Assignment 2: Matrix

Logistics

- The assignment is meant to be done individually.
- The **deadline** for this assignment is **11:59 PM on Oct 27, 2023, PDT**.
- Academic dishonesty is unacceptable and will not be tolerated in this course.
- **Last Modified**: Oct 13, 2022

Get Started

To get started, please download the zip file from Canvas and follow the structure and submission guidelines below.

Repository Structure

After you unzip the zip file downloaded above, you should get the following files and folders:

README.pdf

Problem description and requirements

Assignment_Project

The assignment project folder. It contains `out` and `src` folders.

`out/artifacts/Assignment_Project_jar` is where you should put your JAR file.

In `src/Matrix/Matrix.java`, the template for this assignment is provided. You should implement each method as requested in this description to complete each task.

We also provide a sample test script in `src/SampleTest.java` to let you test your implementation conveniently.

Submission Instructions

- During implementing this assignment:
  - **Please do not modify the repository structure**
  - **Please do not change the template we provide to you**

  A simple way to check two requirements above is to run the sample tests we provided to you.

  Failing to obey this rule may lead to penalty on your final score!
- In this project, you don't have to create a JAR file in your submission. We will use our own script (similar to `src/SampleTest.java`) to test your class `Matrix` assignment.

- In order to run the sample test script in `src/SampleTest.java`, you will need to:
  - Create an artifact configuration for the JAR
    - From the main menu, select `File | Project Structure` and click `Artifacts`.
    - Click `+`, point to `JAR` and select `From modules with dependencies`.
    - To the right of the `Main Class` field, click and select `SampleTest`
    - Apply the changes and close the dialog.
  - Build the JAR artifact
    - From the main menu, select `Build | Build Artifacts`.
    - Point to `SampleTest:jar` and select `Build`.
    - If you now look at the `out/artifacts` folder, you'll find your JAR there.
- After you complete this assignment, please zip up this project folder into `Assignment_Project.zip`.
- Finally, upload it to Canvas under Assignment 2 submission link.
- Sample tests are only for clearer explanation and simple testing during your implementation. **Passing all the sample tests does not mean you will get a full score.** You need to read the description carefully, and think it comprehensively when implementing this assignment.
- Another set of test cases will be used to evaluate your assignment after due. But don't worry. All the test cases will strictly follow this description.

General Description

Matrix is a very important form to represent numerical data. In this assignment, you are required to implement a class called `Matrix` with basic functionalities. In detail, you need implement:

- input/output
- Basic arithmetic operations, such as
  - Addition
  - Subtraction
  - Matrix/Scalar Multiplication
  - Scalar Division
  - Equalization
  - Transposition
- Matrix indexing
- Matrix determinant

Notice:

- **Complete the tasks in sequence**. Task 2, 3 and 4 depend on Task 1 as input and output. Therefore, please complete the tasks of Assignment 2 in sequence.

- Through the whole assignment 2, elements in matrices must be presented and stored in `double` data type.
- Please note that **all the arguments are valid**. In the other word, it is not necessary for you to pre-check the values and raise exceptions.
- Third-party libraries and/or JDKs with built-in matrix data types operations are **forbidden**.
- In order to avoid possible abuse of third-party libraries and/or JDKs with built-in matrix data types operations, we will use OpenJDK-21 for evaluation. It's recommended that you use the same JDK in your implementation process.

Task 1: Matrix Input and Output

In this task, you need to implement basic inputs and outputs for the self-defined class `Matrix`.

Task 1.1: Load the Matrix in

In order to do matrix operation, you first need to load it in. Therefore, in this task, you need to implement the Constructor of class `Matrix` in order to load the matrix in, and store it inside the class.

The printing format follows the format defined below using EBNF form.

```
Matrix = {blank} "[" {blank} Rows {blank} "]" {blank} | {blank} "[" {blank} "]" {blank} ;
Rows = {blank} Row {blank} | {blank} Row {blank} ";" {blank} Rows {blank} ;
Row = {blank} element {blank} | {blank} element {blank} "," {blank} Row {blank} ;
element = double;
blank = " " | "\n" | "\r" | "\t";
```

To make it simple, you should output the matrix in the format of:

```
[Matrix[0][0],Matrix[0][1],...,Matrix[0][n-1];Matrix[1][0],...,Matrix[1][n-1];...;Matrix[m01][0],...,Matrix[n-1][n-1]]
```

Notice that redundant blanks may appear anywhere inside an expression.

