java.util

Class LinkedList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.AbstractSequentialList<E>
        java.util.LinkedList<E>

Type Parameters:

`E` - the type of elements held in this collection

All Implemented Interfaces:

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

```
public class LinkedList<E>
extends AbstractSequentialList<E>
implements List<E>, Deque<E>, Cloneable, Serializable
```

Doubly-linked list implementation of the `List` and `Deque` interfaces. Implements all optional list operations, and permits all elements (including `null`).

All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

Note that this implementation is not synchronized. If multiple threads access a linked list concurrently, and at least one of the threads modifies the list structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more elements; merely setting the value of an element is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the list. If no such object exists, the list should be "wrapped" using the `Collections.synchronizedList` method. This is best done at creation time, to prevent accidental unsynchronized access to the list:

```
List list = Collections.synchronizedList(new LinkedList(...));
```

The iterators returned by this class's `iterator` and `listIterator` methods are *fail-fast*: if the list is structurally modified at any time after the iterator is created, in any way except through the Iterator's own `remove` or `add` methods, the iterator will throw a `ConcurrentModificationException`. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw `ConcurrentModificationException` on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

This class is a member of the Java Collections Framework.

Since:

1.2

See Also:

`List`, `ArrayList`, Serialized Form

Field Summary

Fields inherited from class java.util.AbstractList

modCount

Constructor Summary

Constructors

Constructor and Description
LinkedList() Constructs an empty list.
LinkedList(**Collection**<? extends E> c) Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Method Summary

Methods

Modifier and Type	Method and Description
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**addAll**(**Collection**<? extends E> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
boolean	**addAll**(int index, **Collection**<? extends E> c) Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	**addFirst**(E e) Inserts the specified element at the beginning of this list.
void	**addLast**(E e) Appends the specified element to the end of this list.
void	**clear**() Removes all of the elements from this list.
Object	**clone**() Returns a shallow copy of this LinkedList.
boolean	**contains**(**Object** o) Returns true if this list contains the specified element.
Iterator<E>	**descendingIterator**() Returns an iterator over the elements in this deque in reverse sequential order.
E	**element**() Retrieves, but does not remove, the head (first element) of this list.
E	**get**(int index) Returns the element at the specified position in this list.
E	**getFirst**() Returns the first element in this list.
E	**getLast**()

	Returns the last element in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
ListIterator<E>	**listIterator**(int index) Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
boolean	**offer**(E e) Adds the specified element as the tail (last element) of this list.
boolean	**offerFirst**(E e) Inserts the specified element at the front of this list.
boolean	**offerLast**(E e) Inserts the specified element at the end of this list.
E	**peek**() Retrieves, but does not remove, the head (first element) of this list.
E	**peekFirst**() Retrieves, but does not remove, the first element of this list, or returns null if this list is empty.
E	**peekLast**() Retrieves, but does not remove, the last element of this list, or returns null if this list is empty.
E	**poll**() Retrieves and removes the head (first element) of this list.
E	**pollFirst**() Retrieves and removes the first element of this list, or returns null if this list is empty.
E	**pollLast**() Retrieves and removes the last element of this list, or returns null if this list is empty.
E	**pop**() Pops an element from the stack represented by this list.
void	**push**(E e) Pushes an element onto the stack represented by this list.
E	**remove**() Retrieves and removes the head (first element) of this list.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.
E	**removeFirst**() Removes and returns the first element from this list.
boolean	**removeFirstOccurrence**(Object o) Removes the first occurrence of the specified element in this list (when traversing the list from head to tail).
E	**removeLast**() Removes and returns the last element from this list.
boolean	**removeLastOccurrence**(Object o) Removes the last occurrence of the specified element in this list (when traversing the list from head to tail).
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element.

int	**size**() Returns the number of elements in this list.
Object[]	**toArray**() Returns an array containing all of the elements in this list in proper sequence (from first to last element).
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array.

Methods inherited from class java.util.AbstractSequentialList

iterator

Methods inherited from class java.util.AbstractList

equals, hashCode, listIterator, removeRange, subList

Methods inherited from class java.util.AbstractCollection

containsAll, isEmpty, removeAll, retainAll, toString

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.util.List

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

Methods inherited from interface java.util.Deque

iterator

Constructor Detail

LinkedList

```
public LinkedList()
```

Constructs an empty list.

LinkedList

```
public LinkedList(Collection<? extends E> c)
```

Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Parameters:

c - the collection whose elements are to be placed into this list