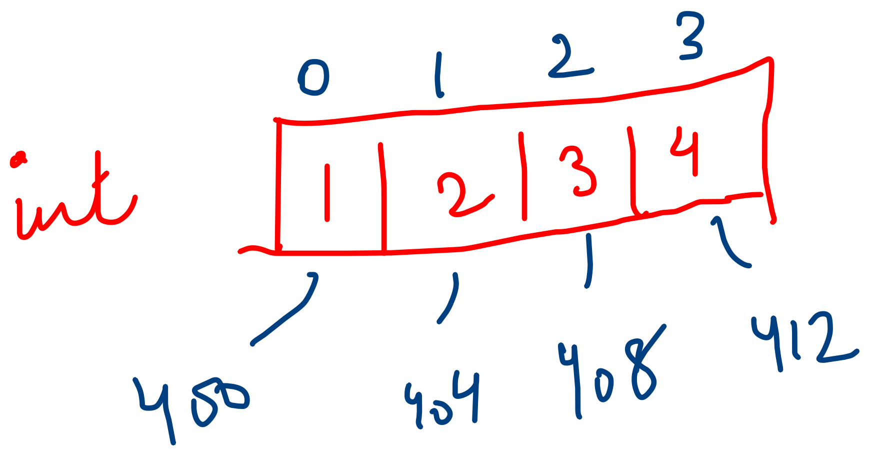
Arrays → always stored continuously.



size of arr = data-type × no. of elements

$4 \times 4 = \underline{16}$

Indexing always from 0.

length of arr = arr.length = no. of elements.

Declaration of array -

representation of arr

array size

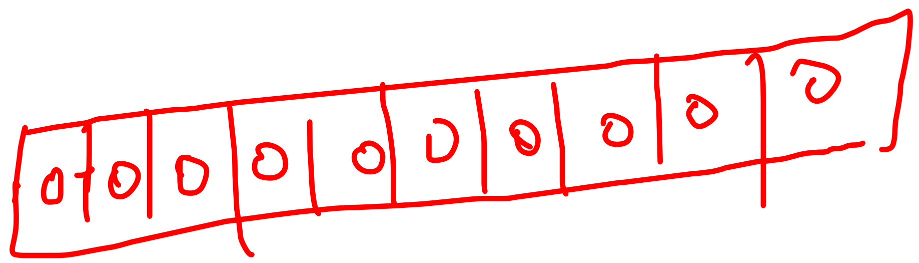
int arr[] = new int [10];

type of array

name of arr

brings continous memory.

data type



RHS = 40 bytes continuous memory.

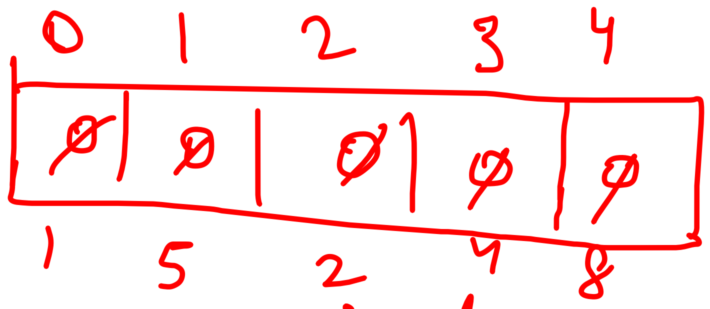
LHS = type & arr name.

```
char ch[] = new char[10]; → 'null'
double [] d = new double[10]; → 0.0
boolean [] b = new boolean[10] → false
```

[] arr ✓
arr[] ✓

Take i/p in array.

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



```
int n = 5;
```

~~=n~~ n

in-built fun.

```
for(int i = 0; i < arr.length; i++){
    arr [i]      = s.nextInt();
}
  Syso(arr[i]);
```

Array work on references.

→ If we try to access an invalid index then it will show error out of bound index.

