Arrays - 1

Array is a collection of similar type of data

Declaration and Initialization

```
int[] arr;            // declaration
arr = new int[size];  // space allocation
```

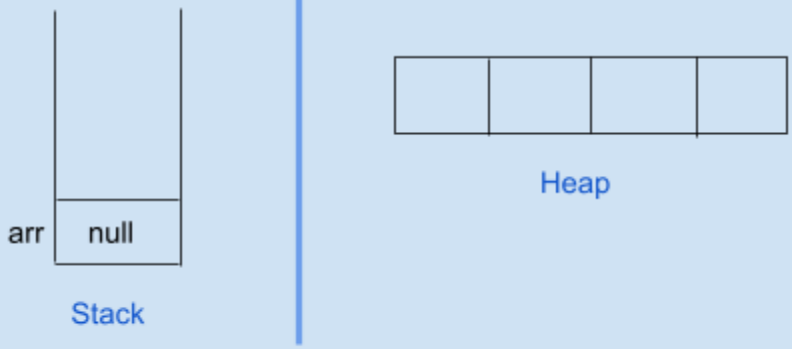
The elements of the array can be accessed by indices in the constant time (O(1) time complexity)

How is the memory allocated in the array ?

Our computer has two type of memories

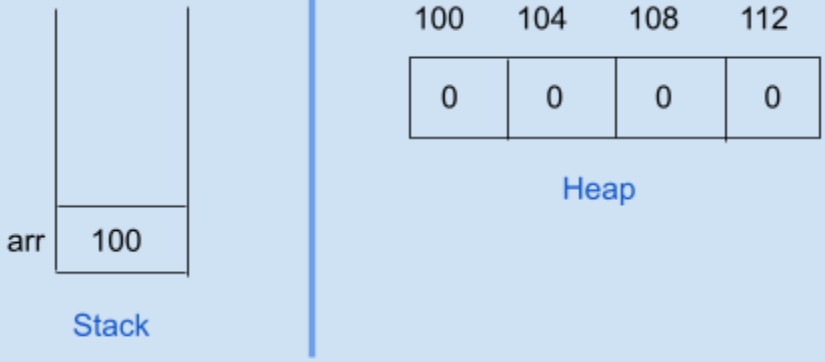
- **Stack**
  - Static memory
- **Heap**
  - Dynamic memory

Step 1: int[] arr;



It basically creates a variable named arr in stack with the value null

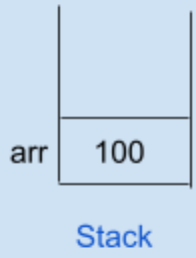
Step 2: arr = new int[4];

