L18

Tuesday, January 25, 2022 3:11 PM

Fundamentals about Arrays, Multi-dimensional arrays
Saurav Hathi

https://www.youtube.com/channel/UCp6MFWao5vWRnyRCxBsKnfw

Fundamentals of Arrays

Q1. In a programming language the data that has to be processed is loaded in memory and held in variables. When we want to process an integer value, we declare an int variable and assign the value to it.
Suppose we want to write a program that prints the total marks scored by students in a class (say, for a total of 30 students).
One way of doing it is to declare 30 int variables, which is unwieldy when we think of multiple classes or a school.

In such cases, when we want to store multiple values of the same data type, we use an array data structure.
A data structure is a particular way of organizing data in a computer so that data can be accessed and modified efficiently.
An array is a kind of data structure that holds a fixed number of values of a single type, each identified by an array index.

For example, an array of size 10 (int[] marksArr = new int[10];) can be visualized as shown below.

0	1	2	3	4	5	6	7	8	9

Essentially an array can be thought of as a sequence of buckets. The first bucket is identified with number 0, the second bucket with 1 and so on. This number is called the index.

An element is stored in a bucket using the bucket's index. For example, if we want to store a value of 341 at the first index, the code is marksArr[0] = 341;.
Similarly if we want to store a value of 425 in the second bucket, the code will be marksArr[1] = 425; and so on.
If we want to store a value of 747 in the last bucket, the code will be marksArr[9] = 747;.

The length of an array is specified at the time of creating the array.
While writing code if we want to access the size of an array, we use the **length** field. For example, marksArr.length gives the value of 10 in our case.
Similarly in the main method, args.length tells us the size of the String array, which is nothing but the number of arguments passed to the main method.

An element of an array is retrieved/accessed using its index. For example, the value stored in the first bucket can be printed using the below code:
System.out.println(marksArr[0]); // remember that array's index starts from 0 and not 1
In Java, we can create an array in the following two ways:

SYNTAX 1 : dataType[] referenceName = new dataType[size];
int[] marksArr = new int[10]; // this creates an empty array of size 10 and initializes all values to 0
or
SYNTAX 2 : dataType[] referenceName = {value1, value2, value3, ...valuen};
int[] marksArr = { 341, 425, 563, 231, 334, 446, 872, 492, 532, 747};

Understand and retype the below code which prints an array of Ramanujan numbers.

q10935/ArrayDemo.java

```java
package q10935;
public class ArrayDemo {
    public static void main(String[] args) {
        int[] ramanujanNumbers = { 1729, 4104, 13832, 20683, 32832 };
        System.out.println("Printing Ramanujan Numbers: ");
        for (int i = 0; i < ramanujanNumbers.length; i++) {
            System.out.println(ramanujanNumbers[i]);
        }
    }
}
```

Multi-dimentional Arrays

Q1. In Java, a multidimensional array is implemented as an array of arrays.

A multidimensional array is an array whose components are themselves arrays. Imagine a medical cabinet like this:

The cabinet has multiple rows of medicines. Each row in the cabinet is an array and the whole cabinet is a multidimensional array.

In Java, the arrays / rows inside a multidimensional array can be of different lengths, like below -

0	1	2	3	4	5	6	7	8	9	→	Array at index 0
47	49	42	45	60	66	→	Array at index 1				
15	16	17	18	19	20	21	22	23	24	25	26	→	Array at index 2
32	10	22	→	Array at index 3							

Each array inside of a multidimensional array can be accessed with an index number as shown in the above figure. The first array is identified with number 0, the second array with 1 and so on.

In Java, we can create a multidimensional array in the following ways:

SYNTAX 1 : dataType[][] referenceName = new dataType[rowsize][];
int[][] marksArr = new int[3][]; // this creates an empty multidimensional array with 3 arrays.

```
marksArr[0] = new int[10]; //this create an empty array of size 10 and adds to marksArray at index 0
marksArr[1] = new int[2]; //this create an empty array of size 2 and adds to marksArray at index 1
marksArr[2] = new int[5]; //this create an empty array of size 5 and adds to marksArray at index 2
```

or

SYNTAX 2 : dataType[][] referenceName = { {$value_1$, $value_2$, $value_3$, ...$value_n$},
{$value_1$, $value_2$, $value_3$, ...$value_p$},
.....
{$value_1$, $value_2$, $value_3$, ...$value_r$}};

```
int[][] marksArr = {
                        {341, 425, 563, 231, 334},
                        {446, 532, 747},
                        {872, 492}
                    };
```

Note: The data types of all the arrays inside a multidimensional array should be the same.

Select all the options where a valid multidimensional array is created.

- [x] `int[][] intArray = new int[2][5];`
- [x]
```
int[][] intArray = new int[3][];
int[] arr1 = new int[5];
int[] arr2 = new int[6];
int[] arr3 = new int[8];

intArray[0] = arr1;
intArray[1] = arr2;
intArray[2] = arr3;
```
- [x]
```
int[] arr1 = new int[5];
int[] arr2 = new int[6];
int[] arr3 = new int[8];

int[][] intArray = {arr1, arr2, arr3};
```
- [x]
```
int[] arr1 = new int[5];

int[][] intArray = {arr1, {10,11,12}, {100,150,200}};
```
- [x] `int[][] intArray = {{10,11,12}};`
- [ ] `int[][] intArray = {{10.0,11,12}};`

Q2. The class MultiDimArrayPrinter with a main method. The program prints a multidimensional array of integers.

