链滴

HashMap

作者：helly

原文链接：https://ld246.com/article/1510292704181

来源网站：链滴

java.util.HashMap

一、特点

1、key和value都允许为空，key只允许有一个为null。

2、无序（这个无序指的是遍历集合的时候取出元素的顺序基本不可能是put的顺序）。

3、线程不安全。

二、容量Capacity和负载因子Load Factor

1 capacity默认初始化容量为16。

2 当hashmap中桶被装满的数量大于容量乘以负载因子的时候会进行rehash。

三、put方法

1 对key的哈希值做hash，然后进行取余操作；

2 根据取余结果查找对应的桶，如果没碰撞直接插入；

3 如果碰撞，插入链表头部，当链表长度过长（默认是8），就把链表转换成红黑树；

4 如果已经存在key相同的节点，就替换；

5 如果桶被装满的数量大于容量乘以负载因子，那么就会进行rehash。

四、get方法

1 根据key的哈希值做hash，然后取余；

2 根据取余结果定位到具体的桶，然后通过equals方法逐个节点比较key是否相同直到找到节点或节不存在。

五、hash(Object key)方法

1 key为null，直接返回0；

2 根据Object.hashCode()方法获取key的hashcode；

3 然后这个hashcode的高16位不变，低16位和高16位做一个异或操作；（保证32位的hashcode都与了后面的取余操作，降低碰撞几率）

取余操作，不是通过取余符号%，而是通过按位与（&）运算。（位运算速度快）

六、rehash死循环问题（JDK1.8之前）

假设

oldTable[i]->node1->node2

rehash为：

newTable[j]->node2->node1

```
e
next = e.next

e.next = newTable(i);
newTable[i] = e;

e = next
```

线程一执行了

```
e //node1
next = e.next //node2
```

然后失去CPU。

线程二获得CPU并执行完rehash，此时

```
newTable[j]->node2->node1
```

线程一又获得CPU了，因为变量e和next都是局部变量，属于线程私有，所以此时

```
e //node1
next = e.next //node2
```

执行了一个节点的插入后产生死循环

```
node1.next = node2;
node2.next = node1;
```

这样next变量永远不可能为null，循环就不会停止。

七、JDK1.8 resize方法优化

每次扩容为之前的两倍，按位与的位数加1，加的这一位只能为0或1,0的话结果不变，1的话原位置+容量。（省去重新计算hash的时间）。

JDK1.8的链表元素不会倒置，因为设置了一个尾指针。

八、HashMap的其他线程不安全问题

1 两个线程同时往同一个桶插入节点时，并发情况下会产生覆盖。

参考：

HashMap源码

Java HashMap工作原理及实现

Java 8系列之重新认识HashMap