

10/8/21

Introduction to Arrays & ArrayList in Java

Why do we need Arrays?

⇒ It was simple when we had to store just five integer numbers and now let's assume we have to store 5000 integer numbers. Is it possible to use 5000 variable? **[NO]**

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

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

⇒ Syntax of an Array:

datatype [] variable_name = new datatype[size];

eg: we want to store roll numbers:

int [] rollnos = new int[5] store 5 roll numbers

OR

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

represent the type of data stored in array.

All the type of data in array should be same!

⇒ Internal working of array:

int [] rollnos; // declaration of array

↳ rollnos are getting defined in stack

rollnos = new int[5]; // initialisation

↳ actual memory allocation happens here
Here, object is being created in heap memory.

declaration of array

compile time

int [] arr

datatype

ref var

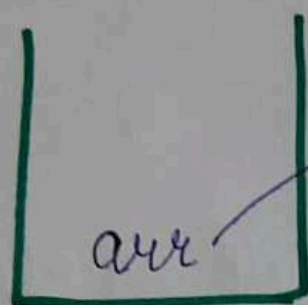
initialisation

runtime

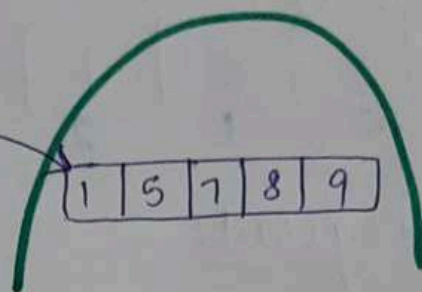
new int [5];

creating object in heap memory

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



Stack



Heap

⇒ Internal Representation of Array:

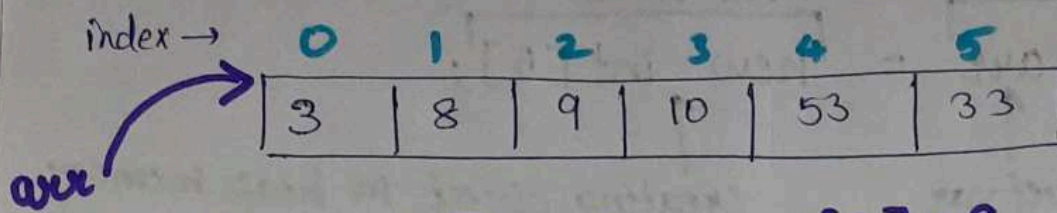
- Internally in Java, memory allocation totally depends on JVM whether it be continuous or not!

Reason 1: Objects are stored in heap memory.

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 continuous (depends on JVM)

⇒ Index of an array:



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

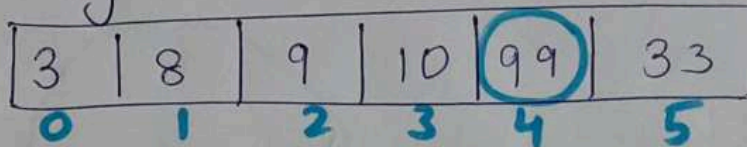
arr[2] = 9
arr[3] = 10

arr[4] = 53
arr[5] = 33

Suppose to change the value of certain index:

arr[4] = 99

New array will be:



⇒ new keyword:

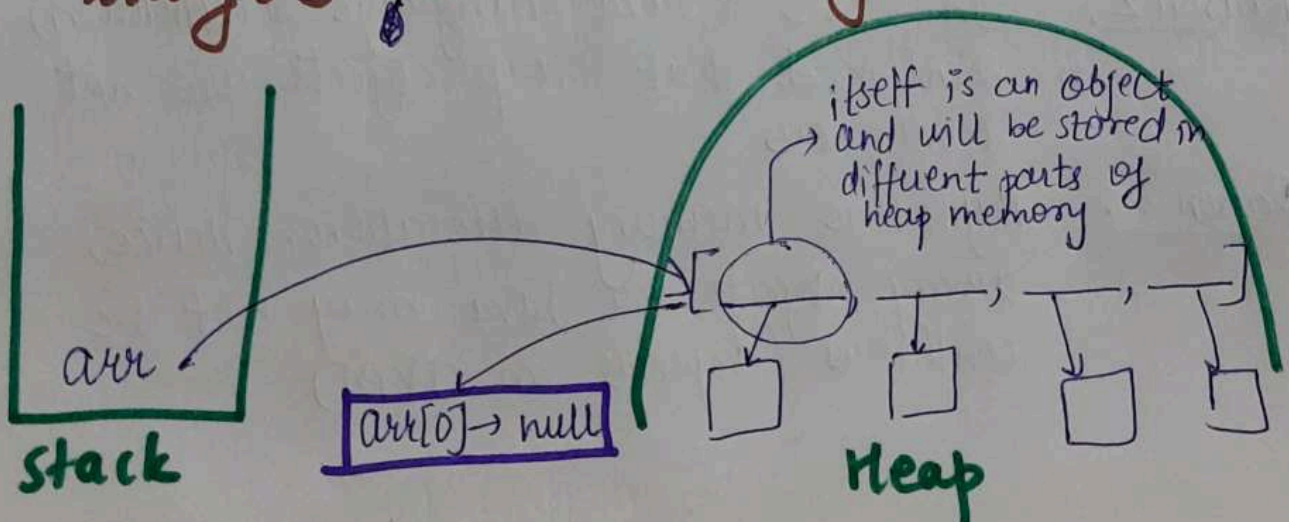
int[] arr = new int[5];

it will create an object in heap memory of array size 5.

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

Array of objects

String[] arr = new String[4];



★ primitive (int, char etc) are stored in stack.

★ All other objects are stored in heap memory.

