**Arrays Revision**

**Day-01**

**18-03-2025**

**=====================**

**Three syntaxes for Arrays:**

**--------------------------------**

**Array Declaration:**

**============**

1) datatype array-name[] = new datatype[size];

2) datatype[] array-name = new datatype[size];

Scanner scan = new Scanner(System.in);

Arrays ==> pre-defined class

was included in "util" package

util is from java library

import java.util.Arrays;

Ex: sort()

Arrays.sort(array-name);

int[] a = {1,2,3,4,5};

a.sort() ==> instance method ==> Incorrect

Arrays.sort(a); ==> Static method

-> The method which is accessing/calling with the object of the class that is called as "Instance method".

-> The method which is accessing/calling with the class name is called as "Static method".

3) datatype[] array-name;

array-name = new datatype[size];

**Array Assignment:**

**---------------------**

datatype[] array-name = {1,2,3,4,5}; // static array

Accessing of Individual elements from the array:

==============================

-> we can use "index"

index or subscript or radix or base

Syntax:

array-name[index-value];

Here:

index-value ==> start at '0'

last element index ==> total size of the array - 1

length:

-------

-> is an attribute or property

of Arrays class

-> we can use to get the size of the array

Syntax:

array-name.length;

Features of Array:

---------------------

1) Arrays are ordered datatype

-> What ever the order the programmer have followed while the definition, the same order can be understood by the JVM while accessing.

2) Arrays are Index based datatype

**Q-1: What happened when we can use the index value from out of index range while accessing of array elements?**

-------------------------------------------------------------------------

Run time error also called as "Exception".

"ArrayIndexOutOfBoundsException" is a class

-> when we can use the index from out of the range, we can get "ArrayIndexOutOfBoundsException".

**Q-2: What happened that I have defined with negative index while array accessing.**

**---------------------------------------------------------------------**

Ans: ArrayIndexOutOfBoundsException

Static Array Definition:

--------------------------

Static array can define without "new" keyword.

/\*

\* Write a java program to define static array.

\*/

public class DefinitionOfArray {

public static void main(String[] args) {

int arr[] = {10,100,11,110,12,120};

System.out.println("The size of the array = "+arr.length);

System.out.println("First = "+arr[0]);

System.out.println("Second = "+arr[1]);

System.out.println("Third = "+arr[2]);

System.out.println("Forth = "+arr[3]);

System.out.println("Fifth = "+arr[4]);

System.out.println("Sixth = "+arr[5]);

// System.out.println("Seventh = "+arr[6]);

// System.out.println(arr[-1]);

}

}

/\*

\* Write a java program to define static array and access

\* all the elements using for loop.

\*/

public class DefinitionOfArray {

public static void main(String[] args) {

int arr[] = {10,100,11,110,12,120};

System.out.println("The Given Array with elements = ");

for(int index = 0;index < arr.length;index++) {

System.out.println("The Element at "+index+" is = "+arr[index]);

}

}

}

Dynamic Array Definition:

-----------------------------

Dynamic array: When we can define the size of the array in the run time of the program that array is called as "Dynamic array".

toString():

-----------

-> can convert the array into the string

Syntax:

Arrays.toString(array-name);

/\*

\* Write a program to define the dynamic array.

\*/

import java.util.Arrays;

import java.util.Scanner;

public class DynamicArray {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the size of the array:");

int size = scan.nextInt();

// array declaration

int[] a = new int[size];

// when we have declared the array with integer type

// and not assigned any values

// jvm can assign default value of integer

// to array according to the size

for(int i = 0;i < size;i++) {

System.out.println("Enter the value for the index:"+i);

a[i] = scan.nextInt();

}

System.out.println("The Array = "+Arrays.toString(a));

}

}

**Q-3:What happened when we can assign the less number of elements to the array.**

**---------------------------------------------------------------------**

Ans:

Assume the array with 6 elements

a[6] = {1,2,3};

import java.util.Arrays;

public class InterviewQuestion {

public static void main(String[] args) {

int[] a = new int[6];

// a = {1,2,3,4,5,6};

a[1] = 11;

a[3] = 33;

a[5] = 55;

System.out.println("The Array = "+Arrays.toString(a));

}

}

-> When the number of elements are less than the given size of the array, then the JVM can store the values which are given according to the index and the remaining positions can fill with default values based on the datatype.

