**/***
**ArrayList and its methods:**

**--> Java ArrayList class uses a dynamic array for storing the elements.**
**--> It is like an array, but there is no size limit. We can add or remove elements anytime.**
**--> So, it is much more flexible than the traditional array. It is found in the java.util package.**
**--> It implements the List interface, so we can use all the methods of the List interface here.**

**Important Points:**
**--> Java ArrayList class can contain duplicate elements.**
**--> Java ArrayList class maintains insertion order.**
**--> Java ArrayList class is non synchronized.**
**--> Java ArrayList allows random access because the array works on an index basis.**
**--> In ArrayList, manipulation is a little bit slower than the LinkedList in Java because a lot of shifting needs to occur if any element is removed from the array list.**
**--> We can not create an array list of the primitive types, such as int, float, char, etc.**
**--> It is required to use the required wrapper class in such cases.**
**Ex:**
**ArrayList<int> al = ArrayList<int>(); // does not work**
**ArrayList<Integer> al = new ArrayList<Integer>(); // works fine**
***/**

```
// import statement is mandatory
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
public class A3
{
   public static void main(String[] args)
   {
      ArrayList a1=new ArrayList();
      // It is used to build an empty array list for Hetrogeneous data
      //with initial capacity "10" if Array list reaches its maximum
      //capacity then a new Arraylist object will be created with
      //New capacity=(current capacity *3/2)+1
      a1.add("Vikram");//string
      a1.add(10);//int
      a1.add('B');//char
      a1.add(10.5);//double
      a1.add(11.5f);//float
      a1.add(25l);//long
      a1.add(true);//boolean
      a1.add(null);//null
      System.out.println(a1);//[Vikram, 10, B, 10.5, 11.5, 25, true, null]

       // ArrayList object creation and Constructors
```

**//ArrayList constructor (1)**

**ArrayList<Integer> al1 = new ArrayList<>(); // It is used to build an empty array list for specified Homogeneous data.**

**// 1. boolean add(E e) - It is used to append the specified element at the end of a list.**

```
al1.add(10);
al1.add(20);
al1.add(30);
al1.add(10);
al1.add(40);
al1.add(50);
System.out.println(al1);//[10, 20, 30, 10, 40, 50]

System.out.println("\n----------*****----------");
```

**// 2. void add(int index, E element) - It is used to insert the specified element at the specified position in a list.**

```
al1.add(1,15); // it will add element 15 at position 1
System.out.println("Array list al1::"+al1);//Array list al1::[10, 15, 20, 30, 10, 40, 50]
```

**//ArrayList constructor (2)**

**// It is used to build an array list that is initialized with the elements of the collection c i.e., ArrayList.**

```
ArrayList<Integer> al2 = new ArrayList<>(al1);
System.out.println("Array list al2::"+al2);//Array list al2::[10, 15, 20, 30, 10, 40, 50]
```

**// It is used to build an array list that has the specified initial capacity.**
**//note*:-we can specify the initial capacity but cannot see the capaity**
**//because capacity is growable**

```
ArrayList<Integer> al3 = new ArrayList<>(5);
```

**// 3. void clear() - It is used to remove all the elements from this list.**

```
al2.clear();
System.out.println("Array list al2::"+al2);//Array list al2::[]
```

**// 4. void ensureCapacity(int requiredCapacity) -**
**//It is used to enhance the capacity of an ArrayList instance.**
**//note*:-we can specify the initial capacity but cannot see the capaity**
**//because capacity is growable**

```
al1.ensureCapacity(12);
```

**// 5. E get(int index) - It is used to fetch the element from the particular position of the list.**

System.out.println("ArrayList al1:"+al1**);//ArrayList al1:[10, 15, 20, 30, 10, 40, 50]**
System.out.println(al1.get(3));**//30**

**// 6. E set(int index, E element) - It is used to replace the specified element in the list, present at the specified position.and this method returns replaced value.**

System.out.println("ArrayList al1:"+al1);**//ArrayList al1:[10, 15, 20, 30, 10, 40, 50]**
System.out.println(al1.set(1,78));**//15 returning the replaced value**
System.out.println("ArrayList al1:"+al1);**//ArrayList al1:[10, 15, 20, 30, 10, 40, 50]**
//System.out.println(al1.set(10,78)); **// IndexOutOfBoundException**

**// 7. boolean isEmpty() - It returns true if the list is empty, otherwise false.**

System.out.println("ArrayList al1:"+al1);**//ArrayList al1:[10, 78, 20, 30, 10, 40, 50]**
System.out.println(al1.isEmpty()); **// false**
System.out.println("ArrayList al2:"+al2);**//ArrayList al2:[]**
System.out.println(al2.isEmpty()); **// true**

**//8. 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.**

System.out.println("ArrayList al1:"+al1); **//ArrayList al1:[10, 78, 20, 30, 10, 40, 50]**
System.out.println(al1.indexOf(10)); **//0 returns first occurance**
System.out.println(al1.indexOf(50)); **//6 returns first occurance**
System.out.println(al1.indexOf(60));**//-1 List does not contain this element.**

**//9. 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.**

System.out.println("ArrayList al1:"+al1); **//ArrayList al1:[10, 78, 20, 30, 10, 40, 50]**
System.out.println(al1.lastIndexOf(10)); **//4 returns last occurance**
System.out.println(al1.lastIndexOf(70)); **//-1 List does not contain this element.**

**//10. public int remove(int indexposition):**
**//It is used to delete an element from the collection.**
**//In argument pass index position that you want to remove.**
**//It returns removed value.**

System.out.println("ArrayList al1:"+al1); **//ArrayList al1:[10, 78, 20, 30, 10, 40, 50]**
System.out.println(al1.remove(2)); **//20 returns removed value**
System.out.println("ArrayList al1:"+al1); **//ArrayList al1:[10, 78, 30, 10, 40, 50]**
System.out.println(al1.remove(4)); **//40 returns removed value**

```
        System.out.println("ArrayList al1:"+al1);//ArrayList al1:[10, 78, 30, 10, 50]

        //System.out.println(al1.remove(10));
       // R.E: IndexOutOfBoundsException, as entered position is not available in ArrayList

        //11. void sort(Comparator<? super E> c)
        //It is used to sort the elements of the list on the basis of the specified comparator.

        System.out.println("ArrayList al1:"+al1); //ArrayList al1:[10, 78, 30, 10, 50]

        // sort the ArrayList in ascending order
        al1.sort(Comparator.naturalOrder());
    System.out.println("Sorted ArrayList al1:"+al1);//Sorted ArrayList al1:[10, 10, 30, 50, 78]

        // sort the ArrayList in reverse order
        al1.sort(Comparator.reverseOrder());
  System.out.println("reverse ArrayList al1:"+al1); //reverse ArrayList al1:[78, 50, 30, 10, 10]

        // 12. int size() - It is used to return the number of elements present in the list.

        System.out.println(al1.size());//5

        // 13. Add Multiple element with Arrays.asList() in Constructor

      ArrayList<String> al4 = new ArrayList<>(Arrays.asList("AB","CD","EG","YZ"));

        // Use of for each loop to print elements of al4

        for (String s : al4)
        {
           System.out.println(s);
        }
    /*  AB
        CD
        EG
        YZ  */
    }
}
```