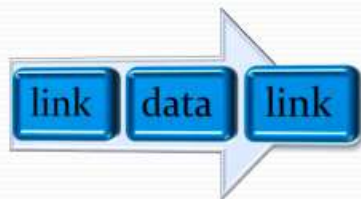
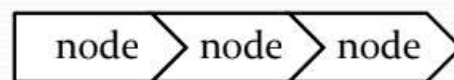


LinkedList Class:

- LinkedList is available since jdk1.2V.
- It allows duplicates, null & insertion order is maintained.
- Default capacity when creating an LinkedList is 0.
- In linked list elements are stored in the form of nodes.
- Each node will have three fields, the data field contains **data** and the link fields contain **references to previous and next nodes**.
- It occupies more memory than ArrayList and Construction time is also high. **Syntax:** `LinkedList ll=new LinkedList();`



Node Structure



Linked List

LinkedList Methods

Methods	Description
Object getFirst();	Returns the first element in this list
Object getLast();	Returns the last element in this list
Object removeFirst();	Removes and returns the first element from this list
Object removeLast();	Removes and returns the last element from this list.
void addFirst(Element e);	Inserts the specified element at the beginning of this list.
addLast(Element e);	Appends the specified element to the end of this list.

```

5 public class ClassA
6 {
7     void meth1()
8     {
9         System.out.println("Implementing LinkedList\n");
10
11         LinkedList<Object> ll=new LinkedList<Object>();
12
13         ll.add(10); // Insertion order is
14         ll.add("Java");// Heterogeneous data is
15         ll.add(null);// null value is
16         ll.add(10);// Duplicates are Allowed
17         ll.add('A'); // It is available from Java 1.2v
18         ll.add(false);// Its default capacity is 0 // In linkedlist elements will be stored in the form of NODES
19         ll.add(99);// Its size increases by DOUBLE
20         ll.add(1); // It is NOT synchronized
21
22         System.out.println(ll);
23     }
24     public static void main(String[] args)
25     {
26         ClassA aobj=new ClassA();
27         aobj.meth1();
28     }

```

```

32
33     System.out.println("\nReteriving the data by using Iterator
34     // write the logic
35
36     System.out.println("\nReteriving the data by using ListIterator
37     ListIterator<Object> li=ll.listIterator();
38     while(li.hasNext())
39     {
40         System.out.print(li.next()+" ");
41     }
42
43     System.out.println();
44
45     while(li.hasPrevious())
46     {
47         System.out.print(li.previous()+" ");
48     }
49 }
50
51 public static void main(String[] args)
52 {
53     ClassA aobj=new ClassA();
54     aobj.meth1();
55 }
56

```

Implementing LinkedList

[10, Java, null, 10, A, false, 99, 1]
size() : 8
get() : false

Reteriving the data in BOTH the directions by using for loop

Reteriving the data by using for-each loop

Reteriving the data by using Iterator

Reteriving the data by using ListIterator
10 Java null 10 A false 99 1
1 99 false A 10 null Java 10

After the end of while loop over cursor is after 1 in above case

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
System.out.println("\nReteriving the data by using Iterato
// write the logic

System.out.println("\nReteriving the data by using ListIt
ListIterator<Object> li=ll.listIterator();
/* while(li.hasNext())
{
    System.out.print(li.next()+" ");
}
System.out.println();
*/
while(li.hasPrevious()) false
{
    System.out.print(li.previous()+" ");
}
}
public static void main(String[] args)
{
    ClassA aobj=new ClassA();
    aobj.meth1();
}
```

Implementing LinkedList
[10, Java, null, 10, A, false, 99, 1]
size() : 8
get() : false
Reteriving the data in BOTH the directions by using for l
Reteriving the data by using for-each loop
Reteriving the data by using Iterator
Reteriving the data by using ListIterator

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
System.out.println("\nReteriving the data by using Iterato
// write the logic

System.out.println("\nReteriving the data by using ListIt
ListIterator<Object> li=ll.listIterator(ll.size());
/* while(li.hasNext())
{
    System.out.print(li.next()+" ");
}
System.out.println();
*/
while(li.hasPrevious())
{
    System.out.print(li.previous()+" ");
}
}
public static void main(String[] args)
{
    ClassA aobj=new ClassA();
    aobj.meth1();
}
```

