Arraylist → dynamic array, generic

Syntax

Arraylist <Data type> arr = new Arraylist <> ();

Integer, Boolean, String, Double, character ... etc...

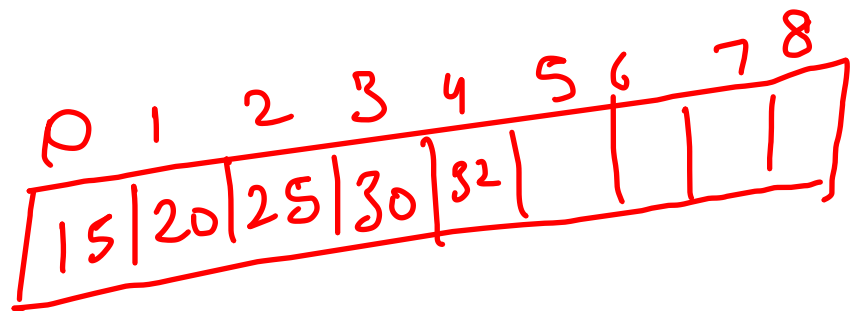
.size() → will show 0 initially.

size & capacity

Capacity → max data stored // By default capacity 10

size → how much stored till now by user.

Insertion in Arraylist



→ .add(val);

To accus a particular elem present at particular index

→ .get(indese);

Add a elem at particular index

→ • add(index, element);

element will get added & all other values will shift by one.

Print all the elements of Array list.

```
for(int i=0; i < list1.size(); i++){
    Syso( list1.get(i) + " " );
```

Remove the element.

→ • remove("index");

change or replace the values.

→ • set("index, element");

Arraylist can also be out of bound, if index out of bound.

```java
 ArrayList <Integer> list1 = new ArrayList<>();
// System.out.println(list1.size());
 list1.add(15);
 list1.add(20);
 list1.add(25);
  //System.out.println(list1.size());
  list1.add(2, 50);
 //  System.out.println(list1.get(2));
 //  System.out.println(list1.get(3));
 list1.remove(1);
 list1.set(1,100);

  for(int i = 0; i < list1.size(); i++){
      System.out.print(list1.get(i) + " ");
  }
  // index out of bound
  System.out.println(list1.get(5));
```

for-each loop.

for-each loop doesn't go to the index of array list, directly access elements.

```
for(int i : list1){
sysо(i + "  ");
}
```

for loop

int i → 0 1 2 3

15	20	50	25

↑

for (elements) each

→ quick traversal

→ only for traversal

→ not for changing or adding values.

→ used with arrays also.

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();

ArrayList<Integer> arr = new ArrayList<>();
for(int i = 0; i < n; i++){
    arr.add(s.nextInt());
}

// reverse print using for loop
for(int i = arr.size() - 1; i >= 0; i--){
    System.out.print(arr.get(i) + " ");
}
System.out.println();

// reverse arraylist
Collections.reverse(arr);

// reverse print using for each
    for(int elem : arr){
        System.out.print(elem + " ");
    }
      System.out.println();
}
```

TC → O(1) → O(n)

SC → O(1)

```java
 Scanner sc = new Scanner(System.in);

int n = sc.nextInt();
int[] a = new int[n];
for(int i = 0; i < n; i++)
    a[i] = sc.nextInt();

int m = sc.nextInt();
int[] b = new int[m];
for(int i = 0; i < m; i++)
    b[i] = sc.nextInt();

TreeSet<Integer> merge = new TreeSet<>();

for(int i : a){
    merge.add(i);
}
for(int i : b){
    merge.add(i);
}

ArrayList<Integer> list = new ArrayList<>(merge);

for(int i : list){
    System.out.print(i + " ");
}

}
```

TC → O(log n)

TreeSet ← Unique, Sorted, TC

→ 1 3 / 2 4 8 → 1 2 3 4 7 8

```java
public static int single(int arr[]){
    int left = 0;
    int right = arr.length - 1;

    while(left < right){
        int mid = (left+right)/2;

        if(mid % 2 == 1) mid--;

        if(arr[mid] == arr[mid+1]){
            left = mid + 2;

        }
        else{
            right = mid;
        }
    }
    return arr[left];
}
```