HashMap面试题整理

JDK8中的HashMap与JDK7的HashMap有什么不一样?

1. JDK8中新增了红黑树，JDK8是通过数组+链表+红黑树来实现的
2. JDK7中链表的插入是用的头插法，而JDK8中则改为了尾插法
3. JDK8中的因为使用了红黑树保证了插入和查询了效率，所以实际上JDK8中的Hash算法实现的复杂度降低了
4. JDK8中数组扩容的条件也发了变化，只会判断是否当前元素个数是否查过了阈值，而不再判断当前put进来的元素对应的数组下标位置是否有值。
5. JDK7中是先扩容再添加新元素，JDK8中是先添加新元素然后再扩容

HashMap中PUT方法的流程?

1. 通过key计算出一个hashcode
2. 通过hashcode与“与操作”计算出一个数组下标
3. 在把put进来的key,value封装为一个entry对象
4. 判断数组下标对应的位置，是不是空，如果是空则把entry直接存在该数组位置
5. 如果该下标对应的位置不为空，则需要把entry插入到链表中
6. 并且还需要判断该链表中是否存在相同的key，如果存在，则更新value
7. 如果是JDK7，则使用头插法
8. 如果是JDK8，则会遍历链表，并且在遍历链表的过程中，统计当前链表的元素个数，如果超过8个，则先把链表转变为红黑树，并且把元素插入到红黑树中

JDK8中链表转变为红黑树的条件?

1. 链表中的元素的个数为8个或超过8个
2. 同时，还要满足当前数组的长度大于或等于64才会把链表转变为红黑树。为什么？因为链表转变为红黑树的目的是为了解决链表过长，导致查询和插入效率慢的问题，而如果要解决这个问题，也可以通过数组扩容，把链表缩短也可以解决这个问题。所以在数组长度还不太长的情况，可以先通过数组扩容来解决链表过长的问题。

HashMap扩容流程是怎样的?

1. HashMap的扩容指的就是数组的扩容， 因为数组占用的是连续内存空间，所以数组的扩容其实只能新开一个新的数组，然后把老数组上的元素转移到新数组上来，这样才是数组的扩容
2. 在HashMap中也是一样，先新建一个2被数组大小的数组
3. 然后遍历老数组上的没一个位置，如果这个位置上是一个链表，就把这个链表上的元素转移到新数组上去
4. 在这个过程中就需要遍历链表，当然jdk7，和jdk8在这个实现时是有不一样的，jdk7就是简单的遍历链表上的没一个元素，然后按每个元素的hashcode结合新数组的长度重新计算得出一个下标，而重新得到的这个数组下标很可能和之前的数组下标是不一样的，这样子就达到了一种效果，就是扩容之后，某个链表会变短，这也就达到了扩容的目的，缩短链表长度，提高了查询效率
5. 而在jdk8中，因为涉及到红黑树，这个其实比较复杂，jdk8中其实还会用到一个双向链表来维护红黑树中的元素，所以jdk8中在转移某个位置上的元素时，会去判断如果这个位置是一个红黑树，那么会遍历该位置的双向链表，遍历双向链表统计哪些元素在扩容完之后还是原位置，哪些元素在扩容之后在新位置，这样遍历完双向链表后，就会得到两个子链表，一个放在原下标位置，一个放在新下标位置，如果原下标位置或新下标位置没有元素，则红黑树不用拆分，否则判断这两个子链表的长度，如果超过八，则转成红黑树放到对应的位置，否则把单向链表放到对应的位置。
6. 元素转移完了之后，在把新数组对象赋值给HashMap的table属性，老数组会被回收到。

为什么HashMap的数组的大小是2的幂次方数?

JDK7的HashMap是数组+链表实现的
JDK8的HashMap是数组+链表+红黑树实现的

当某个key-value对需要存储到数组中时，需要先生成一个数组下标index，并且这个index不能越界。

在HashMap中，先得到key的hashcode，hashcode是一个数字，然后通过hashcode & (table.length - 1) 运算得到一个数组下标index，是通过与运算计算出来一个数组下标的，而不是通过取余，与运算相比于取余运算速度更快，但是也有一个前提条件，就是数组的长度得是一个2的幂次方数。

为什么HashMap在多线程扩容时会出现循环链表的问题?

课上讲 晚上来听Monkey老师直播