，Minor GC与Full GC的触发机制  
7. 了解过JVM调优没，基本思路是什么    如果CPU使用率较高，GC频繁且GC时间长，可能就需要JVM调优了。  
    基本思路就是让每一次GC都回收尽可能多的对象，对于CMS来说，要合理设置年轻代和年老代的大小。8. 该如何确定它们的大小呢？    这是一个迭代的过程，可以先采用JVM的默认值，然后通过压测分析GC日志。        如果看年轻代的内存使用率处在高位，导致频繁的Minor GC，而频繁GC的效率又不高，说明对象没那么快能被回收，这时年轻代可以适当调大一点。    如果看年老代的内存使用率处在高位，导致频繁的Full GC，这样分两种情况：如果每次Full GC后年老代的内存占用率没有下来，可以怀疑是内存泄漏；如果Full GC后年老代的内存占用率下来了，说明不是内存泄漏，要考虑调大年老代。    对于G1收集器来说，可以适当调大Java堆，因为G1收集器采用了局部区域收集策略，单次垃圾收集的时间可控，可以管理较大的Java堆。9. 如何设计存储海量数据的存储系统  
    海量数据的解决方案：    页面上：使用缓存；页面静态化技术；  
    数据库层面：        分离数据库中活跃的数据；        批量读取和延迟修改；        读写分离；        使用NoSQL和Hadoop等技术；        分布式部署数据库；        应用服务和数据服务分离；  
    其他方面：  
        使用搜索引擎搜索数据库中的数据；        进行业务的拆分；    高并发情况下的解决方案：  
        应用程序和静态资源文件进行分离，静态资源可以使用CDN；  
    集群与分布式；  
        使用Nginx反向代理；10. 缓存的实现原理，设计缓存要注意什么  
    将热点数据放在内存中，用户查询时命中内存中的数据而不用到数据库中查询  
注意缓存的一致性，缓存雪崩、击穿、穿透的问题11. 淘宝热门商品信息在JVM哪个内存区域    ？？，不应该在缓存中嘛，然后落地在数据库里，跟JVM有屁关系  
12. 操作系统的页式存储  
    把内存分成大小相同的内存页，然后程序通过页表来查询到自己的存储位置，这样就可以使用不连续的内存来加载程序  
    事实上现在都用虚拟内存的方式，把程序分段加载到虚拟内存中，再把内存分页，通过段表、页表的形式来映射程序在内存中的位置  
13. volatile关键字的如何保证内存可见性  
    volatile修饰的变量保证其每个写操作后都更新到主内存，每个读操作都到主内存中更新，具体的话是在JVM层面，在修饰的变量前后加关键字  
    顺带一提volatile还能防止指令重排，这两者的实现方式都是内存屏障。14. happen-before原则  
    如果前一个操作的执行结果必须对后一个操作可见，那就不允许这两个操作进行重排序，且happen-befor具有传递性  
15. Lucene全文搜索的原理  
    先将全文由分词器进行分词，会提取出关键词和频率，然后这个关键词后面也会跟着一个链表，这个链表记录了有个关键词的文档。我们通过关键词搜索就可以找到这串链表，也就得到了所要的文档了。  
16. 你觉得自己适合哪方面的开发，为什么  
    JavaWeb的后端开发，一个当然是对这方面感兴趣，涉及范围广，然后目前掌握的比较多的也是这方面的内容，技能比较熟练，自己认识的圈子也是这个圈子里的人，如果遇到问题也容易解决，自己也有一套这方面的学习方法，如果去学习其他的话，倒不是走出舒适区的问题，而是对于现阶段的我来说，不能做到在短时间内取得阶段性的成果  
17. 想去哪里实习，杭州？

