ArrayList

java.lang.Object

继承者 java.util.AbstractCollection<E>

继承者 java.util.AbstractList<E>

继承者 java.util.ArrayList<E>

public class ArrayList<E> extends AbstractList<E>

implements List<E>, RandomAccess, Cloneable, java.io.Serializable

已实现的接口：

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

直接已知子类：

AttributeList, RoleList, RoleUnresolvedList

List 接口的大小可变数组的实现。实现了所有可选列表操作，并允许包括 null 在内的所有元素。除了实现 List 接口外，此类还提供一些方法来操作内部用来存储列表的数组的大小。（此类大致上等同于 Vector 类，除了此类是不同步的。）

size、isEmpty、get、set、iterator 和 listIterator 操作都以固定时间运行。add 操作以分摊的固定时间 运行，也就是说，添加 n 个元素需要 O(n) 时间。其他所有操作都以线性时间运行（大体上讲）。与用于 LinkedList 实现的常数因子相比，此实现的常数因子较低。

每个 ArrayList 实例都有一个容量。该容量是指用来存储列表元素的数组的大小。它总是至少等于列表的大小。随着向 ArrayList 中不断添加元素，其容量也自动增长。并未指定增长策略的细节，因为这不只是添加元素会带来分摊固定时间开销那样简单。

在添加大量元素前，应用程序可以使用 ensureCapacity 操作来增加 ArrayList 实例的容量。这可以减少递增式再分配的数量。

注意，此实现不是同步的。如果多个线程同时访问一个 ArrayList 实例，而其中至少一个线程从结构上修改了列表，那么它必须 保持外部同步。（结构上的修改是指任何添加或删除一个或多个元素的操作，或者显式调整底层数组的大小；仅仅设置元素的值不是结构上的修改。）这一般通过对自然封装该列表的对象进行同步操作来完成。如果不存在这样的对象，则应该使用 Collections.synchronizedList 方法将该列表“包装”起来。这最好在创建时完成，以防止意外对列表进行不同步的访问：

List list = Collections.synchronizedList(new ArrayList(...)); 此类的 iterator 和 listIterator 方法返回的迭代器是快速失败的：在创建迭代器之后，除非通过迭代器自身的 remove 或 add 方法从结构上对列表进行修改，否则在任何时间以任何方式对列表进行修改，迭代器都会抛出 ConcurrentModificationException。因此，面对并发的修改，迭代器很快就会完全失败，而不是冒着在将来某个不确定时间发生任意不确定行为的风险。

注意，迭代器的快速失败行为无法得到保证，因为一般来说，不可能对是否出现不同步并发修改做出任何硬性保证。快速失败迭代器会尽最大努力抛出 ConcurrentModificationException。因此，为提高这类迭代器的正确性而编写一个依赖于此异常的程序是错误的做法：迭代器的快速失败行为应该仅用于检测 bug。

此类是 Java Collections Framework 的成员。

构造方法摘要

ArrayList()

构造一个初始容量为 10 的空列表。

ArrayList(Collection<? extends E> c)

构造一个包含指定 collection 的元素的列表，这些元素是按照该 collection 的迭代器返回它们的顺序排列的。

ArrayList(int initialCapacity)

构造一个具有指定初始容量的空列表。

方法摘要

boolean add(E e)

将指定的元素添加到此列表的尾部。

void add(int index, E element)

将指定的元素插入此列表中的指定位置。

boolean addAll(Collection<? extends E> c)

按照指定 collection 的迭代器所返回的元素顺序，将该 collection 中的所有元素添加到此列表的尾部。

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

从指定的位置开始，将指定 collection 中的所有元素插入到此列表中。

void clear()

移除此列表中的所有元素。

Object clone()

返回此 ArrayList 实例的浅表副本。

boolean contains(Object o)

如果此列表中包含指定的元素，则返回 true。

void ensureCapacity(int minCapacity)

如有必要，增加此 ArrayList 实例的容量，以确保它至少能够容纳最小容量参数所指定的元素数。

E get(int index)

返回此列表中指定位置上的元素。

int indexOf(Object o)

