Programming Assignment #2 Matrix [행렬]

Matrices are very useful for solving mathematical problems. They are usually represented as a grid

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

consisting of rows(행) and columns(열).

In this assignment we will **create** classes for performing some simple matrix tasks.

Create a class named **Matrix** that will be used to represent an **integer** matrix.

1. Your class should have the following instance variables and methods
    1. Create an instance variable named **matArray[][]**. This is a **2D integer array** that will contain the numbers in the matrix.
    2. Create two (2) **integer** instance variables named **rows** and **cols**. these are the number of rows and columns in the matrix.
    3. Create a constructor **Matrix(int rows, int cols)** that can create an empty matrix.
    4. Create **accessors** named **getRows()** and **getCols()** that returns the rows and cols.
    5. Create two methods named **setValue(row, col, val)** and **getValue(row, col)**.
        i. setValue can be used to insert a number into the matrix.
        ii. getValue returns an integer number from the matrix.
    6. Create a method named **randomize()**. This method initializes the matrix with **random** numbers between 0 and 100.
    7. Create a method named **add(Matrix)**.
        i. This method accepts a Matrix as a parameter
        ii. In this method the matrix adds **itself** to another **Matrix** and returns the sum(덛셈).
    8. Create a method named **transpose()**.
        i. This method has no parameters
        ii. In this method a matrix transposes itself. It changes the rows to columns **{see hint}**
    9. Create a method named **multiply(Matrix)**.
        i. This method accepts a Matrix as a parameter
        ii. In this method the matrix multiplies **itself** with another **Matrix** and returns the product(곱셈)
    10. Create a method named **equals(Object)**.
        i. This method is used to determine if a matrix is equal to another matrix.
        ii. Matrix is equal if all the values are the same as well as rows and cols are the same. **{use the better equals method}**
    11. Create a method named **toString()**. This method **returns a String** of the matrix like this.

```
[  4 82 60 ]
[ 41 24 63 ]
[  0 78 80 ]
```

2. Square Matrices (정방 행렬) are special types of matrices. .The number of rows is equal to the number of columns.
   Create a new class named **SquareMatrix** that **extends** the Matrix class.
   Your class should have the following instance variables and methods.
    1. Create an instance variable named **dim.** This is the dimension of the square matrix. The dim of the square matrix is equal to the rows and the columns.
    2. Create a constructor **SquareMatrix(int dim)** that accepts the dim as a parameter.
       {hint your constructor should reference the super class's constructor.}
    3. Create an accessor **getDim()** that returns dim.
    4. Create a method named **isDiagonal().**
        i. This method does not accept a parameter and it returns a Boolean.
        ii. If a square matrix is diagonal (대각 행렬) it returns true. **{see hint}**
    5. Create a method named **identity().**
        i. This method returns an identity matrix (단위 행렬) that is the same size as this square matrix. **{see hint}**

3. In our scenario we can think of vectors (벡터) as a 1D matrix. E.g. [9 99 34 96]
   Create a new class named **Vector** that **extends** the Matrix class.
   Your class should have the following instance variables and methods.
    1. Create an instance variable named **dim**. This is the dimension of the vector. The dim of a vector is equal to the number of columns. **dim == col, rows == 1**
    2. Create a constructor **Vector(int dim)** that accepts the dim as a parameter.
       {hint your constructor should reference the super class's constructor.}
    3. Create an accessor **getDim()** that returns dim.
    4. Create a method multiply(Matrix) that Override the **multiply method** of the matrix class.
        i. This method accepts a Matrix as a parameter
        ii. This method multiplies **this vector** with the other **Matrix** and returns the product(곱셈)
    5. **{ hint vector dot product}**

Create a class to test your vectors and matrices. This class will have your main method. E.g

```java
public class test {
    public static void main(String[] args) {
        Matrix testMat = new Matrix(4,2);
        Matrix testMat2 = new Matrix(2,3);
        Matrix testMat3 = new Matrix(2,3);
        SquareMatrix sqMat = new SquareMatrix(3);
        Vector vec = new Vector(4);
        Vector vec2 = new Vector(4);

        testMat.randomize();
        testMat2.randomize();
        testMat3.randomize();
        sqMat.randomize();
        vec.randomize();
        vec2.randomize();

        System.out.println("Test Matrix 1: \n"+testMat);
        System.out.println("Test Matrix 2: \n"+testMat2);
        System.out.println("Test Matrix 3: \n"+testMat3);

        System.out.println("Square Matrix: \n"+sqMat);
        System.out.println("Test Matrix1 * Test Matrix 2: \n"+testMat.multiply(testMat2));
        System.out.println("Test Matrix2 * Square Matrix : \n"+testMat2.multiply(sqMat));
        System.out.println("Test Matrix2 + Test Matrix 3 : \n"+testMat2.add(testMat3));

        sqMat.transpose();
        System.out.println("Transposed Square Matrix: \n"+sqMat);


        System.out.println("Vectors: \n"+vec+"\n"+vec2);
        System.out.println("vector1 * vector2: "+vec.multiply(vec2));
        System.out.println("Square Matrix identity: \n"+sqMat.identity());
        System.out.println("Is Square Matrix diagonal?: "+sqMat.isDiagonal());
        System.out.println("Square Matrix identity diagonal?: "+sqMat.identity().isDiagonal());
    }
}
```

