Page No.
Date :

* Intro To Array & Arraylist *

* Why do we need Arrays ?

⇒ It was simple, when we had to store just five integer numbers and ~~now~~ now let's assume we have to store 5000 integer numbers. ~~It~~ is possible to use 5000 variable ? [No]

To handle these situations, in almost programming language we have a concept called Array

Array is a data structure use to store a collection of data.

Syntax of Array -

datatype [ ] variable-name = new datatype(size);

Eg :- we want to store roll numbers :

int [ ] rollnos = new int[5]

OR

int [ ] rollnos = {51, 82, 13, 15, 16}

*Note :-

- The datatype basically represents what is the data stored inside the array
- All the type of the data in the array should be same

* new Keyword is used to create an object. It will create an object in heap memory of array size size 5.

Page No.
Date :

⇒ Internal working of an array

```
int[] rollnos;  // declaration of array
```
↳ rollnos are getting defined in stack.

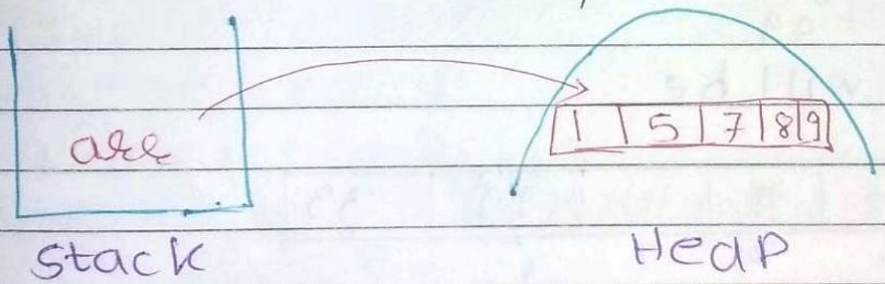
```
rollnos = new int[5]; // initialization
```
↳ actual memory allocation happens here. Here, object is being created in heap memory

declaration of array → compile time

initialization → run time.

```
int[] arr = new int[5];
```

⇒ This above concept is known as Dynamic memory Allocation which means at runtime OR at execution time memory is allocated.



⇒ Internal Representation of Array:-

- Internally in Java, memory allocation totally depends on JVM whether it be continuous or not!

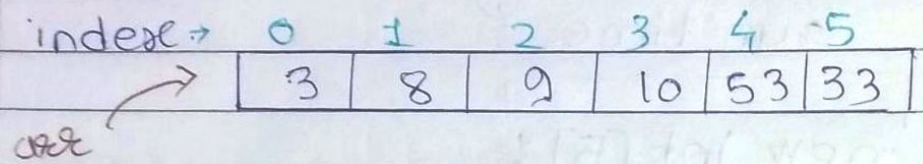
Reason 1: objects are stored in heap memory

Page No.
Date :

Reason 2: In JLS (Java Language specification) it is mentioned that heap objects are not continuous.

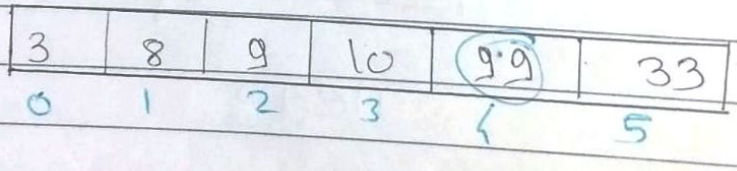
Reason 3: Dynamic memory allocation. Hence, array objects in Java may not be continous (depends on JVM)

