ARRAYS

An Array is an collection of similar data types values.

Syntax-

datatype[] variable_name = new datatype[size]

OR

datatype[] variable_name;

variable_name = new datatype[size]

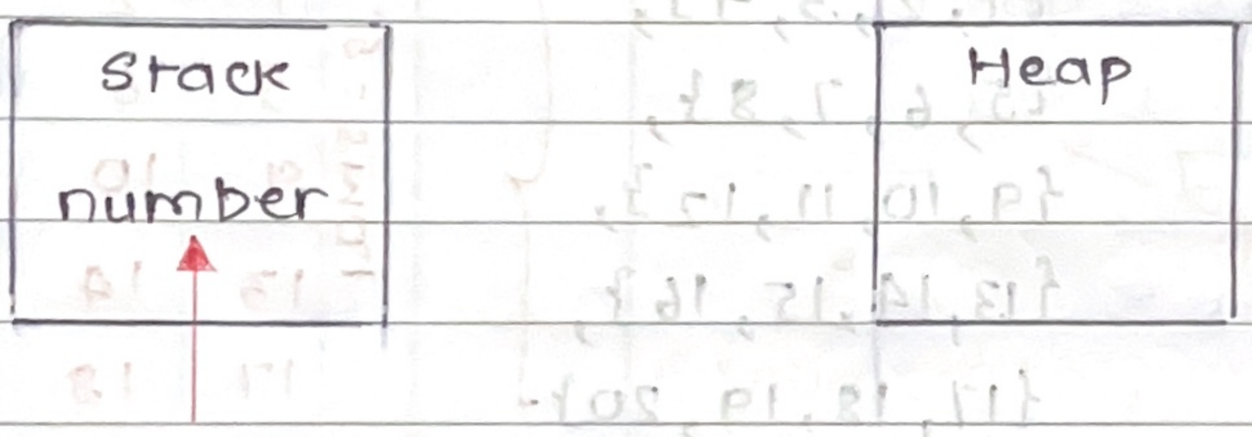
OR

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

- datatype[] variable_name;

This step will initialize the variable and it will getinitialized in the stack during compile time.

int[] number;

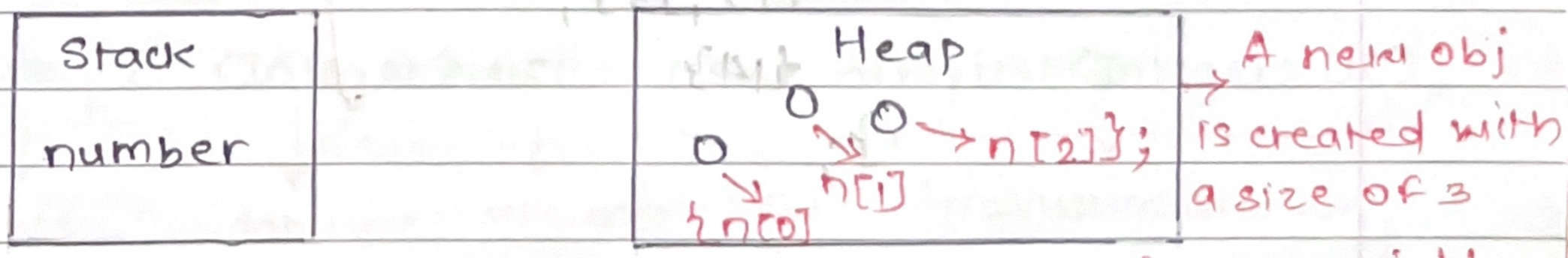


Reference variable initialized

- variable_name = new datatype[size];

In this step new object will be created in the heap memory during runtime. The new keyword is used to create new obj.

n = new int[3];



- If the reference variables have nothing to point to they will return 'null' when called.

- 2D Arrays

  A 2D Array can be visualized as a matrix.

