Class CircularList<E>

```
java.lang.Object
  java.util.AbstractCollection<E>
      java.util.AbstractList<E>
          java.util.AbstractSequentialList<E>
              CircularList<E>
```

All Implemented Interfaces: java.lang.Iterable<E>, java.util.Collection<E>, java.util.List<E>

```
public class CircularList<E>extends
java.util.AbstractSequentialList<E>
```

This class provides a circular doubly linked list data structure of CircularNodes with its corresponding methods. This class does not handle concurrent modifications. A doubly linked list means that each node has an object reference (a link aka pointer) to its previous and next node. A circular doubly linked list refers to the fact that the first node's previous node link points to the last element of the list and the last node's next link points to the first element of the list.

Author: Bryan Davis

Nested Class Summary	
class	**CircularList.CircularListIterator<E>** This class provides an iterator for this CircularList.

Field Summary	
private Circula	**firstNode** The first node of this list

rNode<E>	
private CircularNode<E>	**lastNode** The last node of this list
private int	**modCount** modCount is not used since this class does not handle concurrent modifications
private int	**size** The size of this list (number of nodes)

Constructor Summary

CircularList()
Construct a new CircularList of size 0.

Method Summary	
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
boolean	**addAll**(int index, java.util.Collection<? extends E> c) This method is not supported and must throw an UnsupportedOperationException if called.
E	**get**(int index) Returns the element at the specified position in this list.
CircularNode<E>	**getNode**(int index) Returns the circular node at the specified position in this list.
CircularList.CircularListIterat	**iterator**() Returns a circular list iterator over the elements in this list (in proper sequence).

or<E>	
CircularList.CircularListIterator<E>	**listIterator**(int index) Returns a circular list iterator over the elements in this list (in proper sequence) starting at index.
E	**remove**(int index) Removes the element at the specified position in this list.
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element.
int	**size**() Returns the size of this list.

Methods inherited from class java.util.AbstractList
add, clear, equals, hashCode, indexOf, lastIndexOf, listIterator, removeRange, subList

Methods inherited from class java.util.AbstractCollection
addAll, contains, containsAll, isEmpty, remove, removeAll, retainAll, toArray, toArray, toString

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.List
addAll, contains, containsAll, isEmpty, remove, removeAll, retainAll, toArray, toArray

Field Detail

size

```
private int size
```

The size of this list (number of nodes)

---

firstNode

```
private CircularNode<E> firstNode
```

The first node of this list

---

lastNode

```
private CircularNode<E> lastNode
```

The last node of this list

---

modCount

```
private int modCount
```

modCount is not used since this class does not handle concurrent modifications

Constructor Detail

CircularList

```
public CircularList()
```

Construct a new CircularList of size 0.

Method Detail

add

```
public void add(int index,
                E element)
```

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Specified by:add in interface java.util.List<E>**Overrides:**add in class java.util.AbstractSequentialList<E> **Parameters:**index - index at which the specified element is to be insertedelement - the element to insert **Throws:** java.lang.ClassCastException - if the class of the specified element prevents it from being added to this list java.lang.NullPointerException - if the specified element is null java.lang.IllegalArgumentException - if some property of the specified element prevents it from being added to this list java.lang.IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

---

addAll

```
public boolean addAll(int index,
                      java.util.Collection<? extends E> c)
```

This method is not supported and must throw an UnsupportedOperationException if called.

Specified by:addAll in interface java.util.List<E>**Overrides:**addAll in class java.util.AbstractSequentialList<E> **Throws:** java.lang.UnsupportedOperationException - Throw this error whenever called.

---

get

```
public E get(int index)
```

Returns the element at the specified position in this list.

Specified by:get in interface java.util.List<E>**Overrides:**get in class java.util.AbstractSequentialList<E> **Parameters:**index - index of element to return **Returns:**the element at the specific position in this list **Throws:** java.lang.IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

---

getNode

```
public CircularNode<E> getNode(int index)
```

Returns the circular node at the specified position in this list.

Parameters:index - index of node to return **Returns:**the node at the specific position in this list **Throws:** java.lang.IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

---

iterator

```
public CircularList.CircularListIterator<E> iterator()
```

Returns a circular list iterator over the elements in this list (in proper sequence). This implementation merely returns a list iterator over the list. The iterator returned should have nextNode that points to the first node of this list.

Specified by:iterator in interface java.lang.Iterable<E>**Specified by:**iterator in interface java.util.Collection<E>**Specified by:**iterator in interface java.util.List<E>**Overrides:**iterator in class java.util.AbstractSequentialList<E> **Returns:**a circular list iterator over the elements in this list.

---

listIterator

```
public CircularList.CircularListIterator<E> listIterator(int
index)
```

Returns a circular list iterator over the elements in this list (in proper sequence) starting at index. The iterator returned should have a nextNode pointing at the node at specified index of this list.

Specified by:listIterator in interface java.util.List<E>**Specified by:**listIterator in class java.util.AbstractSequentialList<E> **Parameters:**index - index of first element to be returned from the list iterator (by a call to the next method) **Returns:**a circular list iterator over the elements in this list starting at the node found at index.

---

remove

```
public E remove(int index)
```

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Specified by:remove in interface java.util.List<E>**Overrides:**remove in class java.util.AbstractSequentialList<E> **Parameters:**index - the index of the node to remove **Returns:**the element that was removed from the list **Throws:** java.lang.IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

---

set

```
public E set(int index,
             E element)
```

Replaces the element at the specified position in this list with the specified element.

Specified by:set in interface java.util.List<E>**Overrides:**set in class java.util.AbstractSequentialList<E> **Parameters:**index - index of the element to replaceelement - element to be stored at the specified position **Returns:**the element previously at the specified position **Throws:** java.lang.ClassCastException - if the class of the specified element prevents it from being added to this list java.lang.NullPointerException - if the specified element is null and this list does not permit null elements java.lang.IllegalArgumentException - if some property of the specified element prevents it from being added to this list java.lang.IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

---

size

```
public int size()
```

Returns the size of this list.

Specified by:size in interface java.util.Collection<E>**Specified by:**size in interface java.util.List<E>**Specified by:**size in class java.util.AbstractCollection<E>