04. ARRAYS AND ARRAYLISTS

① Arrays.

→ Java array is an object which contain elements of a similar data type.

→ The elements of an array are stored in a contiguous memory location.

→ Syntax:

or datatype [] var = {val1, val2, val3}

datatype [ ] variable name = new datatype [size];

(stack) (heap)

→ int [] rss // declaration of array. rss is getting defined in the stack

→ rss = new int [5]; // this is actual memory creation

// obj is being created in the memory (heap)

→ eg: int [] rollnos = new int [5]

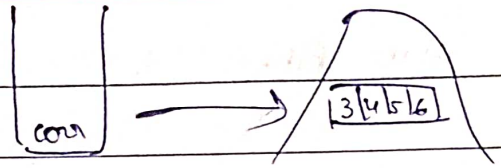
↓ ↓

represent the type of data stored in array

Store 5 things in the array (size)

compile time | run time

The above concept is known as Dynamic memory allocation which means at runtime or execution time, memory is allocated.



(2) Internal representation of Arrays

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

→ Reason1: Objects are stored in heap memory

→ Reason2: In JLS (Java language specification) it is mentioned that heap objects are not continuous

→ Reason3: Dynamic memory allocation. Hence arrays objects in Java may not be continuous (depends on JVM)

(3) Index of an array

index →	0	1	2	3
arr →	3	8	9	10

arr[0] = 3 arr[1] = 8

suppose we need to change value at index 2

arr[2] = 11

new array:

0	1	2	3
3	8	11	10

④ new Keyword.

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

It will create an obj in heap memory of array size 5

→ If we dont provide values in ~~arr~~ array internally by default it stores [0,0,0,0,0] for the above size of array.

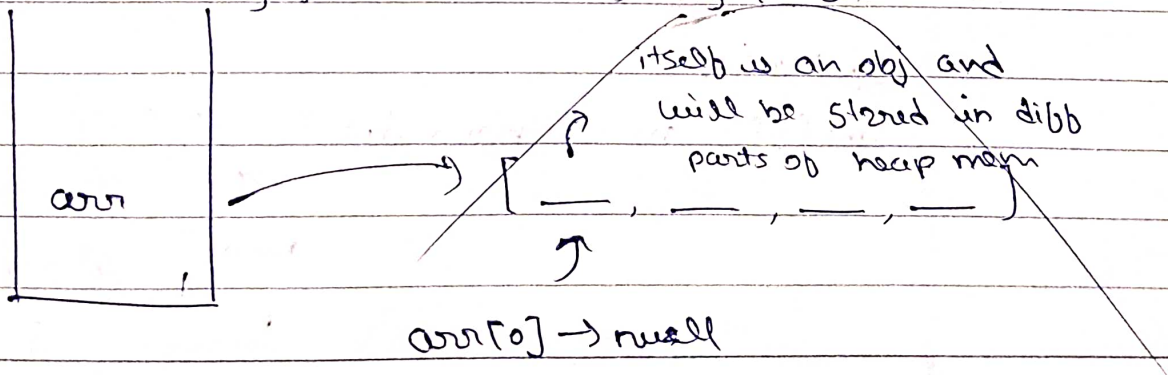
declare string array

```
String[] arr = new String[4];
```

Primitives (int, char, etc) are stored in stack
All other obj (String, array, hashmap) are stored in heap memory

29:44

```
String[] arr = new String[4];
```



⑤ take input

```
int[] arr = new int[5];
for(int i=0; i<arr.length; i++){
    arr[i] = sc.nextInt()
}
```

⑥ for each loop.

```
for (int num : arr) {        ①
    S.out (num + " ");       ②
}
```

① for every element in array print element

② here 'num' represents elements of the array.

Arrays.toString(array) → internally uses for loop and gives the output in proper format

In an array, since we can change the objects they are mutable

Strings are immutable

⑦ 2D array

→ Syntax:

```
int[][] arr = new int [size] [ ]
```

↑ row — mandatory to give size

↑ column — not mandatory

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

```
int [][] arr = new int [3][2];
S.out (arr.length)  // no of rows

                          ①
for (int row=0 ; row < arr.length ; row++){
    for (int col=0 ; col < arr[row].length ; col++ ){
                                          ②
        arr [row][col] = in.nextInt();
}}
```

① Here arr.length = 3 which is defined in int [3][2] . arr.length = no of rows

② Here this for loop will iterate inside the column of that row. The to

OR with toString()

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

OR with for each

```
for (int[] a : arr){
    Sout (Arrays.toString(a));
}
```

(8) Array List

→ It 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.

* Internal working of Arraylist.

→ Size is fixed internally

→ Suppose arraylist gets filled by some amt

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.

→ eg:

```
Arraylist <Integer> list = new Arraylist <>(10);
list.add(1);
list.add(2);
S.out (list).
```