→ First of all lets take a 1D array like this.

```
int[] num = { 2,5,6,9,3 };
```

→ Now write the array vertically like this.

```
int[] num = { 2,
              5,
              6,
              9,
              3 };
```

rows = 5

→ Then replace each element with an array like this

```
int[][] num = {
        {1,2,3,4},
        {5,6,7,8},
        {9,10,11,12},
        {13,14,15,16},
        {17,18,19,20}
};
```

(2D array)

rows = 5	1	2	3	4
	5	6	7	8
	9	10	11	12
	13	14	15	16
	17	18	19	20

colums = 4

total element = 5 x 4 = 20

→ You can also do something like this.

```
int[][] num = {
        {1,2,3,4},
        {5,6,7},
        {8,9,10,11},
        {12,13},
        {14}
};
```

(dynamic array)

rows = 5

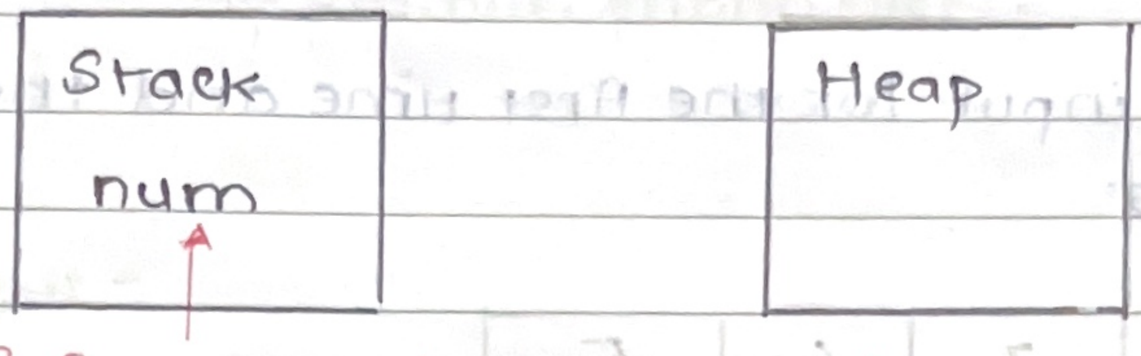
column = dynamic

Syntax -

- `datatype[][] variable_name;`

This step will declare the variable and it will be declared in the stack during compile time.

