Introduction to Arrays and Arraylist

* Why we need Arrays ?

Consider, if we want to store a roll no., we can simply write like

```
int a = 20;
```

But, Imagine → If we have to store 10,000 roll nos.

⇒ Is it possible to write same code 10,000 times ?

Answer is : No

So, For this we use **Array** data structure.

* What is an Array ?

An Array is a group (or collection) of same data types.

* Syntax of an Array

```
datatype [] variable_name = new datatype [size];
```

Ex:→ Store 5 roll numbers:

```
int [] rollnos = new int [5]
```

OR

```
int [] rollnos = {5, 10, 11, 12, 15}
```

↓ represent the data type stored in array

↓ All the type of data in array should be same.

* Internal Working of an Array

- `int [] rollnos;` → declaration of an array
  Here, rollnos are getting defined in stack
- `rollnos = new int[5];` → initialisation
  Actual memory allocation happens here.
  Here, object is being created in heap memory.

* Dynamic Memory Allocation

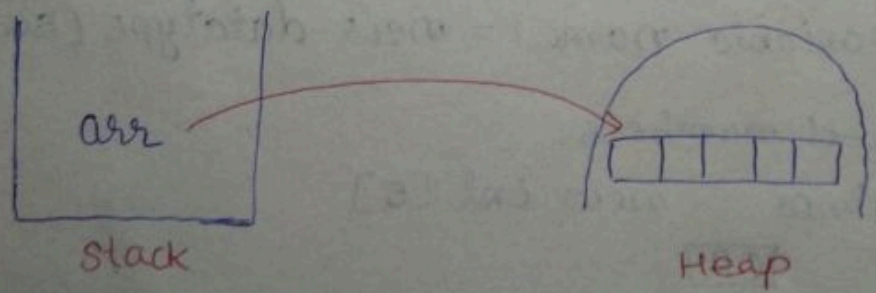
Declaration of array (At Compile time) ↓

Initialisation (At Runtime) ↓

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

- `int []` ↑ data type
- `arr,` ↑ reference variable
- `new int [5];` ↑ creating object in heap memory
  new → a keyword used to create object

* Internal Representation of Array –



Q Memory allocation in Java is Continuous or not ?

⇒ It may be continuous or not. It's totally depends on JVM

Reasons:

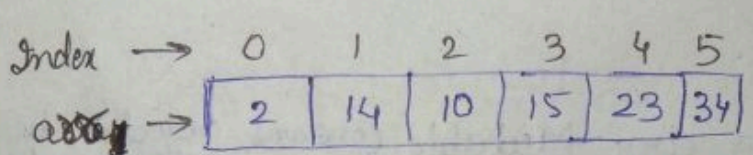
→ objects are stored in heap memory.

→ According to JLS (Java Language Specification)

it is mentioned that heap objects are not continuous.

→ Dynamic memory Allocation.

* Index of an Array



arr[0] = 2
arr[1] = 14
arr[2] = 10
arr[3] = 15
arr[4] = 23
arr[5] = 34

NOTES

⇒ If we don't assign values in the array, Internally, by default it store [0,0,0,0,0,0] like this for above arr.

⇒ i.e., for int → by default value is 0 (zero) for all elements

⇒ for String → by default value is null

* null ⇒ a literal used for reference

* Primitive (int, char, float, etc) are stored in Stack.

* Objects are stored in heap memory.

* Ways to print elements of an array -

1. Using for loop :

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

2. Using for-each loop

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

3. Using toString method of Arrays class

```
System.out.println(Arrays.toString(arr));
```

↓ Internally uses for loop

Best way

* NOTE :- In Java, Arrays are Mutable (means, object can be changed) and Strings are Immutable.

* ### 2D Array

arrays of arrays

row column

Syntax :- `int [][] arr = new int [size][ ]`

OR

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

1 2 3
4 5 6 ⇒
7 8 9

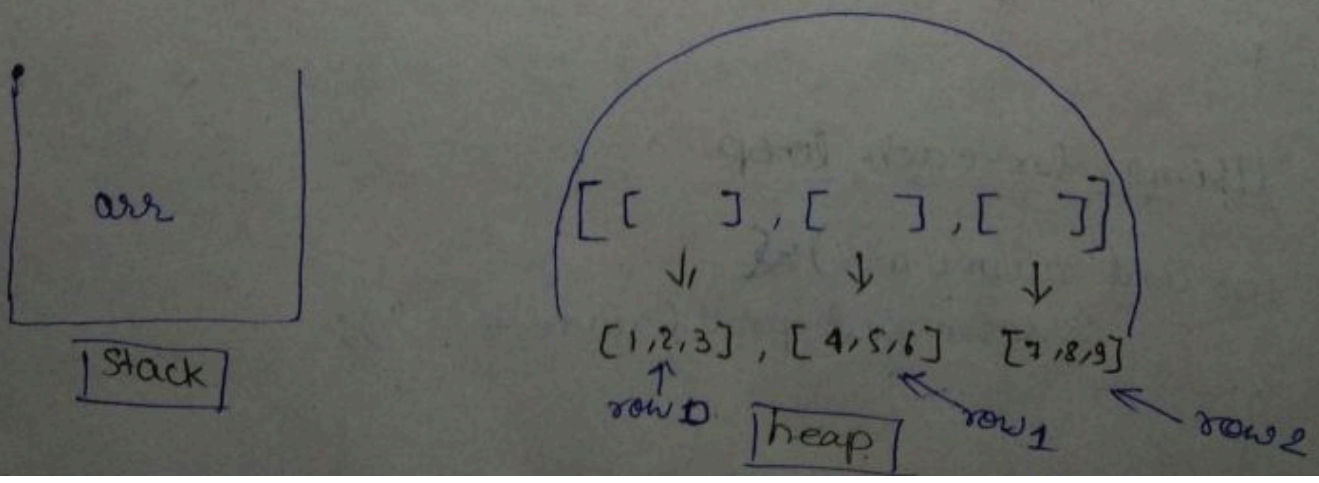
3×3

NOTE

~~Size~~

- It is mandatory to give size of row.
- Not mandatory to give size of column.

Representation



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

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

arr[1] = [4,5,6]

arr[1][0] = 4
arr[1][1] = 5
arr[1][2] = 6

* Why we need Arraylist ?

We have to give size while initialisation. But, what if we don't know the size of array? Then, we use Arraylist.

* Arraylist

→ It is a part of collection framework.
→ It is present in java.util.package.
→ It provides dynamic arrays.
→ It is slower than standard arrays.

Syntax

Arraylist <datatype> list = new Arraylist <> ( );

[like Integer (not int)]
i.e, add wrappers

* Internal Working of Arraylist

⇒ size is fixed internally.
⇒ If Arraylist gets filled by some amount then →
- It will make a new Arraylist of say, double the size of older arraylist.
- Old elements are copied in new Arraylist.
- Old ones are deleted.