Assignment 1: Matrix Manipulations using 2DBlockMatrix

Objective:

The objective of this problem is to be able to parse input from a text file, develop a data-structure for representing 2-dimensional block matrices and perform basic matrix operations. These have to be implemented against 2DBlockMatrix class with specified public methods. Details are given below.

Description:

Input / Output format:

Matrices are provided in a text file whose full-path is given as an argument to one of the static methods - see below.
The format of the file is as follows:

1. The first line of the file contains the location of the first block, i.e., the (row, col) of the block, specified as two integer values separated by a space. Row and column indexes start at 1.
2. Following lines contain space-separated matrix elements in row-wise order that are all floats. The line ends at the first occurrence of a semicolon.
3. The end of each block is marked by a new line containing only "#".
4. The line immediately after "#" contains the location of the next block.
5. End of file is the end of the matrix.
6. You need to parse the file to determine the dimensions of the matrix.

You may assume that the matrices specified are always 2-dimensional.
Example:
Following is a valid 2-dimensional block matrix specified according to the above-specified format:

```
1 1
1 2;
2 4;
#
3 3
1;
#
```

After parsing it you should have the matrix $\begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
The same format has to be followed for generating a string representation of any matrix (see below).

Note: Floating-point numbers in the ***output*** must be rounded off to 2 decimal places. If the output is an integer, you **should not** add decimal point and zeros. (Remember that 0.075 is rounded off to 0.08, while 0.085 is also rounded off to 0.08). However, the ***input matrix in the files*** can be of arbitrary precision.

Implementation contract:

You must implement class `edu.iitd.2020col106.2DBlockMatrix` with the following public methods:

1. `static 2DBlockMatrix build2DBlockMatrix (java.io.InputStream in):` This is a static method on the `2DBlockMatrix` class which takes a reference to a stream from which the matrix --formatted as above-- is read and a reference to the newly created object is returned.
2. `2DBlockMatrix(float[][] array):` Constructor for `2DBlockMatrix` class from a 2-dimensional array of floats.
3. `String toString():` Returns the String representation of a `2DBlockMatrix` object according to the format described above, which when printed (or written to a file) should match the desired result.
4. `2DBlockMatrix transpose():` If the calling object contains a 2-dimensional matrix of size **m x n**. The method should return the **new** transposed 2-dimensional matrix of size **n x m**. The original matrix must remain unchanged.
5. `2DBlockMatrix multiply(2DBlockMatrix other) throws edu.iitd.2020col106.IncompatibleDimensionException:` This method takes another matrix object of type `2DBlockMatrix`. The task is to perform matrix multiplication `this x other`. The method should return the product matrix. If the two matrices are not compatible, you must throw the specified exception.
6. `2DBlockMatrix getSubBlock (int row_start, int col_start, int row_end, int col_end) throws edu.iitd.2020col106.SubBlockNotFoundException`: This method returns

the sub-block of the `2DBlockMatrix` object that belongs to the slice with rows between {row_start, row_end} and columns between {col_start, col_end}. Note that the exception has to be thrown when there is no underlying sub-block in the specified range.

7. `2DBlockMatrix inverse() throws edu.iitd.2020col106.InverseDoesNotExistException`**:** The method should return a new 2DBlockMatrix object containing the inverse of the given matrix. In case the inverse matrix does not exist, you must raise an exception rather than returning anything.

For Bonus Points (for a maximum of additional 10% marks)

You must implement class **`edu.iitd.2020col106.Sparse2DBlockMatrix extends edu.iitd.2020col106.2DBlockMatrix`** that can handle ***sparse block matrices*** of large sizes -- i.e., in the order of **1000 x 1000** (at least) with the same public methods as the superclass. No additional methods are required.
Note: Since Sparse2DBlockMatrix extends the 2DBlockMatrix, all methods in Sparse2DBlockMatrix which take a reference to 2DBlockMatrix object should seamlessly work when replaced with a reference to Sparse2DBlockMatrix.

What is Allowed / Not Allowed?

1. You must work on this assignment **individually**.
2. The programming language to be used is Java.
3. You are not allowed to use any other external libraries for this assignment.
4. Your code must be your own. You are not to take guidance from any general-purpose or problem-specific code meant to solve this or related problem.
5. We will run a plagiarism detection check. Any person found guilty will be awarded a suitable penalty as announced in the class/course website. Please read academic integrity guidelines on the course home page and follow them carefully.

What to Submit?

1. The code should be contained in a zip file. The exact directory structure will be detailed on the moodle submission link. Your submission will be auto-graded.

This means that it is essential to make sure that your code follows the specifications of the assignment **precisely.**

2. Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 16.04 running Java JDK 11.0.5 (https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html).
3. You will be able to check if your submission complies with the proper format and runs as expected in the test environment by submitting to Moodle. The auto-grading in Moodle will run your code and verify if it passes the preliminary test cases.
4. Your submission will only be accepted once it passes the initial test cases. For this, you must at least complete the Minimal submission tests of the assignment (will be announced soon). Further test cases will be used for Evaluation**.**