知识点整理

# 一、java HashMap工作原理及实现

## 1.概述

官方文档是这样描述的：

Hash table based **implementation of the Map interface**. This implementation provides all of the optional map operations, and permits null values and the null key. (The HashMap class is roughly equivalent to Hashtable, except that it is **unsynchronized** and **permits nulls**.) This class makes no guarantees as to the order of the map; in particular, it does not guarantee that the order will remain constant over time.

大体意思就是：HashMap不是线程同步的，允许NULL值，不保证值的顺序，不保证在某个时间点值的顺序保持不变（因为HashMap在进行扩容的时候，需要重新移动元素，这就造成了元素的位置发生了变化。）

## 2.两个重要的参数

在HashMap中有两个重要的参数，容量（capacity）和负载因子（load factor）。

* **Initial capacity** The capacity is **the number of buckets** in the hash table, The initial capacity is simply the capacity at the time the hash table is created.
* **Load factor** The load factor is **a measure of how full the hash table is allowed to get** before its capacity is automatically increased.

简单的说，capacity就是HashMap中bucket数组的大小，loadFactor就是数据填充了bucket多满以后，容量需要扩充，默认情况下是loadFactor是0.75，扩充的时候扩充到原来的二倍。

使用上需要注意：如果在创建HashMap之前，预先知道数据的大小，假设是100，可以初始化HashMap的capacity为(10000+74)/75这样就可以保证不需要进行扩容，以免浪费性能。

## 3.put函数的实现

put函数的大致思路是：

1.根据KEY的hashcode计算hash，然后在打散，计算出index.

2.如果存在冲突，就把拼接成链表。

3.如果链表的长度超过限制，就转换成红黑树。

4.如果当前位置的key已经存在，就替换原来已经存在的oldValue。

5.如果需要进行扩容，就扩容。capacity变换为原来的二倍resize()

## 4.Get函数的实现

如果理解了put函数，get函数的实现就比较简单了。

1.根据KEY的hashcode进行计算hash，然后根据capacity计算出index。

2.如果找到bucket中相应的元素，并且进行key比较，是同一个key，则返回。

3.如果bucket含有多个元素，如果是TreeNode，则进行红黑树的遍历找到key，如果是链表，则遍历链表找到相应的key.

## 5.Hash函数的实现

从上图中可以看出，

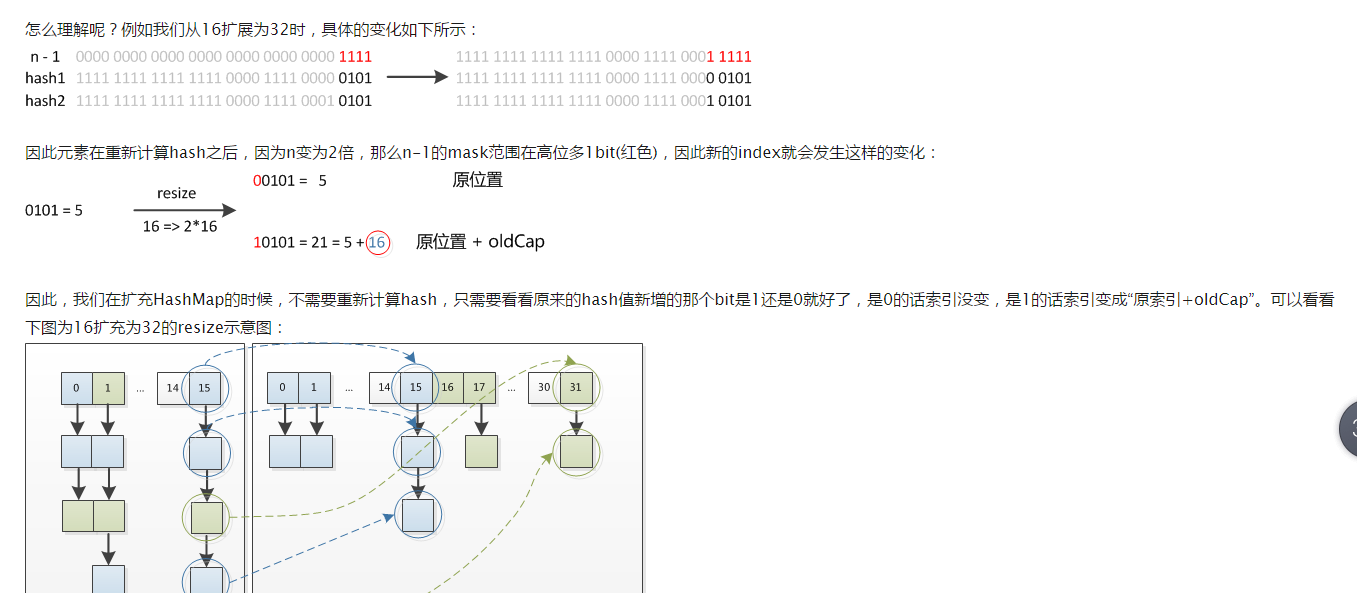
计算出key的hashCode,然后对hashCode和自身的高16bit进行异或处理，这样做的好处是保证高16bit中也参与到计算模打散的过程中。因为capacity是2的n次方，所以进行模n的时候，通过&（与）n-1即可求出模n 的余数。性能提升。

java8之前是利用链表来解决冲突的，java8之后，在链表长度达到8位的时候，转换成红黑树解决冲突，这样时间复杂度就变成了O(1)+O(log n) 相比之前的O(1)+O(n)

## 6.Resize的实现

源码对resize的描述是这样的：

Initializes or doubles table size. If null, allocates in accord with initial capacity target held in field threshold. Otherwise, because we are using power-of-two expansion, the elements from each bin must either **stay at same index**, or **move with a power of two offset** in the new table.

如上图所示的解释。

保证capacity在变成2倍的时候，每个节点根据自己存储的hash，然后判断是否需要移动位置即可。

## 7.总结

综上所述，hashMap主要的亮点如下：

1.resize的时候，扩容二倍，可以重新打散冲突的元素。

2.capaticy的大小为2的n次方。这样在求模取余的时候，可以用 (capacity-1)&hashcode代替hashcode%capacity

3.冲突的解决方式：java8之前的使用链表进行冲突解决，java8后开始使用链表进行解决，当链表超过8后，改成红黑树。