**二面**1. 自我介绍下自己，不超过3分钟（我的自我介绍仍然不超过1分钟）  
2. 你说你熟悉并发编程，那么你说说Java锁有哪些种类，以及区别（果然深度不一样）  
3. 公平锁/非公平锁  
    这个是在ReentrankLock中实现的，synchronized没有，是用一个队列实现的，在公平锁好理解，就是先进这个队列的，也先出队列获得资源，而非公平锁的话，则是还没有进队列之前可以与队列中的线程竞争尝试获得锁，如果获取失败，则进队列，此时也是要乖乖等前面出队才行  
4. 可重入锁  
    如果一个线程获得过该锁，可以再次获得，主要是用途就是在递归方面，还有就是防止死锁，比如在一个同步方法块中调用了另一个相同锁对象的同步方法块  
5. 独享锁/共享锁  
    共享锁可以由多个线程获取使用，而独享锁只能由一个线程获取。  
    对ReentrantReadWriteLock其读锁是共享锁，其写锁是独占锁  
    读锁的共享锁可保证并发读是非常高效的，读写，写读，写写的过程是互斥的。其中获得写锁的线程还能同时获得读锁，然后通过释放写锁来降级。读锁则不能升级  
6. 互斥锁/读写锁  
    上面讲的独享锁/共享锁就是一种广义的说法，互斥锁/读写锁就是具体的实现。  
    互斥锁在Java中的具体实现就是ReentrantLock  
    读写锁在Java中的具体实现就是ReadWriteLock  
7. 乐观锁/悲观锁  
    乐观锁就是乐观的认为不会发生冲突，用cas和版本号实现  
    悲观锁就是认为一定会发生冲突，对操作上锁  
8. 分段锁  
    在1.7的concurrenthashmap中有分段锁的实现，具体为默认16个的segement数组，其中segement继承自reentranklock，每个线程过来获取一个锁，然后操作这个锁下连着的map。  
9. 偏向锁/轻量级锁/重量级锁  
    在jdk1.6中做了第synchronized的优化，偏向锁指的是当前只有这个线程获得，没有发生争抢，此时将方法头的markword设置成0，然后每次过来都cas一下就好，不用重复的获取锁  
    轻量级锁：在偏向锁的基础上，有线程来争抢，此时膨胀为轻量级锁，多个线程获取锁时用cas自旋获取，而不是阻塞状态  
    重量级锁：轻量级锁自旋一定次数后，膨胀为重量级锁，其他线程阻塞，当获取锁线程释放锁后唤醒其他线程。（线程阻塞和唤醒比上下文切换的时间影响大的多，涉及到用户态和内核态的切换）  
    自旋锁：在没有获取锁的时候，不挂起而是不断轮询锁的状态10. 如何保证内存可见性  
    volatile  通过内存屏障  
    synchronized 通过修饰的程序段同一时间只能由同一线程运行，释放锁前会刷新到主内存11. Http请求的过程与原理  
12. 三次握手与四次挥手？  
13. 通过HTTP网络请求过程中的TCP协议  
14. TCP连接的特点  
    相较于UDP来说，更加安全可靠,是面向连接，传输的话是以流的形式传输  
15. TCP连接如何保证安全可靠的  
16. 为什么TCP连接需要三次握手，两次不可以吗，为什么  
    不可以两次握手只能一方确认自己的收发没有问题，而另一方的收没问题，发可能存在问题  
17. AOP的原理  
    静态织入，动态代理  
18. JDK动态代理与cglib实现的区别（这个，醉得很厉害）  
    接口（反射）/继承  
19. 那么你说说代理的实现原理呗  
    1.创建一个接口  
    2.创建一个实现了这个接口的实现类  
    3.创建一个实现了这个接口的代理类，在代理类中实例化实现类，并且调用实现类中的方法  
20. 看过Spring源码没，说说Ioc容器的加载过程吧  
    简单概括：  
    1.刷新预处理  
    2.将配置信息解析，注册到BeanFactory  
    3.设置bean的类加载器  
    4.如果有第三方想再bean加载注册完成后，初始化前做点什么(例如修改属性的值，修改bean的scope为单例或者多例。)，提供了相应的模板方法，后面还调用了这个方法的实现，并且把这些个实现类注册到对应的容器中  
    5.初始化当前的事件广播器  
    6.初始化所有的bean  
    7.广播applicationcontext初始化完成。

21. 了解过字节码的编译过程吗（这个还真不知道）



**三面**

1. 自我介绍，不超过3分钟（这次好像时间更久了，也就2分钟多点）2. 说一下你对哪个项目比较熟悉    数据库项目  
3. 为什么做这个项目  
    当时公司里为了整改balabala  
 4. 项目采用了什么架构，数据库如何设计的  
    简单是MVC架构，数据库  
5. 数据库由哪些表，为什么有这些表6. 主要有哪些核心模块，模块之间如何通信的7. session放在哪里



