目录

[1、 ArrayList 3](#_Toc12705)

[1.1、 空参构造ArrayList() 3](#_Toc16707)

[1.2、 构造函数ArrayList(Collection<? extends E> c) 3](#_Toc18103)

[1.3、 构造函数ArrayList(int) 3](#_Toc3007)

[1.4、 boolean add(E) 3](#_Toc9518)

[1.5、 Void add(int index, E element) 4](#_Toc9528)

[1.6、 boolean addAll(Collection<? extends E> c) 4](#_Toc28233)

[1.7、 boolean addAll(int index, Collection<? extends E> c) 4](#_Toc27898)

[1.8、 Void clear() 5](#_Toc746)

[1.9、 Object clone() 5](#_Toc27649)

[1.10、 boolean contains(Object o) 5](#_Toc19771)

[1.11、 void ensureCapacity(int minCapacity) 5](#_Toc15486)

[1.12、 E get(int index) 5](#_Toc15504)

[1.13、 int indexOf(Object o) 6](#_Toc32444)

[1.14、 Iterator<E> iterator() 6](#_Toc11587)

[1.15、int lastIndexOf(Object o) 7](#_Toc19961)

[1.16、ListIterator<E> listIterator() 7](#_Toc1189)

[1.17、ListIterator<E> listIterator(int) 8](#_Toc32041)

[1.18、E remove(int index) 8](#_Toc18740)

[1.19、boolean remove(Object o) 8](#_Toc25284)

[1.20、boolean removeAll(Collection<?> c) 8](#_Toc431)

[1.21、void sort(Comparator<? super E> c) 9](#_Toc29707)

[1.22、List<E> subList(int fromIndex, int toIndex) 9](#_Toc14799)

[1.23、Object[] toArray() 9](#_Toc13160)

[1.24、T[] toArray(T[] a) 9](#_Toc16802)

[2、 LinkedList 10](#_Toc21202)

[2.1、 构造函数LinkedList() 10](#_Toc9435)

[2.2、 构造函数LinkedList(Collection<? extends E> c) 10](#_Toc12898)

[2.3、 boolean add(E e) 10](#_Toc411)

[2.4、 void clear() 10](#_Toc25720)

[2.5、 Object clone() 10](#_Toc25208)

[2.6、 boolean contains(Object o) 11](#_Toc27683)

[2.7、 Iterator<E> descendingIterator() 11](#_Toc7632)

[3、 11](#_Toc20393)

[4、 11](#_Toc3248)

[5、 11](#_Toc10331)

[6、 HashMap 11](#_Toc15815)

[6.1、构造函数HashMap() 11](#_Toc25126)

[6.2、构造函数HashMap(int initialCapacity) 11](#_Toc11599)

[6.3、构造函数HashMap(int initialCapacity, float loadFactor) 12](#_Toc12407)

[6.3、构造函数HashMap(Map<? extends K, ? extends V> m) 12](#_Toc4295)

[6.4、V put(K key, V value) 12](#_Toc11115)

## ArrayList

### 空参构造ArrayList()

调用此构造函数后，将一个没有成员的数据赋值给成员变量elementData

### 构造函数ArrayList(Collection<? extends E> c)

先将参数集合转换为数组，如果数组的长度为0，则将EMPTY\_ELEMENTDATA赋值给elementData

### 构造函数ArrayList(int)

根据参数，创建指定容量的数组

### boolean add(E)

说明：

elementData -集合底层存储数据的数组

DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA-默认空的object对象数组

Size-集合中存储的数据的数量

modCound-集合结构被修改的次数（比如修改某一索引下的引用，就不包括在内）

newCapacity-elementData 数组扩充后的容量

方法作用：将指定的元素追加到此列表的末尾。

①首先，如果elementData == DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA，则取出 DEFAULT\_CAPACITY与size+1中较大的值，否则取出size+1；

②然后modCount自增长1；

③然后判断第①步中取出的值是否大于elementData 数组的长度，如果是，则进行 elementData 数组扩充

④elementData 数组的新容量newCapacity=老容量+老容量右移1位，并判断新容量是 否超出int类型最大值，如果是，则新容量等于int类型最大值；

