Two Dimensional Arrays

Introduction

- A 2D array is a combination of many 1D arrays.
- It represents a table/matrix with rows and columns of data.
- In this type of array, the position of a data element is referred to by two indices instead of one.

Consider an example of recording temperatures 4 times a day, 4 days in a row. Such data can be presented in a 2D array as below.

```
Day 1 - 11, 12, 5, 2
Day 2 - 15, 6, 10, 6
Day 3 - 10, 8, 12, 5
Day 4 - 12, 15, 8, 6
```

The above data can be represented as a 2D array as below.

```
int[][] T = {{11, 12, 5, 2}, {15, 6, 10, 6}, {10, 8, 12, 5}, {12, 15, 8, 6}};
```

Accessing Values in a Two Dimensional Array

The data elements in a two-dimensional array can be accessed using two indices. One index referring to the main or parent array(inner array) and another index referring to the position of the data element in the inner array. If we mention only one index then it means we are talking about the entire inner array for that index position.

The example below illustrates how it works.

```
int[][] T = {{11, 12, 5, 2}, {15, 6, 10, 6}, {10, 8, 12, 5}, {12, 15, 8, 6}};
System.out.println(T[0]); //array at index 0 in T
```

```
System.out.println(T[1][2]); //Element at index 2 in array, at index
                             // 1 in T
```

When the above code is executed, it produces the following result −

```
[I@6d06d69c                  // address where the 1D array is stored
10
```

Updating Values in Two Dimensional Array

We can update the entire inner list or some specific data elements of the inner list by reassigning the values using the list index.

```
int[][] T = {{11, 12, 5, 2}, {15, 6, 10, 6}, {10, 8, 12, 5}, {12, 15, 8, 6}};
```

This can be done the same way as in 1D arrays. Consider the examples given below:

```
T[0][1]= 1 // Update element at index 1 of array at index 0 of T to 1
T[1][1]= 7 // similarly at 1,1 to 7
```

This would modify the list as:

```
int[][] T = {{11, 1, 5, 2}, {15, 7, 10, 6}, {10, 8, 12, 5}, {12, 15, 8, 6}};
```

Printing a Two Dimensional Array

We can use nested `for` loops to print a 2D List. The outer loop iterates over the inner lists and the inner nested loop prints the elements of the lists. This can be shown as follows:

```
int[][] T= {{11,12,5,2},{15,6,10,6},{0,5,11,13},{10,8,12,5},{12,15,8,6}};
for(int i=0; i<T.length(); i++){ //Every row in T is an array
    for(int j=0; j<T[i].length(); j++){ //Every col of the array row
                                        //is element
```

```
        System.out.print(T[i][j] + " "); //Print element
    }
    System.out.println(); //New Line
}
```

The above code will print T as:

```
11 12 5 2
15 6 10 6
0 5 11 13
10 8 12 5
12 15 8 6
```

Input Of Two Dimensional Arrays

For this we will require the user to specify the number of rows (say `row`) and columns (say `col`) of the 2D array. The user will then enter all the elements of the 2D list in a single line. The java code for this can be written as follows:

```
public static void main(String[] args) {
     Scanner s = new Scanner(System.in);
     int row = s.nextInt();
     int col = s.nextInt();
     int[][] input = new int[row][col];

     for(int i=0; i<row; i++){ //Every row in T is an array
        for(int j=0; j<col; j++){ // each col of row
           input[i][j] = s.nextInt(); //take input
        }

     }
}
```

User Input:

```
4
5
11 12 5 2 15
```

```
6 10 6 0 5
11 13 10 8 12
5 12 15 8 6
```

Now this given input can be printed using previous code for printing the 2D array.

How 2D Arrays are Stored?

When we define this:

```
int[][] T = {{11, 12, 5, 2}, {15, 6, 10, 6}, {10, 8, 12, 5}, {12, 15, 8, 6}};
```

1. Here, T is a variable which stores the address of the 1D array of type array.
2. Further each index of this 1D also contains the address of arrays of type int. And finally these int arrays will contain the integers which are stored by the user.

Problem Statement - Wave print

Given a 2D array, you need to print in this 2D array in a wave fashion. Wave fashion means in a sine wave order, that means, 1st column needs to be printed from top to bottom, then 2nd column needs to be printed from bottom to top, then again next column from top to bottom and so on until we reach the last column.

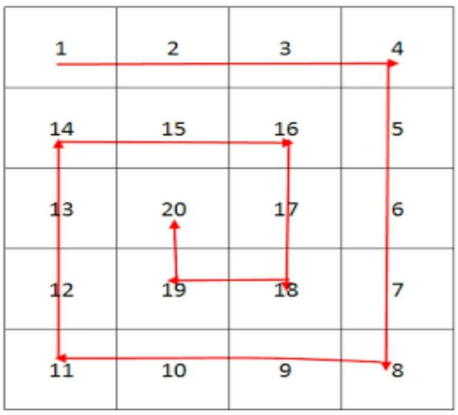
Approach

1. Traverse the 2D array column-wise.
2. Try to keep track of the column which you are printing:
    - If the column number is even, move from top to bottom, i.e., start row from 0 to end.
    - Else if the column is odd, move from bottom to top, i.e., start row number from end and decrement it until it reaches zero.

You may refer to the solution tab for the solution code.

Problem Statement - Spiral print

Given an NxM 2D array, you need to print it in a spiral fashion. See below shown diagram for better understanding



Output : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Approach

1. There needs to be a total of NxM elements printed.
2. The problem can be solved by dividing the matrix into loops or squares or boundaries.
3. It can be seen that the elements of the outer loop are printed first in a clockwise manner then the elements of the inner loop are printed. So printing the elements of a loop can be solved using four loops which prints all the elements.
4. Every ‘for’ loop defines a single direction movement along with the matrix. The first for loop represents the movement from left to right, whereas the second crawl represents the movement from top to bottom, the third represents the movement from the right to left, and the fourth represents the movement from bottom to up.

5. Since there are NxM elements which need to be printed. So, we can wrap all the above code in one single for loop and let it run for until the loop prints NxM elements.

You may refer to the solution tab for the solution code.

Jagged Arrays

Till now we have seen 2D arrays with an equal number of columns in all rows. Jagged arrays are the two-dimensional arrays with a variable number of columns in various rows. Some examples of jagged arrays are shown below:

```
int[][] T1 = {{11,12,5,2},{10,6},{10,8,12},{12}};
int[][] T2 = {{1,2,3}, {1}, {3}, {1,2}};
int[][] T3 = {{1},{},{1,2,1},{0}};
```

The elements in a jagged list can be accessed the same way as shown above using valid indices.

Note: Keep in mind the range of columns for various rows. Indexes out of the range can produce `indexOutOfBound` error.

Java code for taking input of Jagged Arrays

```
public static void main(String[] args) {
     Scanner s = new Scanner(System.in);
     int rows = s.nextInt();
     int[][] input = new int[row][]; // we don't define columns here
                                     // as this will vary for each column

     for(int i=0; i<rows; i++){ //Every row in T is an array
        int cols = s.nextInt(); // number of columns for this row
        for(int j=0; j<cols; j++){ // each col of row
           input[i][j] = s.nextInt(); //take input
        }
```

```
        }
}
```

You can now define the total number of rows and for each row you first need to define the number of columns before entering the elements in that particular column.

Practice Questions

Using above logics for traversal and moving in 2D array, here are few problems for your practice:

- Print row-wise sum
- Find the largest row or column when compared using sum.

See in codezen for a detailed explanation of the problem.