**String Array with Default Values:**

**-------------------------------------**

import java.util.Arrays;

public class InterviewQuestion {

public static void main(String[] args) {

String[] a = new String[5];

a[0] = "Ball";

a[1] = "Bat";

a[2] = "Cat";

System.out.println("The Array = "+Arrays.toString(a));

}

}

**Day-02**

**19-03-2025**

**===================**

**length():**

**======**

-> is a pre-defined or built-in method from the "String" class.

String string-variable-name;

-> length() can be used to find the "length of the string" or "number of characters from string".

Syntax:

string-variable-name.length();

-> an Instance method.

public class LengthVsLengthOf {

public static void main(String[] args) {

String str = "Java Programming Language";

System.out.println("The length of the string = "+str.length());

}

}

**Q: What is the file name of the java program which has given below:**

public class LengthVsLengthOf {

public static void main(String[] args) {

System.out.println("Hi");

}

}

Ans: LengthVsLengthOf.java

**Q: What is the file name of the java program which has given below:**

class LengthVsLengthOf {

public static void main(String[] args) {

System.out.println("Hi");

}

}

Ans: Any name

Note:

-----

-> If the main class is the public type, then the java file name is same as main class name.

-> If the main class is not the public type, then the java file name is any.

length vs length():

--------------------

public class LengthVsLengthOf {

public static void main(String[] args) {

String[] fruits = {"Apple","Banana","Mango","Jack-fruit"};

System.out.println(fruits.length);

// System.out.println(fruits.length());

System.out.println(fruits[0].length());

}

}

=> length is the property can be used to fine the number of elements of the array.

Where as the length() can use to find the size or number of characters in a string.

**Types of Arrays:**

**===========**

-> in two types:

1) Single Dimensional Arrays/One Dimensional Arrays

2) Multi Dimensional Arrays

-> When the array has defined with single subscript is called as "One Dimensional Array".

Syntax:

datatype[] array-name = new datatype[size];

-> When the arrays has defined with more than one subscript is called as "Multi-Dimensional Array".

Syntax:

datatype array-name[][] = new datatype[rows][cols]; ==> two dimensional array

datatype[][][] array-name = new datatype[l][m][n];==> three dimensional array

Two Dimensional Array:

--------------------------

-> The collection of one dimensional arrays is called as "Two dimensional array".

-> In the two dimensional array, each row can be indexed from 0 to total-rows - 1

Syntax:

array-name[index-number] ==> row from 2d array

-> to access individual elements (values at column), we should apply the indexing to the rows

Syntax:

array-name[index-number][index-number] ==> values at column

Syntax for the 2d array definition:

datatype array-name[][] = {{e1,e2,e3,e4,...},{f1,f2,f3,f4,...},{g1,g2,g3,g4,...},{},....{}};

-> To access the individual elements of the array, we can use nested for loop.

Syntax:

for(int row = 0;row < array-name.length; row++){

for(int col = 0; col < array-name[row].length; col++){

Sop(array-name[row][col]);

}

}

/\*

\* Write a java program to define 2d-array

\* and access all the elements.

\*/

import java.util.Arrays;

public class TwoDimenisonalArray {

public static void main(String[] args) {

int a[][] = {{1,2,3},{4,5,6},{7,8,9}};

for(int r = 0; r < a.length; r++) {

for(int c = 0;c < a.length;c++) {

System.out.print(a[r][c]+"\t");

}

System.out.println();

}

}

}

Three Dimensional Array:

----------------------------

-> The collection of two dimensional arrays is called as "Three Dimensional Array".