⑤将老数组中的数据复制到新数组中，并将新数组赋值给elementData ；

⑥最后在elementData [size+1]处插入本次新增数据。

注意：

①通过无参构造创建的集合，在首次添加数据时，elementData 数组容量会设置为10

②当本次需要插入数据的索引值大于elementData 数组容量时，会对elementData 进行 扩容，每次的扩容量为elementData 容量的一半

③添加的对象可以为空，也可以重复添加同一对象

### Void add(int index, E element)

方法作用：在此列表中的指定位置插入指定的元素。

①检查插入的索引位置index是否在0到size之间，如果不是，则抛出索引越界异常；

②modCount自增长1，检查elementData 数组是否需要扩容，需要扩充则扩充；

③将elementData 数组index到size-1索引位置的对象复制到index+1到size位置上， 此操作是调用System.arraycopy()方法完成；

④最后在index插入本次要插入的对象，size自增1.

### boolean addAll(Collection<? extends E> c)

方法作用：按指定集合的Iterator返回的顺序将指定集合中的所有元素追加到此列表的 末尾。

①将传入的集合调用toArray()函数转化为数组a（如果传入的集合为null，这一步将抛 出空指针异常），并取得数组的长度numNew；

②modCount自增长1，检查elementData 数组是否需要扩容，需要扩充则扩充；

③然后调用System.arraycopy()方法将数组a中的对象储存到本集合的末尾；

④最后size增加numNew。

### boolean addAll(int index, Collection<? extends E> c)

方法作用：从指定的位置开始，将指定集合中的所有元素插入到此列表中。

①检查插入的索引位置index是否在0到size之间，如果不是，则抛出索引越界异常；

②将传入的集合调用toArray()方法转化为数组a（如果传入的集合为null，这一步将抛 出空指针异常），并取得数组的长度numNew；

③modCount自增长1，检查elementData 数组是否需要扩容，需要扩充则扩充；

④如果index正好等于size，则直接将数组a中的对象依次放到本集合的末尾；否则 先将elementData 数组从index索引到size-1处的对象向后移动numNew距离，再 将 数组a中的对象依次从elementData 数组index索引处开始存放；

⑤最后size增加numNew。

### Void clear()

方法做用：从列表中删除所有元素。

①modCount自增1；

②将elementData 数组中的所有索引出的引用置为null；

③size置为0；

### Object clone()

方法做用：返回此 ArrayList实例的浅拷贝（克隆出的集合中对象引用与原集合中的相 同）。

①调用Object.clone()方法克隆出新的集合v，再调用Arrays.copyOf()方法将原集合中的 对象复制到新集合中，并将新集合的modCount置为0，最后返回集合v。

### boolean contains(Object o)

方法做用：如果此列表包含指定的元素，则返回 true 。

1. 如果o为空，则遍历elementData 数组，依次判断数组中的对象是否为空，若为空， 则直接返回此索引值；如果o不为空，也遍历elementData 数组，调用o.equals()方法 依次与遍历出的对象比较，如果为true，则直接返回此索引值；如果未找到相等的对象， 返回-1；
2. 如果第①步中的返回值>=0，则返回true，否则返回false。