 8.  如何保存会话状态，有哪些方式、区别如何    cookie 保存在客户端，容易篡改  
    session 保存在服务端，连接较大的话会给服务端带来压力，分布式的情况下可以放在数据库中，  
    优点：  
        1：简单且高性能  
        2：支持分布式与集群  
        3：支持服务器断电和重启  
        4：支持 tomcat、jetty 等运行容器重启  
    缺点：  
        1、需要检查和维护session过期，手动维护cookie；  
        2、不能有频繁的session数据存取；    token 多终端或者app的话一定要这个，  
    随着技术的发展，分布式web应用的普及，通过session管理用户登录状态成本越来越高，因此慢慢发展成为token的方式做登录身份校验，然后通过token去取redis中的缓存的用户信息，随着之后jwt的出现，校验方式更加简单便捷化，无需通过redis缓存，而是直接根据token取出保存的用户信息，以及对token可用性校验，单点登录更为简单。

|  |
| --- |
| JWT的token包含三部分数据：  Header：  头部，通常头部有两部分信息：  声明类型，这里是JWT  加密算法，自定义  我们会对头部进行base64加密（可解密），得到第一部分数据  Payload：  载荷，就是有效数据，一般包含下面信息：  用户身份信息（注意，这里因为采用base64加密，可解密，因此不要存放敏感信>息）  注册声明：  如token的签发时间，过期时间，签发人等  这部分也会采用base64加密，得到第二部分数据  Signature：  签名，是整个数据的认证信息。  一般根据前两步的数据， 再加上服务的>的密钥（secret）  （不要泄漏，最好周期性更换），通过加密算法生成。  用于验证整个数据完整和可靠性（不要泄漏，最好周期性更换），通过加密算法生成。  用于验证整个数据完整和可靠性 |

9. 分布式session如何管理，你有哪些方案  
    Redis做缓存持久化存储session  
    数据库存储session10. 学过数据结构和算法吗（当然），你说说二分搜索的过程  
    二分搜索有一点要求就是数据有已经排序好的，假设是自然排序的，拿到目标数据后查找中间的值，如果大了，就去右边一部分的中间值比较，小了就去左边一部分的中间值11. 说一下快排的过程，写一下伪代码  
    取一个值，然后设置两个指针，一个指针先从后到前开始遍历，遇到小于这个值的就停止，然后另一个指针从前到后遍历，遇到大于这个值的就停止，知道这两个指针相遇，此时交换这个值与相遇的时候指针的值，以这个坐标为边界两边开始递归12. 了解哪设计模式，举例说说在jdk源码哪些用到了你说的设计模式  
    单例：ioc容器  
    模板：ioc、springmvc  
    建造者模式：lombok  
    工厂：ioc  
    代理：aop  
    订阅/发布：消息队列，redis的pub/sub

13. 你有什么问我吗（仍然上面三个问题）

**四面**

1. 来个自我介绍呗，不超过3分钟  
2. 介绍下你最熟悉的项目  
3. 项目使用了什么架构，亮点是什么  
    MVC，用到Lucene，用aop实现了权限的管理  
4. 平时主要学习什么课程  
    Java、数据结构、数学建模  
5. 你目前的研究方向是什么  
6. 家是哪的  
7. 喜欢看什么书8. 大概什么时候能来实习呢

**五面**

1. 自我介绍下吧  
2. 做了哪些项目  
3. 看你在问题中说你在杭州看到很多商贩使用付款二维码，你对支付宝怎么看  
4. 每天有那么多人使用支付宝，这些数据如果给你存储，你会怎么设计呢（不是说HR不问技术问题吗？不愧是阿里的HR）  
5. 为什么想来支付宝实习呢  
6. 技术栈、对移动支付比较好奇，  
7. 你身边同学如何评价你、老师呢  
8. 如果与同事发生了意见的不一致，你会如何解决呢  
    首先就是要确保双方都理解了对方的意思，因为有些是沟通不充分导致的，然后同时综合对比不同意见，可能会对工作内容产生的影响，并且会根据利弊来选择方法