Syntax for the three dimensional array declaration:

datatype[][][] array-name = new datatype[][][];

-> Static three dimensional array:

datatype array-name[][][] = {{2d1},{2d2},{2d3},..};

{{{1d},{1d},{1d},...},{{1d},{1d},{1d},...},{{1d},{1d},{1d},...},...};

-> The syntax for the accessing of three dimensional array elements:

Syntax:

for(int i = 0;i < a.length; i++)

{

for(int j = 0;j < a[i].length;j++){

for(int k = 0;k < a[i][j].length;k++){

Sop(a[i][j][k]);

}

}

}

here:

a.length ==> can give the total number of 2d arrays from the given array

a[i].length ==> can given total number of d arrays from each 1d array

a[i][j].length ==> can given length of each 1d-array

/\*

\* Three Dimensional Array

\*/

public class ThreeDimensionalArray {

public static void main(String[] args) {

int[][][] a = {{{1,2,3,4},{5,6,7,8},{8,7,6,5}},{{4,3,2,1},{0,1,2,3},{4,5,6,7}}};

System.out.println("The Given Three Dimensional Array:");

for(int i = 0;i < a.length;i++) {

for(int j = 0;j <a[i].length;j++) {

for(int k = 0;k < a[i][j].length;k++) {

System.out.print(a[i][j][k]+"\t");

}

System.out.println();

}

System.out.println();

}

}

}

Assignment:

--------------

1) Write a java program to access the dynamic two-dimensional array and print all the elements of the two dimensional array in the matrix form.

int s1 = scan.nextInt();

int s2 = scan.nextInt();

int a[][] = new int[s1][s2];

for(int i = 0;i < s1;i++)

{

for(int j = 0;j < s2;j++)

{

System.out.println("Enter the element:");

a[i][j] = scan.nextInt();

}

}

**Day-03**

**20-03-2025**

**==================**

**1) Matrix Addition**

**---------------------**

Order of matrix: rows X cols

Ex: 1 2 3

4 5 6

-> order : 2 X 3 matrix

1 2 3

4 5 6

7 8 9

-> order : 3 X 3 matrix

**Q: number of elements in the given array declaration**

i) a[3][4] ==> 12 elements

ii) a[2][6] ==> 12 elements

iii) a[3][4][5] ==> 60 elements

Note:

----

The total number of elements in any array is calculated by the following formulae:

indices1 X indices2 X indices3 ....

-> When we want to perform the addition on two matrices,

in this case, both matrices must be with same order.

-> Element by element from each row of two matrices can be added and finally can give the result with same order as like the given matrices.

public class MatrixAdditions {

public static void main(String[] args) {

int[][] a = {{1,3,5,7},{9,7,5,3}};

int b[][] = {{11,22,33,44},{10,20,30,40}};

int m1 = a.length; // number of rows of a

int n1 = b.length; // number of rows of b

int m2 = a[0].length; // number of columns in a row of matrix a

int n2 = b[0].length; // matrix b

int[][] c = new int[m1][m2];

// number of rows of 1st matrix and second matrix are equal and also

// number of columns of 1st and 2nd matrices must be same.

if(m1 == n1 && m2 == n2) {

for(int i = 0;i < m1;i++) {

for(int j = 0;j < m2;j++) {

c[i][j] = a[i][j] + b[i][j];

}

}

}

else {

System.out.println("The addition is not possible.");

}

System.out.println("The sum of two arrays = ");

for(int i = 0;i < m1;i++) {

for(int j = 0;j < m2;j++) {

System.out.print(c[i][j]+"\t");

}

System.out.println();

}

}

}

**2) Matrix Multiplication**

**-----------------------------**

-> For the matrix multiplication, there is a condition:

if number of columns of matrix1 is equals to number of rows of matrix, then the matrix multiplication is possible.

otherwise, matrix multiplication is not possible.

Ex:

matrix1 ==> mxn matrix

matrix2 ==> nxp matrix

here:

number of columns of matrix1 ==> n

number of rows of matrix2 ==> n

n == n ==> multiplication is possible

Ex-2:

matrix1 ==> mxn matrix

matrix2 ==> pxq matrix

here:

number of columns of matrix1 ==> n

number of rows of matrix2 ==> p

n != p ==> multiplication is not possible.

