1. 典型的Top K算法\_找出一个数组里面前K个最大数

方法：一、**最容易想到的是先对数据快速排序，然后输出前k个数字。**

**二、维护一个K大小的数组，从小到大排序**

二、**维护一个K大小的最小堆。**

**三、采用hashmap、HashTable、ConcureetHashMap和最小堆**

1. 维护一个Key为Query字串，Value为该Query出现次数的HashTable，每次读取一个Query，如果该字串不在Table中，那么加入该字串，并且将Value值设为1；如果该字串在Table中，那么将该字串的计数加一即可。最终我们在O(N)的时间复杂度内完成了对该海量数据的处理。
2. 借助堆结构，维护一个K(该题目中是10)大小的小根堆，然后遍历300万的Query，分别和根元素进行对比
3. **总结：**先用Hash表统计每个Query出现的次数，O（N）；然后第二步、采用堆数据结构找出Top 10，N\*O（logK）。所以，我们最终的时间复杂度是：O（N） + N'\*O（logK）。（N为1000万，N’为300万）。
4. **java中TreeMap的实现----红黑树**
5. TreeSet 底层是通过 TreeMap 来实现的（如同HashSet底层是是通过HashMap来实现的一样），因此二者的实现方式完全一样。而 TreeMap 的实现就是红黑树算法。
6. 运行速度都要比Hash集合慢，**他们内部对元素的操作时间复杂度为O(logN)，而HashMap/HashSet则为O(1)**
7. 对于 TreeMap 而言，由于它底层采用一棵“红黑树”来保存集合中的 Entry，这意味这 TreeMap 添加元素、取出元素的性能都比 HashMap 低（红黑树和Hash数据结构上的区别）：当 TreeMap 添加元素时，需要通过循环找到新增 Entry 的插入位置，因此比较耗性能；当从 TreeMap 中取出元素时，需要通过循环才能找到合适的 Entry，也比较耗性能。
8. **红黑树，一种二叉查找树**，但在每个结点上**增加一个存储位表示结点的颜色**，可以是Red或Black。通过对任何一条从根到叶子的路径上各个结点着色方式的限制（**旋转和着色**），红黑树**确保没有一条路径会比其他路径长出俩倍，因而是接近平衡的**。红黑树，**能保证在最坏情况下，基本的动态几何操作的时间均为O（lgn）**。
9. TreeMap添加节点（TreeMap 中使用 Entry 内部类代表节点）的实现，TreeMap 集合的 put(K key, V value) 方法实现了将 Entry 放入排序二叉树中
10. ”**为什么TreeMap采用红黑树而不是二叉查找树？**“

其实这个问题就是在问红黑树相对于排序二叉树的优点。我们都知道排序二叉树虽然可以快速检索，但在最坏的情况下：如果插入的节点集本身就是有序的，要么是由小到大排列，要么是由大到小排列，那么最后得到的排序二叉树将变成链表：所有节点只有左节点（如果插入节点集本身是大到小排列）；或所有节点只有右节点（如果插入节点集本身是小到大排列）。在这种情况下，排序二叉树就变成了普通链表，其检索效率就会很差

1. ”TreeMap、TreeSet 对比 HashMap、HashSet的优缺点？“

**缺点：**  对于 TreeMap 而言，由于它底层采用一棵“红黑树”来保存集合中的 Entry，这意味这 TreeMap 添加元素、取出元素的性能都比 HashMap （O(1)）低：

当 TreeMap 添加元素时，需要通过循环找到新增 Entry 的插入位置，因此比较耗性能（O(logN)）

当从 TreeMap 中取出元素时，需要通过循环才能找到合适的 Entry，也比较耗性能（O(logN)）

**优点：**TreeMap 中的所有 Entry 总是按 key 根据指定排序规则保持有序状态，TreeSet 中所有元素总是根据指定排序规则保持有序状态。

**阿里**

 阿里是四面技术面加一面hr面，每一面的面试官都很nice，感觉都很厉害，能够从面试中学到好多的东西，面试官态度也特别的好，特别耐心，不论从技术上还是面试官上都是体验最好的，最后也成功的收到了阿里的offer，真的感到非常荣幸。

