Iterators in Java :

Enumeration :

> Interface used to get elements of legacy collections(Vector, Hashtable).
> also used to specify the input streams to a SequenceInputStream.
> calling syntax, Enumeration e = V.elements();
> There are two methods in Enumeration interface only —

- public boolean hasMoreElements
  // Test if this enumeration contains more elements
- public Object nextElement(); //returns next element
  // throws NosuchElementException

Example:

```
public class Abhi
{    public static void main(String args[])
     {
            Vector v= new Vector();
            for(int i=0; i<10; i++)
                   v.addElement(i);
            Enumeration e = v.elements();
            while( e.hasMoreElements())
            {
                int i =(Integer)e.nextElement();
                Sop(i +"  ");
}           }
```

Limitations of Enumeration :

> Its for legacy classes (Vector, Hashtable) only. Hence it is not a universal iterator.

> Remove operations can't be performed using Enumeration.

> Only forward direction iterating is possible.

① Iterator

> It is a universal iterator as we can apply it to any collection object.

> can perform both read and ~~write~~ remove operations.

> Iterator is the only cursor available for entire collection framework.

> calling syntax, Iterator i = c. iterator();

> Iterator interface defines three methods :

- public boolean hasNext();
  // returns true if the iteration has more elements

- public Object next(); // return next element in iteration
  > throws NoSuchElementException

- public void remove();
  // Remove the next element in the iteration.
  → ~~Unsuppo~~
  // can be called only once per call to next()

remove()

↳ UnsupportedOperationExceptions:
> If remove operation is not supported by this iterator.

→ IllegalStateException:
- If the next method has not yet been called, or the remove method has been called after the last call to the next method.

Example -

```
public class Abhi
{
    public static void main (String args[])
    {
        ArrayList al = new ArrayList();
        for(int i=0; i<10; i++)
            al.add(i);
        sop(al);
        Iterator itr = al.iterator();
        while(itr.hasNext())
        {
            int i = (Integer)itr.next();
            sop(i+" ");
            if (i%2!=0)
                itr.remove();
        }
        sopln();
        sop(i+" ");
} }
```

Scanned by CamScanner

Limitations of Iterator -
>Only forward direction iteration is possible
> Replacement & addition of new element is not supported by iterator.

ListIterator :

>It is only applicable for ListCollection implemented classes like arraylist, linkedlist etc.

> It provides bi-directional iteration.

> calling syntax, ListIterator ltr = l.listIterator();

> ListIterator interface extends interface interface.

Methods -

>public boolean hasNext();
>public Object next();
>public int nextIndex();
// returns the next element index
// or list size if the list iterator is at the end of the list.

> public boolean hasPrevious();
>public Object previous();
> public int previousIndex();
// -1 if list iterator is at begining of list.

> public void remove

> public void set(Object obj)
// Replaces the last element returned by next() or previous() with specified element.

> public void add(Object obj)
// Inserts the specified element into the list at position before the element that would be returned by next.

> set() method can throw 4 Exceptions:

add() [
- UnsupportedOperationException
- ClassCastException
- IllegalArgumentException
]
- Illegal StateException.

Limitation :

It is the most powerful iterator but it is only applicable for List implemented classes, so it is not a universal operator.

Implementation :

```
public class ListIteratorExample
{
    public static void main(String args[])
    {
      Arraylist al= new ArrayList();
      for(int i=0; i<10; i++)
          al.add(i);
      Sop(al);

      ListIterator Itr= al.listIterator();
      while(Itr.hasNext())
      {
          int i=(Integer) Itr.next();
          Sop(i+" ");
          if(i%2==0)
          {
                i++;
                Itr.set(i);
                Itr.add(i);
          }
      }
       Sopn();
       Sop(al);
    }
}
```