Java Arrays

Project

Table of Contents

Arrays Assignment

Objectives

The objective of the assignment is to:

- Provide practice with the basics of Java, including arrays, classes, methods and fields.
- Provide practice using the Java command line tools.

Specifications

You are going to create a class called `ArrayData` with the following attributes.

***** DO NOT CHANGE THE FIELD OR METHOD DEFINITIONS BELOW. *****
***** YOU ARE REQUIRED TO IMPLEMENT THEM AS GIVEN. *****
YOU CAN ADD ADDITIONAL FIELDS AND METHODS.

Field	Description
`int rows`	Contains the total number of rows. The initial value is 10.
`int columns`	Contains the total number of columns. The initial value is 10.
`int values[][]`	This array contains integers that are used by the class. The size of the array is contained in the fields, `rows` and `columns`.
`int rowData[]`	This array contains the sum (or other operations) of the integers in each row of the `values` array. For example, position 0 in the `rowData` array will contain the sum of the integer values in row 0 of the `values` array. The size of this array is contained in the field, `rows`. NOTE: The results of other operations other than summation maybe performed and placed in this array.
`int colData[]`	This contains the sum (or other operations) of the integers in each column of the `values` array. For example, position 0 in the `colData` array will contain the sum of the integer values in column 0 of the `values` array. The size of this array is contained in the field, `columns`. NOTE: The results of other operations other than summation maybe performed and placed in this array.

Method	Description
`ArrayData()`	Initialize the `rows` and `columns` fields to their default values of 10 each. Initialize the `values`, `rowData` and `colData` arrays to default values of 0.

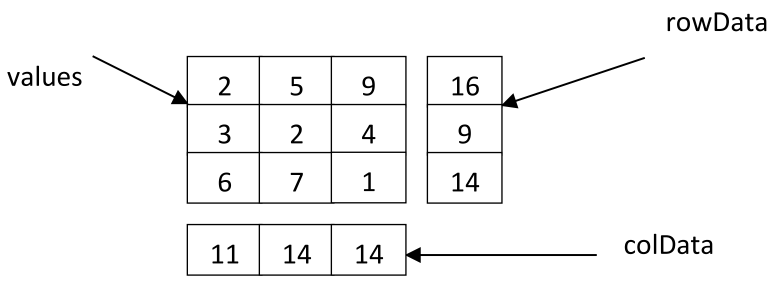
`ArrayData( int nrows, int ncolumns )`	Initialize the rows and columns fields to their new values of `nrows` and `ncolumns` respectively. Initialize the `values`, `rowData` and `colData` arrays to default values of 0.
`ArrayData( int nrows, int ncolumns, int startingValue )`	Initialize the rows and columns fields to their new values of `nrows` and `ncolumns` respectively. Initialize the `values` array to the given starting value called `startingValue`. Initialize the `rowData` and `colData` arrays to 0.
`void generate( int newValue, int total, int minRow, int, maxRow, int minCol, int maxCol )`	This method will randomly choose total positions in the `values` array and change the value to `newValue`. For example, the call `generate(5,10,2,4,2,5)`, will randomly choose 10 positions in the `values` array and set their value to 5. The integers `minRow` and `maxRow` are the minimum and maximum rows that can be used. For example, if `minRow` is 2 and `maxRow` is 5 then you can choose positions in rows 2 to row 5 (rows 2 and 5 are included in the calculations). The same applies to `minCol` and `maxCol`.
`void flip( int num, int val )`	Change the value of `num` positions, in the `values` array, to the new value `val`. The num positions are chosen randomly. It is possible for the same position in the `values` array to be chosen more than once. This is allowed in this assignment.
`void sum()`	This method calculates the sum of each row and places it in the `rowData` array and the sum of each column and places it in the `colData` array. For example, position 0 in the `rowData` array will contain the sum of the integer values in row 0 and so on.
`void occurrence( int num )`	This method will calculate the number of times the integer `num` or a multiple of `num` DOES NOT appear in each row and places it in the `rowData` array. Perform the same action for each column and place the results in the `colData` array. However, for the columns count the number of times integer `num` or a multiple of `num` DOES occur in the columns. For example, if the row contains 2, 4, 7, 8 and the number passed is 4, the answer will be 2 since 2 and 7 are not divisible by 4.
`void standardDeviation()`	This method will calculate the standard deviation of the integers in each row and place it in the `rowData` array. Perform the same action for each column and place the results in the `colData` array. The standard deviation for each row is calculated using the following formula. $\sigma = \sqrt{} (\sum (X - \mu)^2 / N)$ Where σ is the standard deviation μ is the average/mean of the row $\sum$ is the symbol for the “the sum of” X represents the integer value in each row position

	N represents the total number of integers in the row i.e. the length of the row.
`double checkeredOdd()`	This method will sum the values in the odd positions within the `values` array. This is followed by summing the values I the even positions. The method will return the result of dividing the odd sum by the even sum. Always start with the grid position 0,0.
`void product( int min, int max )`	This method will calculate the product of each row and column. The method starts with the rows then the columns i.e. multiply all the numbers in the row or column. If the product (multiplication) in the row is between `min` and `max` (including `min` and `max`) then this method will randomly decrement a member of the row, by 1, until the product is less than `min`. A similar procedure is to be performed for the columns. The results are placed in the `rowData` and `colData` arrays.
`void print()`	Prints the contents of the 3 arrays using the following format. 2 | 5 | 9 || 16 3 | 2 | 4 || 9 6 | 7 | 1 || 14 11 | 14 | 14 Where the bottom row is the values in the `colData` array and the column on the right is the values of the `rowData` array. The columns do not need to be aligned when displayed on the screen.
`void print( int rows, int columns )`	This method is the same as the `print` method but the number of rows and columns printed is set by the parameter values. For example, a call of `print( 2, 1 )` will give the output. 2 || 16 3 || 9 11 Note that the rowData and colData values remain the same as if the entire array had been printed. The columns do not need to be aligned when displayed on the screen.

