Class DList<E>

java.lang.Object
    DList<E>

All Implemented Interfaces:

`java.lang.Iterable<E>`

```
public class DList<E>
extends java.lang.Object
implements java.lang.Iterable<E>
```

Constructor Summary

Constructors

Constructor and Description
DList() Creates an empty doubly-linked list
DList(java.util.Comparator comp)

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	**add**(E item) Add element to the end of the collection
void	**add**(int index, E item) Inserts the item at index position/ This method should behave the same way as the add(int index, E item) method in the ArrayList class including error handling.
boolean	**addAll**(int index, java.util.Collection<E> coll) Inserts all of the elements in the specified collection coll into this list at the specified index and returns true if index is valid.
void	**addInOrder**(E d)

	this method should only be called when current list is sorted insert d in the proper location.
void	**clear**() Removes all data (and associated nodes) from this list
boolean	**contains**(**E** target) Returns true if this list contains item - uses equals() for equality check
boolean	**equals**(java.lang.Object obj)
E	**get**(int index) If index is valid, returns the data at index.
DNode<**E**>	**getHead**() Get the first item in the list
int	**getLength**() Get the length of the list
DNode<**E**>	**getTail**() Get the last item in the list
int	**indexOf**(**E** target) Returns the index of the first occurrence of item.
boolean	**isEmpty**() Returns true if this list is empty, false otherwise
java.util.Iterator<**E**>	**iterator**()
int	**lastIndexOf**(**E** target) Returns the index of the last occurrence of item.
boolean	**remove**(**E** target) Remove the first occurrence of the target in this list and returns true Otherwise return false
E	**remove**(int index) If index is valid remove the element at index position and return the removed data Otherwise, throw IndexOutOfBoundsException
E	**set**(int index, **E** item) Replaces the data at index with item and returns the old (replaced) data if index if valid.
void	**setHead**(DNode<**E**> head) Set the value of the first item in the list
void	**setTail**(DNode<**E**> tail) Set the value of the last item in the list

int	**size**() Returns the number of data elements in this list
java.lang.String	**toString**()
java.lang.String	**toStringBwd**() Returns a print-friendly String representation of this list from back to front (reverse order)

Methods inherited from class java.lang.Object

getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.lang.Iterable

forEach, spliterator

Constructor Detail

DList

```
public DList()
```

Creates an empty doubly-linked list

DList

```
public DList(java.util.Comparator comp)
```

Method Detail

getHead

```
public DNode<E> getHead()
```

Get the first item in the list

Returns:

the first item of the list

setHead

```
public void setHead(DNode<E> head)
```

Set the value of the first item in the list

Parameters:

`head - a DNode`

getTail

```
public DNode<E> getTail()
```

Get the last item in the list

Returns:

`the last item of the list`

setTail

```
public void setTail(DNode<E> tail)
```

Set the value of the last item in the list

Parameters:

`tail - a DNode`

getLength

```
public int getLength()
```

Get the length of the list

Returns:

`the length of the list`

toString

```
public java.lang.String toString()
```

Overrides:

`toString in class java.lang.Object`

toStringBwd

```
public java.lang.String toStringBwd()
```

Returns a print-friendly String representation of this list from back to front (reverse order)

Returns:

a String representation of the reverse of the list

add

```
public void add(E item)
```

Add element to the end of the collection

Parameters:

item - data to add

add

```
public void add(int index,
                E item)
         throws java.lang.IndexOutOfBoundsException
```

Inserts the item at index position/ This method should behave the same way as the add(int index, E item) method in the ArrayList class including error handling.

Parameters:

index - position for new data

item - data to add

Throws:

java.lang.IndexOutOfBoundsException

addAll

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

Inserts all of the elements in the specified collection coll into this list at the specified index and returns true if index is valid. This method runs in O(M+N) where M is the size of coll and N is the size of this list. if index is not valid, returns false.

Parameters:

index - the position of the collection of data

coll - the collection of data to add

Returns:

whether the index is valid

clear

```
public void clear()
```

Removes all data (and associated nodes) from this list

get

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

If index is valid, returns the data at index. Otherwise, returns null. 0-based indexing.

Parameters:

`index - the position of the desired data`

Returns:

`the data at the given index`

set

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

Replaces the data at index with item and returns the old (replaced) data if index if valid. Otherwise, returns null. 0-based indexing.

Parameters:

`index - the position of of the target data`

`item - the new data`

Returns:

`the data being replaced`

contains

```
public boolean contains(E target)
```

Returns true if this list contains item - uses equals() for equality check

Parameters:

`target - the target data`

Returns:

`whether the list contains the item`

indexOf

```
public int indexOf(E target)
```

Returns the index of the first occurrence of item. Returns -1 if this list odes not contain item. DO Not use contains method

Parameters:

`target - the target data`

Returns:

`index the index of the target data`

lastIndexOf

```
public int lastIndexOf(E target)
```

Returns the index of the last occurrence of item. Returns -1 if this list does not contain item.

Parameters:

`target - the target data`

Returns:

`index the last index of the target data`

size

```
public int size()
```

Returns the number of data elements in this list

Returns:

`the length of the list`

isEmpty

```
public boolean isEmpty()
```

Returns true if this list is empty, false otherwise

Returns:

`whether the list is empty`

remove

```
public E remove(int index)
         throws java.lang.IndexOutOfBoundsException
```

If index is valid remove the element at index position and return the removed data Otherwise, throw IndexOutOfBoundsException

Parameters:

`index - position of the data removing`

Returns:

the value of the removed data

Throws:

java.lang.IndexOutOfBoundsException

remove

```
public boolean remove(E target)
```

Remove the first occurrence of the target in this list and returns true Otherwise return false

Parameters:

target - the value of the item removing

Returns:

true or false

equals

```
public boolean equals(java.lang.Object obj)
```

Overrides:

equals in class java.lang.Object

addInOrder

```
public void addInOrder(E d)
```

this method should only be called when current list is sorted insert d in the proper location.

Parameters:

d - the element to add

iterator

```
public java.util.Iterator<E> iterator()
```

Specified by:

iterator in interface java.lang.Iterable<E>