集合

Java集合框架中有哪些类？都有什么特点

- 技术点：集合框架
- 思路：分条解释每种类的特点
- 参考回答：可将Java集合框架大致可分为Set、List、Queue 和Map四种体系

    - Set：代表无序、不可重复的集合，常见的类如HashSet、TreeSet
    - List：代表有序、可重复的集合，常见的类如动态数组ArrayList、双向链表 - LinkedList、可变数组Vector
    - Map：代表具有映射关系的集合，常见的类如HashMap、LinkedHashMap、TreeMap
    - Queue：代表一种队列集合

集合、数组、泛型的关系，并比较

- 技术点：集合、数组、泛型
- 参考回答：
- （1）集合和数组的区别：

    - 数组元素可以是基本类型，也可以是对象；数组长度限定；数组只能存储一种类型的数据元素
    - 集合元素只能是对象；集合长度可变；集合可存储不同种的数据元素

- （2）泛型相比与集合的好处在于它安全简单。具体体现在提供编译时的强类型检查，而不用等到运行；可避免类类型强制转换

ArrayList和LinkList的区别?

- 技术点：List对比
- 参考回答：
    - ArrayList的底层结构是数组，可用索引实现快速查找；是动态数组，相比于数组容量可实现动态增长
    - LinkedList底层结构是链表，增删速度快；是一个双向循环链表，也可以被当作堆栈、队列或双端队列

ArrayList和Vector的区别?

- 技术点：List对比
- 参考回答：

    - ArrayList非线程安全，建议在单线程中才使用ArrayList，而在多线程中可以选择Vector或者CopyOnWriteArrayList；默认初始容量为10，每次扩容为原来的1.5倍

- Vector使用了synchronized关键字，是线程安全的，比ArrayList开销更大，访问更慢；默认初始容量为10，默认每次扩容为原来的2倍，可通过capacityIncrement属性设置

HashSet和TreeSet的区别?

- 技术点：Set对比
- 参考回答：

  - HashSet不能保证元素的排列顺序；使用Hash算法来存储集合中的元素，有良好的存取和查找性能；通过equal()判断两个元素是否相等，并两个元素的hashCode()返回值也相等

  - TreeSet是SortedSet接口的实现类，根据元素实际值的大小进行排序；采用红黑树的数据结构来存储集合元素；支持两种排序方法：自然排序（默认情况）和定制排序。前者通过实现Comparable接口中的compareTo()比较两个元素之间大小关系，然后按升序排列；后者通过实现Comparator接口中的compare()比较两个元素之间大小关系，实现定制排列

HashMap和Hashtable的区别?

- 技术点：Map对比
- 参考回答：

  - HashMap基于AbstractMap类，实现了Map、Cloneable（能被克隆）、Serializable（支持序列化）接口； 非线程安全；允许存在一个为null的key和任意个为null的value；采用链表散列的数据结构，即数组和链表的结合；初始容量为16，填充因子默认为0.75，扩容时是当前容量翻倍，即2capacity

  - Hashtable基于Map接口和Dictionary类；线程安全，开销比HashMap大，如果多线程访问一个Map对象，使用Hashtable更好；不允许使用null作为key和value；底层基于哈希表结构；初始容量为11，填充因子默认为0.75，扩容时是容量翻倍+1，即2capacity+1

HashMap在put、get元素的过程？体现了什么数据结构?

- 技术点：HashMap
- 参考回答：

  - 向Hashmap中put元素时，首先判断key是否为空，为空则直接调用putForNullKey()，不为空则计算key的hash值得到该元素在数组中的下标值；如果数组在该位置处没有元素，就直接保存；如果有，还要比较是否存在相同的key，存在的话就覆盖原来key的value，否则将该元素保存在链头，先保存的在链尾。
  - 从Hashmap中get元素时，计算key的hash值找到在数组中的对应的下标值，返回该key对应的value即可，如果有冲突就遍历该位置链表寻找key相同的元素并返回对应的value
  - 由此可看出HashMap采用链表散列的数据结构，即数组和链表的结合，在Java8后又结合了红黑树，当链表元素超过8个将链表转换为红黑树

如何解决Hash冲突?

- 技术点：Hash冲突
- 参考回答：
    - 开放定址法：常见的线性探测方式，在冲突发生时，顺序查看表中下一单元，直到找出一个空单元或查遍全表
    - 链地址法：将有冲突数组位置生出链表
    - 建立公共溢出区：将哈希表分为基本表和溢出表两部分，和基本表发生冲突的元素一律填入溢出表
    - 再哈希法：构造多个不同的哈希函数，有冲突使用下一个哈希函数计算hash值

如何保证HashMap线程安全？什么原理

- 技术点：ConcurrentHashMap
- 思路：这里回答一种办法，使用ConcurrentHashMap
- 参考回答：ConcurrentHashMap是线程安全的HashMap，它采取锁分段技术，将数据分成一段一段的存储，然后给每一段数据配一把锁，当一个线程占用锁访问其中一个段数据的时候，其他段的数据也能被其他线程访问。在JDK1.8中对ConcurrentHashmap做了两个改进：
    - 取消segments字段，直接采用transient volatile HashEntry[] table保存数据，将数组元素作为锁，对每一行数据进行加锁，可减少并发冲突的概率
    - 数据结构由“数组＋单向链表”变为“数组＋单向链表＋红黑树”，使得查询的时间复杂度可以降低到O(logN)，改进一定的性能。

HashMap是有序的吗？如何实现有序？

- 技术点：LinkHashMap
- 思路：这里回答一种办法，使用LinkedHashMap
- 参考回答：HashMap是无序的，而LinkedHashMap是有序的HashMap，默认为插入顺序，还可以是访问顺序，基本原理是其内部通过Entry维护了一个双向链表，负责维护Map的迭代顺序
- 引申：LinkHashMap有序的底层实现

HashMap是如何扩容的？如何避免扩容？

- 技术点：HashMap
- 参考回答：
    - HashMap几个默认值，初始容量为16、填充因子默认为0.75、扩容时容量翻倍。也就是说当HashMap中元素个数超过16*0.75=12时会把数组的大小扩展为2*16=32，然后重新计算每个元素在数组中的位置
    - 由于每次扩容还需要重新计算元素Hash值，损耗性能，所以建议在使用HashMap时，最好先估算Map的大小，设置初始值，避免频繁扩容

hashcode()的作用，与equal()有什么区别？

- 技术点：Hash值
- 参考回答：hashCode()用于计算对象的Hash值，确认对象在散列存储结构中的存储地址。和equal()的

区别：

- equals()比较两个对象的地址值是否相等 ； hashCode()得到的是对象的存储位置，可能不同对象会得到相同值
- 有两个对象，若equals()相等，则hashcode()一定相等；hashcode()不等，则equals()一定不相等；hashcode()相等，equals()可能相等、可能不等
- 使用equals()比较两个对象是否相等效率较低，最快办法是先用hashCode()比较，如果hashCode()不相等，则这两个对象肯定不相等；如果hashCode()相等，此时再用equal()比较，如果equal()也相等，则这两个对象的确相等