If you are still confused about definition of sparse representation and its expression, please read the appendix, take a view of given test cases and take discussion and lab sessions of Week 3 for more details.

Input and Output

Input

- `m`: row number of the matrix. It should be a positive integer.
- `n`: column number of the matrix. It should be a positive integer.
- `expr`: expression of the matrix. It should be `String`.

Output

There will be no output for this task.

Task 1.2: Print the Matrix out

In this task, you should implement the method `print()` in order to make it possible to print the matrix out.

The printing format follows the similar format with the input format of Task 1.1 defined below using EBNF form.

```
Matrix = "[" Rows "]" | "[" "]" ;
Rows = Row | Row ";" Rows ;
Row = element | element "," Row ;
element = double;
```

To make it simple, you should output the matrix in the format of:

```
[Matrix[0][0],Matrix[0][1],...,Matrix[0][n-1];Matrix[1][0],...,Matrix[1][n-
1];...;Matrix[m01][0],...,Matrix[n-1][n-1]]
```

Where `Matrix[i][j]` means the value of the element on *i*-th row, *j*-th column.

Notice that redundant blanks **may not** appear anywhere inside a valid output expression.

Input and Output

Input

There will be no input

Output

You should output a string following the format defined above.

- Notice that **all the numbers must keep 5 digits after decimal**
  - hint: use `String.format`.

Task 2: Matrix Arithmetic Operations

In this task, you need to implement basic operations for the self-defined class `Matrix`.

Task 2.1: Matrix Addition and Subtraction

Implement methods `add()` and `sub()` for class `Matrix`.

Input and Output

Input

`other`: class `Matrix`

Output

`res`: class `Matrix`

Task 2.2: Matrix Multiplication

Implement method `mul()` for class `Matrix`.

There are 2 circumstances for matrix multiplication:

- Multiplication between 2 matrices
- Multiplication between one matrix and a scalar (which is `double`)

Input and Output

Input

`other`: class `Matrix` or scalar (`double`)

Output

`res`: class `Matrix`

Task 2.3: Matrix Scalar Division

Implement method `div()` for class `Matrix`.

Input and Output

Input

`other`: scalar (`double`)

Output

`res`: class `Matrix`

Task 2.4: Matrix Equalization

Implement method `isEq()` for class `Matrix` to judge whether the given matrix (`other`) is identical with each other.

Input and Output

Input

`other`: class `Matrix`

Output

`isEqual`: `boolean`

Task 2.5: Matrix Transpose

Implement method `transpose` for class `Matrix` to calculate the transposed matrix.

Input and Output

Input

No input for this task

Output

`res`: class `Matrix`

Task 3: Matrix Indexing

In this task, you need to implement method `getElements()` and `setElements()` so that part of the class `Matrix` could get accessed or modified.

The following are tasks you need to do (`x` is an instance of `Matrix` class):

- `x.getElements(row_num)` returns the `row_num`-th row (starting at `0`) which is also a matrix
  - **Input**:
    - `row_num`: integer
  - **Output**:
    - `res`: class `Matrix`
- `x.getElements(row_num, col_num)` returns the element at the `row_num`-th row and `col_num`-th column
  - Please return the result in `String` and **keep it 5 digits after decimal**.
  - **Input**:
    - `row_num`: integer
    - `col_num`: integer
  - **Output**:
    - `res`: String
- `x.setElements(row_num, other)` replaces the `row_num`-th row by the Matrix `other`

    - `other` only has one row
    - **Input**:
        - `row_num`: integer
        - `other`: class `Matrix`
    - **Output**:
        - There will be no output for this task.
- `x.setElements(row_num, col_num, other)` replaces the element at the `row_num`-th row and `col_num`-th column by the `double` float `other`
    - **Input**:
        - `row_num`: integer
        - `col_num`: integer
        - `other`: `double`
    - **Output**:
        - There will be no output for this task.

Please note that **all the arguments are valid**. In the other word, it is not necessary for you to pre-check the values and raise exceptions.

Task 4: Matrix Determinant

In this task, you need to calculate the determinant of the matrix stored in the class `Matrix`.

Input and Output

- Input:
    - There will be input
- Output:
    - `det`: `String`
    - For the value of `det`, please **keep 5 digits after decimal**.
        - hint: use `String.format`.