-> The resultant matrix with the order of:

number-of-rows-matrix1 X number-of-columns-matrix2

Ex:

if matrix1 ==> mxn matrix

matrix2 ==> nxp matrix

result ==> m X p matrix

public class MatrixMultiplication {

public static void main(String[] args) {

int a[][] = {{1,2},{3,4},{5,6}}; // 3 X 2 matrix

int b[][] = {{1,2,3},{4,5,6}}; // 2 X 3 matrix

int m = a.length;

int n = a[0].length;

int p = b.length;

int q = b[0].length;

int c[][] = new int[m][q];

if(n == p) {

for(int i = 0;i < m;i++) {

for(int j = 0;j < p;j++) {

for(int k = 0;k < q;k++) {

c[i][k] += a[i][j] \* b[j][k];

}

}

}

}

else {

System.out.println("Matrix multiplication is not possible.");

}

for(int i = 0;i < c.length;i++) {

for(int j = 0;j<c[i].length;j++) {

System.out.print(c[i][j]+"\t");

}

System.out.println();

}

}

}

**Day-04**

**21-03-2025**

**===============**

Clarification about the below:

---------------------------------

m = a.length; // 3

n = a[0].length; // 3

p = b.length; // 3

q = b[0].length; // 3

n == p

for(int i = 0; i < m; i++) // row of a a[0], a[1], a[2]

{

for(int j = 0;j < p; j++)// column operation for a a[0][0], a[0][1], a[0][2],b[0]

{

for(int k = 0; k < q; k++) // b[0][0], b[1][0], b[2][0]

{

//c[i][k] += a[i][j] \* b[j][k];

c[i][k] = c[i][k] + (a[i][j] \* b[j][k]);

}

}

}

**Compound operator:**

**-------------------------**

x = 10

Sop(x); ==> 10

// x = x + 10;

x += 10;

Sop(x); ==> 20

+=, -=, \*=, /= %= ==> compound operators

-> When we can join arithmetic operators (+, -, \*, / and %) and bitwise operators (&, |, ^, ~, <<, >>, >>>) with assignment operator, we can get "compound operators".

-> When the same variable need to define in left and right side of the assignment operator, to simplify and to perform the same operation we can use "compound operator".

Ex:

a = 100

Sop(a); // 100

//a = a-10;

a -= 10;

Sop(a); // 90

**Matrix Transpose:**

**----------------------**

Syntax:

---------

a = m X n

b = n X m

for(int i = 0; i < m; i++)

{

for(int j = 0;j < n; j++)

{

b[j][i] = a[i][j];

}

}

import java.util.Arrays;

public class MatrixTranspose {

public static void main(String[] args) {

int a[][] = {{1,2,3},{4,5,6}};

int m = a.length;

int n = a[0].length;

int b[][] = new int[n][m];

for(int i = 0;i < m;i++){

for(int j = 0;j < n;j++) {

b[j][i] = a[i][j];

}

}

System.out.println("The Matrix a = ");

for(int i = 0;i < m;i++) {

for(int j = 0;j < n;j++) {

System.out.print(a[i][j]+"\t");

}

System.out.println();

}

System.out.println("The Matrix b after the Transpose = ");

for(int i = 0;i < n;i++) {

for(int j = 0;j < m;j++) {

System.out.print(b[i][j]+"\t");

}

System.out.println();

}

}

}

**Merging of two arrays into one:**

**-------------------------------------**

a = [1,2,3,4]; ==> size = 4 (m)

b = [1,2,3,4,5,6,7,8]; ==> size = 8 (n)

c = [1,2,3,4,1,2,3,4,5,6,7,8] ==> merging ==> size = 12

c[m+n]

import java.util.Arrays;

public class ArrayMerging {

public static void main(String[] args) {

int a[] = {1,3,5,7,9};

int b[] = {1,2,3,4,5,6,7,8,9,10};

int m = a.length;

int n = b.length;

int c[] = new int[m+n];

// writing a into c

for(int i = 0;i < m;i++) {

c[i] = a[i];