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

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

★ strings are immutable. for security reasons

⇒ 2 D Array:

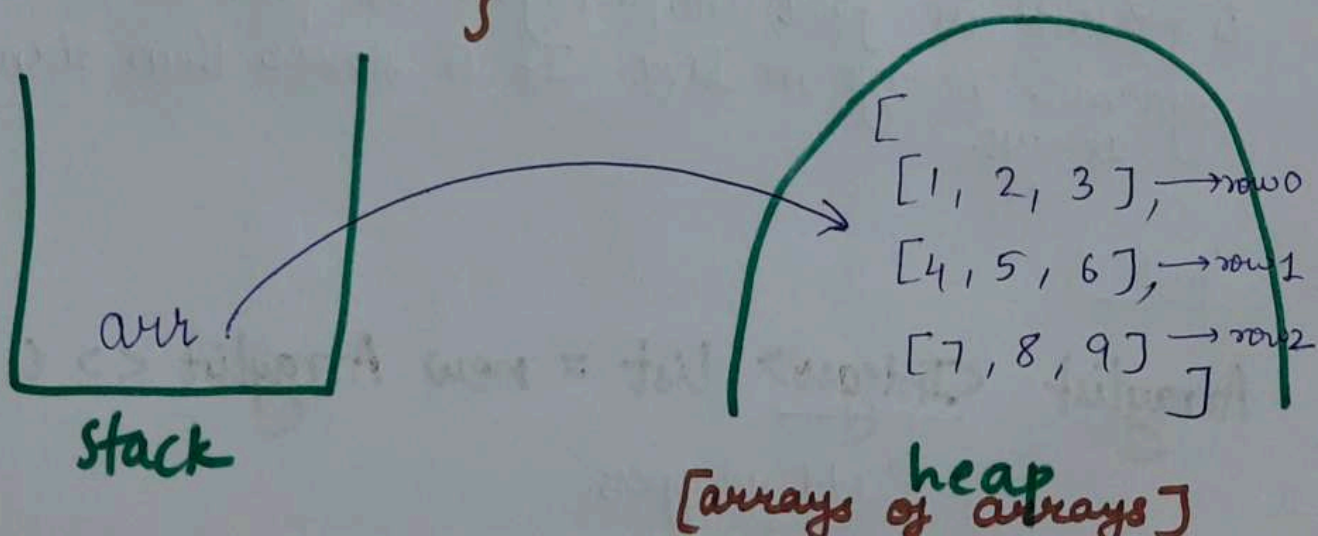
3

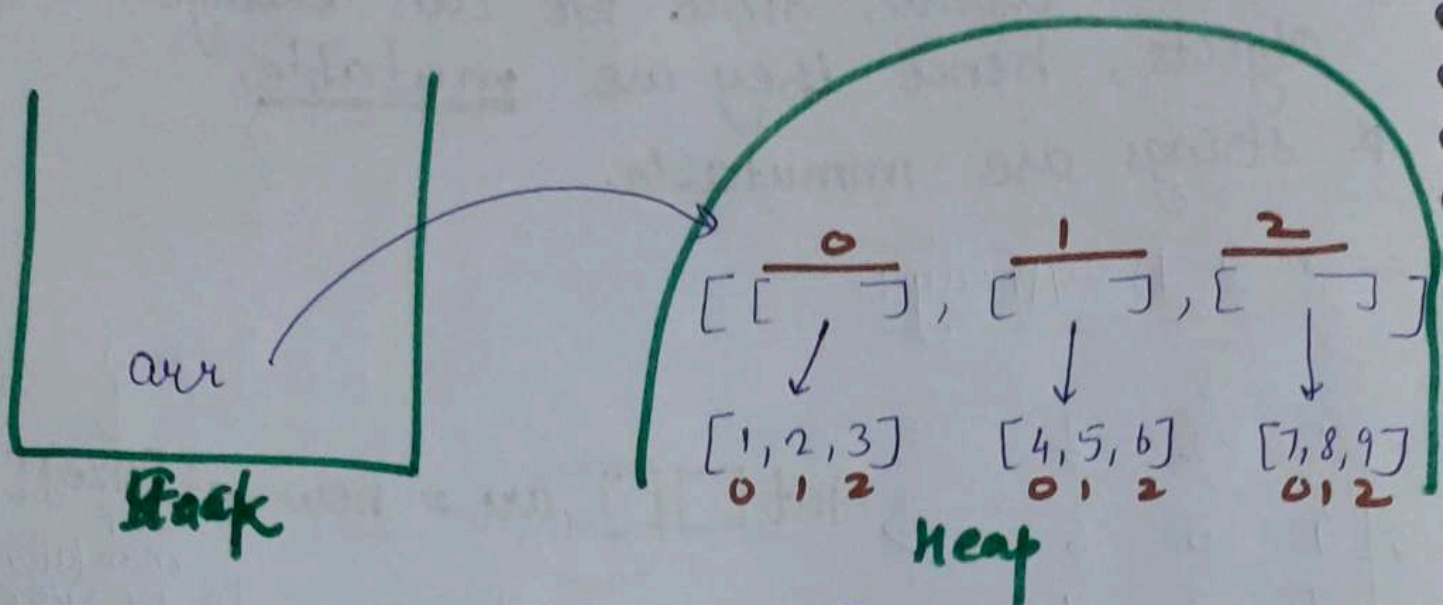
1	2	3
4	5	6
7	8	9

⇒ `int[][] arr = new int[size][]`

row ↓
column ↓
↑
mandatory to give size of row
not mandatory

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





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

⇒ ArrayLists:

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 wrappers.

⇒ 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.

`arr.length` --> gives the size of an array

'null' literal can be only assign to non-primitives .ex : strings
it can not be assign to primitives.