19-05-2020

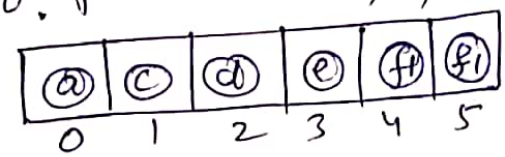
Collection classes in detail Continued —

Let us learn today all about List (interface).

List (interface) —

List is the child of Collection interface. If we want duplicates, insertion order is also preserved, then we have to use List.

By using order (index) we can preserve insertion order also. from 0, 1, 2, 3. — — —



In other words —

→ List is the child interface of Collection

→ 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 differentiate duplicates by using index.

→ We can preserve insertion order by using index, hence index play very important role in list interface.

Methods of list interface: — (Present in List)

→ void add (int index, Object ob)
→ boolean addAll (int index, Collection c)
→ object get (int index)
→ object remove (int index)
→ object set (int index, Object newobject)
→ int indexOf (Object ob)
→ int lastIndexOf (Object ob)
→ ListIterator listIterator ();

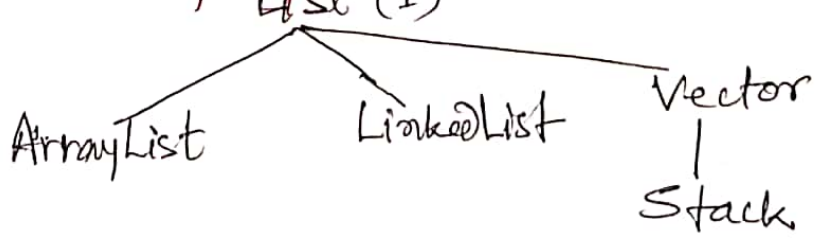
List l;

- l.add("A"); → will add element "A" to the list 0th index
- l.add("B"); → will add B to 1 index

0	1	2
A	B	A

- l.add(int index, Object ob) → will add element ob to the particular index.
- addAll(int index, Collection c) → adds a group of elements to the index.
- remove(int index) — element removal at the specified index of the List.
- l.indexOf("A"); — first occurrence position of element "A" will returned
- l.lastIndexOf("A"); — return last index position of "A" i.e. 2 here
- get(int index) — returns the element present at specified index.
- set(int index, Object ob); — replaces an object present at index, by ob. (Replace)
- ListIterator listIterator() — It is special case of iterator which is used to traverse entire List while getting elements from a List.

Now, Implementing class of List are —



ArrayList

1. Resizable Array or Growable Array
2. Duplicates allowed
3. Insertion Order preserved.
4. Heterogeneous objects can be allowed — that means different class objects can be stored. [except TreeSet and TreeMap] — here not allowed.
5. null insertion is possible

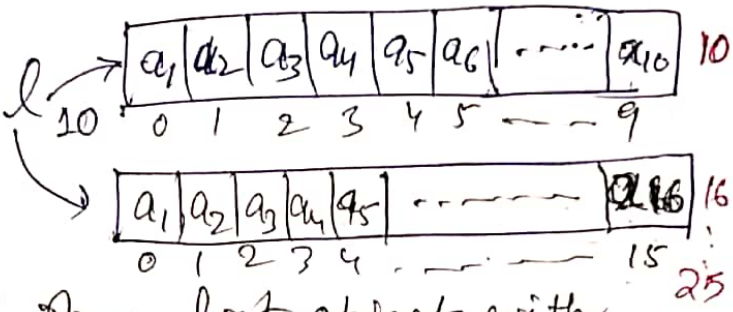
Constructors of ArrayList —

1. ArrayList l = new ArrayList();

When an arraylist filled —

new capacity = (cc * 3/2) + 1

(cc - current capacity)



→ This creates an empty Array list object with default initial capacity 10. Once it reaches its map capacity, a new ArrayList will be created with new capacity = (currentCapacity * 3/2) + 1.

(10 × 3/2) + 1 = 15 + 1 = 16

(16 × 3/2) + 1 = 24 + 1 = 25

. This is the first constructor.

2. ArrayList l = new ArrayList(int initialCapacity);
3. ArrayList l = new ArrayList(Collection c);

. This is 2nd and third important constructors.

Example:— output of any Collection element is printed ~~or~~ in Square brackets because, When we call System.out.println(l);

→ Here l.toString() is already automatically called.

Example :- demo.java → you can also give any name to your class & save with the same name.

```
import java.util.*;
class demo{
    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);   // [A, 10, A, null]
        System.out.println(l);//[A, 10, null] ✓
    l.add(2,"M");  // [A, 10, M, null]
    l.add("N");    // [A, 10, M, null, N]
System.out.println(l); // [A, 10, M, null, N] ✓
    }
}
```

Final output is clearly:-

```
[A, 10, A, null]
[A, 10, null]
[A, 10, M, null, N]
```

① All Collection classes are already implementing Serializable (I) interface Hence, all are Serializable.
So, ArrayList, LinkedList, TreeSet — all are serializable.

② Objects can be easily travel across a network because Objects are serializable in nature.
(Can be sended across a network)

③ Every Collection class is already implementing Cloneable interface.

★ Usually we can use Collections to hold and transfer Objects from one place to another place, to provide support for this requirement every collection already implements Serializable and Cloneable interfaces.

④ In ArrayList and Vector class, one special interface property is present — RandomAccess

In other words, ArrayList and Vector class is implementing RandomAccess interface so that we can access any Random element with the same speed.

Hence if our frequent operation is retrieval operation then ArrayList is the best choice.

RandomAccess (interface)

★ Present in java.util package.
★ It doesn't contain any methods and it is a Marker interface.
★ Only ArrayList and Vector is implemented using RandomAccess interface.

Let us check it with following example:—

```
ArrayList l1 = new ArrayList();
LinkedList l2 = new LinkedList();
System.out.println(l1 instanceOf Serializable); //true
System.out.println(l2 instanceOf Cloneable); //true
System.out.println(l1 instanceOf RandomAccess); //true
System.out.println(l2 instanceOf RandomAccess); // false
```

ArrayList:- Best Choice When:-

If our frequent operation is retrieval operation then ArrayList is the best choice.
Since ArrayList implements RandomAccess interface.

Worst Choice When:-

If our frequent operation is insertion and deletion in the middle of the ArrayList then it is the worst choice.
Since several shift operations are required internally.

In next class we will study - LinkedList (in detail) & Vector.

Thank you