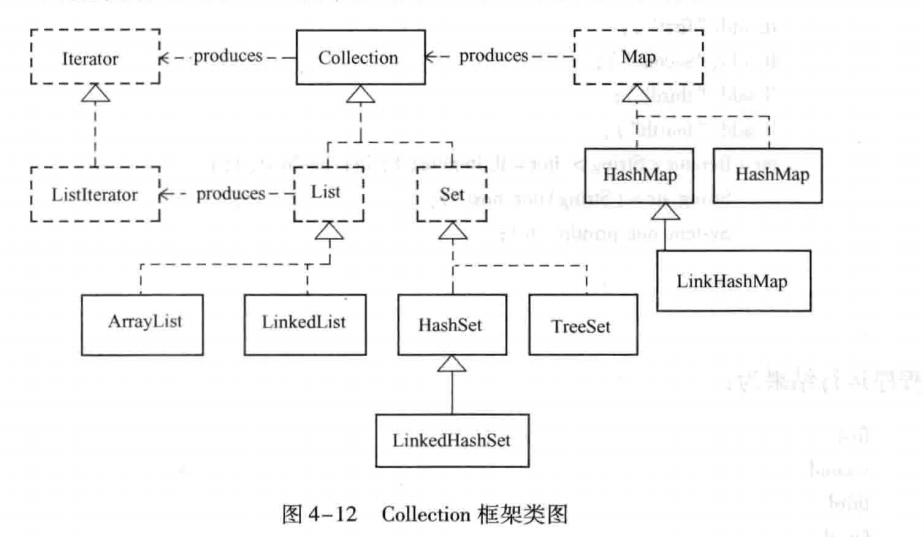
Java容器

一、Collection框架



Collection框架的基本方法有：

* boolean add(E element)
* Iterator<E> iterator()
* int size()
* boolean isEmpty()
* boolean contains(Object o)
* boolean remove(Object o)

二、迭代器

1. Iterator接口包含的方法

* E next()
* boolean hasNext()
* void remove()

2. 迭代器的遍历方法

Iterator<Integer> iter = collection.iterator();

while (iter.hasNext()) {

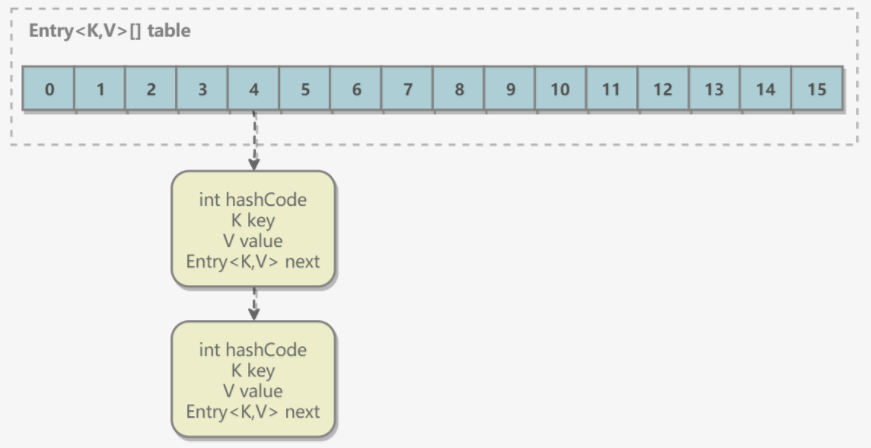
int val = iter.next();

}

三、Map

1. HashMap

在Java中，散列表使用链表数组实现，每个列表被称为桶。查找表中对象的位置时，会先计算它的散列码，然后与桶的总数取余，所得到的结果就是保存这个元素的桶的索引。HashMap不是线程安全的。