面试大概如下：

一面：

    自我介绍

servlet生命周期

**（init、doGet doPost destory）**

HashMap和ConcurrentHashMap相关

**（数组+链表 Segments数组 分段锁机制）**

****Put 第一个问题****,HashMap 是使用下面的算法来计算元素的存放位置的。

int hash = hash(key);

int i = **indexFor(hash, table.length);**

Get方法

**int hash = (key == null) ? 0 : hash(key);**

for (Entry<K,V> e = table[indexFor(hash, table.length)];e != null;e = e.next) {

Object k;

if (e.hash == hash && ((k = e.key) == key || (key != null && key.equals(k))))

return e;

}

类加载机制

**（类加载步骤 双亲委派机制）**

SSH流程

**（Dao Service Action MVC设计模式）**

Redis相关

**（nosql数据库 ，， key-value ，，5中，，在内存中进行数据操作，性能高，但是不适合海量数据存储 适合少量数据的高并发读写 采用单线程）**

事物隔离级别 （ACID）

**（读未提交、读已提交、可重复读、可串行读 数据库锁机制 乐观锁 悲观锁实现）**

SpringIOC

**（控制反转，依赖注入 注入方式 循环依赖 工厂模式）**

    时间有点久，还有别的有点忘了。。。

二面：

    自我介绍

**美团**

    美团是通过内推的，面试地点是在望京，现场面，两面技术面之后过了大概有一个星期收到HR电话确认发offer。

一面：

    自我介绍

    HashMap相关，ConcurrentHashMap相关

多线程通信

**（共享内存）**

进程通信

**（管道、共享内存、消息传递、信号量、socket）**

    项目相关，问的比较细

    还有好多，不过时间有点久忘记了

手写算法，各种排序

**（冒泡排序 哨兵、快速排序 一次划分 左边排序 右边排序）**

**（堆排序 调整堆 for循环创建堆 for循环交换首位节点并调整堆）**

**（归并排序 划分一半，左边归并 右边归并 通过辅助数组进行合并）**

**网易**

    网易面试的是云音乐部门，两面技术面加一轮HR面，都是现场面，面试体验也不错，感觉面试官也很nice。

一面：

    自我介绍

mysql数据库隔离级别

**（四种）**

redis相关

**（key-value nosql）**

    项目各种细节，问得比较细

二面：

类加载机制

**（双亲委派机制）**

数据库索引问了很多，包括底层实现

**（加快检索速度 B+树实现 主索引 辅助索引 MISAM Innodb）**

HR面：

    都是比较常规的问题

**阿里：**

1. 自我介绍
2. 个人项目经历
3. java集合类用过哪些，讲一下TreeMap，讲一下list系列的，ArrayList，LinkedList

**（红黑树 默认对key按照字典顺序进行升序排序 put和get方法时间复杂度）**

1. 讲一下java多线程怎么做的，讲一下相关经历

**（runnable避免了Java中单继承的限制 ； 其次通过一个对象可以创建多个线程，实现资源共享）**

（四种实现方式 ：**Callable和Future，它俩很有意思的，一个产生结果，一个拿到结果。 Callable接口类似于Runnable，从名字就可以看出来了，但是Runnable不会返回结果，并且无法抛出返回结果的异常，而Callable功能更强大一些，被线程执行后，可以返回值，这个返回值可以被Future拿到，也就是说，Future可以拿到异步执行任务的返回值**）

