**DAY-06**

**10-05-2025**

**2D ARRAY OR 2DIMENSIONAL ARRAY**

It consists of rows and columns

int[][]arr=new int[3][]

1. sqaure bracket represent rows
2. Sqaure brackets represent coloums

When we declaring the array we have to declare no of rows and coloums numbers are optional

Each row will act as a individual array

Your Java program initializes a 2D array arr with dimensions 3x2, then reads input for a **triangular portion** of the array, and finally prints it.

import java.util.Scanner;

class Main{

public static void main(String[]args){

Scanner in=new Scanner(System.in);

int[][] arr=new int[3][2];

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

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

arr[i][j]=in.nextInt();

}

}

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

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

System.out.println(arr[i][j]);

}

System.out.println();

}

}

}

**Maximum Consecutive Ones:**

public class Main {

public static void main(String[] args) {

int[] arr = {1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1};

int maxCount = 0;

int currentCount = 0;

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

if (arr[i] == 1) {

currentCount++;

maxCount = Math.max(maxCount, currentCount);

} else {

currentCount = 0;

}

}

System.out.println("Maximum consecutive 1's: " + maxCount);

}

}

**Output**:

MAXIMUM CONSICUTIVE NUMBERS 1’S ARE: 3

**ARRAYS**

Arrays: Collection of similar kind of data in continuous Allocation:

->In c/c++ we have pointers so defnitely elements will be stored in continuous format

->In Java we dont have pointers then there is a chance of elements might store in random places

Syntax:

datatype [] arrname = {array elements};

->Primitive and Non-Primitive:

Primitive data-types can be splitted or divided,but non-primitive data-types can be splitted and divided

Examples(primitive):int,float,char....

Examples(non-primitive):all objects that were created by user,string....

Example:

Class Node{

int data;

Node next;

}

->All the objects will be stored in heap memory, all the references will be stored in stack memory

-> Stack memory having less space compare to heap memory.

Example:

import java.util.\*;

public class Main {

public static void main(String[] args) {

int a = 10;

int b = 20;

int c = 30;

int d = 40;

}

}

All the variables holding same data-type instead of creating multiple variables create an array to store those values : int[] arr = {10,20,30,40};

-> If we create multiple variables to get the data simple we print the variable

when it comes to array if we want to print the data we can print using index values and by default index values will start from 0

->As the heap memory is dynamic the very first value of an array will stored at random place we cant determine

int[] arr = {10,20,30,40}; //declaration and initialization happend at same line.

->To get the length of the array we have a method called "array.length" and in String it is "StringName.length()"

**Camel and Snake case:**

Camel case: Second word first letter should be capital

->arrName.length;

Snake case: separated with underscore "\_"

->String\_name.length();

In Java it is preferred to write in camel case to write production level codes

example(production-level code)://use clear meaningful method/function names

void additionOfTwoNumbers(){

}

\*\*\*Type of Printin Formats:\*\*\*

i)For-loop

import java.util.\*;

public class Main {

public static void main(String[] args) {

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

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

System.out.print(" "+arr[i]);

}

}

} //We can print array values using simple for-loop.

ii)For-Each loop

import java.util.\*;

public class Main {

public static void main(String[] args) {

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

for(int i:arr){

System.out.print(i+" ");

}

}

}

}

->In for-each loop we dont need index values

iii)using toString(); inbuilt-method

public class Main {

public static void main(String[] args) {

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

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

}

}

->toString() is an in-built method to print an array it will accept one argument and that is array itself

\*\*\*Declaring the size of an array and initializing the values later\*\*\*

->Array having fixed size if you declare an array Size it cant be changed

int[] arr = new int[5];//array reference is created

//before = array reference was created with the help of new keyword actual object will be created in the

//heap memory for that particular data-type and size

->if the array type is integer the default values will be zero's

-> if it is a string all the default value are NULL and NULL is a literal

// Example:

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int arr[] = new int[5];

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

//take the values from the user

arr[i] = sc.nextInt();

}

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

System.out.println(arr[i]);

}

// for(int i:arr){

// System.out.println(i);

// }

System.out.print(Arrays.toString(arr));

}

}

->Whenever you are trying to assign or trying to get nth value of array you will get an exception called

"Array out of bound"