}

// writing of b into c

for(int i = 0; i < n;i++) {

c[m+i] = b[i];

}

System.out.println(Arrays.toString(c));

}

}

**Day-05**

**22-03-2025**

**====================**

**Q-1: Write a program using java**

**to find the sum of even elements and odd elements of the array.**

Solution:

---------

int a[] = {19,20,21,22,23,24,25,26,27};

a[0], a[2] % 2 != 0

odd = 19, 21

a[1],a[3] % 2 == 0

even = 20, 22

seven

sodd

Logic:

------

public class SumOfEvenAndOdd {

public static void main(String[] args) {

int a[] = {19,20,21,22,23,24,25,26,27};

int seven = 0,sodd = 0;

for(int i = 0;i < a.length;i++) {

if(a[i] % 2 == 0) {

seven += a[i];

}

else {

sodd = sodd + a[i];// sodd += a[i]

}

}

System.out.println("The Sum of Even elements = "+seven);

System.out.println("The Sum of Odd elements = "+sodd);

}

}

**Q-2: Write a program using java to accept an array as an input**

**and find the second largest element from the given array.**

Solution:

---------

Wrapper classes

int ==> 4-byte

-2^31 to 2^31 - 1

Integer.MIN\_VALUE

Integer.MAX\_VALUE

// finding of minimum and maximum of an integer

public class SecondLargest {

public static void main(String[] args) {

System.out.println("The Least value of integer = "+Integer.MIN\_VALUE);

System.out.println("The Highest value of integer = "+Integer.MAX\_VALUE);

}

}

a[] = {19,89,102,97,79,-1,-102,211};

large = Integer.MIN\_VALUE; //-2147483648

secondlarge = Integer.MIN\_VALUE; //-2147483648

a[0] > large

secondlarge = large;

large = a[0]; // 19

a[1] > large

secondlarge = large; //19

large = a[1]; //89

a[2] > large

secondlarge = large; // 89

large = a[2]; // 102

if(a[3] > large)

fails

else if(a[3] != large && a[3] > secondlarge)

secondlarge = a[3];// 97

if(a[4] > large)

fails

else if(a[4] != large && a[4] > secondlarge)

fails

a[5] > large

Logic:

------

public class SecondLargest {

public static void main(String[] args) {

int a[] = {19,89,102,97,79,-1,-102,211};

int large = Integer.MIN\_VALUE;

int secondLarge = Integer.MIN\_VALUE;

for(int i = 0; i < a.length;i++) {

if(a[i] > large) {

secondLarge = large;

large = a[i];

}

else if(a[i] != large && a[i] > secondLarge) {

secondLarge = a[i];

}

}

System.out.println("The Largest element = "+large);

System.out.println("The Second Largest element = "+secondLarge);

}

}

**Q-3: Write a program using java to accept an array as an input**

**and find the forth largest element from the given array.**

Solution:

---------

a[] = {19,89,102,97,79,-1,-102,211};

if(a.length < 4) ==> not possible

first = 0, second = 0, third = 0, forth = 0;

a[0] > first

forth = third // 89

third = second // 97

second = first // 102

first = a[0] // 211

a[1] > first

forth = third //0

third = second // 0

second = first // 19

first = a[1] // 89

a[2] > first

forth = third

third = second // 19

second = first // 89

first = a[2] // 102

if(a[3] > first)

fails

else if(a[3] != first && a[3] > second)

forth = third // 19

third = second // 89

second = a[3] // 97

if(a[4] > first)

fails

else if(a[4] != first && a[4] > second)

fails

else if(a[4] != second && a[4] > third)

fails

else if(a[4] != third && a[4] > forth)

forth = a[4] // 79

if(a[5] > first)

fails

else if(a[5] != first && a[5] > second)

fails

else if(a[5] != second && a[5] > third)

fails

else if(a[5] != third && a[5] > forth)

fails

Logic:

------

public class ForthLargest {

public static void main(String[] args) {

int a[] = {19,89,102,97,79,-1,-102,211};

int first,second,third,forth;

first = second = third = forth = 0;

for(int i = 0;i < a.length;i++) {

if(a[i] > first) {

forth = third;

third = second;

second = first;

first = a[i];

}

else if(a[i] != first && a[i] > second) {

forth = third;

third = second;

second = a[i];

}

else if(a[i] != second && a[i] > third) {

forth = third;

third = a[i];

}

else if(a[i] != third && a[i] > forth) {

forth = a[i];

}

}

System.out.println("The Forth Largest element in an array = "+forth);

}

}

Assignment:

-----------

1) Write a java program to create the array with even elements and create another array with odd elements by taking the array as input.

a[] = {19,20,21,22,23,24};

aeven = {20,22,24};

aodd = {19,21,23};

2) Write a java program to find the sum of All elements of the given array.

3) Write a java program to find the 3rd largest element of the array.

**Arrays**

**24-03-2025**

**================**

**For Each loop**

**----------------**

-> For loop can be defined in two ways:

1) normal for loop/traditional for loop

Syntax:

for(initialization; condition; update){

loop body statements;

}

2) For each loop/Enhanced for loop

Syntax:

for(datatype iterable: array-name){

for loop body statements;

}

Ex:

int[] a = {10,20,30,40,50};

for(int i : a){

Sop(i);

}

What is Iterable?

------------------

-> A variable which can be defined for loop operations (for/while) is called as "iterable".

-> the iterable in for each loop must be based on the type of the array.

Ex: if the array is integer type then, the iterable must be integer type.

Example - For Each Loop:

----------------------------

public class ForEachLoop {

public static void main(String[] args) {

char ch[] = {'a','e','i','o','u'};

String fruits[] = {"Apple","Banana","Mango","Kiwi","Jack-fruit"};

System.out.println("The Given character array is = ");

for(char c:ch) {

System.out.print(c+"\t");

}

System.out.println();

System.out.println("The Given String array is = ");

for(String s:fruits) {

System.out.print(s+"\t");

}

System.out.println();

}

}

**/\***

**\* Write a program in java to define an array.**

**\* And create the arrays with even elements and odd elements separately.**

**\*/**

import java.util.Arrays;

public class ArrayWithEvenAndOdd {

public static void main(String[] args) {

int a[] = {11,22,33,44,55,66,77,88,99,110};

int ecount = 0,ocount = 0;

for(int i:a) {

if(i % 2 == 0) {

ecount++;

}

else {

ocount++;

}

}

int[] evenArray = new int[ecount];

int[] oddArray = new int[ocount];

int eIndex = 0,oIndex = 0;

for(int i:a) {

if(i%2 == 0) {

evenArray[eIndex++] = i;

}

else {

oddArray[oIndex++] = i;

}

}

System.out.println("The Even Array = ");

System.out.println(Arrays.toString(evenArray));

System.out.println("The Odd Array = ");

System.out.println(Arrays.toString(oddArray));

}

}

**How arrays are mutable?**

**================**

Mutable and Immutable?

----------------------------

String S = "Java";

S[0] = 'J'

S[1] = 'a'

S[2] = 'v'

S[3] = 'a'

S[0] = 'j' ==> Error

int i = 10;

i = 20;

-> When the data cannot allowed for the modification after the definition is called as "Immutable"

Ex: All primitive datatypes, strings are immutable.

-> When the data can allowed for the modification after the definition is called as "Mutable".

Ex: Arrays.

Example for Mutable Vs Immutable:

----------------------------------------

import java.util.Arrays;

public class MutableImmutable {

public static void main(String[] args) {

String s = "Apple";

int[] a = {10,11,30,40,50};

System.out.println("The String = "+s);

System.out.println("The Array = "+Arrays.toString(a));

System.out.println(s.charAt(0));

// s.charAt(0) = 'a';

a[1] = 20;

System.out.println("The Array = "+Arrays.toString(a));

}

}

using sort():

-------------

sort() is a pre-defined method

can be used to arrange the elements of an array in ascending order.