1. 讲一下java线程池

（**ExecutorService 来创建线程池 ThreadPoolExecutor 线程池执行者 参数**：

**1、corePoolSize：核心线程数**

核心线程会一直存活，及时没有任务需要执行

当线程数小于核心线程数时，即使有线程空闲，**线程池也会优先创建新线程处理**

**2、maxPoolSize：最大线程数**

当线程数>=corePoolSize，且任务队列已满时。线程池会创建新线程来处理任务

当线程数=maxPoolSize，且任务队列已满时，线程池会拒绝处理任务而抛出异常

**3、keepAliveTime：线程空闲时间**

当线程空闲时间达到keepAliveTime时，线程会退出，直到线程数量=corePoolSize

4、 线程池按以下行为执行任务

**当线程数小于核心线程数时，创建线程。**

**当线程数大于等于核心线程数，且任务队列未满时，将任务放入任务队列。**

**当线程数大于等于核心线程数，且任务队列已满**

**若线程数小于最大线程数，创建线程**

**若线程数等于最大线程数，抛出异常，拒绝任务**）

1. 将一下springIOC容器初始化的过程

**（classPathXMLApplicationContext fileSystemXMLApplicationContext）**

1. 讲一下session和cookie，原理

**（记录用户状态的，浏览器端，服务器端，4kb，只能存取字符串 ，session存取任何Java对象）**

#### 二面

1. 学习成绩怎么样，排名多少
2. 讲一下java的接口和抽象类的区别
3. 讲一下java线程同步都有哪些方法

**（sysnchnorized volatile reentrantLock threadLocal）**

1. 讲一下equals和==的区别
2. 讲一下类加载的顺序

（5大步骤）

1. 讲一下操作系统中线程和进程
2. 用过哪些框架，那讲一下spring用过哪些内容，ioc和aop，讲一下原理

**（单例、工厂、模板方法、策略模式、代理模式、观察者模式）**

1. 做一套笔试题

#### 三面

1. 学习成绩怎么样，排名多少
2. 你不是科班出身的，计算机相关课程有自学吗，都怎么学习，做过些什么
3. 讲一下你做过的有意思的项目，这其中有什么技术难点呢
4. 操作系统的线程通信有哪些，讲一下全局变量，怎么做，讲一下信号，原理是什么
5. 讲一下类加载的过程，双亲委派模型
6. 数学题，估算7的128次方十进制有多少位

**（50-1）^64 (5\*10)^64 (100/2)^64**

1. 讲一下你周围同学的打算
2. 讲一下**bio和nio**
3. **BIO（Blocking I/O）同步阻塞I/O**

这是最基本与简单的I/O操作方式，其根本特性是做完一件事再去做另一件事，一件事一定要等前一件事做完，这很符合程序员传统的顺序来开发思想，因此BIO模型程序开发起来较为简单，易于把握。

但是BIO如果需要同时做很多事情（例如同时读很多文件，处理很多tcp请求等），就需要系统创建很多线程来完成对应的工作，**因为BIO模型下一个线程同时只能做一个工作**，如果线程在执行过程中依赖于需要等待的资源，那么该线程会长期处于阻塞状态，我们知道在整个操作系统中，线程是系统执行的基本单位，**在BIO模型下的线程 阻塞就会导致系统线程的切换**，从而对整个系统性能造成一定的影响。当然如果我们只需要创建少量可控的线程，那么采用BIO模型也是很好的选择，但如果在需要考虑高并发的web或者tcp服务器中采用BIO模型就无法应对了，**如果系统开辟成千上万的线程，那么CPU的执行时机都会浪费在线程的切换中，使得线程的执行效率大大降低**。此外，关于线程这里说一句题外话，在系统开发中线程的生命周期一定要准确控制，在需要一定规模并发的情形下，**尽量使用线程池来**确保线程创建数目在一个合理的范围之内，切莫编写线程数量创建上限的代码。

1. **NIO（面向buffer和channel、非堵塞的、有Selector）**

**NIO本身是基于事件驱动的思想来实现的，其目的就是解决BIO的大并发问题**，在BIO模型中，如果需要并发处理多个I/O请求，那就需要多线程来支持，**NIO使用了多路复用器机制**，以socket使用来说，多路复用器通过不断轮询各个连接的状态，只有在socket有流可读或者可写时，应用程序才需要去处理它，**在线程的使用上，就不需要一个连接就必须使用一个处理线程了**，而是只是有效请求时**（确实需要进行I/O处理时），才会使用一个线程去处理，这样就避免了BIO模型下大量线程处于阻塞等待状态的情景**。

相对于BIO的流，**NIO抽象出了新的通道（Channel）作为输入输出的通道**，并且提供了**缓存（Buffer）的支持**，在进行**读操作时，需要使用Buffer分配空间，然后将数据从Channel中读入Buffer中，对于Channel的写操作，也需要现将数据写入Buffer，然后将Buffer写入Channel中。**

1. **Selector允许单线程处理多个 Channel。**如果你的应用打开了多个连接（通道），但每个连接的流量都很低，使用Selector就会很方便。例如，在一个聊天服务器中。
2. 分享一下你最近看过比较有意思的文章或开源轮子