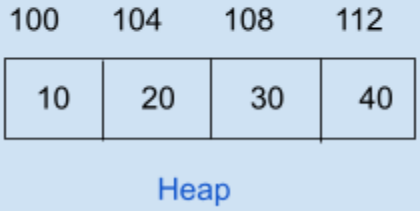


It allocates contiguous heap memory blocks for storing elements

Step 3:

```
arr[0] = 10;
arr[1] =  20;
arr[2] = 30;
arr[3] = 40;
```

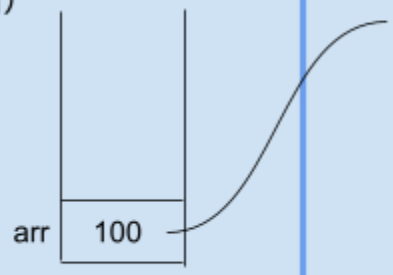


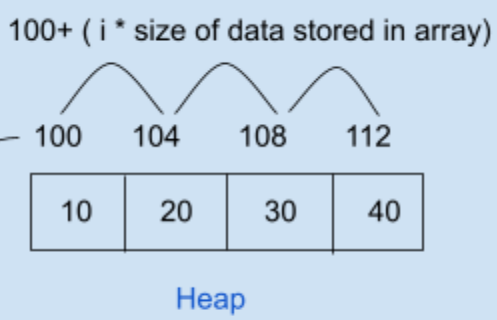


It allocates contiguous heap memory blocks for storing elements

Step 4:

```
for(i=0;i<4;i++){
print(arr[i])
}
```

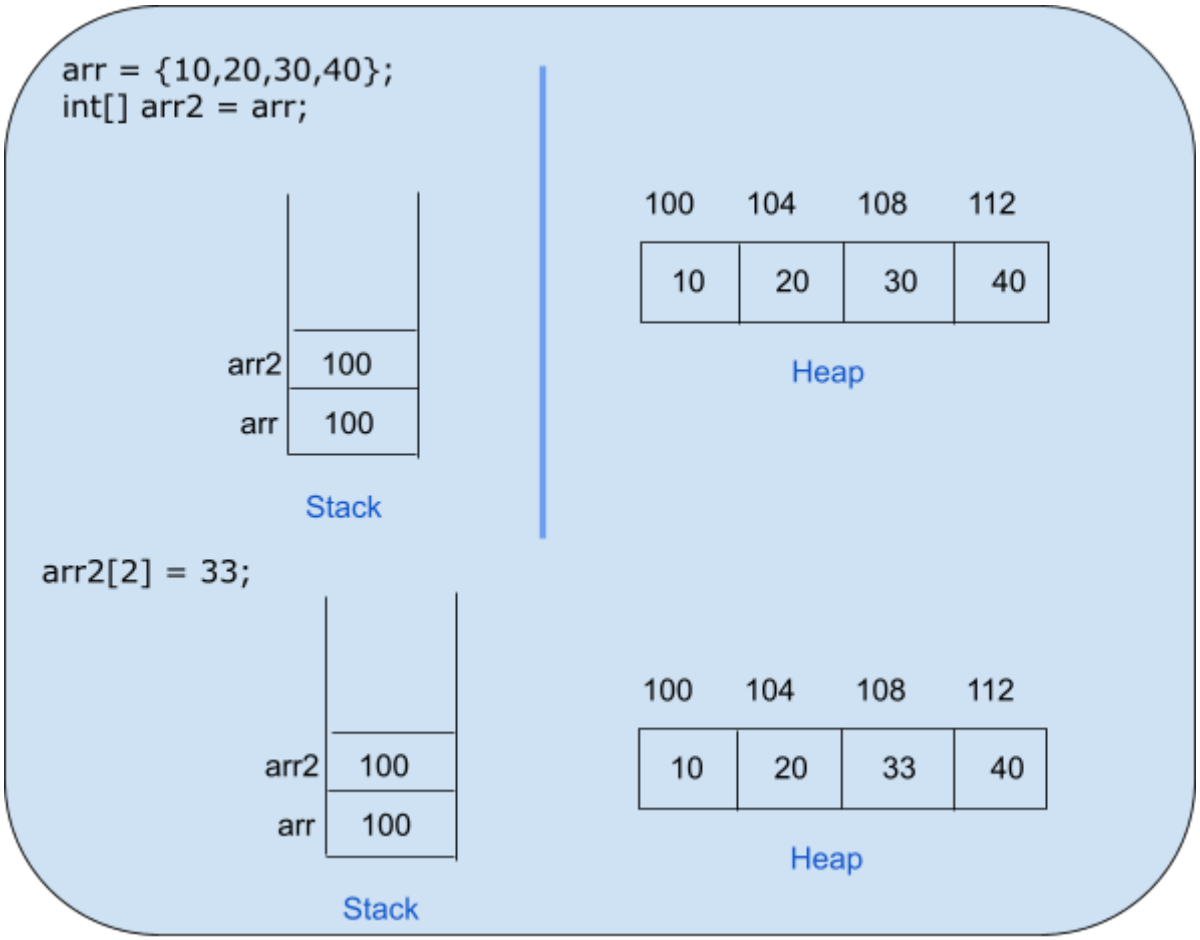




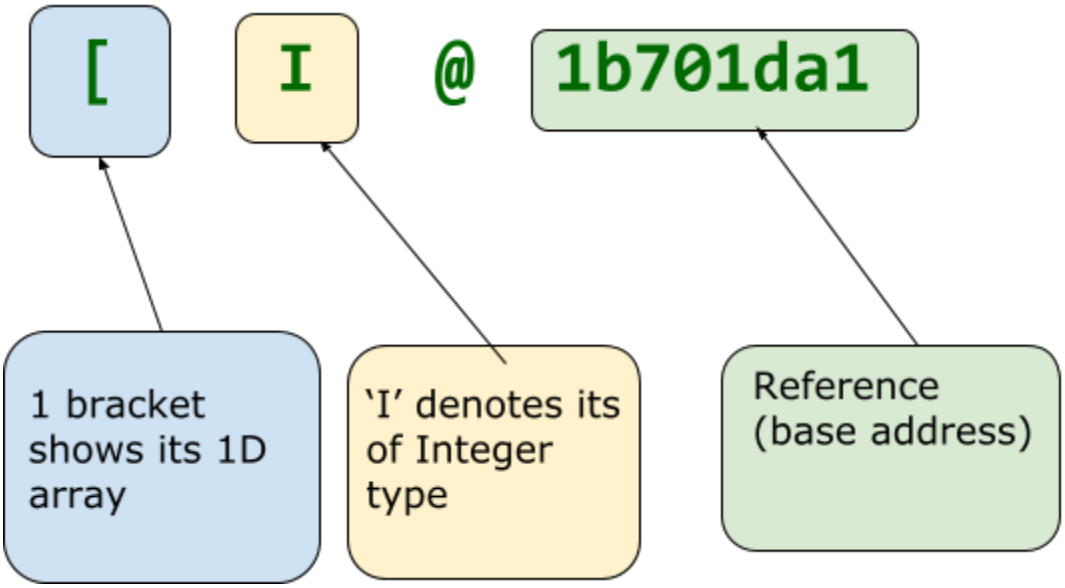
That's how the elements are accessed in constant time

```
int[] arr = {10,20,30,40};
int[] arr2 = arr;

arr[2] = 33;

print(arr2); // 10 20 33 40