Syntax:

import java.util.Arrays;

Arrays.sort(array-name);

Example for sort():

----------------------

import java.util.Arrays;

public class ArraySortMethod {

public static void main(String[] args) {

int[] a = {11,-100,0,22,-10,-1,99,77,79,97,27,19};

System.out.println("The Array without Sorting = ");

System.out.println(Arrays.toString(a));

Arrays.sort(a);

System.out.println("The Array with sorting = ");

System.out.println(Arrays.toString(a));

}

}

**Searching Techniques:**

**===============**

in two ways:

1) Linear search

2) Binary Search

**Linear Search:**

**-----------------**

Assume:

a[] = {11,22,33,44,55,66,77,88,99,110};

key = 77;

key == a[0]

key == a[1]

key == a[2]

**/\***

**\* Write a java program to define the array.**

**\* And also define the searching key.**

**\* And find whether the searching key is identified in the givne array**

**\* or not.**

**\* If identified print the position.**

**\* if not print "not identified".**

**\*/**

public class LinearSearch {

public static void main(String[] args) {

int a[] = {100,99,1,27,19,-10,-1,32,36,27,19};

int key = 109;

boolean flag = false;

int index = 0;

for(int i :a) {

if(i == key) {

System.out.println("The element "+key+" is found at:"+index);

flag = true;

break;

}

index++;

}

if(flag == false) {

System.out.println("The element "+key+" is not found.");

}

}

}

Assignment:

-----------

1) Write a java program to define the array as an input

and find the length of array without "length" property.

2) Write a java program to count number of even elements and odd elements from the given array.

**Day-06**

**25-03-2025**

**===================**

**/\***

**\* Write a java program to define the array.**

**\* And find the length of the given array**

**\* without length property.**

**\*/**

public class ArrayLength {

public static void main(String[] args) {

int a[] = {11,22,33,44,55,66,77,88,99,110};

int count = 0;

for(int i:a) {

count++;

}

System.out.println("The number of elements in the given array = "+count);

}

}

**Write a java program to define the array**

**and print only the element at the 2nd index while iterating on the array.**

public class ArrayLength {

public static void main(String[] args) {

int a[] = {11,22,33,44,55,66,77,88,99,110};

int count = 0;

int index = 0;

for(int i:a) {

if(index == 2) {

System.out.println(a[index]);

break;

}

index++;

}

// System.out.println("The number of elements in the given array = "+count);

}

}

**Binary Search:**

**------------------**

-> To overcome the drawback of linear search, we can use "binary search".

-> Drawback of linear search:

more iterations need to find the element

processing time can be increased.

-> As part of the binary search,

the whole array can be divided into two parts based on the specified condition. And the searching key can search either in left portion of the array or in right portion of the array.

Note:

-----

Binary search is not possible without sorting.

**/\***

**\* Write a java program to define the array**

**\* and also define the searching key**

**\* finally check whether the given searching key is present in the given array**

**\* or not using binary search.**

**\*/**

import java.util.Arrays;

public class BinarySearch {

static void binarySearch(int[] a,int k) {

Arrays.sort(a);

System.out.println("The sorted Array = "+Arrays.toString(a));

int low = 0;

int high = a.length - 1;

boolean flag = false;

while(low <= high) {

int mid = (low + high)/2;

if(a[mid] == k) {

System.out.println("Element Fuund.");

flag = true;

break;

}

else if(a[mid] > k) {

high = mid - 1;

}

else {

low = mid + 1;

}

}

if(flag == false) {

System.out.println("Element not found.");

}

}

public static void main(String[] args) {

int a[] = {19,27,32,26,53,47,39,37,35,28,30,3};

int key = 39;

binarySearch(a,key);

}

}

**/\***

**\* Write a java program to define the array**

**\* and print an array by removing the duplicates.**

**\*/**

import java.util.Arrays;

public class RemoveDuplicates {

public static void removeDuplicates(int[] a) {

Arrays.sort(a); //{0,0,1,1,1,1,1,2,2,3,3,5,7};

int i = 0;

for(int j = 1;j < a.length;j++) {

if(a[i] != a[j]) {

i++;

a[i] = a[j];

