**List<E> Interface.**

Let us start understanding List<E> interface. We will leverage the concept of one of its concrete implementation ArrayList<E> to understand the interface.

First few details for List<E> interface.

List<E> is ordered collection. It means that the insertion of element is maintained. Also a user has precise control of where the element can be inserted. That being said it is conclusive that List<E> provides index access.

NOTE: LinkedList<E> class implements List<E> and it does not provide index based access to elements. So operations cost O(n) where n is total number of elements in List<E>.

Enough theory now let us start coding.

First let us declare our List<E>.

Syntax to use List<E> interface with ArrayList<E> class is as follows:

              List<Integer> list=**new** ArrayList<Integer>();

List<Integer> - we are declaring List<E> of type Integer. Remember you cannot insert anything else then Integer. Compiler will complain (throw Compile time exception) if you do so.

Now we populate the list using add method of List<E> interface.

**boolean** add(E e); - We use add method to insert element in list. As of ArrayList<E> class this method will append the element at the end of the list. Returns true if the element is successfully inserted in list.

       list.add(1);

       list.add(640);

       list.add(23);

Right now we inserted 3 integer values into the list. Let us just printout the list using System.out.println(list). Output is:

[1, 640, 23]

The insertion order is maintained.

Now we will write a program that will display methods of List<E> as implemented by ArrayList<E>

**package** org.collections;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.ListIterator;

/\*\*

 \* This class is used to demonstrate the ArrayList<E> class of List<E> interface

 \* \*/

**public** **class** ListInterfaceImplementation {

**public** **static** **void** main(String[] args) {

              /\*\*

               \* Define a new ArrayList<E> of type Integer.

               \* \*/

              List<Integer> list = **new** ArrayList<Integer>();

              /\*\*

               \* Populate the list using populateList(list) method.

               \* \*/

*populateList*(list);

              /\*\*

               \* Populate the list using display(list) method.

               \* \*/

*display*(list);

              //Now let us start doing some operations on List<E>

              /\*\*

               \* remove(int index) is used to remove element from the index.

               \* \*/

              list.remove(0);

              System.***out***.print("Removes element at index 0 : ");

*display*(list);

              /\*\*

               \* remove(Object o) is used to remove that element.

               \* \*/

              list.remove(**new** Integer(7));

              System.***out***.print("Removes element 7 : ");

*display*(list);

              /\*\*

               \* calls indexOf(object o) method. returns true if it has index returns

               \* false if indexOf(Object o) returns -1

               \* \*/

**boolean** exists = list.contains(**new** Integer(4));

              System.***out***.println("List contains 4 : " + exists);

              System.***out***.println();

              /\*\*

               \* E get(int index);

               \*

               \* returns the element at index 3

               \* \*/

**int** val = list.get(3);

              System.***out***.println("element at index 3 is " + val);

              System.***out***.println();

              /\*\*

               \* E set(int index, E element);

               \*

               \* Removes old element

               \* \*/

              list.set(3, 1000);

              System.***out***.print("Setting new value at index 3 ");

*display*(list);

              /\*\*

               \*  void add(int index, E element);

               \*

               \* adds new element at index 4

               \* \*/

              list.add(4, 400);

              System.***out***.print("Adding new value in list ");

*display*(list);

              /\*\*

               \* int indexOf(Object o);

               \*

               \* returns indexOf object passed as parameter.

               \* return -1 if element is not in list

               \* \*/

**int** index400 = list.indexOf(400);

              System.***out***.println("index of element 400 is " + index400);

              System.***out***.println();

**int** index6 = list.indexOf(6);

              System.***out***.println("index of element 6 is " + index6);

              System.***out***.println();

              /\*\*

               \* int lastIndexOf(Object o);

               \*

               \* returns last index of object specified in parameter.

               \* this method tries to find the element from end of list.

               \* so the first time the element is found is lastIndexof that object.

               \* \*/

**int** newIndex6 = list.lastIndexOf(6);

              System.***out***.println("last index of element 6 is " + newIndex6);

              System.***out***.println();

              /\*\*

               \* Iterator<E> iterator();

               \*

               \* this method returns the iterator.

               \* We use iterator to iterate on list.

