**Java 2D Arrays - Simple Notes 📖**

A **2D array** in Java is an array of arrays, meaning it consists of multiple rows and columns. It's often used to represent matrices, tables, or grids.

**1️⃣ Declaration & Initialization**

**Declaration**

// Declaring a 2D array

datatype[][] arrayName;

**Initialization**

int[][] arr = new int[3][4]; // 3 rows, 4 columns

**Direct Initialization**

int[][] arr = {

{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12}

};

**2️⃣ Accessing Elements**

You can access elements using **row and column indexes**.

System.out.println(arr[1][2]); // Output: 7 (row index 1, column index 2)

**Modify an element:**

arr[2][1] = 20; // Changes value at row 2, column 1 to 20

**3️⃣ Iterating Through a 2D Array**

**Using Nested Loops**

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

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

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

}

System.out.println(); // New line after each row

}

**Using Enhanced for-loop**

for (int[] row : arr) {

for (int num : row) {

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

}

System.out.println();

}

**4️⃣ Common Operations**

**1. Transpose of a Matrix (Swap rows and columns)**

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

for (int j = i + 1; j < arr[i].length; j++) {

int temp = arr[i][j];

arr[i][j] = arr[j][i];

arr[j][i] = temp;

}

}

**2. Row-wise Sum**

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

int sum = 0;

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

sum += arr[i][j];

}

System.out.println("Sum of row " + i + " = " + sum);

}

**3. Column-wise Sum**

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

int sum = 0;

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

sum += arr[i][j];

}

System.out.println("Sum of column " + j + " = " + sum);

}

**5️⃣ Printing a 2D Array Properly**

**Using Arrays.toString()**

import java.util.Arrays;

for (int[] row : arr) {

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

}

**6️⃣ Jagged Arrays (Uneven Columns)**

A **jagged array** is a 2D array where rows have different numbers of columns.

int[][] jaggedArr = {

{1, 2},

{3, 4, 5},

{6}

};

Accessing and iterating is the same as a normal 2D array.

**7️⃣ Rotating a Matrix (90° Clockwise)**

1. **Transpose the matrix** (swap rows and columns).
2. **Reverse each row**.

// Step 1: Transpose

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

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

int temp = arr[i][j];

arr[i][j] = arr[j][i];

arr[j][i] = temp;

}

}

// Step 2: Reverse each row

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

for (int j = 0; j < arr.length / 2; j++) {

int temp = arr[i][j];

arr[i][j] = arr[i][arr.length - 1 - j];

arr[i][arr.length - 1 - j] = temp;

}

}

**8️⃣ Searching in a 2D Array**

**Linear Search**

int target = 7;

boolean found = false;

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

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

if (arr[i][j] == target) {

System.out.println("Element found at (" + i + "," + j + ")");

found = true;

break;

}

}

}

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

**Binary Search (Only for Sorted 2D Arrays)**

public static boolean binarySearch(int[][] matrix, int target) {

int rows = matrix.length, cols = matrix[0].length;

int left = 0, right = rows \* cols - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int midVal = matrix[mid / cols][mid % cols]; // Convert 1D index to 2D

if (midVal == target) return true;

else if (midVal < target) left = mid + 1;

else right = mid - 1;

}

return false;

}

**9️⃣ Sorting a 2D Array Row-wise**

import java.util.Arrays;

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

Arrays.sort(arr[i]); // Sorts each row individually

}

**🔟 Summary Table**

| **Operation** | **Code** |
| --- | --- |
| **Create 2D array** | int[][] arr = new int[3][4]; |
| **Initialize with values** | int[][] arr = {{1,2}, {3,4}}; |
| **Access element** | arr[1][1] |
| **Print 2D array** | Arrays.toString(row) |
| **Transpose Matrix** | Swap arr[i][j] and arr[j][i] |
| **Row-wise sum** | Loop through each row |
| **Column-wise sum** | Loop through each column |
| **Rotate 90° Clockwise** | Transpose + Reverse each row |
| **Search in 2D Array** | Nested loop or Binary Search |
| **Sort Row-wise** | Arrays.sort(arr[i]) |

These are some of the **basic** and **important** operations with **2D arrays in Java**! 🚀 Let me know if you need any clarification! 😊