}

}

System.out.println("The array without duplicates = ");

for(int k = 0;k < i+1;k++) {

System.out.print(a[k]+"\t");

}

System.out.println();

}

public static void main(String[] args) {

int a[] = {1,0,1,2,3,1,0,1,3,5,7,1,2};

removeDuplicates(a);

}

}

**Day-07**

**26-03-2025**

**=========================**

**For each loop on multi-dimensional array**

**===========================**

-> suppose if you want to define for each loop/enhanced for loop on one-dimensional array:

syntax:

for(datatype iterable : array-name){

loop body

}

-> To define the for each loop on multi-dimensional array:

i) 2d-array:

-------------

syntax:

for(datatype[] iterable1 : 2d-array){

for(datatype iterable2 : iterable1){

action of statements;

}

}

Example program:

---------------------

public class ForEachOnTwoDimension {

public static void main(String[] args) {

int[][] a = {{1,2,3},{4,5,6},{7,8,9},{9,8,7}};

for(int[] i:a) {

for(int j:i) {

System.out.print(j+"\t");

}

System.out.println();

}

}

}

ii) 3d-array:

-------------

Syntax:

for(datatype[][] iterable1 : array-name){

for(datatype[] iterable2 : iterable1){

for(datatype iterable3 : iterable2){

some action of statements;

}

}

}

Example Program:

---------------------

public class ForEachOnThreeDimension {

public static void main(String[] args) {

char ch[][][] = {{{'a','b','c'},{'d','e','f'}},{{'g','h','i'},{'j','k','l'}},{{'m','n','o'},{'p','q','r'}}};

for(char[][] ch1:ch) {

for(char[] ch2:ch1) {

for(char ch3:ch2) {

System.out.print(ch3+"\t");

}

System.out.println();

}

System.out.println();

}

}

}

**Selection Sort**

**==========**

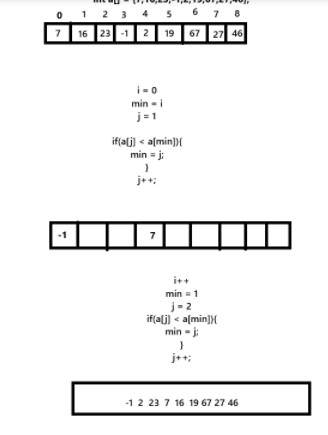
Definition:

------------

1) First find the minimum from the array on the basis of first index (0) then replace minimum value index to 0 and vice versa

2) Then find the minimum with the reference of next index (1) and replace the minimum element with index 1 element.

3) repeat the same for (n-1) number of times to sort the array.



import java.util.Arrays;

public class SelectionSort {

public static void main(String[] args) {

int[] a = {7,16,23,-1,2,19,67,27,46};

System.out.println("The Array before to sort = ");

System.out.println(Arrays.toString(a));

for(int i = 0;i < a.length - 1;i++) {

int min = i;

for(int j = i+1;j < a.length;j++) {

if(a[j] < a[min]) {

min = j;

}

}

int temp = a[i];// a[0]

a[i] = a[min]; // a[0] = a[3]

a[min] = temp; // a[3] = temp;

}

System.out.println("The Array after the sort = ");

System.out.println(Arrays.toString(a));

}

}

**Maximum number of occurrences of consecutive element.**

**{0,1,1,2,2,2,2,2,1,3,1,3,1,1,1}**

**======================================**

-> to write the logic for this,

no sorting is required.

public class MaximumOccurrencesConsecutively {

public static void main(String[] args) {

int a[] = {0,1,1,3,1,2,2,2,2,2,1,3,3,1,1,1};

int count = 1;

int k = 0;

int max = 0;

for(int i = 0;i < a.length-1;i++) {

if(a[i] == a[i+1]) {

count++;

}

else {

count = 1;

}

if(count > max) {

max = count;

k = a[i];

}

}

System.out.println("The Element which occurred consecutively = "+k);

System.out.println("The number of occurrences = "+max);

}

}