1. HashMap的实现原理与细节

public V put(K key, V value) {

if (key == null) //###### 注意这里 2

return putForNullKey(value);

int hash = hash(key.hashCode());

int i = indexFor(hash, table.length);

for (Entry e = table[i]; e != null; e = e.next) {

Object k;

if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {

V oldValue = e.value;

e.value = value;

e.recordAccess(this);

return oldValue;

}

}

modCount++;

addEntry(hash, key, value, i); //###### 注意这里

return null;

}

从HashMap的put方法中可以看到，HashMap底层就是一个数组，然后是使用链表的方法来解决冲突。

Key的哈希值的获取，会先使用key.hashCode方法，再使用hash函数来得到在数组中定位得到index，当index有冲突的时候，使用链表的方式填写，如果是覆盖的情况，会覆盖掉原来的value并且返回被覆盖的旧的value的值。

从put方法中，我们也可以看到，HashMap是支持null作为key或者value的。

\*在JDK8中，解决冲突的链表变成了红黑树，当极端冲突的情况发生时，也能保证至少logn的查找速度

2. HashMap和HashTable的区别

总体来说，HashTable是HashMap的同步版本，但是除了同步与非同步的区别以外，这两者之间还有一些细节上的区别。

HashTable不允许以null作为key或者value，会抛出Exception，而HashMap可以

HashTable的hash算法是除数取余的方法，比较古老，而HashMap用的是位运算

HashTable的所有方法put，get，contain等都是同步的，效率非常之低

HashTable的初始大小是11，然后以2\*old + 1的方式增加；而HashMap的初始大小是16，然后每次以翻倍的方式扩容

3. ConcurrentHashMap