Sri SAI RAM ENGINEERING COLLEGE

INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

Sairam INSTITUTIONS

SAIRAM DIGITAL RESOURCES

UNIT NO 2

INHERITANCE AND INTERFACES

2.8 Array Lists

YEAR	SEM
II	III

CS8392

OBJECT ORIENTED PROGRAMMING (Common to CSE, EEE, EIE, ICE, IT)

COMPUTER SCIENCE & ENGINEERING

Definition:

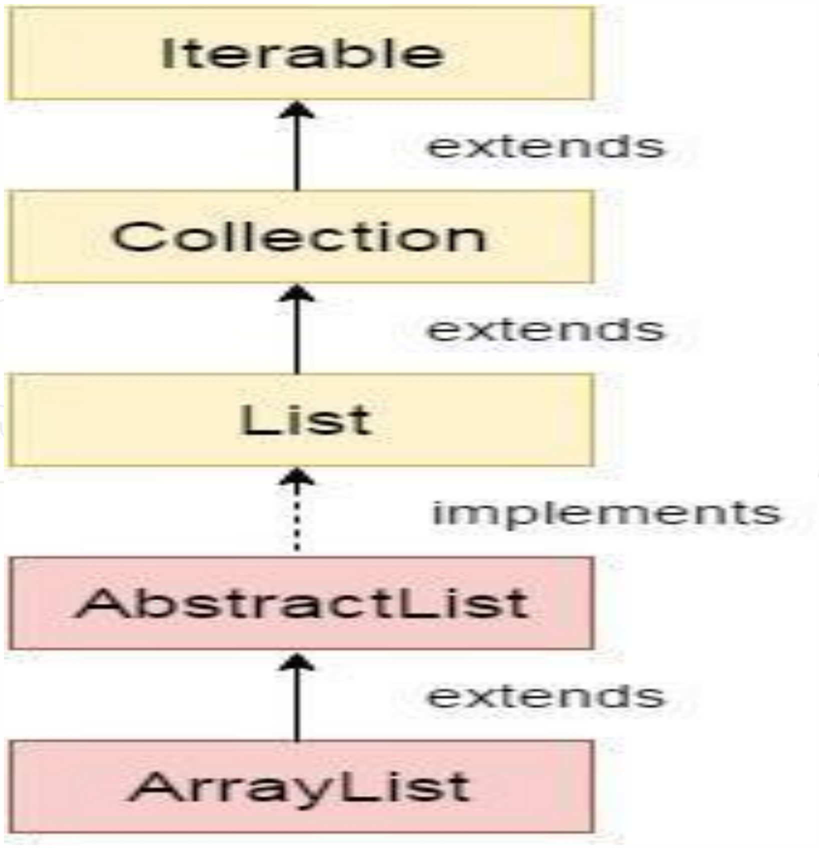
ArrayList is a resizable-array implementation of the List interface. Java ArrayList class uses a dynamic array for storing the elements. It inherits AbstractList class and implements List interface.

The important points about Java ArrayList class are:

O Java ArrayList class can contain duplicate elements.
O Java ArrayList class maintains insertion order.
O Java ArrayList class is non synchronized.
O Java ArrayList allows random access because array works at the index basis.
O In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurred if any element is removed from the array list.

Hierarchy of ArrayList class:

As shown in the below diagram, Java ArrayList class extends AbstractList class which implements List interface. The List interface extends Collection and Iterable interfaces in hierarchical order.


Iterable
extends
Collection
extends
List
implements
AbstractList
extends
ArrayList

Drawback of Array:

The issue with arrays is that they are of fixed length so if it is full we cannot add any more elements to it, likewise if number of elements gets removed from it the memory consumption would be the same as it doesn‘t shrink.

Advantage of ArrayList:

On the other ArrayList can dynamically grow and shrink after addition and removal of elements. Apart from these benefits, ArrayList class enables us to use predefined methods of it which makes our task easy.
Let‘s see the ArrayList example first then we will discuss it‘s methods and their usage.

