Lesson Plan

Multidimensional Array

PW SKILLS

Pre-requisites:

- JAVA syntax, for loops
- 1D arrays

List of concepts involved:

- Multidimensional arrays
- Taking input of 2D array
- Write a program to store roll numbers and marks obtained by 4 students side by side in a matrix.
- Write a program to find the largest element of a given 2D array of integers.
- Write a program to print the sum of all the elements of a 2D matrix.
- Write a program to add two matrices.
- Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.
- Write a program to change the given matrix with its transpose.
- Write a program to rotate the matrix by 90 degrees clockwise.

Multidimensional array: Array of arrays is known as multidimensional arrays.

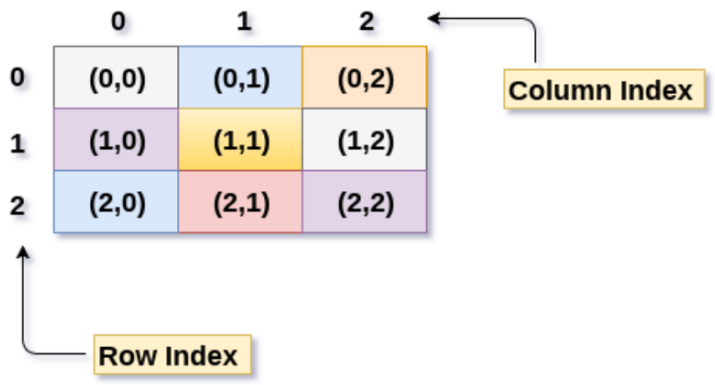
Syntax to declare a N Dimensional array:

data_type[1st dimension][2nd dimension][]..[Nth dimension] array_name = new data_type[size1][size2]....[sizeN];

Syntax to declare a 2Dimensional array of type int:
int[][] arr = new int[rows][column];
where rows imply the number of rows needed for the 2D array and

column implies the number of columns needed.
int[][] arr = new int[4][5];

Here, arr is a two-dimensional array. It can hold a maximum of 20 elements of integer type.
We can think of this array as a table with 4 rows and each row has 5 columns as shown below.



Looping through 2D arrays:

Suppose you want to store 10 at every index of a 2D array of dimensions 4 X 5, you can do so using the following code:

```
for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 5; j++) {
        arr[i][j] = 10;
    }
}
```

Taking 2D array as input from the user:

Following code will show how we can take a 2D array as input from the user.

Code:

```
import java.util.Scanner;
import java.io.*;

public class Main {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);


        System.out.println("Enter the no of rows : ");
        int row = sc.nextInt();
        System.out.println("Enter the number of columns : ");
        int col = sc.nextInt();
        int arr[][] = new int[row][col];
        int total = row * col;


        System.out.println("Please enter " + total + " elements
nows.");


        // read array elements row wise.
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                arr[i][j] = sc.nextInt();
            }
        }
```

```
        // close the scanner
        sc.close();
        System.out.println("The Input array is :");
        for (int i = 0; i < row; i++) {
            for (int j = 0; j < col; j++) {
                System.out.print(arr[i][j] + "\t");
            }
            System.out.println();
        }
    }
}
```

```
Enter the no of rows :
3
Enter the number of columns :
4
Please enter 12 elements nows.
1 2 3 4
4 5 6 7
5 6 7 8
The Input array is :
1       2       3       4
4       5       6       7
5       6       7       8
```

Q1. Write a program to find the largest element of a given 2D array of integers.

- The idea is to traverse the matrix using two nested loops, one for rows and one for columns, and find the maximum element.
- Initialize a variable **maxElement** with a minimum value and traverse the matrix and compare every time if the current element is greater than a **maxElement.**
- If yes then update **maxElement** with the current element.

Below is the implementation of the above approach:

```
// Java code to find max element in a matrix

public class Main{

    final static int N = 4;
    final static int  M = 4 ;
```

```
    // Function to find max element
    // mat[][] : 2D array to find max element
    static int findMax(int mat[][])
    {

        // Initializing max element as INT_MIN
        int maxElement = Integer.MIN_VALUE;

        // checking each element of matrix
        // if it is greater than maxElement,
        // update maxElement
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                if (mat[i][j] > maxElement) {
                    maxElement = mat[i][j];
                }
            }
        }

        // finally return maxElement
        return maxElement;
    }

    public static void main(String args[])
    {
           // matrix
        int mat[][] = { { 1, 2, 3, 4 },
                        { 25, 6, 7, 8 },
                        { 9, 10, 11, 12 },
                        { 13, 14, 15, 16 } };

        System.out.println(findMax(mat)) ;

    }
}
```

Output: 25

Time Complexity: O(N*M), where N and M are the numbers of rows and columns of the given matrix.
Auxiliary Space: O(1), As constant extra space is used.

Q2. Given a 2D array of order M * N, the task is to find out the sum of elements of the matrix.

Examples:

Input: array[2][2] = {{1, 2}, {3, 4}};
Output: 10

Input: array[4][4] = {{1, 2, 3, 4},
{5, 6, 7, 8},
{9, 10, 11, 12},
{13, 14, 15, 16}};
Output: 136

Approach: The sum of each element of the 2D array can be calculated by traversing through the matrix and adding up the elements.