→ indexing can't be -ve, start from [0 -(n-1)]

→ By default all indexes will be zero (0) - int

→ double - 0.0

→ character → null

→ boolean - false.

int n = s.nextInt()

int arr[] = new arr[n];

How data stored in arrays.

int i = 10

$\boxed{10}$ - 4bytes

i

Data types

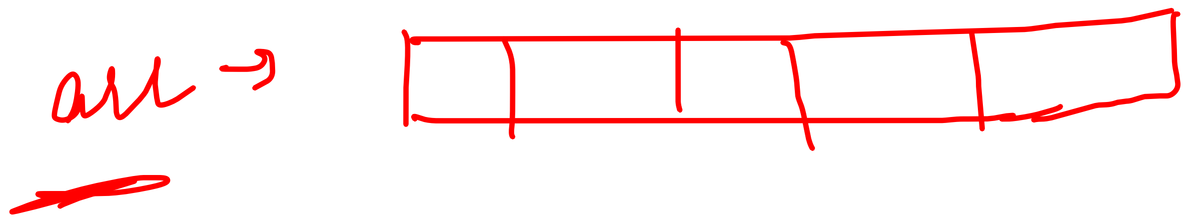
Primitives
- → int
- → char
- → double
- → boolean

etc . .

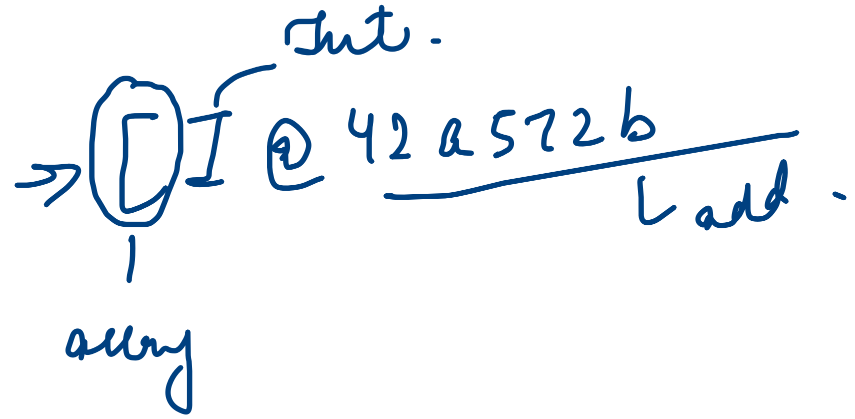
Non-primitives
- → arrays
- → strings
- → Scanner

etc . .

int arr[] = new int [5]



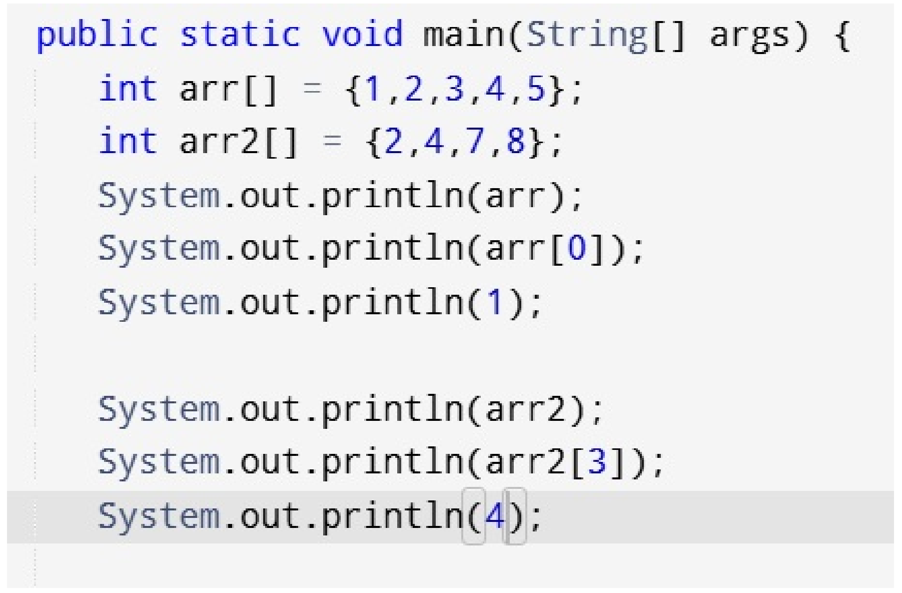
arr is reference, than direct name.

address of array. 