print(arr); // 10 20 33 40
```

Because arr and arr2 are different variables stored in stack but they are pointing to the same heap memory block



```
int[] arr = {1,22,313,44,55,-1,-11};
        System.out.println(arr);

        //output: [I@1b701da1
```

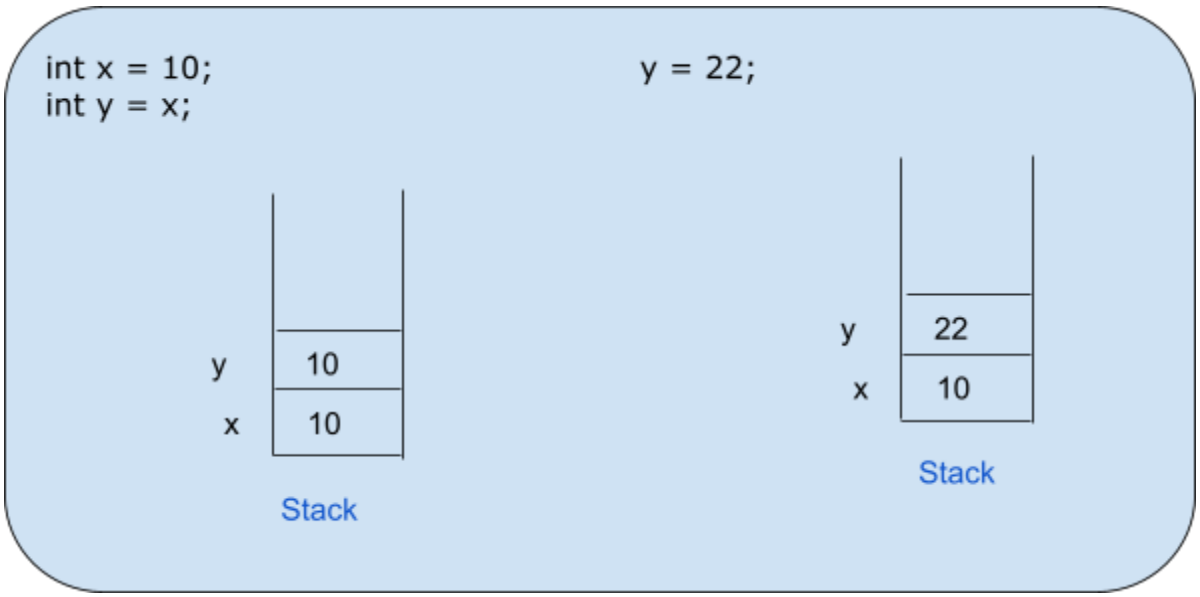


Note:

Non-primitive data types like Arrays, Strings , Scanner stores the reference (base address) rather than the actual value , and at the reference or base address actual value is stored.

```
int x = 10;
int y = x;
y = 22;

print(x,y);   //  10 , 22
```

This happened because the primitive data is stored in stack memory only.



