uc3m | Universidad Carlos III de Madrid

Estructura de Datos y Algoritmos (EDA)
Autor: Profesores EDA

Problema OOP

Implementa una clase, Matrix, que representa una matriz matemática de enteros. La clase debe proporcionar métodos para crear matrices, manejarlas aritméticamente y algebraicamente, y determinar sus propiedades matemáticas (traza, rango, inversa, determinante).

¿Cómo se almacena una matriz MxN en la memoria? ¿Qué atributos se necesitan y de qué tipos son (estáticos o no estáticos)?

Se pide:

1) Añadir un método constructor para crear una matriz de ceros.

2) Añadir un método para inicializar los valores de una matriz con valores aleatorios entre 0 y 10.

3) Añadir un método no estático, **show**, para mostrar los valores de la matriz invocada.

4) Añadir un método no estático, **transpose**, para crear y devolver la transpuesta de la matriz invocada.

5) Añadir un método no estático, **plus**, que toma una matriz como parámetro de entrada, y devolver una nueva matriz que sea la suma de la matriz invocada y la matriz pasada como parámetro. Si las matrices no se pueden sumar, el método debería mostrar un mensaje de error y devolver null.

6) Añadir un método no estático, **minus**, que toma una matriz como parámetro de entrada, y devolver una nueva matriz que es el resultado de la diferencia entre la matriz invocada y la matriz pasada como parámetro. Si las matrices no se pueden restar, el método debería mostrar un mensaje de error y devolver null.

7) Añadir un método no estático, **equal**, que toma una matriz como parámetro de entrada, y verifica si la matriz invocada es igual a la matriz pasada como parámetro.

8) Añadir un método estático que cree y devuelva la matriz identidad NxN.

Solución

```
package unit0;
import java.util.Random;

public class Matrix {
          private final int M;             // number of rows
        private final int N;             // number of columns
        private final int[][] data;   // M-by-N array

        // create M-by-N matrix of 0's
        public Matrix(int M, int N) {
            this.M = M;
            this.N = N;
            data = new int[M][N];
        }

        // create matrix based on 2d array
        public Matrix(int[][] data) {
                Random rn=new Random();
            M = data.length;
            N = data[0].length;
            this.data = new int[M][N];
            for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                        this.data[i][j] = rn.nextInt(10);
        }

        // create and return a random M-by-N matrix with values
between 0 and 1
        public static Matrix random(int M, int N) {
            Matrix A = new Matrix(M, N);
            Random rn=new Random();
            for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                    A.data[i][j] = rn.nextInt();
            return A;
        }

        // create and return the N-by-N identity matrix
        public static Matrix identity(int N) {
            Matrix I = new Matrix(N, N);
            for (int i = 0; i < N; i++)
                I.data[i][i] = 1;
            return I;
        }

        // create and return the transpose of the invoking matrix
        public Matrix transpose() {
            Matrix A = new Matrix(N, M);
            for (int i = 0; i < M; i++)
```

```
                for (int j = 0; j < N; j++)
                    A.data[j][i] = this.data[i][j];
            return A;
        }

        // return C = A + B
        public Matrix plus(Matrix B) {
            Matrix A = this;
            if (B.M != A.M || B.N != A.N) throw new
RuntimeException("Illegal matrix dimensions.");
            Matrix C = new Matrix(M, N);
            for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                    C.data[i][j] = A.data[i][j] + B.data[i][j];
            return C;
        }


        // return C = A - B
        public Matrix minus(Matrix B) {
            Matrix A = this;
            if (B.M != A.M || B.N != A.N) throw new
RuntimeException("Illegal matrix dimensions.");
            Matrix C = new Matrix(M, N);
            for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                    C.data[i][j] = A.data[i][j] - B.data[i][j];
            return C;
        }

        // does A = B exactly?
        public boolean eq(Matrix B) {
            Matrix A = this;
            if (B.M != A.M || B.N != A.N) throw new
RuntimeException("Illegal matrix dimensions.");
            for (int i = 0; i < M; i++)
                for (int j = 0; j < N; j++)
                    if (A.data[i][j] != B.data[i][j]) return false;
            return true;
        }

        // return C = A * B
        public Matrix times(Matrix B) {
            Matrix A = this;
            if (A.N != B.M) throw new RuntimeException("Illegal
matrix dimensions.");
            Matrix C = new Matrix(A.M, B.N);
            for (int i = 0; i < C.M; i++)
                for (int j = 0; j < C.N; j++)
                    for (int k = 0; k < A.N; k++)
                        C.data[i][j] += (A.data[i][k] *
B.data[k][j]);
```

```java
        return C;
    }

    // print matrix to standard output
    public void show() {
        for (int i = 0; i < M; i++) {
            for (int j = 0; j < N; j++)
                System.out.print(data[i][j]);
            System.out.println();
        }
    }

    // test client
    public static void main(String[] args) {
        int[][] d = { { 1, 2, 3 }, { 4, 5, 6 }, { 9, 1, 3} };
        Matrix D = new Matrix(d);
        D.show();
        System.out.println();

        Matrix A = Matrix.random(5, 5);
        A.show();
        System.out.println();
        Matrix B = A.transpose();
        B.show();
        System.out.println();

        Matrix C = Matrix.identity(5);
        C.show();
        System.out.println();

        A.plus(B).show();
        System.out.println();

        B.times(A).show();
        System.out.println();

        // shouldn't be equal since AB != BA in general
        System.out.println(A.times(B).eq(B.times(A)));
        System.out.println();

        Matrix b = Matrix.random(5, 1);
        b.show();
        System.out.println();


        A.times(b).show();

    }
}
```