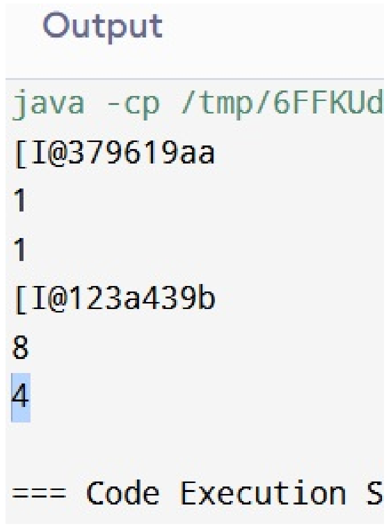
arr – add.

arr[i] – elements

i – index

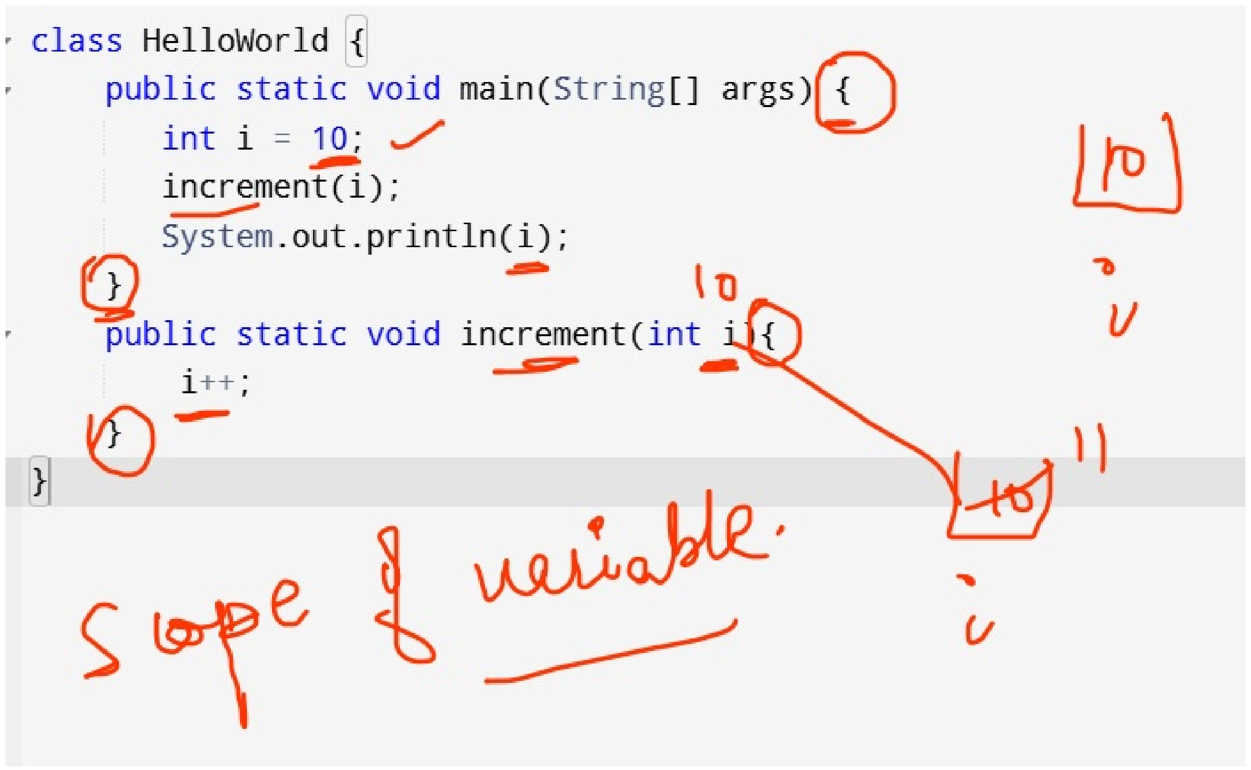


```
public static void main(String[] args) {
    int arr[] = {1,2,3,4,5};
    int arr2[] = {2,4,7,8};
    System.out.println(arr);
    System.out.println(arr[0]);
    System.out.println(1);

    System.out.println(arr2);
    System.out.println(arr2[3]);
    System.out.println(4);
```