#### 四面

1. 你不是科班出身的，计算机基础课程自学了吗，都学了哪些，怎么学的。
2. 讲一下线程和进程。
3. java也是自学的？都是怎么接触和学习一门新的语言的。
4. 讲一下java线程同步都有哪些方法
5. 讲一下java线程通信都有哪些方法
6. 讲一下java线程竞争。
7. wait和sleep的区别
8. java创建线程都有哪些方法，讲一下callable和runnable的区别
9. 讲一下线程池，为什么要使用线程，有什么好处
10. 讲一下二叉树，以及怎么用二叉树排序。
11. 怎么设计一个模型解决生产者消费者问题。
12. 讲一下你最近看过比较有意思的文章。
13. 讲一下双向链表，适用场景

#### hr面

### 阿里 一面

* 去年实习经历
  + 异步调用的使用场景
  + 数据库设计
    - MySQL 索引结构
    - 主键和普通索引的区别
  + 服务化框架
    - 服务注册的原理
    - 负载均衡方案
  + 批量数据处理
    - 如何控制服务器并发量
* 其他项目经历
* 除了 Java 还在学习什么技术
* 二分查找的实现
* 数据库的 ACID
  + 如何理解隔离性，举例如果不隔离会产生什么问题

**隔离级别越高，越能保证数据的完整性和一致性，但是对并发性能的影响也越大。对于多数应用程序，可以优先考虑把数据库系统的隔离级别设为Read Committed，它能够避免脏读取，而且具有较好的并发性能。尽管它会导致不可重复读、虚读和第二类丢失更新这些并发问题，在可能出现这类问题的个别场合，可以由应用程序采用悲观锁或乐观锁来控制。**

* 编程：设计 parseInt(String)
  + **异常处理**
  + **int 的取值范围**

### 阿里 二面

* 再次确认是本科生，有没有读研计划
* Java 集合框架
  + 举例常用数据结构
  + ArrayList/LinkedList 实现对比，使用场景
  + HashMap 实现
* JVM 内存分区、堆的分代
* JVM 垃圾回收算法
* 去年实习经历，印象最深的 bug
* Java 网络编程
  + 有没有用过网络通信框架（没有）
  + **Java IO/NIO 区别**
* 多线程与并发
  + 乐观锁/悲观锁
  + volatile 关键字

### 阿里 三面

* **算法：从大量 ID 中找出出现次数大于 3 次的所有 ID** 
  + ID 取值范围较小：**Bitmap**，每个取值用 2bit
  + ID 取值范围较大：Hash，取模，分片统计，合并

**BitMap解决海量数据寻找重复、判断个别元素是否在海量数据当中等问题。**

先看看这样的一个场景：给一台普通PC，2G内存，要求处理一个包含40亿个不重复并且没有排过序的无符号的int整数，给出一个整数，问如果快速地判断这个整数是否在文件40亿个数据当中？

问题思考：

   40亿个int占（40亿\*4）/1024/1024/1024 大概为14.9G左右，很明显内存只有2G，放不下，因此不可能将这40亿数据放到内存中计算。要快速的解决这个问题最好的方案就是将数据搁内存了，所以现在的问题就在如何在2G内存空间以内存储着40亿整数。一个int整数在java中是占4个字节的即要32bit位，如果能够用一个bit位来标识一个int整数那么存储空间将大大减少，算一下40亿个int需要的内存空间为40亿/8/1024/1024大概为476.83 mb，这样的话我们完全可以将这40亿个int数放到内存中进行处理。

具体思路：

**1个int占4字节即4\*8=32位**，那么我们只需要申请一个int数组长度为 **int tmp[1+N/32]即可存储完这些数据**，其中N代表要进行查找的总数，tmp中的每个元素在内存在占32位可以对应表示十进制数0~31,所以可得到BitMap表:

**tmp[0]:可表示0~31**

**tmp[1]:可表示32~63**

**tmp[2]可表示64~95**

.......