ArrayList Example in Java:

```
importjava.util.*;

publicclassArrayListExample
{
publicstaticvoid main(Stringargs[])
{
ArrayList<String>obj=newArrayList<String>();
obj.add("Ajeet");
obj.add("Harry");
obj.add("Chaitanya");
obj.add("Steve");
obj.add("Anuj");
System.out.println("Currently the array list has following elements:"+obj);
obj.add(0,"Rahul");
obj.add(1,"Justin");
obj.remove("Chaitanya");
obj.remove("Harry");
System.out.println("Current array list is:"+obj); obj.remove(1);
System.out.println("Current array list is:"+obj);
}
}
```

Output:

```
Currently the array list has following elements:[ Ajeet, Harry,Chaitanya,Steve, Anuj]
Current array list is:[ Rahul,Justin, Ajeet,Steve, Anuj]
Current array list is:[Rahul,Ajeet,Steve,Anuj]
```

Methods of ArrayList class:

In the above example we have used methods such as add and remove. However there are number of methods available which can be used directly using object of ArrayList class. Let‘s discuss few of the important methods.

1)add(Object o): This method adds an object o to the arraylist.
obj.add("hello");
This statement would add a string hello in the arraylist at last position.
2)add(int index, Object o): It adds the object o to the array list at the given
index. obj.add(2,"bye");
It will add the string bye to the 2nd index (3rd position as the array list starts with index 0) of array list.
3)remove(Object o): Removes the object o from the
ArrayList. obj.remove("Chaitanya");
This statement will remove the string —Chaitanya‖ from the ArrayList.
4)remove(int index): Removes element from a given
index. obj.remove(3);
It would remove the element of index 3 (4th element of the list – List starts with o).
5)set(int index, Object o): Used for updating an element. It replaces the element present at the specified index with the object o.
obj.set(2,"Tom");
It would replace the 3rd element (index =2 is 3rd element) with the value Tom.
6)int indexOf(Object o): Gives the index of the object o. If the element is not found in the list then this method returns the value -1.
int pos=obj.indexOf("Tom");
This would give the index (position) of the string Tom in the list.

7)Object get(int index): It returns the object of list which is present at the specified
index. String str=obj.get(2);
Function get would return the string stored at 3rd position (index 2) and would be assigned to the string ―str‖. We have stored the returned value in string variable because in our example we have defined the ArrayList is of String type. If you are having integer array list then the returned value should be stored in an integer variable.

8)int size(): It gives the size of the ArrayList – Number of elements of the
list.int numberofitems=obj.size();

9)boolean contains(Object o): It checks whether the given object o is present in the array list if its there then it returns true else it returns false.

```
obj.contains("Steve");
```

It would return true if the string ―Steve‖ is present in the list else we would get false.

10)clear(): It is used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList whose object is obj.

```
obj.clear();
```

Methods of Java ArrayList – summary:

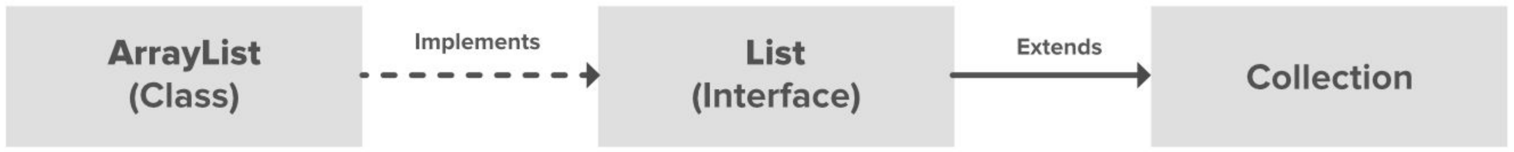
Method	Description
void a dd(int index, Object element)	It is use d to insert the specified element at the specified position index in a list.
boolean addAll(Collection c)	It is used to append all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.
void clear()	It is used to remove all of the elements from this list.
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.
Object[] toArray(Object[] a)	It is used to return an array containing all of the elements in this list in the correct order.

Methods of Java ArrayList – summary:

Method	Description
boolean add(Object o)	It is used to append the specified element to the end of a list.
booleanaddAll(int index,Collection c)	It is used to insert all of the elements in the specified collection into this list, starting at the specified position.
Object clone()	It is used to return a shallow copy of an ArrayList.
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.
void trimToSize()	It is used to trim the capacity of this ArrayList instance to be the list's current size.

ArrayList in Java:

ArrayList is a part of collection framework and is present in java.util package. It provides us with dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed. This class is found in java.util package.



