LinkedList in Java with Example

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

LinkedList is a doubly-linked list implementation of the List and Deque interfaces. LinkedList allows for constant-time insertions or removals using iterators, but only sequential access of elements. In other words, LinkedList can be searched forward and backward but the time it takes to traverse the list is directly proportional to the size of the list.

> At the end of this guide, I have linked all the tutorials that I have shared on LinkedList, refer them in the given order to understand this topic fully.

Example of LinkedList in Java

```
import java.util.*;
public class LinkedListExample {
     public static void main(String args[]) {

       /* Linked List Declaration */
       LinkedList<String> linkedlist = new LinkedList<String>();

       /*add(String Element) is used for adding
        * the elements to the linked list*/
       linkedlist.add("Item1");
       linkedlist.add("Item5");
       linkedlist.add("Item3");
       linkedlist.add("Item6");
       linkedlist.add("Item2");

       /*Display Linked List Content*/
       System.out.println("Linked List Content: " +linkedlist);

       /*Add First and Last Element*/
       linkedlist.addFirst("First Item");
       linkedlist.addLast("Last Item");
       System.out.println("LinkedList Content after addition: " +linkedlist);

       /*This is how to get and set Values*/
       Object firstvar = linkedlist.get(0);
       System.out.println("First element: " +firstvar);
       linkedlist.set(0, "Changed first item");
       Object firstvar2 = linkedlist.get(0);
       System.out.println("First element after update by set method: " +firstvar2);

       /*Remove first and last element*/
       linkedlist.removeFirst();
       linkedlist.removeLast();
       System.out.println("LinkedList after deletion of first and last element: " +linkedlist);

       /* Add to a Position and remove from a position*/
       linkedlist.add(0, "Newly added item");
       linkedlist.remove(2);
       System.out.println("Final Content: " +linkedlist);
     }
}
```

Output:

```
Linked List Content: [Item1, Item5, Item3, Item6, Item2]
LinkedList Content after addition: [First Item, Item1, Item5, Item3, Item6, Item2, Last Item]
First element: First Item
First element after update by set method: Changed first item
```

```
LinkedList after deletion of first and last element: [Item1, Item5, Item3, Item6, Item2]
Final Content: [Newly added item, Item1, Item3, Item6, Item2]
```

Methods of LinkedList class:

For all the examples in the below methods, consider llistobj as a reference for LinkedList<String>.

LinkedList<String> llistobj = new LinkedList<String>();

1) **boolean add(Object item)**: It adds the item at the end of the list.

```
llistobj.add("Hello");
```

It would add the string "Hello" at the end of the linked list.

2) **void add(int index, Object item)**: It adds an item at the given index of the the list.

```
llistobj.add(2, "bye");
```

This will add the string "bye" at the 3rd position(2 index is 3rd position as index starts with 0).

3) **boolean addAll(Collection c)**: It adds all the elements of the specified collection c to the list. It throws NullPointerException if the specified collection is null. Consider the below example –

```
LinkedList<String> llistobj = new LinkedList<String>();
ArrayList<String> arraylist= new ArrayList<String>();
arraylist.add("String1");
arraylist.add("String2");
llistobj.addAll(arraylist);
```

This piece of code would add all the elements of ArrayList to the LinkedList.

4) **boolean addAll(int index, Collection c)**: It adds all the elements of collection c to the list starting from a give index in the list. It throws NullPointerException if the collection c is null and IndexOutOfBoundsException when the specified index is out of the range.

```
llistobj.add(5, arraylist);
```

It would add all the elements of the ArrayList to the LinkedList starting from position 6 (index 5).

5) **void addFirst(Object item)**: It adds the item (or element) at the first position in the list.

```
llistobj.addFirst("text");
```

It would add the string "text" at the beginning of the list.

6) **void addLast(Object item)**: It inserts the specified item at the end of the list.

```
llistobj.addLast("Chaitanya");
```

This statement will add a string "Chaitanya" at the end position of the linked list.

7) **void clear()**: It removes all the elements of a list.

```
llistobj.clear();
```

8) **Object clone()**: It returns the copy of the list.

For e.g. My linkedList has four items: text1, text2, text3 and text4.

```
Object str= llistobj.clone();
 System.out.println(str);
```

Output: The output of above code would be:

[text1, text2, text3, text4]

9) **boolean contains(Object item)**: It checks whether the given item is present in the list or not. If the item is present then it returns true else false.

```
boolean var = llistobj.contains("TestString");
```

It will check whether the string "TestString" exist in the list or not.

10) **Object get(int index)**: It returns the item of the specified index from the list.

```
Object var = llistobj.get(2);
```

It will fetch the 3rd item from the list.