那么接下来就看看十进制数如何转换为对应的bit位：

假设这40亿int数据为：6,3,8,32,36,......，**那么具体的BitMap表示为：可以进行无重复元素的排序**

**优点：**

**排序查询效率高，时间复杂度：O（1）**

占用内存少，比如N=10000000；只需占用内存为N/8=1250000Byte=1.25M。

**缺点：**所有的数据不能重复。即不可对重复的数据进行排序和查找。

* 项目经历，创业项目最后成功上线了吗
* 对实习的期望、未来职业规划

**因为我可能记不清楚问题是在后面的哪次面试面的，所以就一起整理了下。**

**——————————菜鸟一面&其他面————————————————————**

**说实在话去菜鸟也是自己之前没想到的，在XX那里通过了所有的技术面（HR告知）**

**但是因为最后两轮技术面间隔时间比较久，导致系统自动释放简历，最后在公海被**

**菜鸟捞起来了。于是又进行了一次菜鸟的面试**

**面试官：简单介绍下你自己**

**面试官：你能向我解释一下**

**synchronized关键字吗？**

**面试官：你能向我解释一下volatile关键字吗？**

****(可见性、非原子性、禁止指令重排序优化，保证有序性)****

**面试官：Spring有了解过吗？Spring的AOP能向我解释一下吗？**

****（面向切面编程、代理模式来实现、提高代码重用性、降低模块之间的耦合性）****

面试官：你最近在学习的新技术是什么？看的新书呢？看过什么书？

**（java 8 那么大表达式：集合遍历，实现runnable接口 ； Optional 类 null的安全性 ofNullable和orelse ； 函数式编程 stream类 视图 foreach distinct sorted filter map）**

面试官：你获取相应知识的来源是什么？

面试官：你家是哪里的？

面试官：说说项目吧，以及项目中遇见了什么样的问题

面试官：我看你大学做过很多的项目和比赛，说说你的队友对于你的评价

面试官：hashmap的实现原理

**面试官：网络编程会吗？你说一下异步非阻塞（同步非阻塞，这个记不太清楚了）网络应该怎么实现？**

**面试官：Chrome用过吗？两个标签之间是进程关系还是线程关系？为什么是这个关系**

**面试官：设计模式有些什么了解？最近在看什么设计模式？你来说下servlet在多线程的情况下怎么保证单例模式的**

**面试官：Spring涉及到了哪些设计模式？说说你了解的**

****（单例、工厂、模板方法、策略模式、代理模式、观察者模式）****

****观察者模式：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象得到通知并自动更新**。**

**面试官：数据库的索引表熟悉吗？他是怎么实现的？**

（B+树）

**面试官：给你一个100G的全英文字母的数据，你的电脑内存是8G的，你要怎么把这100G的数据进行一个A~Z的排序，假设现在你已经是阿里的员工，你要给你的师弟进行讲解**

**面试官：你之前做项目，是在Linux上配置的还是在Windows上配置的**

**面试官：MySQL是用的命令行还是用的图形化界面**

**面试官：HashMap的value和key存放在什么地方？**

**——————————总结**

我个人习惯于 **将一个知识点 分解为**  **xxx是什么，xxx有什么用，如何实现；缺点是什么（以及为什么有这个缺点，缺点如何补救）**。  
  
Concurrent Mark Sweep (CMS) 垃圾收集器（**希望JAVA垃圾回收器回收垃圾的时间尽可能短**）。

CMS也是**采用分代策略的，用于收集**老年代**的垃圾对象**，**并且分为6个阶段来执行GC**。**在某些阶段，应用的线程会被挂起，而在另外的阶段里，垃圾回收线程可以与应用的线程一起工作。**

**CMS是基于标记-清除算法的，会造成内存碎片。**

**CMS的GC过程有6个阶段（4个并发，2个暂停应用程序）：**

**初次标记**：该阶段进行可达性分析，标记GC ROOT能**直接关联**到的对象，**暂停其它应用。**

**并发标记：**从初次标记收集到的‘根’对象引用开始，遍历所有能被引用的对象。

**并发预清理：**改变当运行第二阶段时，由应用程序线程产生的对象引用，以更新第二阶段的结果。

**重新标记：**由于并发预处理是并发的，对象引用可能发生进一步变化。因此，应用程序线程会再一次被暂停以更新这些变化，并且在进行实际的清 理之前确保一个正确的对象引用视图。这一阶段十分重要，因为必须避免收集到仍被引用的对象。

**并发清理：**所有不再被应用的对象将从堆里清除掉。

**并发重置：**收集器做一些收尾的工作，以便下一次GC周期能有一个干净的状态。

**总结一下：**

**CMS采用了多种方式尽可能降低用户程序停顿时间，挺高了服务器的响应速度，停顿时间降低的同时牺牲了CPU吞吐量 。**

CMS(Concurrent Mark-Sweep)是**以牺牲吞吐量为代价来获得最短回收停顿时间的垃圾回收器**。对于要求**服务器响应速度**的应用上，这种垃圾回收器非常适合