Example: The following implementation demonstrates how to create and use an ArrayList.

```
import java.io.*;
import java.util.*;
class ArrayListExample {
    public static void main(String[] args)
    {
        // Size of the
        // ArrayList
        int n = 5;
          // Declaring the ArrayList with
        // initial size n
ArrayList<Integer> arrli
            = new ArrayList<Integer>(n);
          // Appending new elements at
        // the end of the list
        for (int i = 1; i <= n; i++)
            arrli.add(i);
          // Printing elements
        System.out.println(arrli);
          // Remove element at index 3
        arrli.remove(3);
          // Displaying the ArrayList
        // after deletion
        System.out.println(arrli);
          // Printing elements one by one
        for (int i = 0; i < arrli.size(); i++)
            System.out.print(arrli.get(i) + " ");
    }
}
```

Output:

```
[1, 2, 3, 4, 5]
[1, 2, 3, 5]
1 2 3 5
```

Performing Various Operations on ArrayList:

1. **Adding Elements:** In order to add an element to an ArrayList, we can use the add() method. This method is overloaded to perform multiple operations based on different parameters. They are:

add(Object): This method is used to add an element at the end of the ArrayList.
add(int index, Object): This method is used to add an element at a specific index in the ArrayList.

```
import java.util.*;
public class GFG {
      public static void main(String args[])
    {
        ArrayList<String> al = new ArrayList<>();
          al.add("Sairam");
         al.add(“Sairam");
         al.add(1, "For");
         System.out.println(al);
    }
}
```

Output:

```
[Sairam, For, Sairam]
```

2. Changing Elements: After adding the elements, if we wish to change the element, it can be done using the set() method. Since an ArrayList is indexed, the element which we wish to change is referenced by the index of the element. Therefore, this method takes an index and the updated element which needs to be inserted at that index.

```
import java.util.*;
 public class GFG {
      public static void main(String args[])
    {
        ArrayList<String> al = new ArrayList<>();
        al.add("Sairam");
        al.add("Sairam");
        al.add(1, "Sairam");
        System.out.println("Initial ArrayList " + al);
        al.set(1, "For");
        System.out.println("Updated ArrayList " + al);
    }
}
```

Output:

```
Initial ArrayList [Sairam, Sairam, Sairam]
Updated ArrayList [Sairam, For, Sairam]
```

3. Removing Elements: In order to remove an element from an ArrayList, we can use the remove() method. This method is overloaded to perform multiple operations based on different parameters. They are:

remove(Object): This method is used to simply remove an object from the ArrayList. If there are multiple such objects, then the first occurrence of the object is removed.

remove(int index): Since an ArrayList is indexed, this method takes an integer value which simply removes the element present at that specific index in the ArrayList. After removing the element, all the elements are moved to the left to fill the space and the indices of the objects are updated.

```
import java.util.*;
public class GFG {
      public static void main(String args[])
    {
       ArrayList<String> al = new ArrayList<>();
        al.add("Sairam");
        al.add("Sairam");
        al.add(1, "For");
        System.out.println("Initial ArrayList " + al);
        al.remove(1);
        System.out.println("After the Index Removal " + al);
        al.remove("Sairam");
        System.out.println("After the Object Removal " + al);
    }
}
```

Output:

```
Initial ArrayList [Sairam, For, Sairam]
After the Index Removal [Sairam, Sairam]
After the Object Removal [Sairam]
```

4. Iterating the ArrayList: There are multiple ways to iterate through the ArrayList. The most famous ways are by using the basic for loop in combination with a get() method to get the element at a specific index and the advanced for loop.

```
import java.util.*;
 public class GFG {
    public static void main(String args[])
    {
        ArrayList<String> al = new ArrayList<>();
        al.add("Sairam");
        al.add("Sairam");
        al.add(1, "For");
        // Using the Get method and the
        // for loop
        for (int i = 0; i < al.size(); i++) {

            System.out.print(al.get(i) + " ");
        }
         System.out.println();
          // Using the for each loop
        for (String str : al)
        System.out.print(str + " ");
    }
}
```

