**Collection Framework – Part\_03**

* **Collection(I):**

If we want to represent a group of individual objects as a single entity then we should go for collection.

Collection interface defines the most common methods which are applicable for any collection object.

Methods:

boolean add(Object o);

boolean addAll(Collection c);

boolean remove(Object o);

boolean removeAll(Collection c);

boolean retainAll(Collection c);

To remove all objects except those present in c.

void clear();

boolean contains(Object o);

boolean containsAll(Collection c);

boolean isEmpty();

int size();

Object[] toArray();

Iterator iterator();

Note: There is concrete class which implements Collection interface directly.

* **List(I):**

List is child interface of Collection interface.

If we want to represent a group of individual objects as a single entity, where duplicates are allowed and insertion order must be preserved, then we should go for List.

We can preserve insertion order with index and we can differentiate duplicate objects by using index. Hence index will play very important role in List.

List interface defines the following specific methods.

void add(int index, Object o);

boolean addAll(int index, Collection c);

Object get(int index);

Object remove(int index);

Object set(int index, Object new);

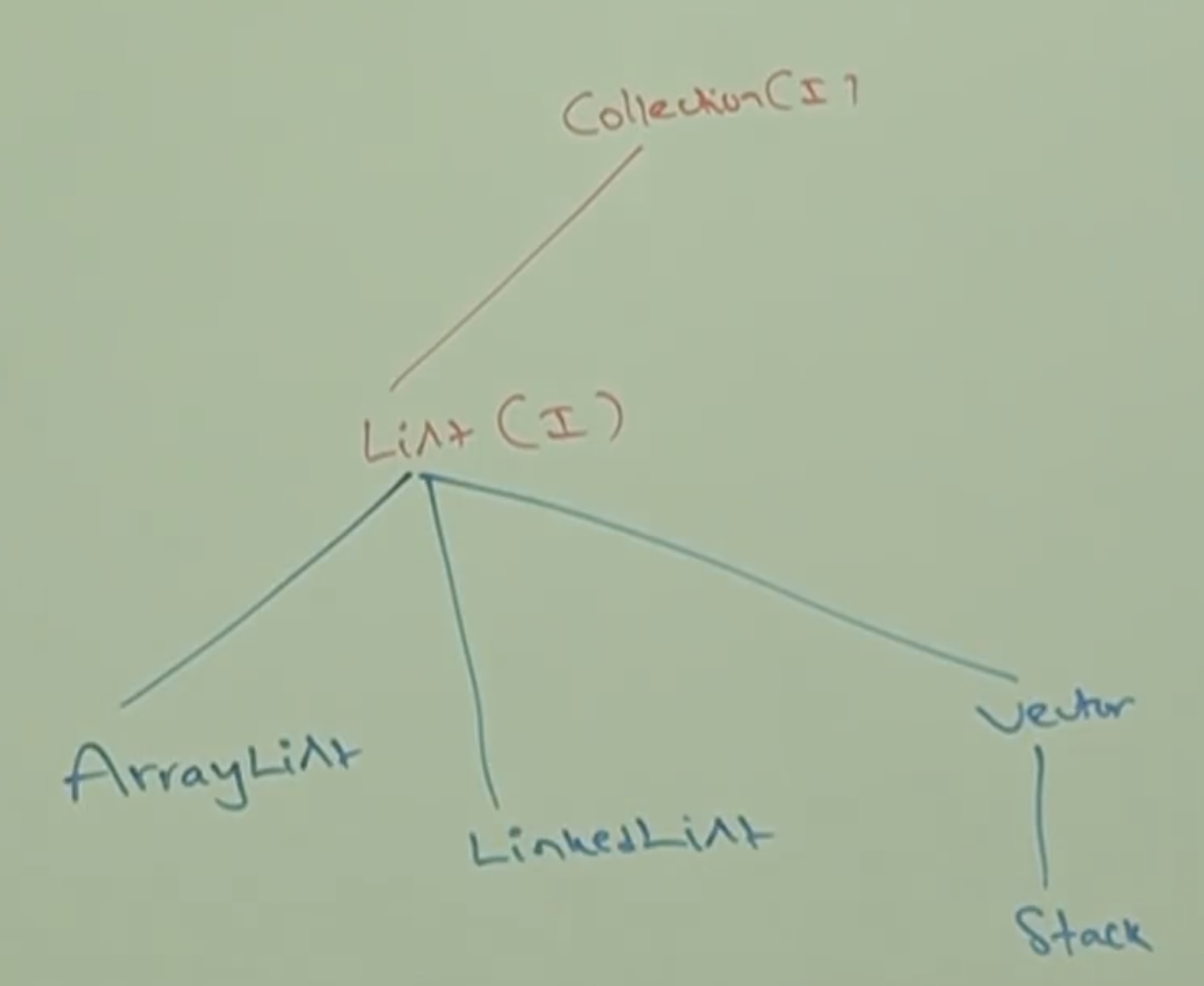
To replace the element present at specified index with provided Object and returns old object.

int indexOf(Object o);

Returns index of first occurrence of ‘o’

int lastIndex(Object o);

ListIterator listIterator();



* **ArrayList:**

1. The underlying data structure is resizable Array or growable Array.
2. Duplicates are allowed.
3. Insertion order is preserved.
4. Heterogeneous objects are allowed. (Except TreeSet and TreeMap, everywhere heterogeneous objects are allowed).
5. null insertion is provided.

**Constructors:**

ArrayList l = new ArrayList();

Creates an empty ArrayList object with default initial capacity 10. Once ArrayList reaches its max capacity then a new ArrayList object will be created with

New Capacity = (Current Capacity \* 3/2) +1;

ArrayList l = new ArrayList(int initialCapacity);

Creates an empty ArrayList object with specified initial capacity.

ArrayList l = new ArrayList(Collection c);

Creates an equivalent ArrayList object for the given Collection.

**Example:**

import java.util.\*;

class ArrayListDemo{

public static void main(String[] args){

ArrayList l = new ArrayList();

l.add(“A”);

l.add(10);

l.add(“A”);

l.add(null);

System.out.println(l); // [A, 10, A, null]

l.remove(2);

System.out.println(l); // [A, 10, null]

l.add(2, “M”);

l.add(“N”);

System.out.println(l); // [A, 10, M, null, N]

}

}

**Note:**

Whenever we try to print the reference variable, it will call the toString() method, the toString() method is overridden in following way for collection and map.

Collection: [ comma separated collection values]

[A, B, C, 10, 20, null]

Map: {key=value, key=value, …}

{100 = Saravana, 101= Pandiyan}