`int[][] num;`

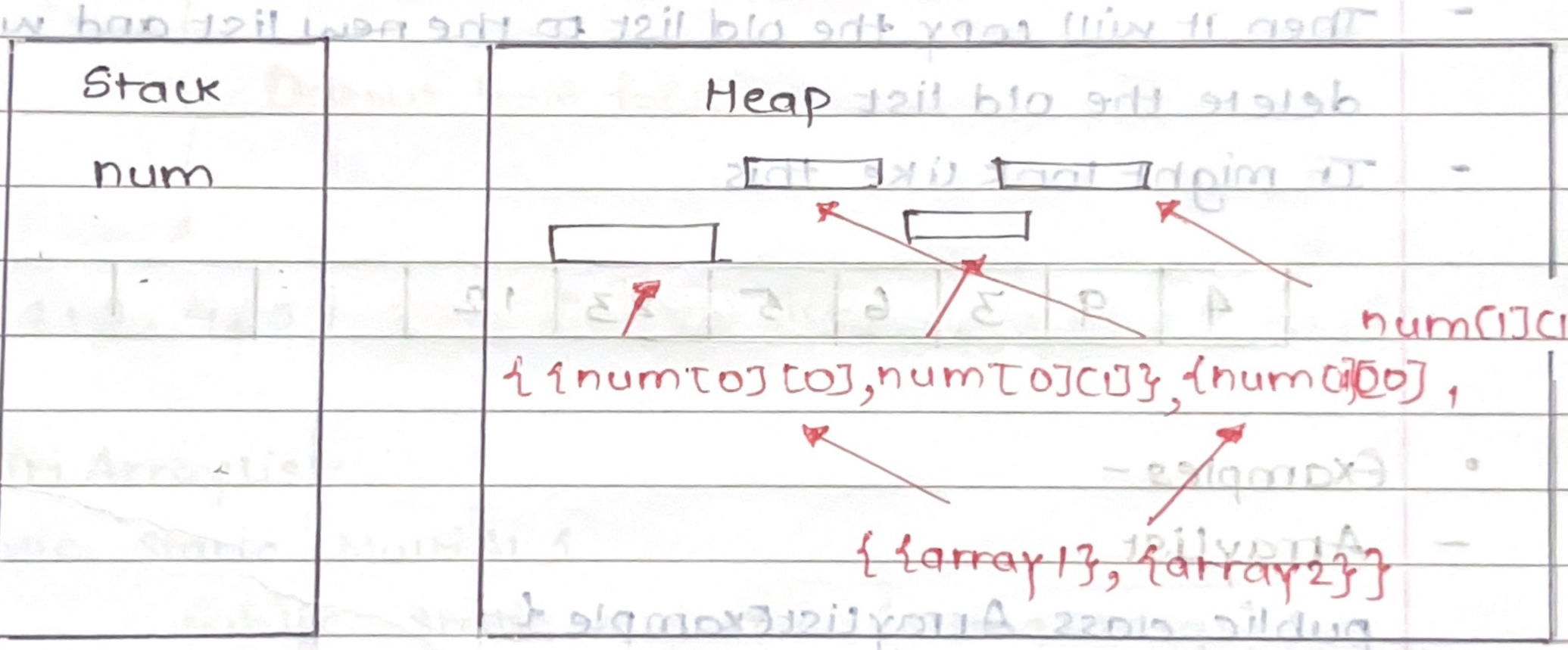


Reference variable declared.

- `variable_name = new datatype[row_size][column_size];`

In this step new object will be declared / created / initialized in the heap memory during run time.

`num = new int[2][2];`



- Arraylist

The Arraylist class is a resizable array.

Syntax -

`Arraylist<Integer> list = new Arraylist<Integer>(10);`

- class
- datatype (wrapper class) [not primitive]
- reference variable
- new obj.
- Arraylist constructor
- Initial size

- Methods -
- add() - Adds a new value to ArrayList
- set(index, value) - Updates a existing value for specified index
- get(index) - Used to retrive existing value for a specified index.

Example -
- You provide the input for the first time and the initial size is set to 5

4	9	3	6	5

- Then you decide you want to add another four element to the Arraylist,
- But we dont ~~know~~ have enough space, So what will Java do?
- The answer is it will create a new list with a new size (it depends) enough to acomodate new element.
- Then it will copy the old list to the new list and will delete the old list
- It might look like this

4	9	3	6	5	23	12				

- Examples -
- Arraylist

```
public class ArrayListExample {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);
        //syntax
        ArrayList<Integer> list = new ArrayList<>(5);
        // input
        for(int i=0; i<5; i++) {
            list.add(in.nextInt());
        }
```

```
//output
For(int i=0; i<5; i++){
        System.out.println(list.get(i));
//pass index here, list[index] syntax will not work.
}
    System.out.println(list);
  }
}
```

Output -

1
2
3
4
5
} Input values

1
2
3
4
5
} Output from forloop

[1, 2, 3, 4, 5] → Print list

- Multi Arraylist

```
public static MultiAL{
        public static void main(String[] args){
            Scanner in = new Scanner(System.in);
            ArrayList<ArrayList<Integer>> list = new ArrayList<>();
            //initialization
            For(int i=0; i<3; i++){
                list.add(new ArrayList<>());
            }
```

```
//add elements
For (int i=0; i<3; i++) {
    For (int j=0; j<3; j++) {
        list.get(i).add(in.nextInt());
    }
}
System.out.println(list);
}
}
```

Output-

1
2
3
4
5
6
7
8
9

Input values

[[1,2,3],[4,5,6],[7,8,9]] ⟶ print of list