11) **Object getFirst()**: It fetches the first item from the list.

```
Object var = llistobj.getFirst();
```

12) **Object getLast()**: It fetches the last item from the list.

```
Object var= llistobj.getLast();
```

13) **int indexOf(Object item)**: It returns the index of the specified item.

```
llistobj.indexOf("bye");
```

14) **int lastIndexOf(Object item)**: It returns the index of last occurrence of the specified element.

```
int pos = llistobj.lastIndexOf("hello);
```

integer variable pos will be having the index of last occurrence of string "hello".

15) **Object poll()**: It returns and removes the first item of the list.

```
Object o = llistobj.poll();
```

16) **Object pollFirst()**: same as poll() method. Removes the first item of the list.

```
Object o = llistobj.pollFirst();
```

17) **Object pollLast()**: It returns and removes the last element of the list.

```
Object o = llistobj.pollLast();
```

18) **Object remove()**: It removes the first element of the list.

```
llistobj.remove();
```

19) **Object remove(int index)**: It removes the item from the list which is present at the specified index.

```
llistobj.remove(4);
```

It will remove the 5th element from the list.

20) **Object remove(Object obj)**: It removes the specified object from the list.

```
llistobj.remove("Test Item");
```

21) **Object removeFirst()**: It removes the first item from the list.

```
llistobj.removeFirst();
```

22) **Object removeLast()**: It removes the last item of the list.

```
llistobj.removeLast();
```

23) **Object removeFirstOccurrence(Object item)**: It removes the first occurrence of the specified item.

```
llistobj.removeFirstOccurrence("text");
```

It will remove the first occurrence of the string “text” from the list.

24) **Object removeLastOccurrence(Object item)**: It removes the last occurrence of the given element.

```
llistobj.removeLastOccurrence("String1);
```

It will remove the last occurrence of string “String1”.

25) **Object set(int index, Object item)**: It updates the item of specified index with the give value.

```
llistobj.set(2, "Test");
```

It will update the 3rd element with the string “Test”.

26) **int size()**: It returns the number of elements of the list.

```
llistobj.size();
```

LinkedList Tutorials

Here are the tutorials that I have shared on LinkedList.

LinkedList Basics

- **How to iterate LinkedList**

Add/Remove

- **Adding an element to LinkedList**
- **Add element at specific index in LinkedList**
- **Add element at the beginning and end of LinkedList**
- **Adding an element to the front of LinkedList**
- **Remove First and last elements from LinkedList**
- **Remove element from specific index**
- **Remove specified element from LinkedList**
- **Remove All elements from LinkedList**
- **Append all the elements of a List to LinkedList**

Get/Search

- **Get first and last elements from LinkedList**
- **Get element from specific index of LinkedList**
- **Search element in LinkedList**
- **Get Sub list of LinkedList**

Iterator/ListIterator

- **LinkedList Iterator example**
- **LinkedList ListIterator example**
- **Iterate a LinkedList in reverse Order**

Other Tutorials

- **Replace element with a new value in LinkedList**
- **Check whether a particular element exists in LinkedList**
- **Clone a LinkedList to another LinkedList**
- **Get the index of last occurrence of an element in LinkedList**

- LinkedList push() and pop() methods
- LinkedList poll(), pollFirst() and pollLast() methods
- LinkedList peek(), peekFirst() and peekLast() methods

Conversion

- Convert LinkedList to ArrayList
- Convert LinkedList to Array

Differences

- LinkedList vs ArrayList

Reference

- LinkedList Documentation

Comments

user says
JULY 5, 2016 AT 2:14 PM

Exception in thread "main" java.lang.Error: Unresolved compilation problems:
The type LinkedList is not generic; it cannot be parameterized with arguments
The type LinkedList is not generic; it cannot be parameterized with arguments

Reply

April Randolph says
JANUARY 3, 2017 AT 7:42 PM

I'm compiling receiving the same message.

Reply

Brutus says
MARCH 1, 2017 AT 11:08 AM

It works fine for me, never just copy/paste everything, do it yourself and in an IDE and you will find no problem at all.
on the code there are some symbols forgotten like " which are essential for strings. and you might do it with a class instead, for example create a class Person and create the linkedlist.

btw Chaitanya keep on the good work

Reply

Arun says

JUNE 23, 2017 AT 11:05 PM

When to go for an arraylist and when to go for LinkedList ? Can anyone expain

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

POST COMMENT

Java Tutorial

- Java Tutorial
- OOPs Concepts

Java Collections

- ArrayList
- LinkedList
- ArrayList vs LinkedList
- Vector
- ArrayList vs Vector
- HashMap
- TreeMap
- LinkedHashMap
- HashSet
- TreeSet
- LinkedHashSet
- Hashtable