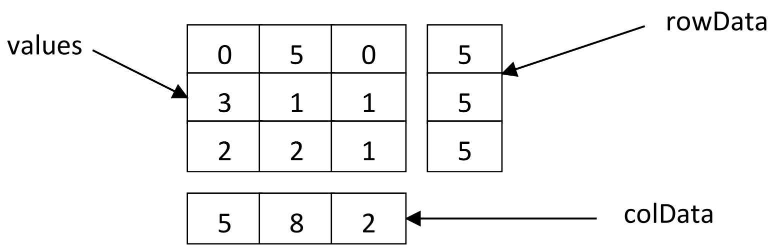
You must implement each of the methods described in the previous table. You must also use the fields described. You can create other methods and fields to complete the program. An example of how to use the class is given below.

```
public class TestArrayData
{
      public static void main( String args[] )
      {
            ArrayData s = new ArrayData();

            s.generate( 5, 10, 2, 4, 2, 5 );
            s.sum();
            s.print();

            s.product( 2, 4 );
            s.sum();
            s.print();

            s.occurrence( 2 );
            s.print();

            s.print( 4, 2 );
      } // main
} // TestArrayData
```

To give you an idea of what the arrays would look like, assume that the total number of rows is 3 and the columns is 3. The arrays will have the following format.



If the rows sums must be equal to 5, the grid may look like the following.



The row sums are now equal to 5 and the column sums adjusted to their new values.

To generate a simple random value, use the following code snippet.

```
import java.util.Random;

public class ArrayData
{
      private Random rand = new Random();

      public int getRandNumber( int maxNum )
```

```
        {
                // Get a number between 0 and maxNum (excluding maxNum).
                return( rand.nextInt( maxNum ) );
        }
} // ArrayData
```

The `nextInt` method returns an integer value that lies between 0 and `maxNum` (including 0 and excluding `maxNum`).

Grading Components

This section describes a few key areas that are being considered during the grading process. The grading rubric contains all the areas that are being graded.

In-Code Documentation

In-code documentation refers to the commenting of code to ensure that the functionality and purpose of the code is understood. The minimum comments that should be included are:

- At the top of each .java file, there should be a description of what the class does as well as the name of programmer(s).
- At the top of each function/method in the .java file there should be a description of the purpose of the function, the values passed to the function and what is returned by the function.
- Next to each field there should be a description of its purpose.
- Utilise sensible names for classes, fields and functions/methods that reflect their purpose.

Grading Rubric

This project is worth 8% of the total course mark. The grading scheme is given below.

Area	Excellent	Good	Average	Unsatisfactory
Data structures and algorithms chosen (30)	Excellent use of the appropriate data structures and algorithms. Excellent coding practices have been used with no redundant or	Good use of the appropriate data structures and algorithms. Good coding practices have been used with little redundant	Average use of the appropriate data structures and algorithms. Considerable redundant or unnecessary code is present.	Poor use of the appropriate data structures and algorithms. Redundant and unnecessary code is largely present.

	unnecessary code. Algorithms are efficient. No unnecessary data is used. **(24-30 marks)**	or unnecessary code. Algorithms are mostly efficient. Small amounts of unnecessary data is used. **(16-23 marks)**	Algorithms are generally not efficient. Significant amounts of unnecessary data is used. **(8-15 marks)**	Algorithms are not efficient. Most of the data that is used is unnecessary. **(0-7 marks)**
Implementation and Execution (30)	Over 80% of the required functionality has been implemented and executes correctly. **(24-30 marks)**	Between 50% - 80% of the required functionality has been implemented and executes correctly. **(16-23 marks)**	Between 25%-49% of the required functionality has been implemented and executes correctly. **(8-15 marks)**	Less than 25% of the required functionality has been implemented and executes correctly. **(0-7 marks)**
Compiling (10)	The software compiles with no errors and less than 3 warnings. **(10 marks)**	The software compiles with 1-6 errors and 4-6 warnings. **(7-9 marks)**	The software compiles with 7-12 errors and 7-10 warnings. **(4-6 marks)**	The software compiles with greater than 12 errors and greater than 10 warnings. **(0-3 marks)**
Naming Conventions (10)	Over 80% of the code uses the correct naming conventions. **(10 marks)**	Between 51%-80% of the code uses the correct naming conventions. **(7-9 marks)**	Between 26%-50% of the code uses the correct naming conventions. **(4-6 marks)**	Between 0%-25% of the code uses the correct naming conventions. **(0-3 marks)**
In-Code Documentation (10)	Over 80% of the code has been fully documented with comments laid out in a readable and understandable form. **(10 marks)**	Between 50%-80% of the code has been fully documented with comments mostly laid out in a readable and understandable form. **(7-9 marks)**	Between 25%-49% of the code has been fully documented with some of the comments laid out in a readable and understandable form. **(4-6 marks)**	Less than 25% of the code has been fully documented with very few of the comments laid out in a readable and understandable form. **(0-3 marks)**
Total (100)				

Deliverables

The final deadline for this assignment is 7th October 2022 at midnight. It should be submitted on eLearninig.

The deliverables are:

- The Java code in the .java files. Do not include the .class files.
- The completed design document which can be found on eLearning, should be completed and included with the submission.