**Java List Interface**

List Interface is the subinterface of Collection. It contains index-based methods to insert and delete elements. It is a factory of ListIterator interface.

**List Interface declaration**

1. public interface List<E> extends Collection<E>

**Methods of Java List Interface**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(int index, E element) | It is used to insert the specified element at the specified position in a list. |
| boolean add(E e) | It is used to append the specified element at the end of a list. |
| boolean addAll(Collection<? extends E> c) | It is used to append all of the elements in the specified collection to the end of a list. |
| boolean addAll(int index, Collection<? extends E> c) | It is used to append all the elements in the specified collection, starting at the specified position of the list. |
| void clear() | It is used to remove all of the elements from this list. |
|  |  |
| boolean equals(Object o) | It is used to compare the specified object with the elements of a list. |
| int hashcode() | It is used to return the hash code value for a list. |
| E get(int index) | It is used to fetch the element from the particular position of the list. |
| boolean isEmpty() | It returns true if the list is empty, otherwise false. |
| int lastIndexOf(Object o) | It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element. |
| Object[] toArray() | It is used to return an array containing all of the elements in this list in the correct order. |
| T[] toArray(T[] a) | It is used to return an array containing all of the elements in this list in the correct order. |
| boolean contains(Object o) | It returns true if the list contains the specified element |
| boolean containsAll(Collection<?> c) | It returns true if the list contains all the specified element |
| int indexOf(Object o) | It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element. |
| E remove(int index) | It is used to remove the element present at the specified position in the list. |  |
| boolean remove(Object o) | It is used to remove the first occurrence of the specified element. |  |
| boolean removeAll(Collection<?> c) | It is used to remove all the elements from the list. |  |
| void replaceAll(UnaryOperator operator) | It is used to replace all the elements from the list with the specified element. |  |
| void retainAll(Collection<?> c) | It is used to retain all the elements in the list that are present in the specified collection. |  |
| E set(int index, E element) | It is used to replace the specified element in the list, present at the specified position. |  |
| void sort(Comparator<? super E> c) | It is used to sort the elements of the list on the basis of specified comparator. |  |
| Spliterator spliterator() | It is used to create spliterator over the elements in a list. |  |
| List<E> subList(int fromIndex, int toIndex) | It is used to fetch all the elements lies within the given range. |  |
| int size() | It is used to return the number of elements present in the list. |  |

**Java List Example**

1. import java.util.\*;
2. public class ListExample{
3. public static void main(String args[]){
4. List<String> al=new ArrayList<String>();
5. al.add("Amit");
6. al.add("Vijay");
7. al.add("Kumar");
8. al.add(1,"Sachin");
9. System.out.println("An element at 2nd position: "+al.get(2));
10. for(String s:al){
11. System.out.println(s);
12. }
13. }
14. }

Output:

An element at 2nd position: Vijay

Amit

Sachin

Vijay

Kumar

**Java ListIterator Interface**

ListIterator Interface is used to traverse the element in a backward and forward direction.

**ListIterator Interface declaration**

1. public interface ListIterator<E> extends Iterator<E>

**Methods of Java ListIterator Interface:**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void add(E e) | This method inserts the specified element into the list. |
| boolean hasNext() | This method returns true if the list iterator has more elements while traversing the list in the forward direction. |
| E next() | This method returns the next element in the list and advances the cursor position. |
| int nextIndex() | This method returns the index of the element that would be returned by a subsequent call to next() |
| boolean hasPrevious() | This method returns true if this list iterator has more elements while traversing the list in the reverse direction. |
| E previous() | This method returns the previous element in the list and moves the cursor position backward. |
| E previousIndex() | This method returns the index of the element that would be returned by a subsequent call to previous(). |
| void remove() | This method removes the last element from the list that was returned by next() or previous() methods |
| void set(E e) | This method replaces the last element returned by next() or previous() methods with the specified element. |

**Example of ListIterator Interface**

1. import java.util.\*;
2. public class ListIteratorExample1{
3. public static void main(String args[]){
4. List<String> al=new ArrayList<String>();
5. al.add("Amit");
6. al.add("Vijay");
7. al.add("Kumar");
8. al.add(1,"Sachin");
9. ListIterator<String> itr=al.listIterator();
10. System.out.println("Traversing elements in forward direction");
11. while(itr.hasNext()){
13. System.out.println("index:"+itr.nextIndex()+" value:"+itr.next());
14. }
15. System.out.println("Traversing elements in backward direction");
16. while(itr.hasPrevious()){
18. System.out.println("index:"+itr.previousIndex()+" value:"+itr.previous());
19. }
20. }
21. }

Output:

Traversing elements in forward direction

index:0 value:Amit

index:1 value:Sachin

index:2 value:Vijay

index:3 value:Kumar

Traversing elements in backward direction

index:3 value:Kumar

index:2 value:Vijay

index:1 value:Sachin

index:0 value:Amit

**Example of ListIterator Interface: Book**

1. import java.util.\*;
2. class Book {
3. int id;
4. String name,author,publisher;
5. int quantity;
6. public Book(int id, String name, String author, String publisher, int quantity) {
7. this.id = id;
8. this.name = name;
9. this.author = author;
10. this.publisher = publisher;
11. this.quantity = quantity;
12. }
13. }
14. public class ListIteratorExample2 {
15. public static void main(String[] args) {
16. //Creating list of Books
17. List<Book> list=new ArrayList<Book>();
18. //Creating Books
19. Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
20. Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);
21. Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
22. //Adding Books to list
23. list.add(b1);
24. list.add(b2);
25. list.add(b3);
26. //Traversing list
27. for(Book b:list){
28. System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
29. }
30. }
31. }

Output:

101 Let us C Yashwant Kanetkar BPB 8

102 Data Communications & Networking Forouzan Mc Graw Hill 4

103 Operating System Galvin Wiley 6