接下来就写下三次面试题目：  
阿里一面：

1.个人介绍

2.项目介绍，项目介绍首先讲最好的项目，因为后面的项目可能都没时间去讲。

3.数据库 联合索引 用法

**（主键索引、普通索引、唯一索引、联合索引、全文索引）**

4.Spring IOC初始化过程

5.ConcurrentHashMap实现原理

6.CAS操作

**（通过调用本地方法实现，比较并交换，三个参数，内存值、期望值、新的值）**

7.ReentrantLock和Synchronized区别

**（一个是类，一个是关键字，一个基于JDK实现，一个基于JVM实现，提供公平锁，一个可以实现中断线程tryLock或者lockInterruptbly）**

8.CMS垃圾回收过程

（**Concurrent Mark Sweep：并发的标记清除算法的垃圾回收器，起作用是尽可能地降低垃圾回收的时间，减少应用程序的暂停等待时间；主要作用于老年代的gc，主要分为6个阶段，初始标记阶段，并发标记阶段，清除预处理阶段，重新标记阶段，并发清除阶段，并发重置阶段，会产生内存碎片**）

9.Full GC次数太多了，如何优化。

10.直接内存 如何 管理的

11.线程池 参数。

（**corePoolSize、maxPoolSize、threadAliveTime、unit、任务队列**）

**（当线程数<corePoolSize时，哪怕有空闲线程，也会创建新的；**

**当线程数<maxPoolSize时，且任务队列未满，会将任务放到任务队列中去；**

**当线程数<maxPoolSize时，且任务队列已满，则会创建新的线程；**

**当线程数=maxPoolSize时，抛出异常、拒绝任务执行；）**

线程创建的过程。有几种方法。

**（四种）**

线程资源如何回收的。

**（线程私有资源，不用进行垃圾回收；线程共享资源，需要进行垃圾回收）**

阿里二面

1.自我介绍

2.项目介绍

3.堆和栈介绍

4.线程安全

5.乐观锁，悲观锁

**（Java中的悲观锁、数据库中的悲观锁）**

1. TCP三次握手
2. socket通信有关，select epoll

8.项目中的难点有哪些

我答得是：使用一个组合的设计模式 去解决文章多级分类的问题。

hr面试

1.自我介绍。

2.项目介绍。项目中难点。得过啥奖没。

3.项目有什么不足？

4.前面的面试 发挥怎么样？

5.前面的面试难度怎样？

愿大家都能拿到自己理想的offer。

1. **外链接、内连接**

内连接：利用内连接可获取两表的公共部分的记录

外连接：外连接分为两种，一种是左连接（Left JOIN）和右连接（Right JOIN）  
 (1)左连接（Left JOIN）：即图3公共部分记录集C＋表A记录集A1。

1. 右连接（Right JOIN）：即图3公共部分记录集C＋表B记录集B1。
2. **随机数生成 Random nextInt（）**

为了后续推广中，给其他用户起到促进作用，需要添加机器人数据，对初始数据累加随机数据。  
1——10的几率：60%  
10——20:20%  
20——30:10%  
30——40:5%  
40——50%：2%  
50——100:3%

在前面的方法介绍中，nextInt(int n)方法中生成的数字是均匀的，也就是说该区间内部的每个数字生成的几率是相同的。那么如果生成一个[0,100)区间的随机整数，则每个数字生成的几率应该是相同的，而且由于该区间中总计有100个整数，所以每个数字的几率都是1%。按照这个理论，可以实现程序中的几率问题。

示例：随机生成一个整数，该整数以55%的几率生成1，以40%的几率生成2，以5%的几率生成3。实现的代码如下：

int n5 = r.nextInt(100);int m; //结果数字if(n5 < 55){ //55个数字的区间，55%的几率

m = 1;

