HashTable

一、基本介绍

1）定义

哈希表（HashTable）又叫做散列表，是根据关键码值（即键值对）而直接访问的数据结构。

```
public class Hashtable<K,V>
    extends Dictionary<K,V>
    implements Map<K,V>, Cloneable, java.io.Serializable
```

2）主要属性

table：Entry[] 数组类型，实际上 Entry 是一个单向链表。每一个 Entry 都是一个键值对，哈希表的"key-value键值对"都是存储在 Entry 数组中的。
count：HashTable的大小，它是Hashtable保存的键值对的数量（不是HashTable 容器的大小，是所有 Entry 键值对的总数）。
threshold：HashTable 容量的阈值，用于判断是否需要调整Hashtable的容量。threshold的值=“容量*加载因子”。
loadFactor：加载因子，默认0.75f。
modCount：HashTable 被改变的次数，用来实现 fail-fast 机制。

3）构造方法

```
//默认构造函数，容量为11，加载因子为0.75。
public Hashtable(int initialCapacity, float loadFactor)
//用指定初始容量和默认的加载因子 (0.75) 构造一个新的空哈希表。
public Hashtable(int initialCapacity) {  this(initialCapacity, 0.75f); }
//用指定初始容量和指定加载因子构造一个新的空哈希表。
public Hashtable() {  this(11, 0.75f);}
//构造一个与给定的 Map 具有相同映射关系的新哈希表。
public Hashtable(Map<? extends K, ? extends V> t) {
    this(Math.max(2*t.size(), 11), 0.75f);
    putAll(t);
}
```

二、主要方法

1、put 方法

① 确保 value 值不为空。

② 计算key的hash值，直接使用 key.hashCode()，表明 key 不为空，否则也会 NPE。

③ 确认 key 在 table[] 中的索引位置，具体的算法是：key 的 hash 值去除最高为，然后和 table 的长度取余。

④ 链式迭代查找逻辑，可以看出采用的数据结构是（数组 + 链表）

⑤ 当前的容量超过阈值，进行扩容操作，扩容的大小为：newCapacity = (oldCapacity << 1) + 1，即容量扩大两倍+1

2、get 方法

相比于 put 方法，get 方法比较简单，计算 key 的 hash 值，然后判断在 table 中的索引位置，迭代链表后返回，具体的源码如下：

```
public synchronized V get(Object key) {
        Entry<?,?> tab[] = table;
        int hash = key.hashCode();
        int index = (hash & 0x7FFFFFFF) % tab.length;
        for (Entry<?,?> e = tab[index] ; e != null ; e = e.next) {
            if ((e.hash == hash) && e.key.equals(key)) {
                return (V)e.value;
            }
        }
        return null;
    }
```

三、底层实现

键值对数组