Sample Output:

```
Test Matrix 1:
[ 10 91 ]
[ 18  1 ]
[ 22 13 ]
[  6 72 ]

Test Matrix 2:
[ 53 52 13 ]
[  8 18 60 ]

Test Matrix 3:
[  4 35 88 ]
[ 26 23 47 ]

Square Matrix:
[  4 82 60 ]
[ 41 24 63 ]
[  0 78 80 ]
```

```
Test Matrix1 * Test Matrix 2:
[ 1258 2158 5590 ]
[  962  954  294 ]
[ 1270 1378 1066 ]
[  894 1608 4398 ]

Test Matrix2 * Square Matrix :
[ 2344 6608 7496 ]
[  770 5768 6414 ]

Test Matrix2 + Test Matrix 3 :
[  57  87 101 ]
[  34  41 107 ]

Transposed Square Matrix:
[  4 41  0 ]
[ 82 24 78 ]
[ 60 63 80 ]
```

```
Vectors:
[ 80 25 86 63 ]

[  9 99 34 96 ]

vector1 * vector2: [ 12167 ]

Square Matrix identity:
[  1  0  0 ]
[  0  1  0 ]
[  0  0  1 ]

Is Square Matrix diagonal?: false
Square Matrix identity diagonal?: true
```

Hints

$$\mathbf{A}+\mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

transpose

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & . & . & . & a_{1n} \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ a_{m1} & . & . & . & a_{mn} \end{bmatrix} \qquad \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} a_{11} & . & . & . & a_{m1} \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ a_{1n} & . & . & . & a_{mn} \end{bmatrix}$$

multiply

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix} \quad \mathbf{AB} = \begin{pmatrix} (\mathbf{AB})_{11} & (\mathbf{AB})_{12} & \cdots & (\mathbf{AB})_{1p} \\ (\mathbf{AB})_{21} & (\mathbf{AB})_{22} & \cdots & (\mathbf{AB})_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{AB})_{n1} & (\mathbf{AB})_{n2} & \cdots & (\mathbf{AB})_{np} \end{pmatrix}$$

Diagonal matrix

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}$$

Identity Matrix

Identity is a diagonal matrix where the diagonal is 1

$$I_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Vector dot product

$$c = ab^T$$ (only if a and b have same number of elements)

You need to transpose the vector.

$$c = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}_{1\times 3} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}_{3\times 1}$$

Submission :

You have to submit the source code and the documentation file compressed into a zip/jar file.

Written Documentation

In the documentation you should include a description of the implementation methodology as well as an explanation of the program design and structure. **Also you have to include print screen or screen shot or snapshot of your program's output in the document.**

- Run the program using the test class given.
- Capture screenshots of the results.
- Run the program with Random matrices of different dimensions.
- Capture screenshots of the results.

Online Submission

Submit your assignment via the course on canvas.instructure.com. All students should have already been enrolled onto Canvas and be given access to the course.

Select Assignment #2 from the Assignment's menu and use the button to submit the assignment.

The compressed file should have the name "**ClassNo-OOP-StudentIdNumber-Surname**".

For example: **1-OOP-2008012409-kim**

2-OOP-2009003987-Bae

Hard-copy submission:

You have to submit a hard-copy (printout) of Program, output and documentation prior to the start of the class on: **Thursday, May 12, 2016 for Class 1.**

Friday, May 13, 2016 for Class 2.

Scoring Criteria:

- You will get 100 points if the program satisfies all the requested features and runs smoothly.
- Note: **50 Points Online Submission** and **50 Points Hard-Copy**.
- For missing features (e.g. not repeatedly asking for input) points will be deducted.
- In case you do not meet the deadline,
    - 50% of your score will be deducted for a delay within 24 hours
    - 75% of your score will be deducted for a delay within 48 hours
    - 0 points will be given for a delay of more than 48 hours.

Assistance and Explanations:

If you need assistance or more explanations you can visit me at

- IT/BT Room 402-2
- Mondays and Wednesdays from 2pm – 4pm.