Arrays and Arraylists

Array

↳ data structure used to store a collection of data

→ all elements in an array share the same data type.

Syntax of an array.

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

OR

```
datatype[] variable_name = { value1, value2, ..... valueN };
```

Internal working of an array.

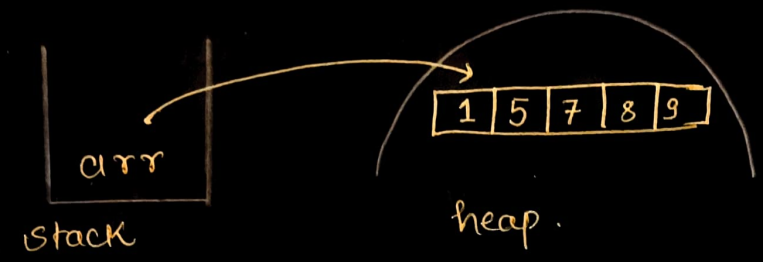
```
int [] rollno.s ;
// declaration of array
// roll no.s are defined into the stack.

rollnos = new int [5];
```

↳ actual memory allocation happen here.
↳ object is being created in heap memory.

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

- `int [] arr` → declaration of array → compile time
- `new int [5]` → initialisation and object creation in heap → run time.

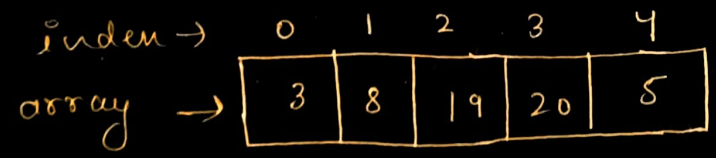
The concept of allocating memory during run time or execution time is called dynamic memory allocation.



* if reference variables have nothing to point to, they will return 'null' when called.

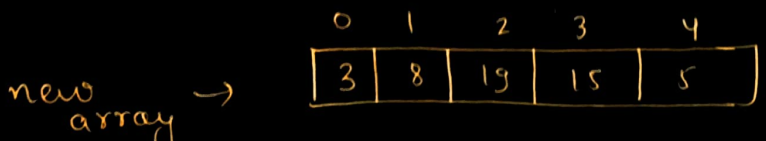
1) Array objects are in heap.
2) Heap objects are not continuous.
   array objects in Java may not be continuous. (not sure)
3) Dynamic memory allocation.

Index of an array :



if we need to change value at particular index.

```
arr[4] = 15;
```



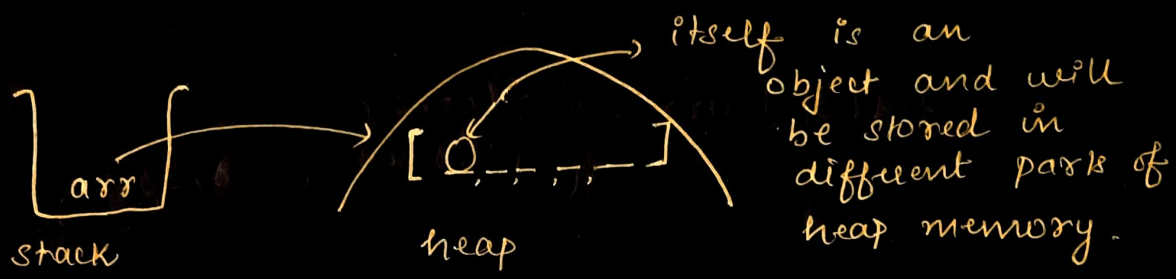
new keyword:

↳ to create an object in heap memory

↳ if values aren't provided, it stores default values.

[0, 0, 0, 0, 0] → for int/long

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

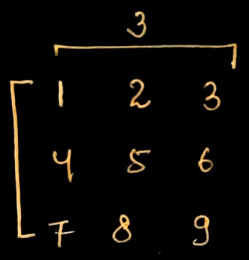


* primitives are stored in stack.
* All other objects are stored in heap memory.

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

* Array objects are mutable
* String are immutable.

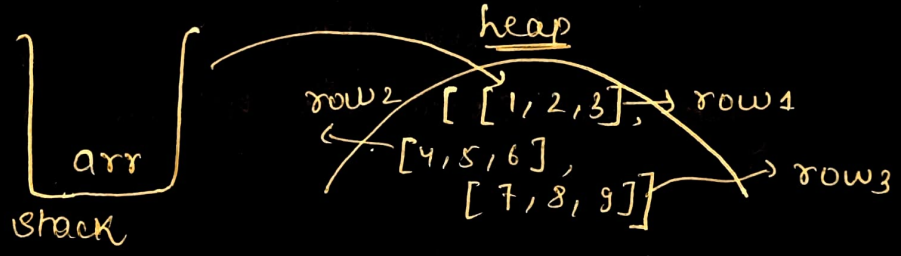
2-D Arrays.



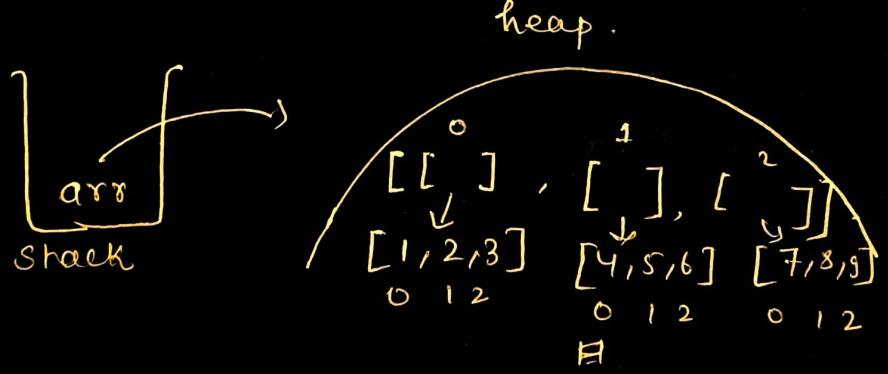

```
int[][] arr = new int[size][];
```

- `size` → row → mandatory to give size of row
- `[]` → column → not mandatory

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



arrays of arrays.




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

ArrayLists

- Part of collection framework 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 <>();
```

↑ wrapper classes only.

Internal Working of Arraylist.

- size of arraylist 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.