**Java Collections – Iterator/ListIterator**

Both Iterator and ListIterator are used to iterate through elements of a collection class. Using Iterator we can traverse in one direction (forward) while using ListIterator we can traverse the collection class on both the directions(backward and forward).

**Iterator**

Iterator is used for iterating (looping) various collection classes such as [HashMap](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/" \o "HashMap in Java with Example" \t "_blank), [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/" \o "ArrayList in java with example programs – Collections Framework" \t "_blank), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/" \o "LinkedList in Java with Example" \t "_blank) etc.

**Iterator Example:**

import java.util.ArrayList;

import java.util.Iterator;

public class IteratorDemo2 {

public static void main(String args[]){

ArrayList names = new ArrayList();

names.add("Chaitanya");

names.add("Steve");

names.add("Jack");

//Adding Integer value to String ArrayList

names.add(new Integer(10));

Iterator it = names.iterator();

while(it.hasNext()) {

String obj = (String)it.next();

System.out.println(obj);

}

}

}

**Output:**

ChaitanyaException in thread "main"

Steve

Jack

java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String

at beginnersbook.com.Details.main(Details.java:18)

# ListIterator in Java with examples

ListIterator allows us to traverse the list in both directions (forward and backward).

**ListIterator Example**

In this example we are traversing an [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/" \o "ArrayList in java with example programs – Collections Framework" \t "_blank) in both the directions.

import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class ListIteratorExample {

public static void main(String a[]){

ListIterator<String> litr = null;

List<String> names = new ArrayList<String>();

names.add("Shyam");

names.add("Rajat");

names.add("Paul");

names.add("Tom");

names.add("Kate");

//Obtaining list iterator

litr=names.listIterator();

System.out.println("Traversing the list in forward direction:");

while(litr.hasNext()){

System.out.println(litr.next());

}

System.out.println("\nTraversing the list in backward direction:");

while(litr.hasPrevious()){

System.out.println(litr.previous());

}

}

}

**Output:**

Traversing the list in forward direction:

Shyam

Rajat

Paul

Tom

Kate

Traversing the list in backward direction:

Kate

Tom

Paul

Rajat

Shyam

**Methods of ListIterator**

1) void add(E e): Inserts the specified element into the list (optional operation).  
2) boolean hasNext(): Returns true if this list iterator has more elements when traversing the list in the forward direction.  
3) boolean hasPrevious(): Returns true if this list iterator has more elements when traversing the list in the reverse direction.  
4) E next(): Returns the next element in the list and advances the cursor position.  
5) int nextIndex(): Returns the index of the element that would be returned by a subsequent call to next().  
6) E previous(): Returns the previous element in the list and moves the cursor position backwards.  
7) int previousIndex(): Returns the index of the element that would be returned by a subsequent call to previous().  
8) void remove(): Removes from the list the last element that was returned by next() or previous() (optional operation).  
9) void set(E e): Replaces the last element returned by next() or previous() with the specified element (optional operation).