Output

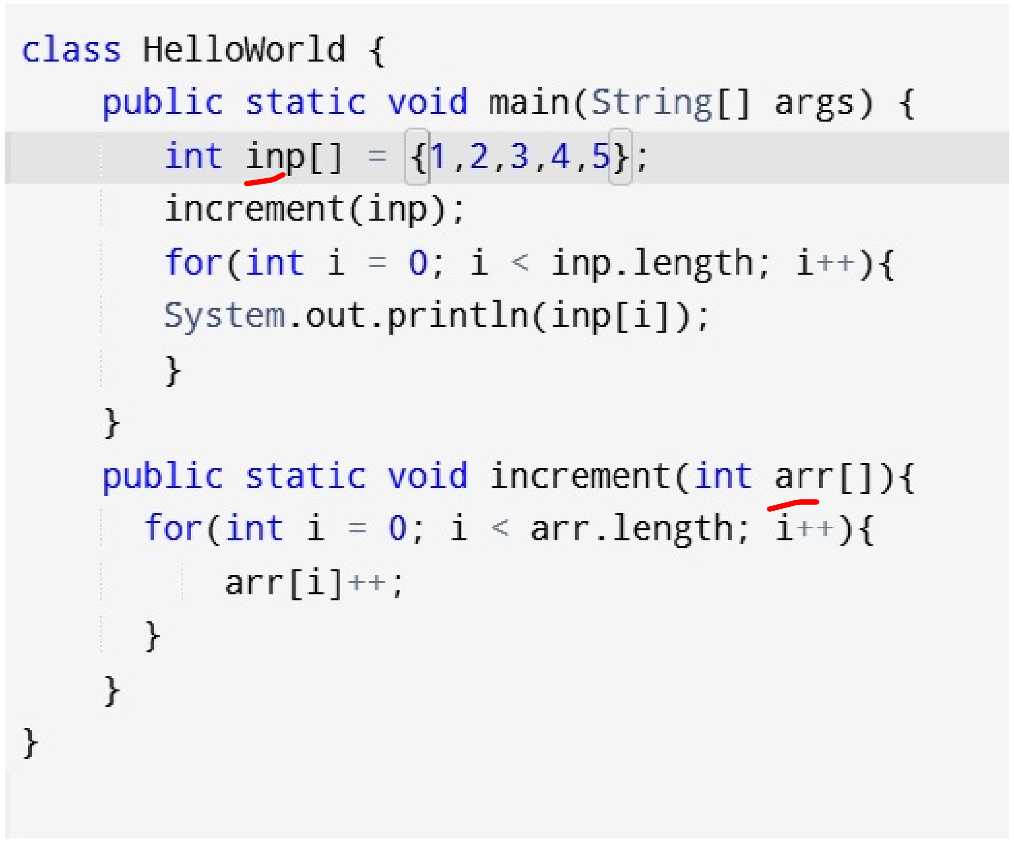
```
java -cp /tmp/6FFKUd
[I@379619aa
1
1
[I@123a439b
8
4

=== Code Execution S
```



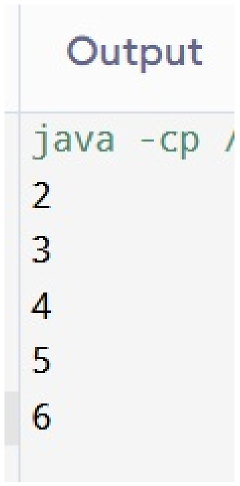
```
class HelloWorld {
    public static void main(String[] args) {
        int i = 10;
        increment(i);
        System.out.println(i);
    }
    public static void increment(int i){
        i++;
    }
}
```

Scope of variable.

