**Collection Framework – Part\_06**

* **Limitations of Enumeration:**

We can Enumeration concept only for legacy classes and it is not a universal cursor.

By using Enumeration, we can get only read access and we can’t perform remove operation.

To overcome above limitations, we should go for iterator.

* **Iterator:**

We can apply Iterator concept for any Collection object and hence it is universal cursor.

By using Iterator, we can perform read/remove operations.

We can create Iterator object by using iterator() method of Collection interface.

public Iterator iterator();

Exampe:

Iterator itr = c.iterator();

c 🡪 Any collection object.

**Methods:**

public boolean hasNext();

public Object next();

public void remove();

**Example:**

import java.util.\*;

class IteratorDemo{

public static void main(String[] args){

ArrayList l = new ArrayList();

for(int i=0;i<=10;i++){

l.add(i);

}

System.out.println(l); //[0,1,2,…..10]

Iterator itr = l.iterator();

while(itr.hasNext()){

Integer I = (Integer) itr.next();

if(I % 2 == 0){

System.out.println(l); //0,2,4,6,8,10

else

itr.remove();

}

}

System.out.println(l); //[0,1,2,4,6,8,10]

}

}

**Limitations of Iterator:**

1. By using Enumeration and Iterator, we can always move only towards forward direction and we can’t move towards backward direction. These are single direction cursors, but not bi-directional cursor.
2. By using Iterator, we can perform only read and remove operations and we can’t perform replacement and addition of new objects.

To overcome above limitations, we should go for ListIterator.

* **ListIterator:**

1. By using ListIterator, we can move either to the forward direction or to the backward direction and hence it is bi-directional cursor.
2. By using ListIterator, we can perform replacement and addition of new object in addition to read and remove operations.

We can created ListIterator, by using listIterator() of List interface.

public ListIterator listIterator();

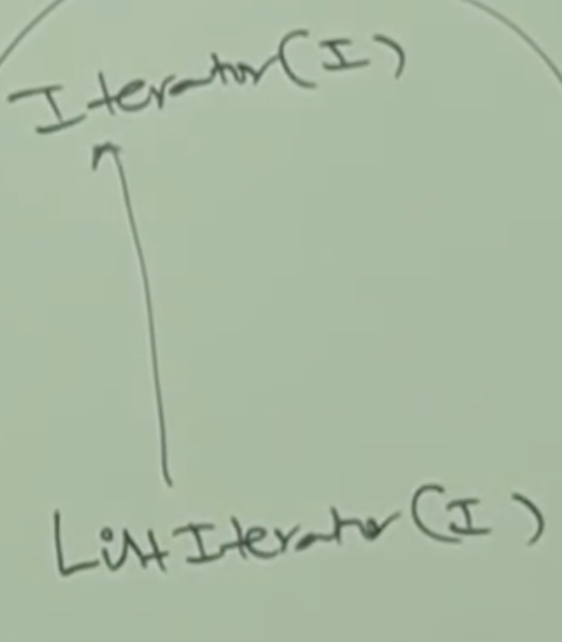
Example:

ListIterator ltr = l.listIterator();

l 🡪 Any List object.

**Methods:**

ListIterator is the child interface of Iterator and hence all methods present in Iterator by defaut available to the ListIterator.



ListIterator defines the following 9 methods.

**Forward Movement Methods:**

public boolean hasNext();

public Object next();

public int nextIndex();

**Backward Movement Methods:**

public boolean hasPrevious();

public Object previous();

public int previousIndex();

**Extra Operations:**

public void remove();

public void add(Object o);

public void set(Object o);

**Example:**

import java.util.\*;

class ListIteratorDemo{

LinkedList l = new LinkedList();

l.add(“balakrishna”);

l.add(“venki”);

l.add(“chiru”);

l.add(“nag”);

System.out.println(l);[balakrishna,venki,chiru,nag]

ListIterator itr = l.listIterator();

while(itr.hasNext()){

String s = (String) itr.next();

If(s.equals(“venki”)){

itr.remove(); //[balakrishna,chiru,nag]

} else if(s.equals(“nag”)){

ltr.add(“chaitu”); //[balakrishna,chiru,nag,chaitu]

} else if(s.equals(“chiru”)){

ltr.set(“charan”);//[balakrishna,charan,nag,chaitu

}

}

System.out.println(l); //[balakrishna,charan,nag,chaitu]

}

Note:

The most powerful cursor is ListIterator. But, its limitation is, it is applicable only for List objects.

* **Comparison Table of 3 cursors:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Property** | **Enumeration** | **Iterator** | **ListIterator** |
| 1 | Where we can apply? | Only for legacy classes | For any Collection object. | Only for list object. |
| 2 | Is it legacy? | YES(1.0V) | NO(1.2V) | NO(1.2V) |
| 3 | Movement | Single Direction (only forward direction) | Single Direction(only forward direction) | Bidirectional |
| 4 | Allowed operations | Only read | Read/Remove | Read/Remove  Replace/Add |
| 5 | How we can get? | By using elements() of vector class | By using iterator() of Collection(I) | By using listIterator() of List(I) |
| 6 | Methods | 2 methods:  hasMoreElements()  nextElement() | 3 Methods  hasNext()  next()  remove() | 9 methods |

* **Internal implementation of cursors:**

import java.util.\*;

class CursorDemo{

public static void main(String[] args){

Vector v = new Vector();

Enumeration e = v.element();

Iterator itr = v.iterator();

ListIterator litr = v.listIterator();

System.out.println(e.getClass().getName());

System.out.println(itr.getClass().getName());

System.out.println(litr.getClass().getName());

}

}

Output:

java.util.Vector$1

java.util.Vector$Itr

java.util.Vector$1ListItr

Note:

We are not creating and getting the objects of above interfaces, we getting the reference of the class which implements the above interfaces.