* Index of an array:



arr[0] = 3 arr[2] = 9 arr[4] = 53
arr[1] = 8 arr[3] = 10 arr[5] = 33

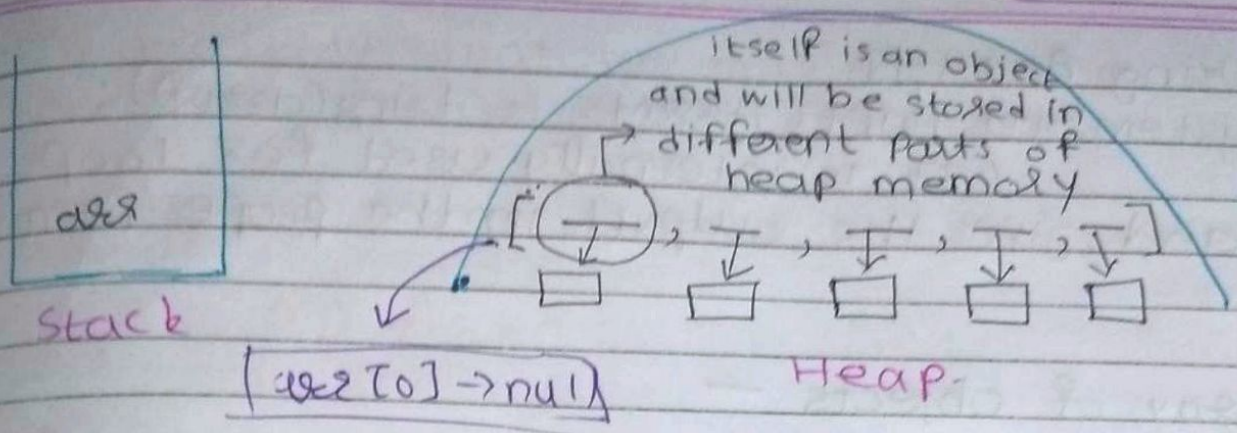
Suppose to change the value of certain index:
arr[4] = 99
New array will be.



⇒ If we don't provide values in the array, internally by default it stores [0,0,0,0,0] for above size of array

String arr[] = new String[4];

Primitives are stored in the stack memory only



* Primitives (int, char etc) are stored in stack
* All other objects are stored in Heap memory.

- Input an array :-
  1. Input using for loops.

```
for(int i=0; i<arr.length; i++){
        arr[i] = in.nextInt();
}
```

~~- Inp~~

- Printing an array
  - Using for loops.

```
for(int i=0; i<arr.length; i++){
    ~~arr[i] =~~
    System.out.print(arr[i] + " ");
}
```

  - Using for each loops.

```
for(int j : arr){
    System.out.print(j + " ");
}
```

// Here j represents the elements of an array

- Using Arrays.

```
System.out.Print(Arrays.toString(arr));
        // It is internally used for loop
and gives the output in the proper format
```

* Array of objects —

```
String[] str = new String[4];
for(int i=0 ; i<str.length; i++){
    str[i] = in.next();
}
System.out.Print(Arrays.tostring(str));
```

* Passing in functions —

```
{
int[] nums = {3,4,5,12};
System.out.Print(Arrays.toString(nums));
change(nums);
System.out.Print(Arrays.tostring(nums));
}

static void change(int[] arr){
    arr[0] = 99;
}
```

output :-

```
[3,4,5,12]
[99,4,5,12]
```

* In an array since we can change the objects, hence they are mutable. (changeble)

* Strings are immutable (not changeble)

2D Arrays

3.