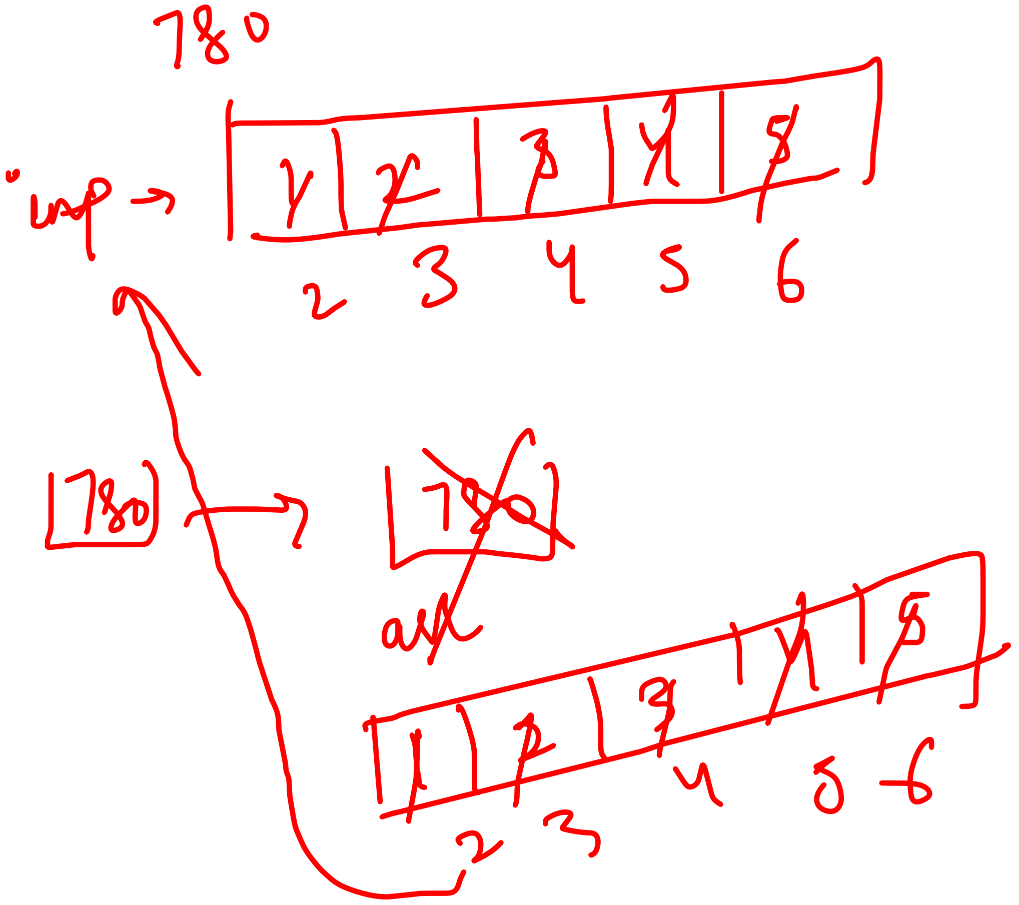
o/p - 10



```
class HelloWorld {
    public static void main(String[] args) {
        int inp[] = {1,2,3,4,5};
        increment(inp);
        for(int i = 0; i < inp.length; i++){
        System.out.println(inp[i]);
        }
    }
    public static void increment(int arr[]){
      for(int i = 0; i < arr.length; i++){
          arr[i]++;
      }
    }
}
```



```
Output

java -cp /
2
3
4
5
6
```



∴ Same memory shared by both main & increment.

Garbage Collector

→ array has a garbage collector, so that memory which is not in use is deallocated.