### void ensureCapacity(int minCapacity)

方法做用：如果需要，增加此 ArrayList实例的容量，以确保它可以至少保存minCapacity 个元素。

### E get(int index)

方法做用：返回此列表中指定位置的元素。

### int indexOf(Object o)

方法做用：返回此列表中指定元素的第一次出现的索引，如果此列表不包含元素，则返 回-1。

### Iterator<E> iterator()

方法做用：以正确的顺序返回该列表中的元素的迭代器。

迭代器中的成员变量：

①int cursor：下一个要返回元素的索引值，初始值为0

②int lastRet：最后一个已经返回元素的索引值。初始值为-1

③int expectedModCount：集合被修改的次数，初始值为modCound

迭代器中的方法：

①boolean hasNext()：集合中是否存在下一个元素

当cursor等于size时返回false，否则返回true

②E next()：返回集合中下一个元素

首先检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常；

判断cursor是否大于等于size，如果是直接抛出NoSuchElementException异常；

判断cursor是否大于等于elementData.lengh，如果是直接抛出 ConcurrentModificationException异常；

将cursor的值赋值给lastRet，然后cursor自增1；

最后返回elementData[lastRet]。

③void remove()：删除集合中最后被迭代器返回的元素

判断lastRet是否小于0，如果是，直接抛出IllegalStateException异常；

检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常；

调用此迭代器所属集合的remove(lastRet)删除集合中最后被迭代器返回的元素（注 意，remove(lastRet)方法中进行了modCound++）；

将lastRet的值赋值给cursor，将lastRet赋值为-1，并将最新的modCound值赋值 给expectedModCount。

④void forEachRemaining(Consumer<? super E> consumer)：对每个剩余元素执行给定的操 作，直到所有元素都被处理或动作引发异常。

判断消费者参数consumer是否为空，若是直接抛出NullPointerException异常；

判断此时迭代器中是否还有剩余的元素，及判断cursor是否大于等于size，是的话 直接返回；

将cursor赋值给i，判断cursor是否大于等于elementData.lengh，如果是直接抛出 ConcurrentModificationException异常；

进入while循环，条件是i != size && modCount == expectedModCount；循环体是 consumer对elementData[i]的操作，之后i++；

将i赋值给cursor，将i-1赋值给lastRet，及lastRet指向集合中最后一个元素的索 引；

最后检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常。

### 1.15、int lastIndexOf(Object o)

方法做用：返回此列表中指定元素的最后一次出现的索引，如果此列表不包含元素， 则返回-1。

### 1.16、ListIterator<E> listIterator()

方法做用：返回列表中的列表迭代器（按适当的顺序）。

迭代器中的继承父类的成员变量：

①int cursor：下一个要返回元素的索引值，初始值为0

②int lastRet：最后一个已经返回元素的索引值。初始值为-1

③int expectedModCount：集合被修改的次数，初始值为modCound

迭代器中的方法：

①boolean hasPrevious()：当前迭代器cursor是否指向0索引位置

当cursor等于0时返回false，否则返回true

②int nextIndex()：返回迭代器cursor的值，及下一个元素的索引值

③int previousIndex()：返回迭代器cursor-1的值

④E previous()：返回列表中的上一个元素，并向后移动光标位置。

检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常。

判断cursor-1是否小于0，如果是，则直接抛出NoSuchElementException异常；

判断cursor-1是否大于等于elementData.lengh，如果是直接抛出 ConcurrentModificationException异常；

将cursor的值减1，在将cursor的值赋值给lastRet，最后返回elementData[lastRet]。

⑤void set(E e)：用 指定的元素替换由 next()或 previous()返回的最后一个元素。

判断lastRet是否小于0，如果是，则直接抛出IllegalStateException异常；

检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常。

调用此迭代器所属的集合的set(lastRet, e)，将传入的e储存到lastRet索引位置。

⑥void add(E e)：将指定的元素插入到下一个元素所在的位置。

检查集合是否被除此迭代器以外的对象做了修改，及此迭代器维护的 expectedModCount是否还等于当前集合的modCound，如果不相等，直接抛出 ConcurrentModificationException异常。