The code in the main method uses a for-each loop to iterate over the multidimensional array intArr and prints the values.

Understand and retype the code below.

q10946/MultiDimArrayPri

```java
package q10946;
public class MultiDimArrayPrinter {
    public static void main(String[] args) {
        int[][] int2DArr = {
                            {1},
                            {2, 3},
                            {4, 5, 6},
                            {7, 8, 9, 10}
                        };
        for (int[] arr : int2DArr) {
            for (int value : arr) {
                System.out.print(value + " ");
            }
            System.out.println();
        }
    }
}
```

Q3. The class MultiDimArrayPrinter prints a multidimensional array of integers.

The code in the main method uses for loops with loop counters to iterate over the multidimensional array int2DArr and print the values.

Understand and retype the code below:

q10947/MultiDimArrayPri

```java
package q10947;
public class MultiDimArrayPrinter {
    public static void main(String[] args) {
        int[][] int2DArr = {
                            {1},
                            {2, 3},
                            {4, 5, 6},
                            {7, 8, 9, 10}
                           };
        for (int i = 0; i < int2DArr.length; i++) {
            for (int j = 0; j < int2DArr[i].length; j++) {
                System.out.print(int2DArr[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Program to find Addition of Two matrices

Q4. Write a class AdditionOfMatrix with a **public** method add which returns the sum of its arguments. If the two arguments are not matrices of the same size, then the method should return null.

Consider the following examples for your understanding:

Matrix 1:
Enter number of rows: 2
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 3 4
Matrix 2:
Enter number of rows: 2
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 4 5
Enter row 2: 6 7
Sum of the two given matrices is:
5 7
9 11
Matrix 1:
Enter number of rows: 2
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 3 4
Matrix 2:
Enter number of rows: 2
Enter number of columns: 3
Enter 3 numbers separated by space
Enter row 1: 1 2 3
Enter row 2: 4 5 6
Addition of different sized matrices is not possible

q11053/AdditionOfMatrix

```java
package q11053;
public class AdditionOfMatrix {

    /**
     * Computes the addition of the two given matrices if the sizes of the matrices are the same.
     * Otherwise, returns null.
     *
     * @return the resultant matrix, null if addition is not possible
     */
    public int[][] add(int[][] matrix1, int[][] matrix2) {
        // Write the code

        int row=matrix1.length,col=matrix1[0].length;

        int row1=matrix2.length, col1=matrix1[0].length;


        int sum[][] = new int[row][col];

            for(int i = 0; i<row; i++){
                for(int j = 0; j<col; j++){

                    if(matrix1[i][j] != matrix2[i][j]){

                    sum[i][j] = matrix1[i][j]+matrix2[i][j];

                    } else{
                        return null;
                    }
                }
        }

        return sum;

    }
}
```

Write a Java program to display Transpose of the given Matrix

Q5. Write a class TransposeMatrix with a **public** method transposeMatrix that takes one parameter matrix1 of type int[][] which returns the transpose of the given matrix.

Consider the following example for your undertanding:
Matrix:
Enter number of rows: 3
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 3 4
Enter row 3: 5 6
Transpose of given matrices is:

1 3 5
2 4 6

q11105/TransposeMatrix.j q11105/TransposeMain.ja

```java
package q11105;
public class TransposeMatrix {
/**
     * Computes the transpose of the given matrix.
     *
     * @return the resultant matrix
     */

public int[][] transposeMatrix(int[][] matrix1) {

        // Write the code
        int row = matrix1.length;

        int col = matrix1[0].length;

        int tranMatrix[][] = new int[col][row];

        for(int i = 0; i<row; i++){

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

                tranMatrix[j][i] = matrix1[i][j];
            }
        }
        return tranMatrix;

    }
}

```

Program to find Multiplication of Two matrices

Q6. Write a class MultiplicationOfMatrix with a **public** method multiplication which returns the multiplication result of its arguments. if the first argument column size is not equal to the row size of the second argument, then the method should return null.

Consider the following example for your understanding
Matrix 1:
Enter number of rows: 3
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 4 5
Enter row 3: 7 8
Matrix 2:
Enter number of rows: 2
Enter number of columns: 3
Enter 3 numbers separated by space
Enter row 1: 1 2 3
Enter row 2: 4 5 6
Multiplication of the two given matrices is:
9 12 15
24 33 42
39 54 69
Matrix 1:
Enter number of rows: 2
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 3 4
Matrix 2:
Enter number of rows: 3
Enter number of columns: 2
Enter 2 numbers separated by space
Enter row 1: 1 2
Enter row 2: 4 5
Enter row 3: 2 3
Multiplication of matrices is not possible

q11106/MultiplicationOfM q11106/MultiplicationOfM

```java
package q11106;
public class MultiplicationOfMatrix{
    public int[][] multiplication(int[][] matrix1, int[][] matrix2) {
        /*Return the result if the matrix1 coloumn size is equal to matrix2 row size and print the result.
        * @Return null.
        */
        // Write your logic here for matrix multiplication
        int row=matrix1.length, col=matrix1[0].length;

        int row1=matrix2.length, col1=matrix2[0].length;

        int multi[][] = new int[row][col1];

        for(int i = 0; i<row; i++){

            for(int j = 0; j<col1; j++){

                for(int k = 0; k<row1; k++){

                if(row1 == col){

                    multi[i][j] += matrix1[i][k] * matrix2[k][j];

                } else {

                   return null;
                  }
                }
            }
        }
        return multi;
    }
}
```