3 | 1 2 3 |
  | 4 5 6 |
  | 7 8 9 |

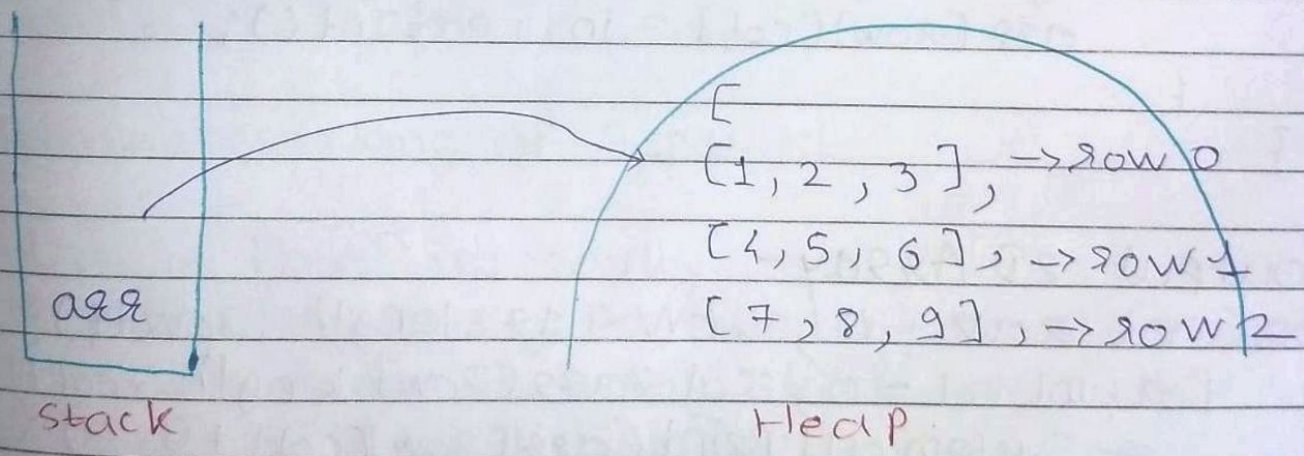
row column

⇒ int [][] arr = new int [size] [ ]

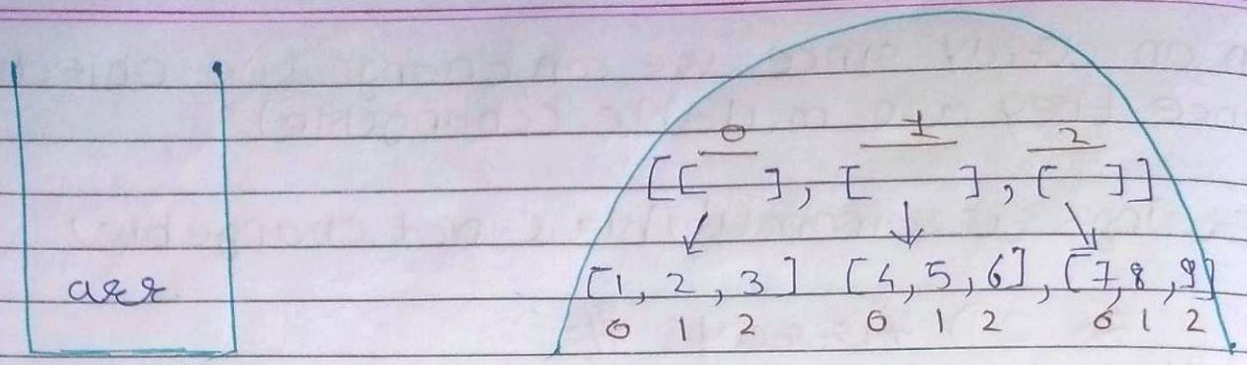
↑ mandatory to give size of row.

OR

```
int[][] arr = {
                {1,2,3},
                {4,5,6},
                {7,8,9}
              }
```



Imagine this as an arrays of arrays.



arr[0] = [1, 2, 3]
arr[0][2] = 3.

* Input 2d Array —

```
int[][] arr = new int[3][2];

for(int row=0; row<arr.length; row++){
    // for each col in every row
    for(int col=0; col<arr[row].length; col++){
        arr[row][col] = in.nextInt();
    }
}
```

* output 2D Array -

```
for(int row=0; row<arr.length; row++){
    for(int col=0; col<arr[row].length; col++){
        System.out.Print(arr[row][col] + " ");
    }
    System.out.Println();
}
```

OR

```
for (int row=0; row <arr.length ; row++){
  System.out.Print (Arrays.toString (arr[row]));
}
```

OR

```
for(int[] ar : arr){
   System.out.Print(Arrays.toString(ar));
}
```

* ArraysList

'Arraylist is a part of collection framework and is present in java.util.package. It provides us with dynamic arrays in Java. It is slower than standard arrays.'

Syntax:-

```
ArrayList <Integer> List = new Arraylist<>();
```

(add wrapper here.

*Internal working of Arraylist

- Size is fixed internally
- Suppose Arraylist gets filled by some amount
  a) it will make an arraylist of say double the size of arraylist initially
  b) old elements are copied in the new arraylist
  c) old ones are deleted.