Below is the implement the above approach-

```
// Java code for the above approach
import java.io.*;

class Main{
  // Get the size m and n
  static int M = 4;
  static int N = 4;

  // Function to calculate sum
  // of elements in 2d array
  static int sum(int arr[][])
  {
    int i, j;
    int sum = 0;

    // Finding the sum
    for (i = 0; i < M; ++i) {
      for (j = 0; j < N; ++j) {
        // Add the element
        sum = sum + arr[i][j];
      }
    }
    return sum;
  }

  public static void main (String[] args)
  {
    int i, j;
    int arr[][]= new int[M][N];

    // Get the matrix elements
    int x = 1;
    for (i = 0; i < M; i++)
      for (j = 0; j < N; j++)
        arr[i][j] = x++;
```

```
    // Get sum
    System.out.println(sum(arr));
  }
}
```

Output: 136

Time Complexity: O(M*N)
Auxiliary Space: O(1)

Q3. Given two N x M matrices. Find a N x M matrix as the sum of given matrices each value at the sum of values of corresponding elements of the given two matrices.

Approach: Below is the idea to solve the problem.

- Iterate over every cell of matrix (i, j), add the corresponding values of the two matrices and store in a single matrix i.e. the resultant matrix.

Follow the below steps to Implement the idea:

- Initialize a resultant matrix res[N][M].
- Run a for loop for counter i as each row and in each iteration:
- Run a for loop for counter j as each column and in each iteration:
- Add values of the two matrices for index i, j and store in res[i][j].
- Return res.
- Below is the Implementation of the above approach.

```
class Main {
    static final int N = 4;

    // This function adds A[][] and B[][], and stores
    // the result in C[][]
    static void add(int A[][], int B[][], int C[][])
    {
        int i, j;
        for (i = 0; i < N; i++)
            for (j = 0; j < N; j++)
                C[i][j] = A[i][j] + B[i][j];
    }

    // Driver code
    public static void main(String[] args)
    {
        int A[][] = { { 1, 1, 1, 1 },
                      { 2, 2, 2, 2 },
                      { 3, 3, 3, 3 },
                      { 4, 4, 4, 4 } };
```

```
        int B[][] = { { 1, 1, 1, 1 },
                      { 2, 2, 2, 2 },
                      { 3, 3, 3, 3 },
                      { 4, 4, 4, 4 } };

        // To store result
        int C[][] = new int[N][N];
        int i, j;
        add(A, B, C);

        System.out.print("Result matrix is \n");
        for (i = 0; i < N; i++) {
            for (j = 0; j < N; j++)
                System.out.print(C[i][j] + " ");
            System.out.print("\n");
        }
    }
}
```

Output:

Result matrix is
2 2 2 2
4 4 4 4
6 6 6 6
8 8 8 8

Time Complexity: O(n^2).
Auxiliary Space: O(n^2). since n^2 extra space has been taken for storing results

Q4. Write a program to print the transpose of the matrix entered by the user and store it in a new matrix.

Transpose of a matrix is the process of swapping the rows to columns. For 2x3 matrix,

Matrix
a11 a12 a13
a21 a22 a23

Transposed Matrix
a11 a21
a12 a22
a13 a23

Example: Program to Find Transpose of a Matrix

```
public class Transpose {

    public static void main(String[] args) {
        int row = 2, column = 3;
        int[][] matrix = { {2, 3, 4}, {5, 6, 4} };

        // Display current matrix
        display(matrix);

        // Transpose the matrix
        int[][] transpose = new int[column][row];
        for(int i = 0; i < row; i++) {
            for (int j = 0; j < column; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }

        // Display transposed matrix
        display(transpose);
    }

    public static void display(int[][] matrix) {
        System.out.println("The matrix is: ");
        for(int[] row : matrix) {
            for (int column : row) {
                System.out.print(column + "    ");
            }
            System.out.println();
        }
    }
}
```

Output:

The matrix is:
2 3 4
5 6 4

The matrix is:
2 5
3 6
4 4

In the above program, display() function is only used to print the contents of a matrix to the screen.

Here, the given matrix is of form 2x3, i.e. row = 2 and column = 3.

For the transposed matrix, we change the order of transposed to 3x2, i.e. row = 3 and column = 2. So, we have transpose = int[column][row]

The transpose of the matrix is calculated by simply swapping columns to rows:

transpose[j][i] = matrix[i][j];

Q5. Write a program to change the given matrix with its transpose.

- This approach works only for square matrices (i.e., – where no. of rows are equal to the number of columns).
- This algorithm is also known as an "in-place" algorithm as it uses no extra space to solve the problem.

