Java里的集合类型的数据结构和实现

1. java集合类存放在Java.util包中，主要有三种：set，list，map/

Collection是集合List，Set，Queue的基本接口。

Iterator是迭代器，是用来遍历集合数据的。

Map是映射表的基础接口

1. List数据类型

List是一个有序的collection。他有三个实现类，ArrayList，Vector和LinkedList。

1. ArrayList是通过数组实现的，随机访问快。缺点是，在重点插入删除的时候代价较高。线程不安全的，因为多个线程可以对它进行修改。
2. Vector

和ArrayList一样，不过他是线程安全的，支持线程的同步。

1. LinkList

是java中封装好的链表，很适合数据的动态插入和删除。访问和遍历速度较慢，可以当作堆栈，队列，双向队列使用。

Set是一个独一无二的集合类型，里面的值不能重复。而对象相等性的本质是对象的hashcode值相等，根据内存地址计算。

Set下面有HashSet

1. HashSet表里存放的是哈希值，从而获得了当前要查询的内容的位置。有利于判断查询效率的提升。插入时，先比较hashcode方法获得的值，如果一样，再比较equals方法。不一样就链表顺延，还一样就不懂。
2. TreeSet。用二叉树的原理来排序一个有序的集合，效果是得到的集合总是有序的。如果是对象想要实现，需要覆写compaer函数。
3. LinkHashSet（继承HashSet和LinkedHashMap来实现）

Map(主要有Hash Map）重点。

HashMap（非线程安全）：

数组加链表加红黑树。查找时先根据hash值定义到数组下标。再顺着找下去。

当链表中元素大于8的时候，链表转化成红黑树，红黑树的查询效率为logn。

负载因子决定了size达到多少时扩容。

HashMap是“懒加载”，在构造器中值保留了相关保留的值，并没有初始化table<Node>数组，当我们向map中put第一个元素的时候，map才会进行初始化！

关键为题：

HaspMap的resize函数

size必须是2的整数次方原因

get和put方法流程

resize方法

影响HashMap的性能因素（key的hashCode函数实现、loadFactor、初始容量）

HashMap key的hash值计算方法以及原因（见上面hash函数的分析）

HashMap内部存储结构：Node数组+链表或红黑树

table[i]位置的链表什么时候会转变成红黑树（上面源码中有讲）

HashMap主要成员属性：threshold、loadFactor、HashMap的懒加载

HashMap的get方法能否判断某个元素是否在map中

HashMap线程安全吗，哪些环节最有可能出问题，为什么？

HashMap的value允许为null，但是HashTable和ConcurrentHashMap的valued都不允许为null，试分析原因？

HashMap中的hook函数（后面讲解LinkedHashMap时会讲解，这可作为HashMap的延伸知识，增加面试官对你的印象）

CurrentHashMap：线程安全的。

TreeMap：排好序的map