返回此列表中首次出现的指定元素的索引，或如果此列表不包含元素，则返回 -1。

boolean isEmpty()

如果此列表中没有元素，则返回 true

int lastIndexOf(Object o)

返回此列表中最后一次出现的指定元素的索引，或如果此列表不包含索引，则返回 -1。

E remove(int index)

移除此列表中指定位置上的元素。

boolean remove(Object o)

移除此列表中首次出现的指定元素（如果存在）。

protected void removeRange(int fromIndex, int toIndex)

移除列表中索引在 fromIndex（包括）和 toIndex（不包括）之间的所有元素。

E set(int index, E element)

用指定的元素替代此列表中指定位置上的元素。

int size()

返回此列表中的元素数。

Object[] toArray()

按适当顺序（从第一个到最后一个元素）返回包含此列表中所有元素的数组。

<T> T[] toArray(T[] a)

按适当顺序（从第一个到最后一个元素）返回包含此列表中所有元素的数组；返回数组的运行时类型是指定数组的运行时类型。

void trimToSize()

将此 ArrayList 实例的容量调整为列表的当前大小。

从类 java.util.AbstractList 继承的方法

equals, hashCode, iterator, listIterator, listIterator, subList

从类 java.util.AbstractCollection 继承的方法

containsAll, removeAll, retainAll, toString

从类 java.lang.Object 继承的方法

finalize, getClass, notify, notifyAll, wait, wait, wait

从接口 java.util.List 继承的方法

containsAll, equals, hashCode, iterator, listIterator, listIterator, removeAll, retainAll, subList

java.lang.Object

继承者 java.util.AbstractCollection<E>

继承者 java.util.AbstractList<E>

继承者 java.util.AbstractSequentialList<E>

继承者 java.util.LinkedList<E>

public class LinkedList<E>

extends AbstractSequentialList<E>

implements List<E>, Deque<E>, Cloneable, java.io.Serializable

类型参数：

E - 在此 collection 中保持的元素的类型

所有已实现的接口：

Serializable, Cloneable, Iterable<E>, Collection<E>, Deque<E>, List<E>, Queue<E>

List 接口的链接列表实现。实现所有可选的列表操作，并且允许所有元素（包括 null）。除了实现 List 接口外，LinkedList 类还为在列表的开头及结尾 get、remove 和 insert 元素提供了统一的命名方法。这些操作允许将链接列表用作堆栈、队列或双端队列。

此类实现 Deque 接口，为 add、poll 提供先进先出队列操作，以及其他堆栈和双端队列操作。

所有操作都是按照双重链接列表的需要执行的。在列表中编索引的操作将从开头或结尾遍历列表（从靠近指定索引的一端）。

注意，此实现不是同步的。如果多个线程同时访问一个链接列表，而其中至少一个线程从结构上修改了该列表，则它必须 保持外部同步。（结构修改指添加或删除一个或多个元素的任何操作；仅设置元素的值不是结构修改。）这一般通过对自然封装该列表的对象进行同步操作来完成。如果不存在这样的对象，则应该使用 Collections.synchronizedList 方法来“包装”该列表。最好在创建时完成这一操作，以防止对列表进行意外的不同步访问，如下所示：

List list = Collections.synchronizedList(new LinkedList(...));此类的 iterator 和 listIterator 方法返回的迭代器是快速失败 的：在迭代器创建之后，如果从结构上对列表进行修改，除非通过迭代器自身的 remove 或 add 方法，其他任何时间任何方式的修改，迭代器都将抛出 ConcurrentModificationException。因此，面对并发的修改，迭代器很快就会完全失败，而不冒将来不确定的时间任意发生不确定行为的风险。

注意，迭代器的快速失败行为不能得到保证，一般来说，存在不同步的并发修改时，不可能作出任何硬性保证。快速失败迭代器尽最大努力抛出 ConcurrentModificationException。因此，编写依赖于此异常的程序的方式是错误的，正确做法是：迭代器的快速失败行为应该仅用于检测程序错误。

此类是 Java Collections Framework 的成员。

字段摘要

从类 java.util.AbstractList 继承的字段

modCount

构造方法摘要

LinkedList()

