CSE 215: Programming Language II Lab
Daily Lab Task 2

Note: Think of this as a practice for the lab midterm.

Consider the following UML diagram and implement it:

Matrix
- numberOfRows: int - numberOfColumns: int - matrix: double[][]
+ Matrix(constant: int, rows: int, columns: int) + Matrix(matrix: double[][]) + getMatrix(): double[][] + getNumberOfRows(): int + getNumberOfColumns(): int + setMatrix(newMatrix: double[][]): void + generateRandomMatrix(rows: int, columns: int): Matrix + generateRandomMatrix(rows: int, columns: int, lowerBound: double, upperBound: double): Matrix + multiplyMatrixByConstant(constant: int): Matrix + toString(): String

a. The `Matrix` class has three private attributes: two integers and a two dimensional array of `double` data type.
b. It has two constructors.
   i. `Matrix(constant, rows, columns)` takes the number of rows and columns and a constant as input parameter and fills the array with that constant. For example, consider the following:

3	3	3
3	3	3
3	3	3

new Matrix(3, 3, 3) should create the above matrix

1	1	1	1
1	1	1	1
1	1	1	1

new Matrix(1, 3, 4) should create the above matrix

   ii. `Matrix(matrix)` takes a two dimensional array of double data type as input parameter. **This constructor must also update the values of `numberOfRows` and `numberOfColumns` data fields from the dimensions of the passed array.**

c. There are 3 getters and 1 setter methods.
d. `generateRandomMatrix()` has two versions:

i. `generateRandomMatrix(rows, columns)` takes the number of rows and columns as parameters and returns a Matrix object randomly filled between 7.5 (inclusive) and 42.0 (exclusive).

ii. `generateRandomMatrix(rows, columns, lowerBound, upperBound)` takes two additional parameters, `lowerBound` and `upperBound`, and returns a `Matrix` object filled with numbers between `lowerBound` (inclusive) and `upperBound` (exclusive).

e. `multiplyMatrixByConstant(constant)` multiplies each of the elements of the array stored in the `matrix` data field of a `Matrix` object by the provided constant parameter. Consider the example below:

3.5	2.5
1.75	7.0

Initial Matrix

7.0	5.0
3.5	14.0

Matrix object returned after calling `multiplyMatrixByConstant(2)`

1.0	8.0	9.0
2.0	7.0	8.0
3.0	6.0	7.0
4.0	5.0	6.0

Initial Matrix

5.0	40.0	45.0
10.0	35.0	40.0
15.0	30.0	35.0
20.0	25.0	30.0

Matrix object returned after calling `multiplyMatrixByConstant(5)`

f. `toString()` returns the String representation of the array stored in the matrix data field. The format should resemble a regular matrix such as this (this is a sample for a 3x3 matrix):

```
| 1.00000 | 2.00000 | 3.00000 |
| 5.00000 | 8.00000 | 4.00000 |
| 5.66766 | 2.47874 | 5.20001 |
```

Perform the following operations in your main method:

a. Call the version of `generateRandomMatrix()` that only takes rows and columns as input parameters with values of your choice and store the returned `Matrix` object in a variable called `matrix1`.

b. Call the other version of `generateRandomMatrix()` and generate a 3x7 matrix filled with random values between 6.0 and 79.5. Store the returned `Matrix` object in a variable called `matrix2`.

c. Use `toString()` method to print `matrix1` and `matrix2`.

d. Call `multiplyMatrixByConstant()` on `matrix1` and `matrix2` with values of your choice. Store the returned matrices in variables called `matrix3` and `matrix4`.

e. Use `toString()` method to print `matrix3` and `matrix4`.

Hints: Assume that you have to print out upto 5 digits after the decimal point wherever applicable, even if the examples above used a different number of digits after the decimal point.