LinkedList<E> :
----
public class LinkedList<E> extends AbstractSequentialList<E> implements List<E>, Deque<E>, Cloneable, Serializable

* It is an implemented class of List<E> interface available from JDK 1.2V.
* It can accept duplicate, null, homogeneous and hetrogeneous elements.
* It stores the element based on the index position but in a **non contiguous** memory location.
* It uses Doubly LinkedList data structure.
* This linkage facility provides an efficient way to insert and delete the element anywhere in the List.
* It stores the data in Node class format.
* Default capacity is 0, Every time we will add an element one new node will be created internally.
* Each node is interlinked with another node.
* It also implements Deque<E> interface from JDK 1.6V so perform basic queue operation.
* Methods are not synchronized.
* Iterator is Fail Fast Iterator.
* IT IS MAINLY USED TO PERFORM FREQUENT INSERTION AND DELETION IN THE MIDDLE OF THE LIST.
* It may iterate elements more slowly than ArrayList because It search the element by using node reference
* It will traverse the list from the beginning or the end, whichever is closer to the specified index.

Method searching criteria in the LinkedList :

Methods		Criteria
indexOf()	:	Head Node
lastIndexOf()	:	Tail Node
contains()	:	Head Node
get()	:	Head OR Tail (Closer end)

Constructor :
----
* We have two types of constructor in LinkedList :

1) LinkedList l1 = new LinkedList();
   Will create empty LinkedList with default capacity is 0.

2) LinkedList l2 = new LinkedList(Collection coll);
   To provide loose coupling

Methods :
----
1) public void addFirst(Object obj)
2) public void addLast(Object obj)

3) public Object getFirst()
4) public Object getLast()

5) public Object removeFirst()
6) public Object removeLast()

```java
package com.ravi.linked_list;


import java.util.Iterator;
import java.util.LinkedList;
public class LinkedListDemo
{
 public static void main(String args[])
 {
      LinkedList<Object> list=new LinkedList<>();
      list.add("Ravi");
      list.add("Vijay");
      list.add("Ravi");
      list.add(null);
      list.add(42);

      System.out.println("1st Position Element is :"+list.get(1));

      //Iterator interface

       Iterator<Object> itr = list.iterator();
       itr.forEachRemaining(System.out::println); //JDK 1.8

  }
}
```

Note : from the above program It is clear that LinkedList follwos index position

---

```java
package com.ravi.linked_list;

import java.util.*;
public class LinkedListDemo1
{
    public static void main(String[] args)
    {
        final int ITERATION = 100000;

        long startTime = System.currentTimeMillis();

        ArrayList<Integer> al = new ArrayList<>();

        for(int i=0; i<=ITERATION; i++)
        {
            al.add(0, i);
        }

        long endTime = System.currentTimeMillis();

        System.out.println("Time taken by ArrayList class :"+(endTime - startTime)+" ms");

           startTime = System.currentTimeMillis();

        LinkedList<Integer> list = new LinkedList<>();

        for(int i=0; i<=ITERATION; i++)
        {
            list.add(0, i);
        }

        endTime = System.currentTimeMillis();

        System.out.println("Time taken by LinkedList class :"+(endTime - startTime)+" ms");

    }
}
```

From the above program, It is clear that LinkedList is very good choice for insertion and deletion of the element from the middle of the List.

```java
package com.ravi.linked_list;


import java.util.Arrays;
//Methods of LinkedList class
import java.util.LinkedList;
import java.util.List;
public class LinkedListDemo2
{
  public static void main(String[] argv)
  {
        LinkedList<String> list = new LinkedList<>();

        list.addFirst("Ravi");  //    Rahul
        list.add("Rahul");
        list.addLast("Anand");

        System.out.println(list.getFirst());
        System.out.println(list.getLast());

        list.removeFirst();
        list.removeLast();

        System.out.println(list); //[Rahul]

        System.out.println("...........................");

       List<String> listOfName = Arrays.asList("Ravi","Rahul","Ankit", "Rahul");

       LinkedList<String> names = new LinkedList<>(listOfName);
       names.forEach(System.out::println);
  }
}
```

```java
package com.ravi.linked_list;

//ListIterator methods (add(), set(), remove())
import java.util.*;
public class LinkedListDemo3
{
    public static void main(String[] args)
    {
        LinkedList<String> city = new LinkedList<> ();
         city.add("Kolkata");
         city.add("Bangalore");
         city.add("Hyderabad");
         city.add("Pune");
         System.out.println(city);


        ListIterator<String> lt = city.listIterator();

         while(lt.hasNext())
          {
            String cityName =  lt.next();

            if(cityName.equals("Kolkata"))
            {
                lt.remove();
            }
            else if(cityName.equals("Hyderabad"))
            {
                lt.add("Ameerpet");
            }
            else if(cityName.equals("Pune"))
            {
                lt.set("Mumbai");
            }
        }
        city.forEach(System.out::println);

    }
}
```