               \* hasNext()

               \*                   returns true if next element exists

               \*                   returns false if end of list reached

               \* \*/

              System.***out***.print("Printing throught iterator ");

              Iterator<Integer> iterator = list.iterator();

**while** (iterator.hasNext()) {

                     System.***out***.print(iterator.next() + " ");

              }

              System.***out***.println();

              System.***out***.println();

              /\*\*

               \* ListIterator<E> listIterator();

               \*

               \* this method

               \* Few methods are listed below.

               \* boolean hasNext()

               \*                   returns true if next element exists

               \*                   returns false if end of list reached

               \*

               \* E next()

               \*                   returns the next element in list

               \*

               \* boolean hasPrevious()

               \*                   returns true if previous element exists

               \*                   returns false if start of list reached

               \*

               \* E previous()

               \*                   returns the previous element

               \*

               \* int nextIndex()

               \*                   returns the nextIndex in list

               \*

               \* int previousIndex()

               \*                   returns the previousIndex in list

               \*

               \* \*/

              ListIterator<Integer> listItr = list.listIterator();

              /\*\*

               \* Traverse through listIterator() in forward direction

               \* \*/

              System.***out***.print("Printing throught ListIterator next() ");

**while** (listItr.hasNext()) {

                     System.***out***.print(listItr.next() + " ");

              }

              System.***out***.println();

              System.***out***.println();

              /\*\*

               \* Traverse through listIterator() in backward direction

               \* \*/

              System.***out***.print("Printing throught ListIterator previous() ");

**while** (listItr.hasPrevious()) {

                     System.***out***.print(listItr.previous() + " ");

              }

              System.***out***.println();

              System.***out***.println();

              /\*\*

               \* List<E> subList(int fromIndex, int toIndex);

               \*

               \* returns the partial view of the list. It is just the view

               \* so if any thing is modified in sublist it will affect the

               \* original list too

               \* \*/

              List<Integer> subList = list.subList(2, 6);

              subList.add(100000);

              System.***out***.print("Modified the subList & original list is modified too");

*display*(list);

              List<Integer> newList = **new** ArrayList<Integer>(list.subList(2, 6));

              System.***out***.print("New sublist created ");

*display*(newList);

              newList.add(19);

              System.***out***.print("Element added in new sublist ");

*display*(newList);

              System.***out***.print("New element in sublist but not in old list ");

*display*(list);

       }

       /\*\*

        \* This method takes argument as list and displays it as a String.

        \* \*/

**static** **void** display(List<Integer> list) {

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

              System.***out***.println();

       }

       /\*\*

        \* This method is used to populate list with some Integer values and null.

        \* \*/

**static** **void** populateList(List<Integer> list) {

              list.add(1);

              list.add(20);

              list.add(13);

              list.add(17);

              list.add(4);

              list.add(7);

              list.add(2);

              list.add(6);

              list.add(6);

              list.add(18);

              list.add(**null**);

              list.add(**null**);

       }

}

Output::

[1, 20, 13, 17, 4, 7, 2, 6, 6, 18, null, null]

Removes element at index 0 : [20, 13, 17, 4, 7, 2, 6, 6, 18, null, null]

Removes element 7 : [20, 13, 17, 4, 2, 6, 6, 18, null, null]

List contains 4 : true

element at index 3 is 4

Setting new value at index 3 [20, 13, 17, 1000, 2, 6, 6, 18, null, null]

Adding new value in list [20, 13, 17, 1000, 400, 2, 6, 6, 18, null, null]

index of element 400 is 4

index of element 6 is 6

last index of element 6 is 7

Printing throught iterator 20 13 17 1000 400 2 6 6 18 null null

Printing throught ListIterator next() 20 13 17 1000 400 2 6 6 18 null null

Printing throught ListIterator previous() null null 18 6 6 2 400 1000 17 13 20

Modified the subList & original list is modified too[20, 13, 17, 1000, 400, 2, 100000, 6, 6, 18, null, null]

New sublist created [17, 1000, 400, 2]

Element added in new sublist [17, 1000, 400, 2, 19]

New element in sublist but not in old list [20, 13, 17, 1000, 400, 2, 100000, 6, 6, 18, null, null]