GeeksforGeeks Courses
A computer science portal for geeks

Hire with us! Login

Randomly select items from a List in Java

Get first and last elements from ArrayList in Java

Split a list into two halves in Java

How to Remove Duplicates from ArrayList in Java

How to get ArrayList from Stream in Java 8

How to convert LinkedList to Array in Java?

How to make an ArrayList read only in Java

Join two ArrayLists in Java

Find common elements in two ArrayLists in

Java

Find first and last element of ArrayList in java

Convert an Iterator to a List in Java

LinkedList in Java

Linked List are linear data structures where the elements are not stored in contiguous locations and every element is a separate object with a data part and address part. The elements are linked using pointers and addresses. Each element is known as a node. Due to the dynamicity and ease of insertions and deletions, they are preferred over the arrays. It also has few disadvantages like the nodes cannot be accessed directly instead we need to start from the head and follow through the link to reach to a node we wish to access.
To store the elements in a linked list we use a doubly linked list which provides a linear data structure and also used to inherit an abstract class and implement list and deque interfaces.

In Java, LinkedList class implements the list interface. The LinkedList class also consists of various constructors and methods like other java collections.

Constructors for Java LinkedList:

1. LinkedList(): Used to create an empty linked list.
2. LinkedList(Collection C): Used to create a ordered list which contains all the elements of a specified collection, as returned by the collection’s iterator.

```java
// Java code for Linked List implementation

import java.util.*;

public class Test
{
    public static void main(String args[])
    {
        // Creating object of class linked list
        LinkedList<String> object = new LinkedList<String>();

        // Adding elements to the linked list
        object.add("A");
        object.add("B");
        object.addLast("C");
        object.addFirst("D");
        object.add(2, "E");
        object.add("F");
        object.add("G");
        System.out.println("Linked list : " + object);

        // Removing elements from the linked list
        object.remove("B");
        object.remove(3);
        object.removeFirst();
        object.removeLast();
        System.out.println("Linked list after deletion: " + object);

        // Finding elements in the linked list
        boolean status = object.contains("E");

        if(status)
            System.out.println("List contains the element 'E' ");
        else
            System.out.println("List doesn't contain the element 'E'");

        // Number of elements in the linked list
        int size = object.size();
        System.out.println("Size of linked list = " + size);
```

```
            // Get and set elements from linked list
            Object element = object.get(2);
            System.out.println("Element returned by get() : " + element);
            object.set(2, "Y");
            System.out.println("Linked list after change : " + object);
        }
    }
```

Output:

```
Linked list : [D, A, E, B, C, F, G]
Linked list after deletion: [A, E, F]
List contains the element 'E'
Size of linked list = 3
Element returned by get() : F
Linked list after change : [A, E, Y]
```

Methods for Java LinkedList:

1. **add(int index, E element):** This method Inserts the specified element at the specified position in this list.
2. **add(E e):** This method Appends the specified element to the end of this list.
3. **addAll(int index, Collection c):** This method Inserts all of the elements in the specified collection into this list, starting at the specified position.
4. **addAll(Collection c):** This method 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.
5. **addFirst(E e):** This method Inserts the specified element at the beginning of this list.
6. **addLast(E e):** This method Appends the specified element to the end of this list.
7. **clear():** This method removes all of the elements from this list.
8. **clone():** This method returns a shallow copy of this LinkedList.
9. **contains(Object o):** This method returns true if this list contains the specified element.
10. **descendingIterator():** This method returns an iterator over the elements in this deque in reverse sequential order.
11. **element():** This method retrieves, but does not remove, the head (first element) of this list.
12. **get(int index):** This method returns the element at the specified position in this list.

13. **getFirst():** This method returns the first element in this list.
14. **getLast():** This method returns the last element in this list.
15. **indexOf(Object o):** This method returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
16. **lastIndexOf(Object o):** This method returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
17. **listIterator(int index):** This method returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list.
18. **offer(E e):** This method Adds the specified element as the tail (last element) of this list.
19. **offerFirst(E e):** This method Inserts the specified element at the front of this list.
20. **offerLast(E e):** This method Inserts the specified element at the end of this list.
21. **peek():** This method retrieves, but does not remove, the head (first element) of this list.
22. **peekFirst():** This method retrieves, but does not remove, the first element of this list, or returns null if this list is empty.
23. **peekLast():** This method retrieves, but does not remove, the last element of this list, or returns null if this list is empty.
24. **poll():** This method retrieves and removes the head (first element) of this list.
25. **pollFirst():** This method retrieves and removes the first element of this list, or returns null if this list is empty.
26. **pollLast():** This method retrieves and removes the last element of this list, or returns null if this list is empty.
27. **pop():** This method Pops an element from the stack represented by this list.
28. **push(E e):** This method Pushes an element onto the stack represented by this list.
29. **remove():** This method retrieves and removes the head (first element) of this list.
30. **remove(int index):** This method removes the element at the specified position in this list.
31. **remove(Object o):** This method removes the first occurrence of the specified element from this list, if it is present.
32. **removeFirst():** This method removes and returns the first element from this list.
33. **removeFirstOccurrence(Object o):** This method removes the first occurrence of the specified element in this list (when traversing the list from head to tail).
34. **removeLast():** This method removes and returns the last element from this list.
35. **removeLastOccurrence(Object o):** This method removes the last occurrence of the specified element in this list (when traversing the list from head to tail).
36. **set(int index, E element):** This method replaces the element at the specified position in this list with the specified element.

37. **size():** This method returns the number of elements in this list.
38. **spliterator():** This method Creates a late-binding and fail-fast Spliterator over the elements in this list.
39. **toArray():** This method returns an array containing all of the elements in this list in proper sequence (from first to last element).
40. **toArray(T[] a):** This method 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.

This article is contributed by Mehak Narang.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recommended Posts:

Java.util.LinkedList.poll(), pollFirst(), pollLast() with examples in Java

Java.util.LinkedList.peek() , peekfirst(), peeklast() in Java

Java.util.LinkedList.offer(), offerFirst(), offerLast() in Java

Java.util.LinkedList.indexOf(), lastIndexof() in Java

Java.util.LinkedList.get(), getFirst(), getLast() in Java

LinkedList set() Method in Java

LinkedList get() Method in Java

LinkedList pop() Method in Java

LinkedList add() Method in Java

LinkedList contains() Method in Java

Reverse a LinkedList in Java

ArrayList vs LinkedList in Java

LinkedList toArray() method in Java with Example

LinkedList removeLastOccurrence() method in Java with Example

LinkedList addLast() Method in Java

Improved By : Chinmoy Lenka

Article Tags : Java Linked List Java - util package Java-Collections java-LinkedList java-list

Practice Tags : Linked List Java Java-Collections

36

1.7

☐ To-do ☐ Done

Based on **59** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments