



# JAVA ARRAY



# Today's checklist

1. Introduction to Arrays
2. Syntax , accessing elements of Arrays
3. Printing Output and Taking Input
4. Length operator
5. Basic problems
6. Memory Allocation of Arrays in Java

# What is an array?



You have store roll no.s of 100 students

```
int rno = 76;
```

```
int rno2 = 88;
```

```
int rno3 = 13;
```

If I have to store a collection of same data type variables then  
I use arrays.

# Syntax and Declaration

```
int x;
```

```
int[] y = new int[100];
```

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

```
arr[0] = 100;
```

```
arr[1] = 20;
```

```
arr[2] = 30;
```

```
arr[3] = 40;
```

```
arr[4] = 50;
```

```
arr[0] = 99;
```

arr

0	1	2	3	4
99 100	20	30	40	50

# How to access Elements in Array ?



square brackets  $\rightarrow$  `arr[i]`

# Printing Output & Taking Input



*loops.*

# Length operator

*arr.length*

## Ques:

**Q1** : Given an array of marks of students, if the mark of any student is less than 35 print its roll number. [roll number here refers to the index of the array.]

	0	1	2	3	4	5	6
arr	81	17	45	36	31	100	60

```
for(    ){  
    if(arr[i] < 35) sout(i);  
}
```



## Ques:

**Q2** : Are the following array declarations correct?

`int a (25);` ✗ `int[] a;`

`int size = 10, b[size];` → ✗ `int size = 10; int b[size];`

`int c = {0,1,2};` ✗ `int[] c = {0, 1, 2};`

## Ques:

**Q3 :** Which element of the array does this expression reference?

$\nearrow$  array's name  
**num[4]**  $\rightarrow$  5<sup>th</sup> element from the start  
 $\downarrow$   
index

# Predict the output :

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        ✓int[] num = new int[26];
```

```
        ✓num[0] = 100;
```

```
        ✓num[25] = 200;
```

```
        ✓int temp = num[25];
```

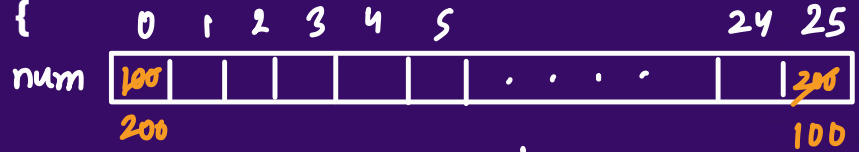
```
        ✓num[25] = num[0];
```

```
        ✓num[0] = temp;
```

```
        ✓System.out.println("\n" + num[0] + " " + num[25]);
```

```
    }
```

```
}
```



swap

```
•  
• 200 100  
•
```

## Ques:

**Q4 :** Calculate the sum of all the elements in the given array.

	0	1	2	3	4	5	6
arr	81	17	45	36	31	100	60

```
int sum = 0;
```

# Linear search

**Q5** : Find the element 'x' in the array . Take array and x as input.

```
int x = sc.nextInt();
```

# Ques:

**Q6 :** Find the maximum value out of all the elements in the array.

arr = { 10, 8, 12, 4, 6, 23, 8 }

0 1 2 3 4 5 6

→

M-I :

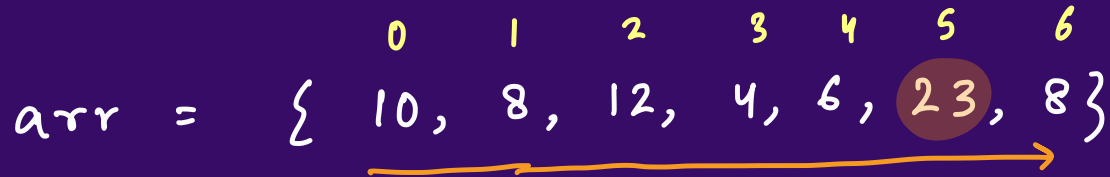
```
int mx = arr[0];  
for(int i=1; i<n; i++){  
    if(arr[i] > mx) mx = arr[i];  
}
```

mx  
↓  
10  
12  
23

## Ques:

**Q6 :** Find the maximum value out of all the elements in the array.

arr = { 10, 8, 12, 4, 6, 23, 8 }



M-II:  
int mx = -1;  
for(int i=0, i<n; i++){  
 if(arr[i] > mx) mx = arr[i];  
}

**Ques:** *(Easy-med)*

**Q7 :** Find the second largest element in the given Array.



# MCQ :

What is the difference between the 5's in these two expressions?

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

`num[5] = 11;`

↗ size of num

↘ 5<sup>th</sup> index

1. first is particular element, second is type
- ✓ 2. first is array size, second is particular element
3. first is particular element, second is array size
4. both specify array size

# MCQ :

What would happen if you assign a value to an element of an array whose **subscript** exceeds the size of the array?

↓  
index

1. the element will be set to 0
2. nothing, it's done all the time
3. other data may be overwritten
- ✓ 4. error message from the compiler

`int[] arr = new int[5];` → 0 to 4

`arr[6] = 7;`

↓

index out of bound

# State TRUE or FALSE :

1. The array `int[] num = new int[26]` has twenty-six elements. *True*
2. The expression `num[1]` designates the first element in the array *False*  
*Second element*
3. It is necessary to initialize the array at the time of declaration. *False*
4. The expression `num[27]` designates the 28th element in the array. *True*

**Ques:** H.W. rakha

**Q5:** Count the number of elements in given array greater than a given number x.

# Point out the errors(if any) in the following code: *h.w*

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int size = sc.nextInt();  
        int[] arr = new int[size];  
        for (int i = 0; i < size; i++) {  
            arr[i] = sc.nextInt();  
            System.out.print(arr[i] + " ");  
        }  
    }  
}
```

# Passing Array to Methods

```
public static void main(String[] args) {  
    ✓ int x = 5;  
    ✓ System.out.println(x);  
    ✓ change(x);  
    ✓ System.out.println(x);  
}  
  
public static void change(int 5x) {  
    ✓ x = 10;  
}
```

$\boxed{5}$   
x

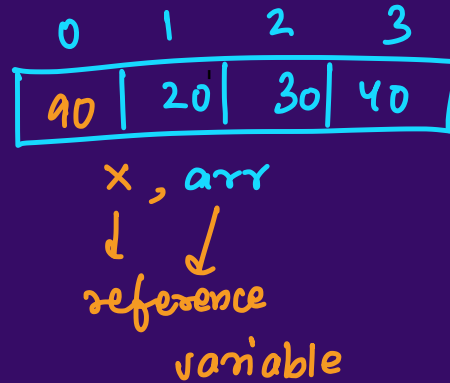
$\boxed{\begin{smallmatrix} 10 \\ \cancel{5} \end{smallmatrix}}$   
x

Output

· 5  
· 5

# Passing Array to Methods

```
✓ public static void main(String[] args) {  
    ✓ int[] arr = {10,20,30,40};  
    ✓ System.out.println(arr[0]);  
    ✓ change(arr);  
    ✓ System.out.println(arr[0]);  
}  
  
public static void change(int[] x) {  
    ✓ x[0] = 90;  
}
```



Output

• 10

• 90

Pass by Reference

# MCQ:

When you pass an array as an argument to a Method, what actually gets passed?

- ✓ 1. address of the array (In Java Reference variable gets passed)
- 2. values of the elements of the array ✗
- 3. address of the first element of the array
- 4. number of elements of the array ✗

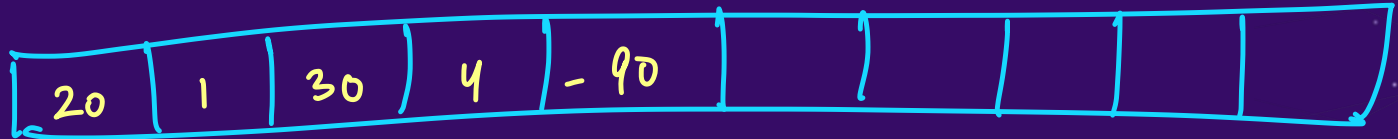
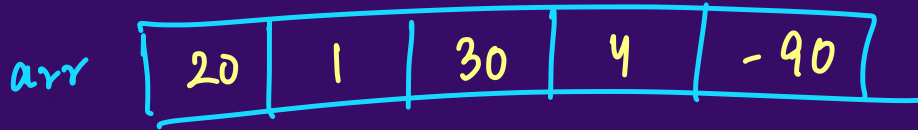


# ArrayList in Java

Unlimited size Array



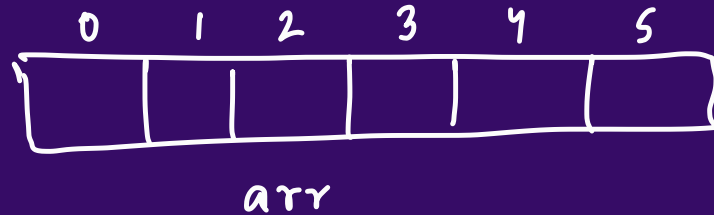
↓  
Why? → In Java, arrays they have fixed size  
So, we use **dynamic arrays** using OOP



# Basic Operations on ArrayList

&  
Arrays

```
ArrayList<Integer> arr = new ArrayList<>(6);
```



```
arr.add(90);
```

↓  
pushes 90 to  
the end of the list

index  
↑  

```
arr.add(0, 30)
```

 initialised this value

```
arr.set(0, 50)
```

  
index  
↓  
update/modify

```
arr.get(3);
```

# Ques :

**Q8 :** Find the **doublet** in the Array whose sum is equal to the given value x. (Two Sum)

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

# Two Pointers

**Q9**: Write a program to reverse the array without using any extra array.

$arr = \{ 10, 20, 30, 40, 50, 60, 70 \}$

$i$   $j$

$\hookrightarrow \{ 70, 60, 50, 40, 30, 20, 10 \}$

$$\Rightarrow \begin{aligned} i+j &= n-1 \\ j &= n-1-i \end{aligned}$$

`swap(arr[i], arr[j])`

# Two Pointers

**Q9**: Write a program to reverse the array without using any extra array.

arr = { 60, 50, 40, 30, 20, 10 }

0 1 2 3 4 5

j i

```
while (i <= j) {  
    swap(arr[i], arr[j])  
    i++;  
    j--;  
}
```

## Ques :

**Q10** : Rotate the given array 'a' by k steps, where k is non-negative. Without using extra Array

**Note** : k can be greater than n as well where n is the size of array 'a'.

$a = \{10 \ 20 \ 30 \ 40 \ 50 \ 60 \ 70\}$

$k=5$

$n=7$

$n-k$



$30 \ 40 \ 50 \ 60 \ 70 \ 10 \ 20$

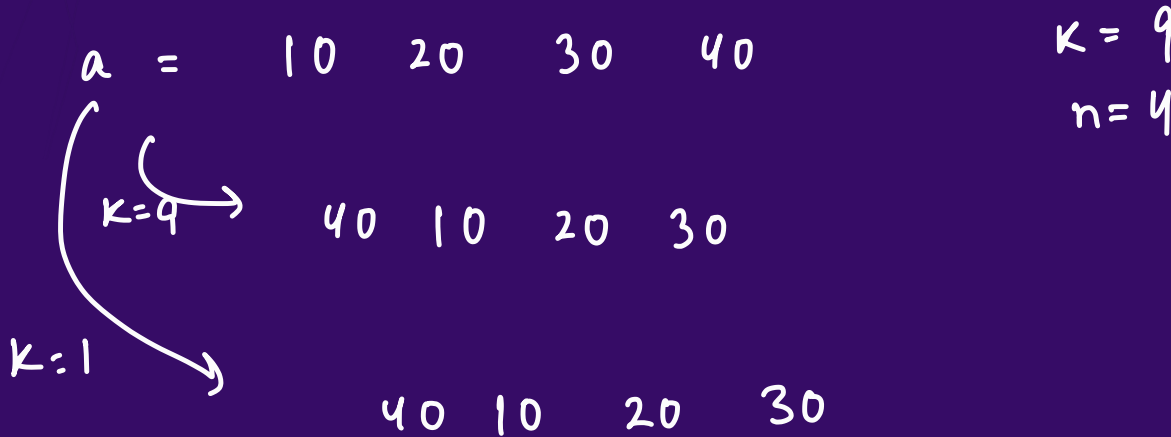
30	40	50	60	70	10	20
----	----	----	----	----	----	----

# Ques :

**Q10** : Rotate the given array 'a' by k steps, where k is non-negative.

$$K = K \% n$$

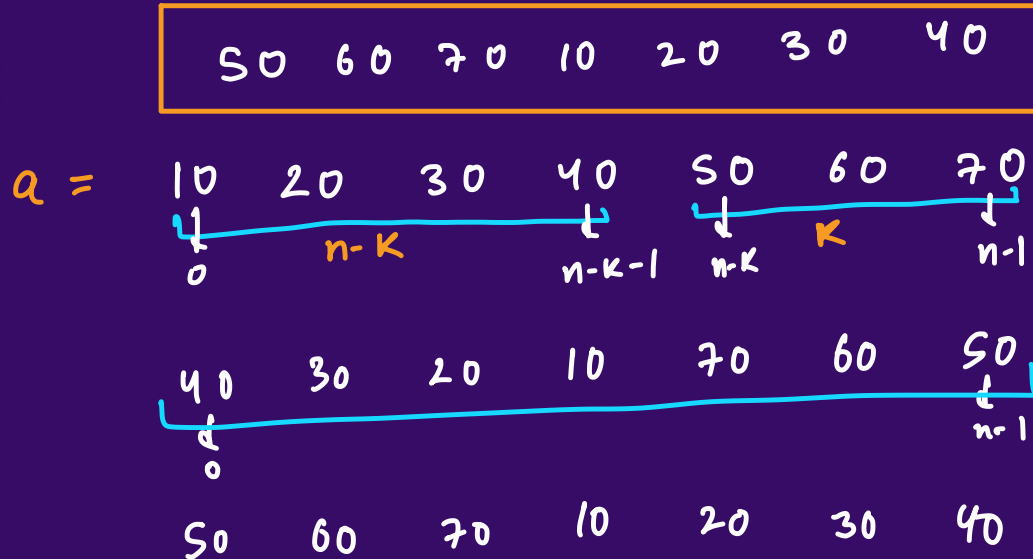
**Note** : k can be greater than n as well where n is the size of array 'a'.



# Ques :

**Q10** : Rotate the given array 'a' by k steps, where k is non-negative.

**Note** : k can be greater than n as well where n is the size of array 'a'.



k=3  
n=7



# Ques :

**Q11.** Sort the array of 0's and 1's .

arr = { 1 0 0 1 1 0 0 0 1 0 }

M-I: Arrays.sort(arr);

M-II: int noOfOnes = 0, noOfZeros = 0;

→ Calculate no. of 0's & 1's .  
↓  
Update the array with 0's & 1's

Two Pass Sol<sup>n</sup>

**Ques :** M-2 : One Pass Sol<sup>n</sup>

**Q11.** Sort the array of 0's and 1's .

0 1 2 3 4 5 6 7 8 9  
arr = { 0 0 0 0 0 0 1 1 1 1 }

while (i < j)

i  
j

```
if (arr[i] == 0) i++;  
if (arr[j] == 1) j--;  
if (arr[i] == 1 && arr[j] == 0) {  
    swap(arr[i], arr[j]);  
    i++;  
    j--;  
}
```

# Hint

- 1) 2 pointer approach
- 2) Swapping

# Ques :

**Q11.** Sort the array of 0's and 1's .

arr = { 0, 0, 0, 1, 1, 1 }

j      i

while (i < j)

if (arr[i] == 0) i++;

if (arr[j] == 1) j--;

if (arr[i] == 1 && arr[j] == 0) {

    swap(arr[i], arr[j]);

    i++;

    j--;

}

# Ques :

**Q12.** Sort the array of 0's , 1's and 2's . (Dutch Flag Algorithm)

Ex-  $arr = \{ 0, 1, 2, 0, 1, 2, 1, 2, 0, 0 \}$

noo, noo, not  
4 3 3

M-I → two-pass solution

**Ques :** 3-pointer approach  $\rightarrow$  lo, mid, hi

**Q12.** Sort the array of 0's, 1's and 2's. (Dutch Flag Algorithm)

Method-2 : Dutch Flag Algorithm

arr = { 0, 1, 2, 0, 1, 2, 1, 2, 0, 0 }

↑  
l  
↓  
m  
↓  
h

Steps : Break array in 4 parts

0 0 0 ... 0    1 1 1 ... 1    unsorted part    2 2 2 2 2

↓                      ↓                      ↓                      ↓                      ↓

0                      0-1                      lo                      mid-1                      mid                      hi                      hi+1                      n-1

# Ques :

Q12. Sort the array of 0's , 1's and 2's . (Dutch Flag Algorithm)



```
if(arr[mid] == 0){
    swap(arr[lo], arr[mid])
    lo++, mid++
}
if(arr[mid] == 1) mid++
```

```
if(arr[mid] == 2){
    swap(arr[mid], arr[hi])
    hi--
}
```

do se pehle  
Saare 0 move  
Chahiye

hi ke baad  
Saare 2  
move chahiye  
mid se pehle

# Ques :

**Q12.** Sort the array of 0's , 1's and 2's . (Dutch Flag Algorithm)

arr = { 0, 0, 0, 0, 1, 1, 1, 2, 2, 2 }

l ↑  
          ↓  
          h  
              ↓  
              m

**Ques :** # Hint  $\rightarrow$  3 pointer  $\rightarrow i, j, k \rightarrow 0$   
 $\begin{matrix} \downarrow & \downarrow & \downarrow \\ a & b & c \end{matrix}$

**Q13.** Merge two sorted arrays in one single array.

a

0	1	2	3
11	33	42	71

b

0	1	2	3
26	54	69	81

j

i

c

0	1	2	3	4	5	6	7
11	26	33	42	54	69	71	

k

```
if(a[i] <= b[j]){  
    | c[k] = a[i];  
    | i++; k++;  
    |  
    3
```



# Ques :

Q13. Merge two sorted arrays .

	0	1	2	3
a	11	33	42	62

i

	0	1	2	3	4
b	26	54	69	81	94

j

	0	1	2	3	4	5	6	7	8
c	11	26	33	42	54	62	69	81	94

k

# Ques :

Next greater element  $\Rightarrow$  Stack



## Q14. Next greatest element.

arr = 

12	8	60	37	2	49	16	28	21
----	---	----	----	---	----	----	----	----

ans = 

							21	-1
--	--	--	--	--	--	--	----	----

nge = 21 28

int nge = arr[n-1];

```
for(int i = n-2; i >= 0; i--) {  
    ans[i] = nge;  
    nge = max(nge, arr[i]);  
}
```

◀ **THANK YOU** ▶