Follow the given steps to solve the problem:

- Run a nested loop using two integer pointers i and j for 0 <= i < N and i+1 <= j < N
- Swap A[i][j] with A[j][i]

Below is the implementation of the above approach:

```
class Main {
    static final int N = 4;

    // Finds transpose of A[][] in-place
    static void transpose(int A[][])
    {
        for (int i = 0; i < N; i++)
            for (int j = i + 1; j < N; j++) {
                int temp = A[i][j];
                A[i][j] = A[j][i];
                A[j][i] = temp;
            }
    }
    public static void main(String[] args)
    {
        int A[][] = { { 1, 1, 1, 1 },
                      { 2, 2, 2, 2 },
                      { 3, 3, 3, 3 },
                      { 4, 4, 4, 4 } };

        transpose(A);

        System.out.print("Modified matrix is \n");
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++)
                System.out.print(A[i][j] + " ");
            System.out.print("\n");
        }
    }
}
```

Output:

Modified matrix is
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4

Time Complexity: O(N^2).
Auxiliary Space: O(1)

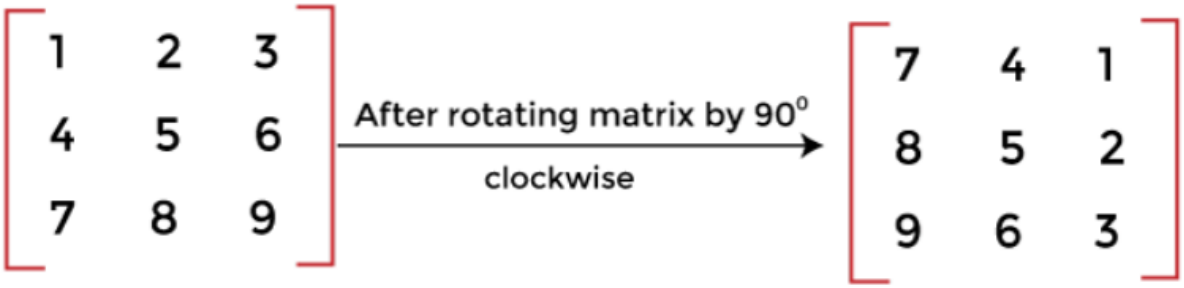
Q6. Write a program to rotate the matrix by 90 degrees clockwise.

The rotation of a matrix involves two steps:

- First, find the transpose of the given matrix.
- Swap the elements of the first column with the last column (if the matrix is of 3*3). The second column remains the same.

It is an efficient solution.

Note: Matrix must have the same number of rows and columns.



Let's understand through an example. Suppose, the matrix is:Let's find the transpose of the matrix.

$$\begin{bmatrix} 5 & 12 & 45 \\ 9 & 23 & 8 \\ 10 & 34 & 19 \end{bmatrix}$$

To get the rotated matrix, swap the first column with the last column.

$$\begin{bmatrix} 45 & 12 & 5 \\ 8 & 23 & 9 \\ 19 & 34 & 10 \end{bmatrix}$$

The above matrix is rotated by 90 degrees.

If the given matrix is 4*4 matrix, swap the first column with the last column and the second column with the third column. For example, consider the following figure.

Step 1: Find the transpose of the matrix.

```
for(int i=0; i<matrix.size(); i++)
{
for(int j=i; j<matrix[i].size(); j++)
{
//checks if i is not equal to j because in transpose matrix
diagonal elements will not swap
if(i≠j)
{
//swapping elements
int temp = matrix[i][j];
matrix[i][j]=marix[j][i];
matrix[j][i]=temp;
}
}
}
```

Step 2: Swap the elements.

```
for(int i=0; i<n; i++)  //n is the number of rows
{
for(int j=i; j<c/2; j++)   //c is the number of columns
{
if(i≠j)
{
//swapping elements of the first column with last column
int temp = matrix[i][j];
matrix[i][j]=marix[i][c-j-1];
matrix[i][c-j-1]=temp;
}
}
}
```

Let's implement the above logic in a Java program.

RotateMatrix.java

```
public class RotateMatrix
{
//function to rotate the matrix by 90 degrees clockwise
static void rightRotate(int matrix[][],int n)
{
//determines the transpose of the matrix
for(int i=0;i<n;i++)
{
for(int j=i;j<n;j++)
{
int temp = matrix[i][j];
matrix[i][j] = matrix[j][i];
matrix[j][i] = temp;
}
}
//then we reverse the elements of each row
for(int i=0;i<n;i++)
{
//logic to reverse each row i.e 1D Array.
int low = 0, high = n-1;
while(low < high)
{
int temp = matrix[i][low];
matrix[i][low] = matrix[i][high];
matrix[i][high] = temp;
low++;
high--;
}
}

System.out.println("The Right Rotated Matrix is: ");
//prints matrix after rotation
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
System.out.print(matrix[i][j]+" " +"\t");
}
System.out.println();
}
}
//driver code
public static void main(String args[])
{
int n=3;
//initializing a 3*3 matrix
int matrix[][] = new int[][]{ {1,2,3}, {4,5,6} , {7,8,9} };
System.out.println("The Original Matrix is: ");
```

```
//prints matrix
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
//prints the elements of the matrix
System.out.print(matrix[i][j]+" " +"\t");
}
System.out.println();
}
//function calling
rightRotate(matrix, n);
}
}
```

Output: