K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A1	**Roll No.: 16010123012**
Experiment / assignment / tutorial No. 4	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE : Multi-dimensional Arrays (Jagged Array)

AIM: Write a program which stores information about n players in a two dimensional array. The array should contain the number of rows equal to the number of players. Each row will have a number of columns equal to the number of matches played by that player which may vary from player to player. The program should display player number (index +1), runs scored in all matches and its batting average as output. (It is expected to assign columns to each row dynamically after getting value from the user.)

Expected OUTCOME of Experiment:

CO1:Apply the features of object oriented programming languages. (C++ and Java)

CO2:Explore arrays, vectors, classes and objects in C++ and Java

Books/ Journals/ Websites referred:

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

Pre Lab/ Prior Concepts:

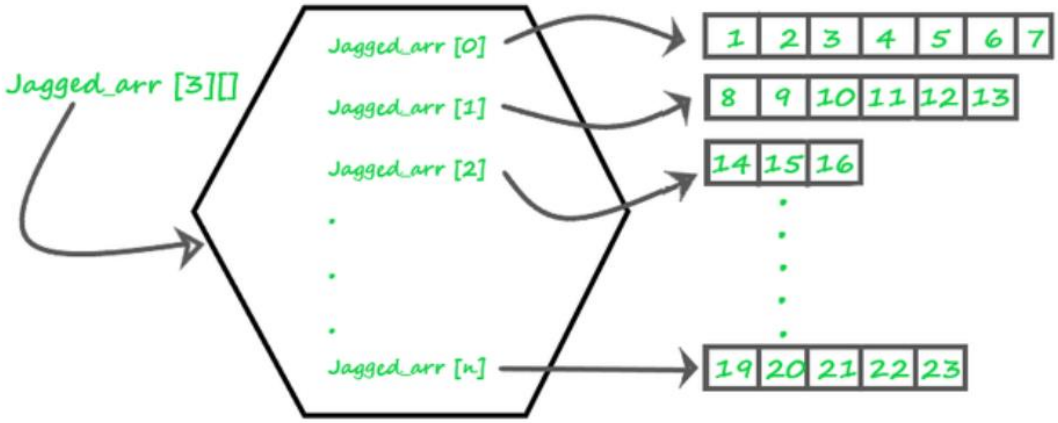
Arrays

Multi-Dimensional Array:

10 12 43 11 22
20 45 56 1 33
30 67 32 14 44
40 12 87 14 55
50 86 66 13 66

60 53 44 12 11
A multi-dimensional array is one that can hold all the values above. You set them up like this:
int[ ][ ] numbers = new int[6][5];
The first set of square brackets is for the rows and the second set of square brackets is for the columns. In the above line of code, we're telling Java to set up an array with 6 rows and 5 columns.
aryNumbers[0][0] = 10;
aryNumbers[0][1] = 12;
aryNumbers[0][2] = 43;
aryNumbers[0][3] = 11;
aryNumbers[0][4] = 22;
So the first row is row 0. The columns then go from 0 to 4, which is 5 items.
Jagged Array:

A jagged array, also known as a "ragged array," is an array of arrays where each "inner" array can have different lengths. This contrasts with a rectangular array (or a multi-dimensional array), where every inner array must have the same length. Jagged arrays are useful when dealing with data structures that naturally vary in size, such as lists of lists or matrices with different numbers of columns.



Theory

1. **Definition**: A jagged array is an array whose elements are arrays, possibly of different lengths. This means that the length of each inner array can vary.
2. **Memory Layout**: Unlike a rectangular array where memory allocation is continuous, each inner array in a jagged array is a separate array stored at different locations in memory.

3. **Usage**: Jagged arrays are often used in scenarios where the data is inherently irregular. For example, they can be useful in representing data structures like adjacency lists in graphs, where different nodes have different numbers of neighbors.
4. **Advantages**:
   - **Space Efficiency**: Only the required space is allocated for each sub-array, saving memory when dealing with irregular data.
   - **Flexibility**: Allows more flexibility in managing arrays of varying lengths.
5. **Disadvantages**:
   - **Complexity**: Increased complexity in managing and accessing elements.
   - **Performance**: Potentially lower performance due to non-contiguous memory allocation

Syntax :

```
// Declare a jagged array with 3 elements
int[][] jaggedArray = new int[3][];
```

Algorithm:

1. Input Number of Players:

2. Initialize Jagged Array:

   - Creating a 2D jagged array 'runs' with 'NumberOfPlayers' rows. Each row will represent a player's matches.

3. Collect Data for Each Player:

   - For each player (loop from 0 to 'NumberOfPlayers - 1'):

     1. Input Matches Played:

        - Initialize the 'runs' array for the current player with the size equal to 'matches'.

     2. Input Runs for Each Match:

        - For each match (loop from 0 to 'matches - 1'):

          - Read and store the runs in the 'runs' array for the current player.

4. Display Player Statistics:

   - For each player (loop from 0 to NumberOfPlayers - 1):

1. Calculate Sum of Runs:

   - For each match (loop through the 'runs' array for the current player):

     - Print the runs scored in the current match.

2. Calculate and Print Average Runs:

   - Calculate the average runs by dividing `sum` by the number of matches.

Implementation details:

```
import java.util.Scanner;

public class JaggedArray {
    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter number of players: ");
      int NumberOfPlayers = sc.nextInt();

      int[][] runs = new int[NumberOfPlayers][];
      for(int i = 0; i < NumberOfPlayers; i++) {
        System.out.print("Enter number of matches played by player " + (i
+ 1) + ": ");
        int matches = sc.nextInt();
        runs[i] = new int[matches];

        for (int j = 0; j < matches; j++) {
          System.out.print("Enter runs scored by player " + (i + 1) + "
in match " + (j + 1) + ": ");
          runs[i][j] = sc.nextInt();
        }
      }
      System.out.println("\nPlayer Stats:");
        for (int i = 0; i < NumberOfPlayers; i++) {
            System.out.print("Player " + (i + 1) + ": Runs - ");

            int sum = 0;
            for (int j = 0; j < runs[i].length; j++) {
                System.out.print(runs[i][j] + " ");
                sum += runs[i][j];
            }
            float average = (float) sum / runs[i].length;
```

```
            System.out.printf("\tAverage: %.2f%n", average);
        }
    }
}
```

Output:

```
PS D:\Java\A1\Exp4> javac JaggedArray.java
PS D:\Java\A1\Exp4> java JaggedArray
Enter number of players: 3
Enter number of matches played by player 1: 4
Enter runs scored by player 1 in match 1: 23
Enter runs scored by player 1 in match 2: 12
Enter runs scored by player 1 in match 3: 17
Enter runs scored by player 1 in match 4: 35
Enter number of matches played by player 2: 3
Enter runs scored by player 2 in match 1: 34
Enter runs scored by player 2 in match 2: 65
Enter runs scored by player 2 in match 3: 7
Enter number of matches played by player 3: 3
Enter runs scored by player 3 in match 1: 43
Enter runs scored by player 3 in match 2: 78
Enter runs scored by player 3 in match 3: 50

Player Stats:
Player 1: Runs - 23 12 17 35    Average: 21.75
Player 2: Runs - 34 65 7        Average: 35.33
Player 3: Runs - 43 78 50       Average: 57.00
```

```
PS D:\Java\A1\Exp4> javac JaggedArray.java
PS D:\Java\A1\Exp4> java JaggedArray
Enter number of players: 3
Enter number of matches played by player 1: 2
Enter runs scored by player 1 in match 1: 45
Enter runs scored by player 1 in match 2: 87
Enter number of matches played by player 2: 2
Enter runs scored by player 2 in match 1: 32
Enter runs scored by player 2 in match 2: 56
Enter number of matches played by player 3: 2
Enter runs scored by player 3 in match 1: 65
Enter runs scored by player 3 in match 2: 10

Player Stats:
Player 1: Runs - 45 87  Average: 66.00
Player 2: Runs - 32 56  Average: 44.00
Player 3: Runs - 65 10  Average: 37.50
```

Conclusion: We have successfully completed this experiment and learned how to create multidimensional or jagged arrays in java.

Date: 20/08/2024 **Signature of faculty in-charge**

Post Lab Descriptive Questions:

Q.1 Write a program for Given an array arr[] of size N. The task is to find the sum of the contiguous subarray within a arr[] with the largest sum.

```
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the size of the array: ");
      int n = sc.nextInt();
      int[] arr = new int[n];
      System.out.println("Enter the elements of the array:");
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
      }
      int max = arr[0];
      int current = arr[0];
      for (int i = 1; i < n; i++) {
```

```
            current = Math.max(arr[i], current + arr[i]);
            max = Math.max(max, current);
        }
        System.out.println("Maximum sum of the contiguous subarray is: " + max);
    }
}
```

```
Enter the size of the array: 9
Enter the elements of the array:
1 2 3 4 5 6 7 8 9
Maximum sum of the contiguous subarray is: 45

=== Code Execution Successful ===
```

Q.2.Create a jagged array of integers. This array should consist of two 2-D arrays. First 2-D array should contain 3 rows having length of 4, 3, and 2 respectively. Second 2-D array should contain 2 rows with length 3 and 4 respectively.

```
class arr {
    public static void main(String[] args) {
    int arr1[][]=new int[3][];
    arr1[0]=new int[4];
    arr1[1]=new int[3];
    arr1[2]=new int[2];
    int arr2[][]=new int[2][];
    arr2[0]=new int[3];
    arr2[1]=new int[4];
    System.out.println("Array 1");
    for(int i=0;i<arr1.length;i++){
    for(int j=0;j<arr1[i].length;j++)
    System.out.print(arr1[i][j]+" ");
    System.out.println();
    }
    System.out.println("\nArray 2");
    for(int i=0;i<arr2.length;i++){
    for(int j=0;j<arr2[i].length;j++)
    System.out.print(arr2[i][j]+" ");
    System.out.println();
    }
    }
```

}

```
Array 1
0 0 0 0
0 0 0
0 0

Array 2
0 0 0
0 0 0 0

=== Code Execution Successful ===
```

Q.3. Consider the following code

int number[] = new int[5];

After execution of this statement, which of the following are true?

(A) number[0] is undefined

(B) number[5] is undefined

(C) number[4] is null

(D) number[2] is 0

(E) number.length() is 5

(i) (C) & (E)

(ii) (A) & (E)

(iii) (E)

(iv) (B), (D) & (E)

Ans: (iv) (B), (D) & (E)