**Batch: C3 Roll No.: 16010123217**

**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**];**

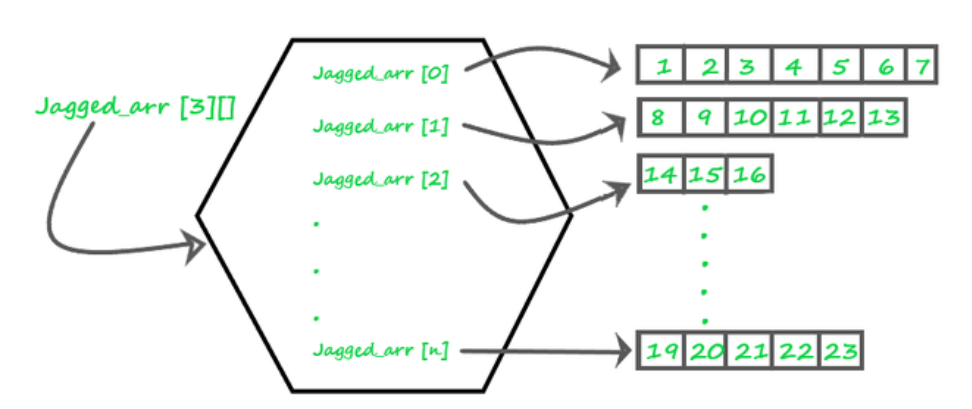
The first set of square brackets is for the rows and the second set of square brackets is for the columns. In the above line of code, we're telling Java to set up an array with 6 rows and 5 columns.

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][];**

**Class Diagram:**

Class Name:

PlayerStats

Attributes:

None (as the class doesn't have instance variables; all variables are local to the main method).

Methods:

main(String[] args) : void — The entry point of the program that contains the logic for storing and processing the players' runs and calculating their batting averages.

**Algorithm:**

1. **Start**

2. **Initialize Scanner**

* Create a Scanner object s1 to read input from the user.

3. **Input Number of Players**

* Prompt the user to enter the number of players n.
* Store this value in an integer variable n.

4. **Initialize Jagged Array**

* Declare a 2D jagged array runs with n rows.

5. **Input Runs for Each Player**

* **For** each player i from 0 to n-1:
  1. **Input Number of Matches**:
     + Prompt the user to enter the number of matches played by player i+1.
     + Store this value in an integer variable matches.
  2. **Allocate Array for Matches**:
     + Initialize the ith row of runs with a new array of size matches.
  3. **Input Runs for Each Match**:
     + **For** each match j from 0 to matches-1:
       - Prompt the user to enter the runs scored in match j+1.
       - Store this value in runs[i][j].

6. **Calculate and Display Results**

* **For** each player i from 0 to n-1:
  1. **Initialize total to 0**.
  2. **Display Runs Scored**:
     + Print "Player i+1 runs: ".
     + **For** each match j from 0 to matches-1:
       - Display runs[i][j].
       - Add runs[i][j] to total.
  3. **Calculate Batting Average**:
     + Calculate the batting average as average = total / number of matches if number of matches > 0, else set average to 0.0.
  4. **Display Batting Average**:
     + Print the calculated batting average for player i+1.

7. **Close Scanner**

* Close the Scanner object s1 to release resources.

8. **End**

**Implementation details:**

import java.util.Scanner;

public class PlayerInfo{

    public static void main(String[] args){

        Scanner s1 = new Scanner(System.in);

        System.out.print("Enter the number of players: ");

        int n = s1.nextInt();

        int[][] runs = new int[n][];

        for (int i = 0; i < n; i++){

            System.out.print("Enter the number of matches played by player "+(i + 1)+": ");

            int matches = s1.nextInt();

            runs[i] = new int[matches];

            System.out.println("Enter the runs scored in each match:");

            for (int j = 0; j < matches; j++) {

                System.out.print("Match "+(j + 1)+": ");

                runs[i][j] = s1.nextInt();

            }

        }

        for (int i = 0; i < n; i++) {

            int total = 0;

            System.out.print("\nPlayer "+(i + 1)+" runs: ");

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

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

                total += runs[i][j];

            }

            double average = (runs[i].length > 0)?(double)total/runs[i].length:0.0;

            System.out.println("\nBatting Average: " + average);

            System.out.println();

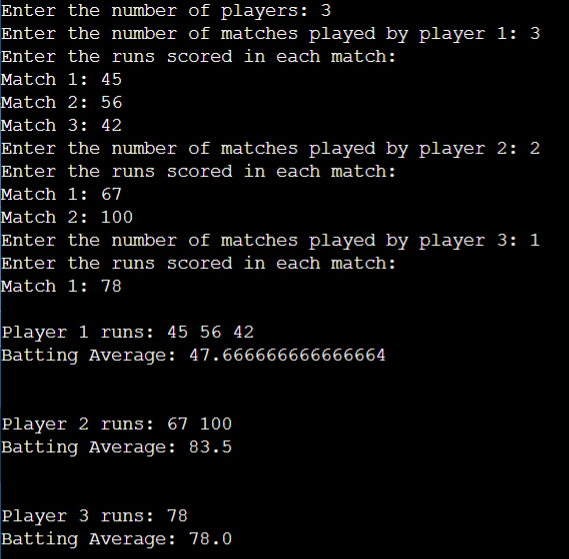
        }

        s1.close();

    }

}

**Output:**



**Conclusion:**

This experiment showed how jagged arrays can handle data with varying lengths in rows. We used them to track player stats, compute totals and averages, and sort the results. We also practiced finding the maximum sum of contiguous subarrays. This helped us see the flexibility and usefulness of jagged arrays in real-world scenarios.

**Date: 29/08/2024 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.*\**;

class Largestsum {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter size of array: ");

        int n = sc.nextInt();

        System.out.println("Please enter the array here: ");

        int arr[] = new int[n];

        for(int i = 0;i<n;i++){

            arr[i] = sc.nextInt();

        }

        int max=0,max\_end=0;

        for (int i=0;i<n;i++){

            max\_end += arr[i];

            if (max\_end > max){

                max = max\_end;

            }

            if (max\_end <0 ){

                max\_end=0;

            }

        }

        System.out.println("The sum of contiguous subarray is " + max);

        sc.close();

    }

}

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**.**

import java.util.*\**;

class q2exp2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter number of 2D array: ");

        int n = sc.nextInt();

        int arr[][][] = new int[n][][];

        for (int i=0;i<n;i++){

        System.out.println("Enter Size of "+(i+1)+" 2D array: ");

        int m = sc.nextInt();

        arr[i] = new int[m][];

        for(int j=0;j<m;j++){

            System.out.println("Enter Size of "+ (j+1) + " inside

            array: ");

            int o = sc.nextInt();

            arr[i][j] = new int[o];

            System.out.println("Enter the elements of array: ");

            for(int k = 0;k<o;k++){

                arr[i][j][k] = sc.nextInt();

            }

        }

        }

    System.out.println("These are your 2D arrays:-");

    for (int i = 0;i<n;i++){

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

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

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

            }

            System.out.println();

        }

        System.out.println();

    }

    sc.close();

    }

}

**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)

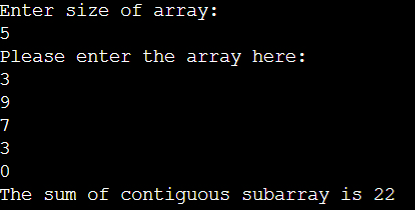
**Ans:**  (iv) (B), (D) & (E)

**Q.4.**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.

**Code:**Same as Q.1

**Output:**

1)



2)

