DAY-12

~~Data & Space~~ ~~Complexities~~

Arrays

Store group of values or data.

Syn: datatype [] Variablename = new datatype [size];

datatype represent what kind of data is stored

int [] rollno = new int [5] → Store 5 rollno
Array obj

int [] rollno = {51, 20, 17, 26, 29}
↳ Ref variable

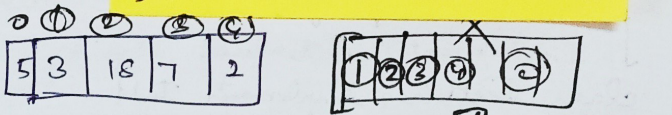
Cannot mismatch the data.

Obj in Heap memory

int [] roll; Declaration of array
roll = new int [5]; Initialize
Runtime

Dynamic Memory allocation
→ At the runtime, memory is allocated.

→ Array obj are in heap
→ heap obj are not continuous
Depends on JVM



→ Memory does not allocate Continuously

int [] arr = new int [5];
↳ Create an obj

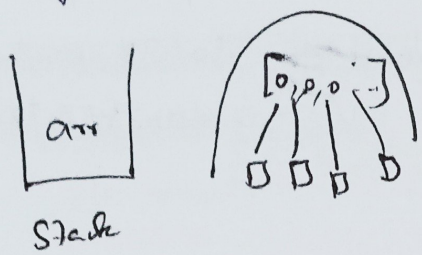
Eg: int [] nos;
nos = new int [5];
S.o.p (nos[1]);
o/p: 0

If values are not declared in it.
It assume 0 as a default value

String [] arr = new String [4]
S.o.p (arr[0]); o/p: Null

Primitives are stored in stack memory, all other obj are in heap
int, char, bool

Eg: String[] arr = new String[5];



For Loop

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

o/p: 1 2 3 4 5
[1,2,3,4,5]

Also for String

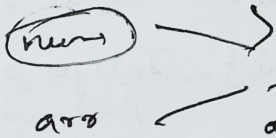
a b c d e

[a,b,c,d,e]

String are Immutable, Arrays are mutable in Java

Eg: Passing in Functions

```
import java.util.Arrays;
class PassingFunctions{
    public static void main(String[] args){
        int[] nums = {3,4,5};
        S.o.p(Arrays.toString(nums));
        change(nums);
        S.o.p(Arrays.toString(nums));
    }
    static void change(int[] arr){
        arr[0] = 94;
    }
}
```

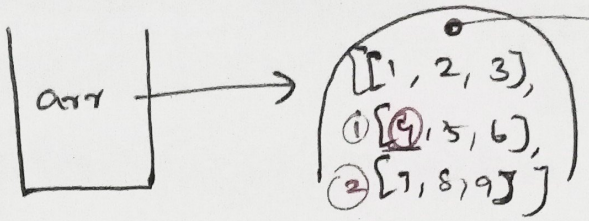


o/p: [24,2,4]

o/p:
[3,4,5]
[94,4,5]

2D Array

`int [][] arr = new int [row][col];`



Hard to do 3D in Java
We can do this in Numpy

arr[1] → [4, 5, 6]

arr[1][0] → 4

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

Eg:
```
int [ ] [ ] arr2D = {
        {1, 2, 3},
        {4, 5},
        {6, 7, 8, 9}
    };
```
Diff dimensions can be applied

ArrayList:

Part of collection framework & present in java.util package. Slower than array.

Syn: `ArrayList<Integer> list = new ArrayList<>();`
- `ArrayList` → Class
- `Integer` → Add wrapper class
- `list` → Variable
- `new ArrayList<>()` → Creates new obj
- `ArrayList<>` → Constructor
- `()` → Initial Size

Some Methods:

add() : , set(index, value) : Update existing value for specified index

get(index) : Retrieve the specific value

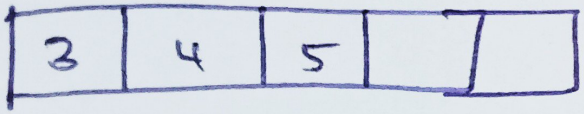
Internally size is fixed
Same as Array
Obj oriented Prog.

Eg:
```
import java.util.ArrayList;
public class ArrayListEg{
    public static void main (String [] args){
        ArrayList<Integer> list = new ArrayList<>(10);
        list.add(67);
        list.add(49);
        list.add(50);
        S.o.p(list.contains(55));
        S.o.p(list);
    }
}
```
we can add more than 10

o/p:
```
False
[67, 49, 50]
```
49 → True

Internal Working

Design Int(5) → Given value 3



Then decide to add values upto 4 element. No Space there.

It will create new list with new size. Copy the old to new list & delete old list.