Page No.
Date :

* Muti Dimensional Arraylist.

- Syntax :-

```
Arraylist < Arraylist <Integer>> List = new
                                Arraylist <> ();
```

```
// Initialization
 for (int i=0 ; i < 3; i++)
 {
    List.add (new nextInt
    List.add (new Arraylist <>());
 }

// Input
 for (int i=0; i<3 ; i++){
    For (int j=0 ; j<3; j++){
       list.get(i).add (in.nextInt());
    }
 }
```

* Linear Search in Java *

Page No.
Date :

Searching: It is a process of finding a given value position in a list of values.

* Linear / Sequential Search:

- It is a simple search algorithm
- In sequential search, we compare the target value with all other elements given in the list.

e.g. arr = [18, 12, 19, 77, 29, 50] (unsorted array)

0 1 2 3

start

target = 77

In above example, the target value is compared with all the elements in array in sequential / linear way.

Time Complexity:

→ Best Case: O(1) → constant

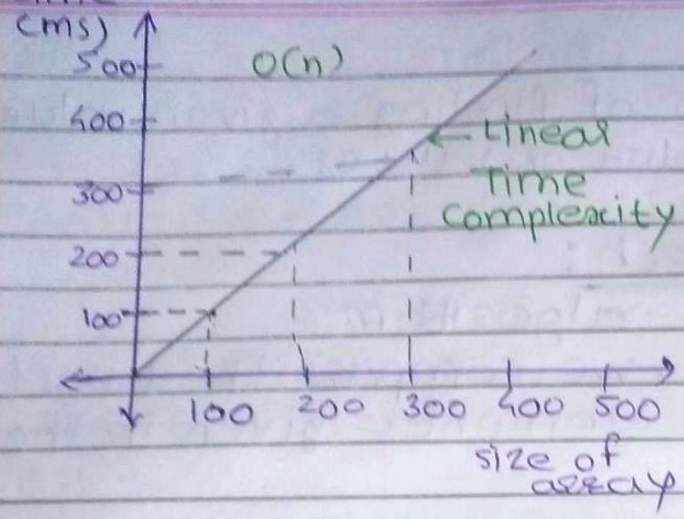
⇒ How many checks will the loop make in best case i.e. the element will be found at 0^{th} index i.e. only one comparison will be made for best case.

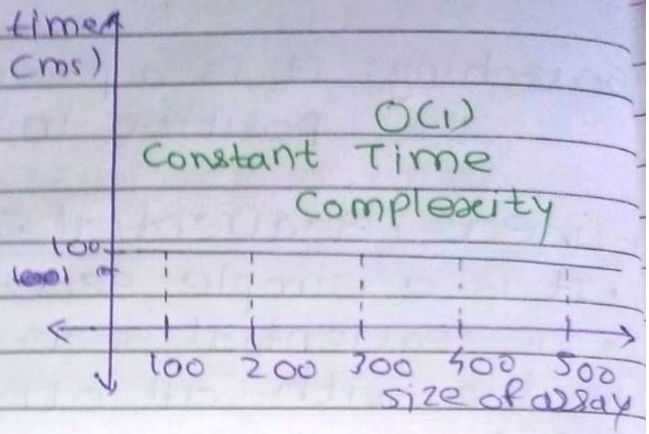
→ Worst Case: O(n)

⇒ worst case, here it will go through every element and then it says element not found.

⇒

size of array	No. of comparisons	time (ms)
100	100	100 ms
200	200	200 ms
n	n	





worst case

Best Case

Ex –

```
Pubic Static void main (string[] args){
 int [] arr = { 12, 13, 88, 67, 50};
 int target = 88;
 System.out.Print ln (linearSearch (arr, target));
}
Static int linearSearch (int[] arr, int target){
    if (arr.length ==0){
        return -1;
    }
    for (int i=0; i<arr.length; i++){
        int element = arr [i];
     if (element == target){
        return i;
      }
    }
     return -1;
},
```