}else if(n5 < 95){//[55,95)，40个数字的区间，40%的几率

m = 2;

}else{

m = 3;

}

#### 1. 菜鸟网络（三面挂）：

一面（电话）：

1. synchronized是否可以由同一个函数多次调用（一个函数递归调用）；

**（可重入性）**

1. hashMap是否线程安全，与HashTable的区别；
2. ConcurrentHashMap中的扩容是否需要对整个表上锁；

**（对单个segment进行扩容）**

1. 浏览器输入url的过程；

**（建立连接、请求报文，响应报文、释放连接）**

1. MySQL索引的结构，什么是B+树；

**（B+，多路平衡查找树，所有关键字都在叶子节点中、指针相连）**

1. 什么是红黑树，讲一讲如何修复；

**（二叉查找树，节点中多设置了一位用来存储节点颜色，保证最坏情况下，查找效率）**

二面（电话）：

1. 垃圾回收的方式；
2. **哪些可以作为GC ROOT；**

**1、 虚拟机栈（栈帧中的本地变量表）中引用的对象。**

**2、 本地方法栈中JNI（即一般说的native方法）引用的对象。**

**3、 方法区中的静态变量和常量引用的对象。**

1. 快速排序、堆排序、选择排序；
2. BeanDefination是什么；
3. AOP的原理；
4. MySQL的索引；
5. HashMap与HashTable的区别；

三面（现场）：

1. 项目里的难点、特色；
2. 根据项目中的难点想出更优的解决办法；
3. 聊人生、理想；
4. 有没有看过Spring的源码；
5. 现在要你为广州市设计消防系统，两分钟思考一下，如果规划消防站的设立；

第三现场面后被挂，当时觉得莫名其妙，没写代码没问基础，知识抓着项目细节问优化，现在想想当时一些问题回答的有些避重就轻，以为是遮掩过去了，可能还是被抓住了。

**2.迭代集合时有效的去除元素**

在迭代过程中修改集合的唯一正确方法是使用Iterator.remove() 例如，

**Iteratoritr = list.iterator();while(itr.hasNext()) {// do somethingitr.remove();}**

一个最常见的错误代码是

for(Integer i: list) {list.remove(i);}

通过运行上面的代码，您将得到ConcurrentModificationException。这是因为已经生成了一个迭代器（在for语句中）来遍历列表，但同时该列表由Iterator更改。remove（）。在Java中，“一个线程通常不允许修改集合，而另一个线程正在迭代它。”

**3.如何将List转换为int []？**

最简单的方法可能是在Apache Commons Lang库中使用ArrayUtils。

int[] array = ArrayUtils.toPrimitive(list.toArray(new Integer[0]));

在JDK中，没有捷径。请注意，您不能使用列表。toArray（），因为那会转换List到int[] 。正确的方法如下，

int[] array = new int[list.size()];

for(int i=0; i < list.size(); i++) {

array[i] = list.get(i);

}

**4.如何将int []转换为List？**

最简单的方法可能仍然是在Apache Commons Lang库中使用ArrayUtils，如下所示。

List list = Arrays.asList(ArrayUtils.toObject(array));

在JDK中，也没有捷径。

int[] array = {1,2,3,4,5};List list = new ArrayList();for(int i: array) {list.add(i);}

**6.将List转换为Set的最简单方法是？**

有两种方法可以这样做，具体取决于你想如何平等定义。第一部分代码将一个列表放入HashSet中。主要通过hashCode（）来识别重复。在大多数情况下，它会起作用。但是如果您需要指定比较方式，最好使用第二段代码，您可以在其中定义自己的比较器。

**Set set = new HashSet(list);**

Setset = new TreeSet(aComparator);**set.addAll(list);**

**7.如何从ArrayList中删除重复的元素？**

这个问题与上面的问题很相关。

如果你不关心ArrayList中元素的排序，聪明的方法是将列表放入一个集合中去除重复，然后将其移回列表中。这是代码

ArrayList\*\* list = ... // 初始化一个带有重复元素的列表

**Set set = new HashSet(list);**

**list.clear();**

**list.addAll(set);**