# HashMap

#### HashMap通过（hash值%hash表的长度）来确定entry数组的下标（注：（hash值%hash表的长度）==（hash值&（hash表长度-1）））

#### 在java1.8之后，HashMap引入了红黑树

新增TreeNode内部类（继承了LinkedHashMap中的entry类）

//一个桶内元素到达8后就会由链表转化为红黑树

**static final int *TREEIFY\_THRESHOLD*** = 8;

//一个桶内的元素少于6个时会有红黑树转化为链表结构  
**static final int *UNTREEIFY\_THRESHOLD*** = 6;

//只有当hash表的大小超过64后，桶内元素超过临界值才会进行树形转化，否则hash表会扩容，重新分配桶内元素。  
**static final int *MIN\_TREEIFY\_CAPACITY*** = 64;

树化过程：

1. 将Node单链表包装成TreeNode双向链表
2. 构造二叉树
3. 平衡二叉树变为红黑树（但这不意味这原先的单链表结构消失了）