Implementing LinkedList
[10, Java, null, 10, A, false, 99, 1]
size() : 8
get() : false
Reteriving the data in BOTH the directions by using for lc
Reteriving the data by using for-each loop
Reteriving the data by using Iterator
Reteriving the data by using ListIterator
1 99 false A 10 null Java 10 |

```

1 package com.pack1;
2
3 import java.util.LinkedList;
4 import java.util.ListIterator;
5
6 public class ClassA
7 {
8     void meth1()
9     {
10         System.out.println("Implementing LinkedList\n");
11
12         LinkedList<Object> ll=new LinkedList<Object>();
13
14         ll.add(10); // Insertion order is maintained
15         ll.add("Java");// Heterogeneous data is allowed
16         ll.add(null);// null value is allowed
17         ll.add(10);// Duplicates are Allowed
18         ll.add('A'); // It is available from Java 1.2v
19         ll.add(false);// Its default capacity is 0 // In linkedlist elements will be stored in the form of NODES
20         ll.add(99);// Its size increases by DOUBLE
21         ll.add(1); // It is NOT synchronized
22
23         System.out.println(ll);
24         System.out.println("size() : "+ll.size());
25         System.out.println("get() : "+ll.get(5));
26
27         System.out.println("\nReteriving the data in BOTH the directions by using for loop");
28         // write the logic
29
30         System.out.println("\nReteriving the data by using for-each loop");
31         // write the logic
32
33         System.out.println("\nReteriving the data by using Iterator");
34         // write the logic
35
36         System.out.println("\nReteriving the data by using ListIterator");
37         ListIterator<Object> li=ll.listIterator(ll.size());
38         /* while(li.hasNext())
39         {
40             System.out.print(li.next()+" ");
41         }
42
43         System.out.println();
44         */
45
46         while(li.hasPrevious())
47         {
48             System.out.print(li.previous()+" ");
49         }
50     void meth2()
51     {
52         // TASK : WAP to pass userdefined class Object into linkedlist & reterive it back.
53     }
54     public static void main(String[] args)
55     {
56         ClassA aobj=new ClassA();
57         aobj.meth1();
58     }
59 }

```



```

3=import java.util.ArrayList;
4 import java.util.LinkedList;
5
6 public class LinkedList_Time
7 {
8     private static Object arr[];
9
10    static
11    {
12        arr=new Object[100000];
13        for(int i=0;i<arr.length;i++) // using for loop to pass 100000 Objects
14            arr[i]=new Object();
15    }
16    void ArrayListTime()
17    {
18        long start;
19        long end;
20
21        ArrayList<Object> al=new ArrayList<Object>();
22        start=System.currentTimeMillis();//Its a static method gives the current system time in long millisec
23        for(Object obj1:arr)
24        {
25            al.add(obj1);
26        }
27        end=System.currentTimeMillis();
28
29        System.out.println("ArrayList Construction Time"+(end-start));
30    }
31    void LinkedListTime()
32    {
33        long start, end;
34        LinkedList <Object>ll=new LinkedList<Object>();
35        start=System.currentTimeMillis();
36        for(Object obj2:arr)
37        {
38            ll.add(obj2);
39        }
40        end=System.currentTimeMillis();
41        System.out.println("LinkedList Construction Time"+(end-start));
42    }
43    void meth1()
44    {
45        for(Object o:arr)
46            System.out.println(o);
47    }

```

```

47= public static void main(String[] args)
48 {
49     LinkedList_Time lt=new LinkedList_Time();
50     lt.ArrayListTime();
51     lt.LinkedListTime();
52     //lt.meth1();
53 }
54 }

```

The screenshot shows an IDE with a code editor on the left and a console output on the right. The code in the editor is the same as the previous block. The console output shows two lines: "ArrayList Construction Time6" and "LinkedList Construction Time6". Both lines are highlighted with a blue selection box and a red checkmark to their right.

We are getting the same time, but LinkedList should take more time than array List because data transfer in nodes execute the program for one more time

The screenshot shows the same IDE as before, but the console output now shows "ArrayList Construction Time6" and "LinkedList Construction Time7". The "LinkedList Construction Time7" line is highlighted with a blue selection box and a red checkmark to its right.

- 1) What is a Java LinkedList and how is it different from an ArrayList?
- 2) How do you add an element to the beginning of a Java LinkedList?
- 3) How do you add an element to the end of a Java LinkedList?
- 4) How do you remove the first element of a Java LinkedList?
- 5) How do you remove the last element of a Java LinkedList?
- 6) How do you iterate through a Java LinkedList?
- 7) How do you get the size of a Java LinkedList?
- 8) How do you check if a Java LinkedList contains a specific element?
- 9) How do you get the first element of a Java LinkedList?
- 10) How do you get the last element of a Java LinkedList?

Write a Java program that creates a LinkedList of strings and then removes all strings that contain the letter "a". Finally, print out the LinkedList to the console.