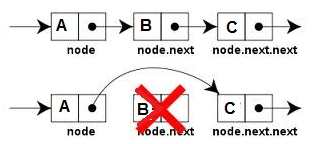
**Java LinkedList Class**

**Introduction**

A LinkedList is ordered by index position, like ArrayList, except that the elements are doubly-linked to one another. This linkage gives you new methods (beyond what you get from the List interface) for adding and removing from the beginning or end, which makes it an easy choice for implementing a stack or queue. Linked list has concept of nodes and data. Here Node is storing values of next node while data stores the value it is holding. Below diagram shows how LinkedList storing values. There are three elements in LinkedList A, B and C. We are removing element B from middle of the LinkedList which will just change node value of element A’s node to point to node C.



Keep in mind that a LinkedList may iterate more slowly than an ArrayList, but it's a good choice when you need fast insertion and deletion.

LinkedList has the two constructors shown here:

iLinkedList( )

LinkedList(Collection<? extends E> c)

The first constructor builds an empty linked list. The second constructor builds a linked list that is initialized with the elements of the collection c.

Important methods of LinkedList class:

|  |  |
| --- | --- |
| **Method** | **Description** |
| addFirst() or offerFirst( ) | To add elements to the start of a list |
| addLast( ) or offerLast( ) or add() | To add elements to the end of the list |
| getFirst( ) or peekFirst( ) | To obtain the first element of the list |
| getLast( ) or peekLast( ) | To obtain the last element of the list |
| removeFirst( ) or pollFirst( ) or remove() | To remove the first element of the list |
| removeLast( ) or pollLast( ) | To remove the last element of the list |
|  |  |

Java Program to demonstrate use of all above methods described above. Here we are creating LinkedList named myLinkedList and adding objects using add(), addFirst() and addLast() methods as well as using index based add() method, then printing all the objects. Then modifying the list using remove(), removeLast() and remove(Object o) methods. Then we demonstrate use of getFirst() and getLast() methods. Output of program is shown below the java code.

**Java Code**

[view plaincopy to clipboardprint?](http://www.w3resource.com/java-tutorial/java-linkedlist.php)

1. **package** linkedlist;

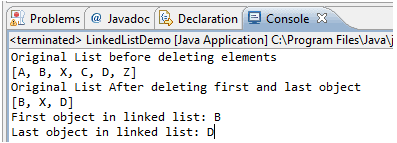

5. **import** java.util.LinkedList;


9. **public** **class** LinkedListDemo {


13. **public** **static** **void** main(String[] args) {
15. LinkedList<String> myLinkedList = **new** LinkedList<String>();
17. myLinkedList.addFirst("A");
19. myLinkedList.add("B");
21. myLinkedList.add("C");
23. myLinkedList.add("D");
25. myLinkedList.add(2, "X");//This will add C at index 2
27. myLinkedList.addLast("Z");
29. System.out.println("Original List before deleting elements");


33. System.out.println(myLinkedList);
35. myLinkedList.remove();
37. myLinkedList.removeLast();
39. myLinkedList.remove("C");
41. System.out.println("Original List After deleting first and last object");
43. System.out.println(myLinkedList);
45. System.out.println("First object in linked list: "+ myLinkedList.getFirst());
47. System.out.println("Last object in linked list: "+ myLinkedList.peekLast());
49. }
51. }

**Output**



**Summary**

* LinkedList is good for adding elements to the ends, i.e., stacks and queues
* LinkedList perform best while removing objects from middle of the list
* LinkedList implements List, Deque and Queue interfaces so LinkedList can be used to implement queues and stacks.