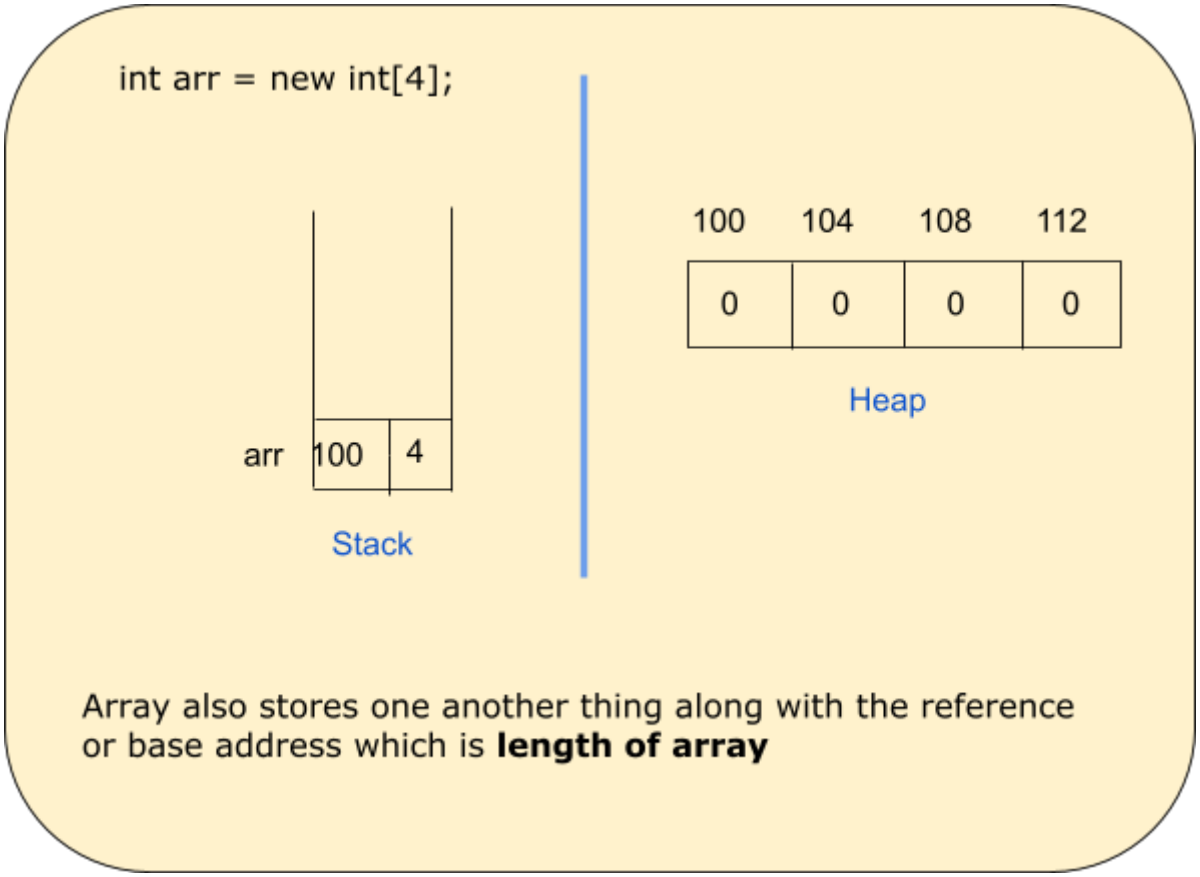
```java
public static void increment(int i){
        i++;
    }
    public static void main(String[] args) {

        int i = 10;
        increment(i);
        System.out.println(i);

        //output; 10

    }
```

Because here value 10 is passed to the increment function not the reference to i
So main's i and increment's i are different variables with separate memories.

```java
public static void increment(int[] arr){
        for (int i = 0; i < arr.length; i++) {
            arr[i] += 1;
        }
    }
    public static void printArray(int[] arr){
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i]+" ");
        }
    }
    public static void main(String[] args) {

        int arr[] = {1,2,3};
        increment(arr);
        printArray(arr);

        //output: 2 3 4

    }
```

Here we are passing variable arr to increment function , in java every thing is pass

by value so value of arr which is reference and length of array is copied in arr variable of increment function, now increment function is normally updating the value at his arr then also value of arr in main changes because both of them are actually pointing to same memory block.



Q. Pair sum

Q. Intersection of two arrays