Collection集合对null元素的支持总结

# HashMap与Hashtable

注意：Hashtable中的t是小写字母，不是大写，不是HashTable。

HashMap支持null元素，键key和值value都支持。

Hashtable的key和value都不支持null元素。Neither the key nor the value can be <code>null</code>.

示例：

Map<Integer,String> map = **new** HashMap<>();

**map.put(null, null);**

System.***out***.println(map.get(**null**));//null

map.put(1, "aaa");

**map.put(null, "bbb");**

System.***out***.println(map.get(**null**));//bbb

//Hashtable

Hashtable<Integer,String> map2 = **new** Hashtable<>();

map2.put(0, **null**);// java.lang.NullPointerException

map2.put(**null**, "bbb");// java.lang.NullPointerException

## 源代码分析：

HashMap的put方法源代码：

/\*\*

\* Associates the specified value with the specified key in this map.

\* If the map previously contained a mapping for the key, the old

\* value is replaced.

\*

\* **@param** key key with which the specified value is to be associated

\* **@param** value value to be associated with the specified key

\* **@return** the previous value associated with <tt>key</tt>, or

\* <tt>null</tt> if there was no mapping for <tt>key</tt>.

\* **(A <tt>null</tt> return can also indicate that the map**

**\* previously associated <tt>null</tt> with <tt>key</tt>.)**

\*/

**public** V put(K key, V value) {

**return** **putVal(*hash*(key), key, value, false, true);**

}

/\*\*

\* Implements Map.put and related methods

\*

\* **@param** hash hash for key

\* **@param** key the key

\* **@param** value the value to put

\* **@param** onlyIfAbsent if true, don't change existing value

\* **@param** evict if false, the table is in creation mode.

\* **@return** previous value, or null if none

\*/

**final** V putVal(**int** hash, K key, V value, **boolean** onlyIfAbsent,

**boolean** evict) {

Node<K,V>[] tab; Node<K,V> p; **int** n, i;

**if** ((tab = table) == **null** || (n = tab.length) == 0)

n = (tab = resize()).length;

**if** ((p = tab[i = (n - 1) & hash]) == **null**)

tab[i] = newNode(hash, key, value, **null**);

**else** {

Node<K,V> e; K k;

**if** (p.hash == hash &&

((k = p.key) == key || (key != **null** && key.equals(k))))

e = p;

**else** **if** (p **instanceof** TreeNode)

e = ((TreeNode<K,V>)p).putTreeVal(**this**, tab, hash, key, value);

**else** {

**for** (**int** binCount = 0; ; ++binCount) {

**if** ((e = p.next) == **null**) {

p.next = newNode(hash, key, value, **null**);

**if** (binCount >= ***TREEIFY\_THRESHOLD*** - 1) // -1 for 1st

treeifyBin(tab, hash);

**break**;

}

**if** (e.hash == hash &&

((k = e.key) == key || (key != **null** && key.equals(k))))

**break**;

p = e;

}

}

**if** (e != **null**) { // existing mapping for key

V oldValue = e.value;

**if** (!onlyIfAbsent || oldValue == **null**)

e.value = value;

afterNodeAccess(e);

**return** oldValue;

}

}

++modCount;

**if** (++size > threshold)

resize();

afterNodeInsertion(evict);

**return** **null**;

}

**Hashtable的put方法的源代码**：

/\*\*

\* Maps the specified <code>key</code> to the specified

\* <code>value</code> in this hashtable. **Neither the key nor the**

**\* value can be <code>null</code>. <p>**

\* The value can be retrieved by calling the <code>get</code> method

\* with a key that is equal to the original key.

\*

\* @param key the hashtable key

\* @param value the value

\* @return the previous value of the specified key in this hashtable,

\* or <code>null</code> if it did not have one

\* @exception NullPointerException if the key or value is

\* <code>null</code>

\* @see Object#equals(Object)

\* @see #get(Object)

\*/

public **synchronized** V put(K key, V value) {

// Make sure the value is not null

if (value == null) {

throw new NullPointerException();

}

// Makes sure the key is not already in the hashtable.

Entry<?,?> tab[] = table;

int hash = key.hashCode();

int index = (hash & 0x7FFFFFFF) % tab.length;

@SuppressWarnings("unchecked")

Entry<K,V> entry = (Entry<K,V>)tab[index];

for(; entry != null ; entry = entry.next) {

if ((entry.hash == hash) && entry.key.equals(key)) {

V old = entry.value;

entry.value = value;

return old;

}

}

addEntry(hash, key, value, index);

return null;

}

# LinkedList

LinkedList支持**null元素**。

**LinkedList**<Integer> li = new LinkedList<Integer>();

li.add(null); li.add(3);

System.out.println(li.get(0));//null

System.out.println(li.get(1));//3

# Queue

一般不支持null元素，LinkedList也实现了Queue，但是LinkedList支持null元素。