2. HashMap的插入，删除底层原理？

插入时，先计算对象所对应的hashCode，然后获取桶的下标，当该桶中已经有元素存在时，使用头插法将该元素插在链表的最前端。

从JDK1.8开始，一个桶插入的链表长度大于等于8时，会将链表转变成红黑树。当删除后链表的长度小于等于6时，会将红黑树转变成链表。

3. HashMap插入长度超过阈值后怎么办？

HashMap的默认大小tableSize为16，默认装载因子loadFactor为0.75，默认阈值threshold = 16 \* 0.75 = 12，当键值对的数量超过阈值时会触发扩容。

扩容时使用resize()实现，令容量扩大到原来的两倍，但是需要把oldTable的所有键值对重新插入newTable中。

4. 为什么HashMap的容量一定是2的幂次？

* 在寻找桶下标时，可以直接通过位运算hashCode & (tableSize - 1)来得到下标
* 在扩容后，原下标和新下标只有一位的差异，能够保持新老数组索引一致
* 会使得索引更加均匀：对于低位部分来说任何一位的变化都会对结果产生影响

5. HashMap，HashTable的区别？

* HashMap继承自AbstractMap类，而HashTable继承自Dictionary类
* HashTable是线程安全的，而HashMap不是线程安全的
* HashTable只有contains方法（等价于containsValue），而HashMap有containsKey和containsValue方法
* HashMap允许Key为空而HashTable不允许
* HashMap要重新计算key值而HashTable直接使用对象的hashCode
* HashTable默认大小是11，增加的方式是2 \* Size + 1；而HashMap默认大小为16，增加方式为加倍。

6. 为什么HashMap的key允许空值，而HashTable却不允许？

因为在HashTable中没有ContainsKey()方法，因为如果一个线程在调用ContainsKey()方法的时候，其他线程可能会把key给删除掉，导致不一致。

HashTable中没有ContainsKey()使得，如果通过map.get(key)得到了NULL值，则无法判断到底是因为没有这个key还是本身的value为NULL。

四、List与Set

1. HashSet和TreeSet的区别

* HashSet是基于散列表的集合；TreeSet是基于红黑树的集合，是一个有序集合
* HashSet允许一个值为NULL，而TreeSet不允许值为NULL
* HashSet要求放入的对象必须实现HashCode()方法，而TreeSet要求放入的对象必须实现CompareTo()方法
* 效率上来说TreeSet的查找效率低于HashSet

2. LinkedHashSet

* 使用双向链表维护插入的顺序
* 具有HashSet的查找效率

3. ArrayList和LinkedList的区别

* ArrayList：基于动态数组实现，支持随机访问，插入删除元素需要移动大量数据
* LinkedList：基于双向链表实现，只能顺序访问，但是可以快速插入和删除元素
* Vector：类似于ArrayList，但是是线程安全的

4. ArrayList的实现

* 初始化：数组的默认大小为10，支持通过initialCapacity指定初始值
* 扩容：当容量不够时，使用grow()方法进行扩容，新容量的大小为旧容量的1.5倍
* 删除：需要调用 System.arraycopy() 将 index+1 后面的元素都复制到 index 位置上
* 序列化：ArrayList 基于数组实现，并且具有动态扩容特性，因此保存元素的数组不一定都会被使用，那么就没必要全部进行序列化。

*ArrayList在序列化的时候会调用writeObject，直接将size和element写入ObjectOutputStream；反序列化时调用readObject，从ObjectInputStream获取size和element，再恢复到elementData。为什么不直接用elementData来序列化，而采用上诉的方式来实现序列化呢？原因在于elementData是一个缓存数组，它通常会预留一些容量，等容量不足时再扩充容量，那么有些空间可能就没有实际存储元素，采用上诉的方式来实现序列化时，就可以保证只序列化实际存储的那些元素，而不是整个数组，从而节省空间和时间。*

5. Vector的实现

* 同步：实现方法上使用了synchronized关键字进行同步
* 扩容：Vector 的构造函数可以传入 capacityIncrement 参数，它的作用是在扩容时使容量capacity增长capacityIncrement。如果这个参数的值小于等于0，扩容时每次都令capacity为原来的两倍。