Here there is no ConcurrentModificationException because ListIterator is modifying the structure by it's own method hence there is no problem because it is internal structure modification.

```java
package com.ravi.linked_list;


//Insertion, deletion, displaying and exit

import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

public class LinkedListDemo4
{
public static void main(String[] args)
    {
     List<Integer> linkedList = new LinkedList<>();
     Scanner scanner = new Scanner(System.in);

       while (true)
        {
         System.out.println("Linked List: " + linkedList); //[]
         System.out.println("1. Insert Element");
         System.out.println("2. Delete Element");
           System.out.println("3. Display Element");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");

         int choice = Integer.parseInt(scanner.nextLine());
         switch (choice)
            {
             case 1:
                 System.out.print("Enter the element to insert: ");
                 int elementToAdd = Integer.parseInt(scanner.nextLine());
                 linkedList.add(elementToAdd);
                 break;
             case 2:
                 if (linkedList.isEmpty())
                   {
                     System.out.println("Linked list is empty. Nothing to delete.");
                     }
                   else
                   {
                     System.out.print("Enter the element to delete: ");
                     int elemenetToDelete = Integer.parseInt(scanner.nextLine());

          boolean remove =  linkedList.remove(Integer.valueOf(elemenetToDelete));

                       if(remove)
                       {
                           System.out.println("Element "+elemenetToDelete+ " is deleted
 Successfully" );
                       }
                       else
                       {
                           System.out.println("Element "+elemenetToDelete+" not available is
 the LinkedList");
                       }

                 }
                 break;
               case 3:
                  System.out.println("Elements in the linked list.");
                  linkedList.forEach(System.out::println);
                   break;
             case 4:
                 System.out.println("Exiting the program.");
                 scanner.close();
                 System.exit(0);
             default:
                 System.out.println("Invalid choice. Please try again.");
         }
     }
 }
 }
```

```java
package com.ravi.linked_list;


import java.util.LinkedList;

public class LinkedListDemo5
{
    public static void main(String[] args)
    {
        LinkedList<String> train = new LinkedList<>();

        train.add("Coach A");
        train.add("Coach B");
        train.add("Coach C");
        System.out.println("Initial Train: " + train);

        train.addFirst("Engine");
        System.out.println("After adding Engine at front: " + train);


        train.addLast("Guard Coach");
        System.out.println("After adding Guard Coach at end: " + train);


        train.add(2, "Pantry Coach");
        System.out.println("After adding Pantry Coach in middle: " + train);


        train.removeFirst();
        System.out.println("After removing first coach: " + train);


        train.removeLast();
        System.out.println("After removing last coach: " + train);


        train.remove(1);
        System.out.println("After removing Pantry Coach from middle: " + train);

    }
}
```

```java
package com.ravi.linked_list;


import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

record Product(Integer productId, String productName)
{

}

public class LinkedListDemo6 {

    public static void main(String[] args)
    {
        LinkedList<Product> listOfProduct = new LinkedList<Product>();
        listOfProduct.add(new Product(1, "ApplePhone"));
        listOfProduct.add(new Product(2, "MiPhone"));
        listOfProduct.add(new Product(3, "VivoPhone"));

        System.out.println("Is list empty :"+listOfProduct.isEmpty());

        Iterator<Product> iterator = listOfProduct.iterator();
        iterator.forEachRemaining(prod ->
System.out.println(prod.productName().toUpperCase()));

        Product product = listOfProduct.get(1);
        System.out.println(product.productName());

    }

}
```

```java
package com.ravi.linked_list;

import java.util.LinkedList;

public class LinkedListDemo7
{
    public static void main(String args[])
    {
         LinkedList<String> list= new LinkedList<>(); //generic
         list.add("Item 2");//2
         list.add("Item 3");//3
         list.add("Item 4");//4
         list.add("Item 5");//5
         list.add("Item 6");//6
         list.add("Item 7");//7

         list.add("Item 9"); //10

         list.add(0,"Item 0");//0
         list.add(1,"Item 1"); //1



         list.add(8,"Item 8");//8
          list.add(9,"Item 10");//9
          System.out.println(list);


           list.remove("Item 5");

             System.out.println(list);

              list.removeLast();
               System.out.println(list);

                list.removeFirst();
               System.out.println(list);

             list.set(0,"Ajay"); //set() will replace the existing value
             list.set(1,"Vijay");
             list.set(2,"Anand");
             list.set(3,"Aman");
             list.set(4,"Suresh");
             list.set(5,"Ganesh");
             list.set(6,"Ramesh");
             list.forEach(x -> System.out.println(x));

    }

}
```

Set<E> interface :
----
* Set<E> interface is the sub interface of Collection<E> available from JDK 1.2V.

* Set interface does not follow indexing technique, Actually It uses **hashing** technique.

* Set interface never accept duplicate elements, Here our best friend is equals(Object obj) method which will verify whether two objects are **identical** or not, If identical then It will accept only one.

* Set<E> interface does not follow index so ListIterator interface will not work with Set<E> interafce.

* Set<E> interface uses all the methods of Collection<E> interface, few more methosa are added from JDK 9V.

Set interface Hierarchy :