构造一个空列表。

LinkedList(Collection<? extends E> c)

构造一个包含指定 collection 中的元素的列表，这些元素按其 collection 的迭代器返回的顺序排列。

方法摘要

boolean add(E e)

将指定元素添加到此列表的结尾。

void add(int index, E element)

在此列表中指定的位置插入指定的元素。

boolean addAll(Collection<? extends E> c)

添加指定 collection 中的所有元素到此列表的结尾，顺序是指定 collection 的迭代器返回这些元素的顺序。

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

将指定 collection 中的所有元素从指定位置开始插入此列表。

void addFirst(E e)

将指定元素插入此列表的开头。

void addLast(E e)

将指定元素添加到此列表的结尾。

void clear()

从此列表中移除所有元素。

Object clone()

返回此 LinkedList 的浅表副本。

boolean contains(Object o)

如果此列表包含指定元素，则返回 true。

Iterator<E> descendingIterator()

返回以逆向顺序在此双端队列的元素上进行迭代的迭代器。

E element()

获取但不移除此列表的头（第一个元素）。

E get(int index)

返回此列表中指定位置处的元素。

E getFirst()

返回此列表的第一个元素。

E getLast()

返回此列表的最后一个元素。

int indexOf(Object o)

返回此列表中首次出现的指定元素的索引，如果此列表中不包含该元素，则返回 -1。

int lastIndexOf(Object o)

返回此列表中最后出现的指定元素的索引，如果此列表中不包含该元素，则返回 -1。

ListIterator<E> listIterator(int index)

返回此列表中的元素的列表迭代器（按适当顺序），从列表中指定位置开始。

boolean offer(E e)

将指定元素添加到此列表的末尾（最后一个元素）。

boolean offerFirst(E e)

在此列表的开头插入指定的元素。

boolean offerLast(E e)

在此列表末尾插入指定的元素。

E peek()

获取但不移除此列表的头（第一个元素）。

E peekFirst()

获取但不移除此列表的第一个元素；如果此列表为空，则返回 null。

E peekLast()

获取但不移除此列表的最后一个元素；如果此列表为空，则返回 null。

E poll()

获取并移除此列表的头（第一个元素）

E pollFirst()

获取并移除此列表的第一个元素；如果此列表为空，则返回 null。

E pollLast()

获取并移除此列表的最后一个元素；如果此列表为空，则返回 null。

E pop()

从此列表所表示的堆栈处弹出一个元素。

void push(E e)

将元素推入此列表所表示的堆栈。

E remove()

获取并移除此列表的头（第一个元素）。

E remove(int index)

移除此列表中指定位置处的元素。

boolean remove(Object o)

从此列表中移除首次出现的指定元素（如果存在）。

E removeFirst()

移除并返回此列表的第一个元素。

boolean removeFirstOccurrence(Object o)

从此列表中移除第一次出现的指定元素（从头部到尾部遍历列表时）。

E removeLast()

移除并返回此列表的最后一个元素。

boolean removeLastOccurrence(Object o)

从此列表中移除最后一次出现的指定元素（从头部到尾部遍历列表时）。

E set(int index, E element)

将此列表中指定位置的元素替换为指定的元素。

int size()

返回此列表的元素数。

Object[] toArray()

返回以适当顺序（从第一个元素到最后一个元素）包含此列表中所有元素的数组。

<T> T[]

toArray(T[] a)

返回以适当顺序（从第一个元素到最后一个元素）包含此列表中所有元素的数组；返回数组的运行时类型为指定数组的类型。

从类 java.util.AbstractSequentialList 继承的方法

iterator

从类 java.util.AbstractList 继承的方法

equals, hashCode, listIterator, removeRange, subList

从类 java.util.AbstractCollection 继承的方法

containsAll, isEmpty, removeAll, retainAll, toString

从类 java.lang.Object 继承的方法

finalize, getClass, notify, notifyAll, wait, wait, wait

从接口 java.util.List 继承的方法

containsAll, equals, hashCode, isEmpty, iterator, listIterator, removeAll, retainAll, subList

从接口 java.util.Deque 继承的方法

iterator