调用此迭代器所属的集合的add(cursor, e)，将传入的e储存到cursor索引位置；

cursor自增1，lastRet赋值为-1，将modCound值赋值给expectedModCount。

### 1.17、ListIterator<E> listIterator(int)

方法做用：从列表中的指定位置开始，返回列表中的元素（按正确顺序）的列表迭代器。

### 1.18、E remove(int index)

方法做用：删除该列表中指定位置的元素。

①判断index >= size，如果是，则抛出IndexOutOfBoundsException异常；

②modCount++；

③调用System.arraycopy()方法将index索引之后的对象往前挪动一个索引的位置；

④将size的值减1，然后将elementData[size]置为空；

⑤最后将elementData[index]处的旧值返回。

### 1.19、boolean remove(Object o)

方法做用：从列表中删除指定元素的第一个出现（如果存在）。

### 1.20、boolean removeAll(Collection<?> c)

方法做用：从此列表中删除指定集合中包含的所有元素。

①判断传入的集合c是否为空。如果是，抛出NullPointerException异常；

②见原码。

### 1.21、void sort(Comparator<? super E> c)

方法做用：使用传入的比较器c对此列表进行重排序。

当集合中元素个数小于32个时，采用插入排序算法；

当集合中元素个数大于31个时，需要阅读java.util.TimSort的原码。

### 1.22、List<E> subList(int fromIndex, int toIndex)

方法做用：返回此列表中指定的 fromIndex （含）和 toIndex之间的视图。

注意：方法返回的List<E>并不是一个新创建的list，而是包含了原集合中fromIndex （含）和 toIndex索引之间的视图，对list的操作实际是对原集合的操作。

### 1.23、Object[] toArray()

方法做用：以正确的顺序（从第一个到最后一个元素）返回一个包含此列表中所有元素的数组。

注意：返回的数组与原集合中的elementData 数组的相对应的每个索引指向同一个引用。

### 1.24、T[] toArray(T[] a)

方法做用：以正确的顺序返回一个包含此列表中所有元素的数组（从第一个到最后一个元素）; 返回的数组的运行时类型是指定数组的运行时类型。

注意：当参数数组a的长度小于集合的长度，返回的数组为新new的数组，其中包含所有集合中的元素；否则，返回的还是数组a本身。

## LinkedList

### 构造函数LinkedList()

方法做用：构造一个空列表。

### 构造函数LinkedList(Collection<? extends E> c)

方法做用：按照传入集合c的迭代器返回的顺序，构造一个包含指定集合c所有元素的列表。

调用boolean addAll(Collection<? extends E> c)方法将传入的集合中的元素插入到本集合的链表中。

### boolean add(E e)

方法做用：将指定的元素追加到此列表的末尾。

### void clear()

方法做用：从列表中删除所有元素。

注意：此方法将集合中所有数据节点的值和引用的前后节点全部置为null，让所有节点之间断开关系，方便于垃圾回收。

### Object clone()

方法做用：返回此 LinkedList的浅版本。

源码中为什么要将克隆出来的集合的首尾节点置为空，再将被克隆集合的所有元素添加到克隆集合中，这不是多此一举吗？

### boolean contains(Object o)

方法做用：如果此列表包含指定的元素，则返回 true 。

此方法就是调用集合自身的int indexOf(Object o)方法，查询传入对象在集合中的索引值，只要索引值不等于-1，则返回true。

### Iterator<E> descendingIterator()

方法做用：以逆序返回此集合中的元素的迭代器。

## 3、

## 4、

## 5、

## HashMap

### 6.1、构造函数HashMap()

方法作用：将哈希表的加载因子loadFactor赋值为默认值0.75。

### 6.2、构造函数HashMap(int initialCapacity)

方法作用：将哈希表的加载因子loadFactor赋值为默认值0.75，并根据传入的initialCapacity计算出下次数组扩容时，元素数量需要达到的值threshold的值，计算的底层算法使threshold的值只能等于2整数次幂，切最大值为2的30次幂。

### 6.3、构造函数HashMap(int initialCapacity, float loadFactor)

方法作用：将哈希表的加载因子loadFactor赋值为传入的loadFactor，并根据传入的initialCapacity计算出要调整大小的下一个大小值threshold的值，计算的底层算法使threshold的值只能等于2整数次幂，切最大值为2的30次幂。

### 6.3、构造函数HashMap(Map<? extends K, ? extends V> m)

方法作用：将哈希表的加载因子loadFactor赋值为默认值0.75，其他内容暂留

### 6.4、V put(K key, V value)

方法作用：将指定的 key映射到此 key value中指定的value，及向集合中添加键值对。