C212/A592 Spring 17 Lab 4

Intro to Software Systems

Instructions:

- Review the requirements given below and Complete your work. Please submit all files through Canvas.
- The grading scheme is provided on Canvas

Matrix Library

We will be creating a class which will be given a 2-D array when instantiated, and then implement various methods to perform operations on the 2-D array.

Create the class with the following API

```
public class Matrix {

        private double[][] matrix;
        private final int NUMROW;
        private final int NUMCOL;

        public Matrix(double[][] m) { }
        public String toString() { }
        public Matrix getRowMatrix(int m) { }
        public Matrix getColMatrix(int m) { }
        public Matrix transpose() { }
        public Matrix upperDiagonal() { }
        public Matrix lowerDiagonal() { }
        public Matrix diagonalMatrix() { }
        public void main(String[] args) { }

}
```

1. *Matrix(double[][] m)*
    a. Initialize the NUMROW and NUMCOL variables. These are the lengths of the rows and columns of *m.*
    b. The constructor should initialize the *double[][] matrix* field
    c. Do not assume the input to the constructor is a NxN matrix
    e.g. the constructor and other methods need to work on a 4x4 matrix or a 4x2 matrix

2. *toString()*

a. returns a string value of the matrix. To receive full points, string must be in this form: Each value should have a comma and a space except the last one. There should be no space before the closing bracket.

4x4 matrix
[0.0, 1.0, 2.0, 3.0]
[4.0, 5.0, 6.0, 7.0]
[8.0, 9.0, 10.0, 11.0]
[12.0, 13.0, 14.0, 15.0]

4x2 matrix
[0.0, 1.0]
[2.0, 3.0]
[4.0, 5.0]
[6.0, 7.0]

3. *Matrix getRowMatrix(int m) and Matrix get*Col*Matrix(int m)*
   a. returns a new matrix which is the m^{th} row or column. These methods can be used with ToString() method to print the row and column of interest. If the m^{th} row or column doesn't exist, methods should warn the user and prompt to enter the correct row or column number (by providing the size of the matrix). Example: For the given matrix

      4x4 matrix
      [0.0, 1.0, 2.0, 3.0]
      [4.0, 5.0, 6.0, 7.0]
      [8.0, 9.0, 10.0, 11.0]
      [12.0, 13.0, 14.0, 15.0]

      getRowMatrix(0) should return [0.0, 1.0, 2.0, 3.0], and
      getColMatrix(0) should return [0.0, 4.0, 8.0, 12.0].

4. *Matrix transpose()*

   a. Transpose should work for square and non-square matrices
   b. Returns the transpose of the instance field *matrix* as a new Matrix object
   c. Include these two matrices below in your test cases for this method.

   The **transpose** of a matrix is a new matrix whose rows are the columns of the original. (This makes the columns of the new matrix the rows of the original). Here is a matrix and its transpose:

$$\begin{pmatrix} 5 & 4 & 3 \\ 4 & 0 & 4 \\ 7 & 10 & 3 \end{pmatrix}^{T} = \begin{pmatrix} 5 & 4 & 7 \\ 4 & 0 & 10 \\ 3 & 4 & 3 \end{pmatrix}$$

Another way to look at the transpose is that the element at row r column c in the original is placed at row c column r of the transpose. The element $\mathbf{a_{rc}}$ of the original matrix becomes element $\mathbf{a_{cr}}$ in the transposed matrix.

$$\begin{pmatrix} 5 & 4 \\ 4 & 0 \\ 7 & 10 \\ -1 & 8 \end{pmatrix}^{T}_{4x2} = \begin{pmatrix} 5 & 4 & 7 & -1 \\ 4 & 0 & 10 & 8 \end{pmatrix}_{2x4}$$

5. *Matrix lowerDiagonal()* and *Matrix upperDiagonal()*
    a. The *lowerDiagonal*() should return a new matrix where all values below the main diagonal are 0. For example, given the following 4x4 matrix:

        4x4 matrix
        [3.0, 1.0, 2.0, 3.0]
        [4.0, 5.0, 6.0, 7.0]
        [8.0, 9.0, 1.0, 1.0]
        [1.0, 1.0, 1.0, 1.0]

        It should return the following as output:

        4x4 matrix
        [3.0, 1.0, 2.0, 3.0]
        [0.0, 5.0, 6.0, 7.0]
        [0.0, 0.0, 1.0, 1.0]
        [0.0, 0.0, 0.0, 1.0]

    b. Likewise, for *upperDiagonal()* all values above the main diagonal are 0. For example, given the following 4x4 matrix:

        4x4 matrix
        [3.0, 1.0, 2.0, 3.0]
        [4.0, 5.0, 6.0, 7.0]
        [8.0, 9.0, 1.0, 1.0]
        [1.0, 1.0, 1.0, 1.0]

        It should return the following as output:

        4x4 matrix
        [3.0, 0.0, 0.0, 0.0]
        [4.0, 5.0, 0.0, 0.0]

[8.0, 9.0, 1.0, 0.0]
[1.0, 1.0, 1.0, 1.0]

c. The main diagonal of a matrix (square or non-square) are when the indices in the matrix *i* and *j* are such that *i* = *j.*

6. *Matrix diagonalMatrix()*
    a. A diagonal matrix is a matrix were the all values are 0 except the main diagonal
    b. Using only *lowerDiagonal()* and *upperDiagnoal(),* return a diagonal matrix of the instance field *matrix*

7. As always, use the *main* as a test client to have test cases for grading.