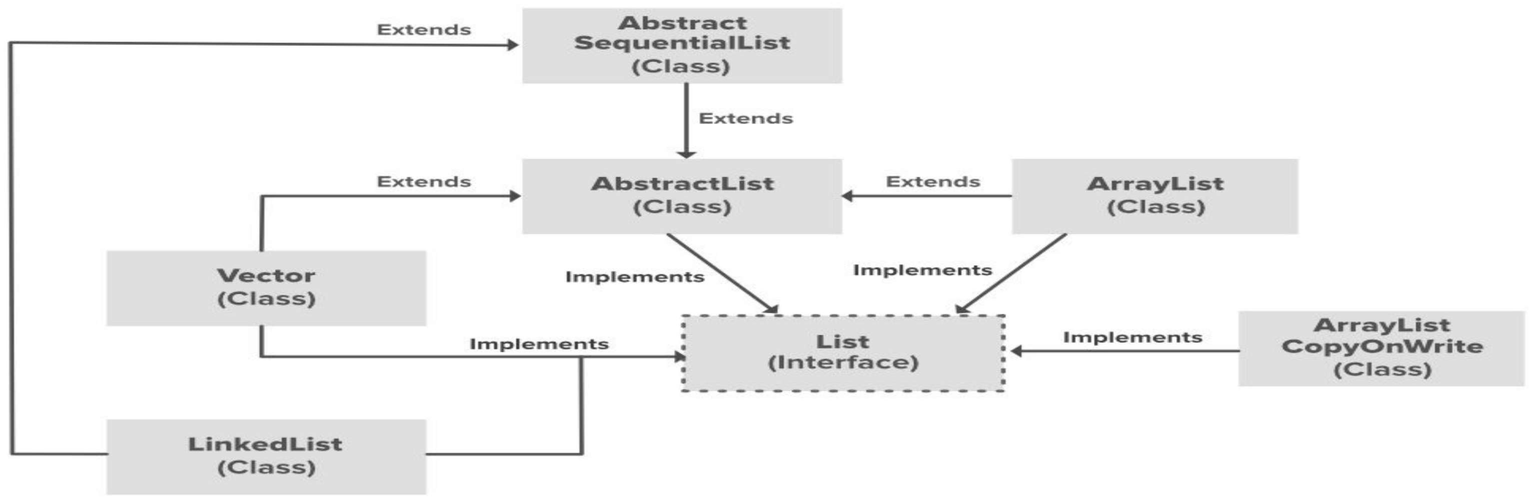
<u>Output:</u>

```
Sairam For Sairam
Sairam For Sairam
```

Important Features:

•ArrayList inherits AbstractList class and implements List interface.
•ArrayList is initialized by the size. However, the size is increased automatically if the collection grows or shrinks if the objects are removed from the collection.
•Java ArrayList allows us to randomly access the list.
•ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases.
•ArrayList in Java can be seen as a vector in C++.
•ArrayList is not Synchronized. Its equivalent synchronized class in Java is Vector.

Let's understand the Java ArrayList in depth. Look at the below image:



In the above illustration, **AbstractList, CopyOnWriteArrayList and the AbstractSequentialList** are the classes which implement the list interface. A separate functionality is implemented in each of the mentioned classes. They are:
AbstractList: This class is used to implement an unmodifiable list, for which one needs to only extend this AbstractList Class and implement only the get() and the size() methods.
CopyOnWriteArrayList: This class implements the list interface. It is an enhanced version of **ArrayList** in which all the modifications(add, set, remove, etc.) are implemented by making a fresh copy of the list.
AbstractSequentialList: This class implements the **Collection interface** and the AbstractCollection class. This class is used to implement an unmodifiable list, for which one needs to only extend this AbstractList Class and implement only the get() and the size() methods.

How ArrayList work internally?

Since ArrayList is a dynamic array and we do not have to specify the size while creating it, the size of the array automatically increases when we dynamically add and remove items. Though the actual library implementation may be more complex, the following is a very basic idea explaining the working of the array when the array becomes full and if we try to add an item:

- Creates a bigger sized memory on heap memory (for example memory of double size).
- Copies the current memory elements to the new memory.
- New item is added now as there is bigger memory available now.
- Delete the old memory.

Constructors in the ArrayList:

In order to create an ArrayList, we need to create an object of the ArrayList class. The ArrayList class consists various constructors which allows the possible creation of the array list. The following are the constructors available in this class:

1.ArrayList(): This constructor is used to build an empty array list. If we wish to create an empty ArrayList with the name **arr**, then, it can be created as:

```
ArrayList arr = new ArrayList();
```

2.ArrayList(Collection c): This constructor is used to build an array list initialized with the elements from the collection c. Suppose, we wish to create an arraylist arr which contains the elements present in the collection c, then, it can be created as:

```
ArrayList arr = new ArrayList(c);
```

3.ArrayList(int capacity): This constructor is used to build an array list with initial capacity being specified. Suppose we wish to create an ArrayList with the initial size being N, then, it can be created as:

```
ArrayList arr = new ArrayList(N);
```

Note: You can also create a generic ArrayList:

```
// Creating generic integer ArrayList
ArrayList<Integer> arrli = new ArrayList<Integer>();
```