1. 集合
2. HashMap初始大小是多少?怎么扩容的？

Jdk1.7默认是16,key的个数超过容量长度\*加载因子

的数量，会扩容为原来的2倍。Jdk1.8在扩容的时候采用

expectedSize / 0.75 + 1 公式，为了提高效率可以采用Maps.newHashMapWithExpectedSize().

1. Arrays.sort 是怎么实现排序的?

实现Comparator接口，或用lambda表达式,，实现倒序排序。

快速排序算法，先选择一个基准，从指针end往前移,遇到大于end值，进行交换，然后从指针start往后移动，遇到小于start值，进行交换，

递归完成排序。

1. HashMap实现原理？

数组+链表,HahMap 静态内部类Entry属性有key,value,next,hash 4个属性; 记录存储个数size,,阈值为初始容量\*负载因子，负载因子；

key通过hash函数确定hash值，hash值与容量 &运算确定位置，通过遍历链表查找value;对于自定义对象作为key,需要重写equals和hashcode方法

1. ConcurrentHashMap 比HashTable 性能优的地方?

HashTable对整个集合加锁，为了防止ConcurrentModificationException抛出，ConcurrentHashMap 使用volatile 关键字，直接修改内存的值，注销直接的缓存，禁止程序执行结果的重排序，写操作由于读操作，采取锁分段技术。

1. 为什么要重写hashcode方法？

equals继承Object,对于值对象，比较的是两个对象的值，对于引用对象比较的是两个对象的地址。如果你重写了equals 方法，而没有重新hashcode 方法，那么会出现明明两个对象是相同的，却因它的hash值不同，让这两个对象也不一样。

1. 多线程

线程池工作队列饱和，有哪些策略？

1. Abort策略，队列满了之后，新提交任务时，抛出异常。
2. CallerRuns策略，调节机制，既不抛弃任务也不抛出异常，在exectord的线程中去执行。
3. Discard策略，新提交的任务被抛弃。

4．DiscardOldest策略，抛弃对头的任务，提交新的任务。

三、Spring

动态代理实现的两种方式，以及他们的区别？  
SpirngAOP利用动态代理实现日志打印,事务管理，通过JDK和CGLIB两种方式实现，JDK需要被代理类实现一个接口，代理类实现同样的接口，CGLIB是代理类继承被代理类，作为代理对象，实现AOP的功能。

Spring如何实现事务管理?

Spring事务管理分为编程式事务管理和基于AOP技术实现的声明式事务管理，声明式事务管理只能作用在方法上，

切点精确到哪个类，切面精确到哪个方法,创建代理类,调用方法调用的是代理对象的方法,编程式事务管理能作用在代码块上。

Spring可以设置事务的传播行为，隔离级别，事务的只读，以减轻

事务对数据库的压力，遇到异常时，发生事务回滚。

1. linux命令

统计一个文件下，一个字符串出现的次数?

在命令 模式下输入 :%s/objStr//gn

1. redis

redis持久化的两种方式?

RDB和AOF,RDB 定时将操作日志写入硬盘，AOF是实时的将操作日志写入磁盘。

1. hibernate

hibernate的find和load方法的区别

find会先在一级缓存中查询，如果查不到再去数据库去查询，load可以取到一级缓存和二级缓存的数据。

1. 数据库

索引的实现原理

分为Clustered Index(主键索引) 和Secondary Index（辅助索引）,基于B+Tree的数据结构，Secondary Index 的索引树存的是Clustered Index的索引值，为了减小update,insert,delete操作的复杂度。

主键查询----直接在Clustered B+Tree上查询。

辅助索引查询---先在Secondary B+Tree上查询到主键，然后在ClusteredB + Tree上查询。

Insert---在Clustered B+Tree上插入数据，然后在SecondaryB +Tree上插入主键。

DELETE---在Clustered B+Tree上删除数据，在所有SecondaryB+Tree上删除主键。

UPDATE非主键列---在ClusteredB+Tree更新数据。

UPDATE主键列---在ClusteredB+Tree假删除原来的记录，插入新的记录,Secondary B+Tree删除原来的数据，插入新的数据。

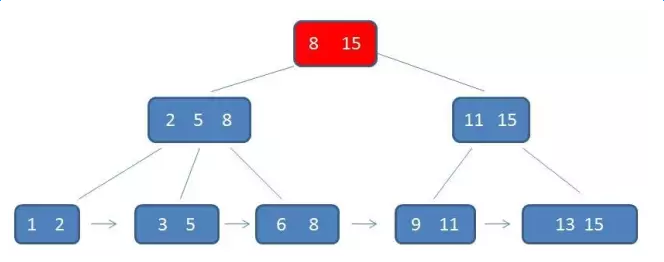
UPDATE辅助索引的键值---在ClusteredB+Tree上更新数据，在每一个SecondaryB+Tree删除原来的原来的主键，新增原来的主键。

1. JVM调优
2. 算法

折半查找

public static int searchByHalf(int key,int[] array){  
 int pos = -1;  
 int start = 0,end = array.length - 1;  
 int mid = (start + end) / 2;  
 while(start <= end){  
 if(array[mid] == key){  
 pos = mid;  
 break;  
 }else if(array[mid] > key){  
 end = mid - 1;  
 }else {  
 start = mid + 1;  
 }  
 mid = (start + end) / 2;  
  
 }  
 return pos;  
}

B+Tree



1. 单节点可以存储更多的元素，使得查询磁盘IO次数更少。
2. 所有查询都走到叶子节点，查询性能稳定。
3. 所有叶子节点形成有序列表，便于范围查询。