Ejercicios matrices

Deberas crear una clase MatrixOperations que implemente los siguientes metodos estaticos:

Ejercicio 1: metodo sum. De entrada dos matrices NxM devuelva la suma de las dos matrices.

Ejercicio 2: metodo product. De entrada dos matrices NxM y MxP que calcule y devuelva el producto de ambas matrices.

Ejercicio 3: metodo transpose. De entrada una matriz y que calcule la matriz traspuesta.

Ejercicio 4: metodo determinant. Metodo que calcule el determinante de una matriz NxN

Ejercicio 5: metodo isMatrixSymmetric. Metodo que recibe una matriz como parametro y devuelve si es simetrica o no.

Ejercicio 6: isMatrixDiagonal. Metodo que recibe una matriz como parametro y devuelve si es diagonal o no.

Ejercicio 7: isMatrixRowEchelonForm. Metodo que recibe una matriz y devuelve si esta en forma escalonada o no.

```
public class MatrixOperations {
    // Ejercicio 1
    public static int[][] sum(int[][] matrix1, int[][] matrix2) {
        int n = matrix1.length;
        int m = matrix1[0].length;
        int[][] result = new int[n][m];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }

        return result;
    }

    // Ejercicio 2
    public static int[][] product(int[][] matrix1, int[][] matrix2) {
        int n = matrix1.length;
        int m = matrix1[0].length;
        int p = matrix2[0].length;

        int[][] result = new int[n][p];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < p; j++) {
                for (int k = 0; k < m; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        return result;
    }

    // Ejercicio 3
    public static int[][] transpose(int[][] matrix) {
        int n = matrix.length;
        int m = matrix[0].length;

        int[][] result = new int[m][n];
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            result[j][i] = matrix[i][j];
        }
    }

    return result;
}

// Ejercicio 4
public static int determinant(int[][] matrix) {
    int n = matrix.length;

    if (n == 1) {
        return matrix[0][0];
    }

    int det = 0;
    int sign = 1;

    for (int i = 0; i < n; i++) {
        int[][] subMatrix = getSubMatrix(matrix, 0, i);
        printMatrix(subMatrix);
        det += sign * matrix[0][i] * determinant(subMatrix);
        sign *= -1;
    }

    return det;
}

private static int[][] getSubMatrix(int[][] matrix, int row, int col) {
    int n = matrix.length;
    int[][] subMatrix = new int[n - 1][n - 1];
    int rowIndex = 0;
    int colIndex;

    for (int i = 0; i < n; i++) {
        if (i == row) {
            continue;
        }

        colIndex = 0;
        for (int j = 0; j < n; j++) {
            if (j == col) {
                continue;
            }

            subMatrix[rowIndex][colIndex] = matrix[i][j];
            colIndex++;
        }

        rowIndex++;
    }

    return subMatrix;
```

```
}

// Ejercicio 5
public static boolean isMatrixSymmetric(int[][] matrix) {
    int n = matrix.length;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] != matrix[j][i]) {
                return false;
            }
        }
    }

    return true;
}

// Ejercicio 6
public static boolean isMatrixDiagonal(int[][] matrix) {
    int n = matrix.length;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i != j && matrix[i][j] != 0) {
                return false;
            }
        }
    }

    return true;
}

// Ejercicio 7
public static boolean isMatrixRowEchelonForm(int[][] matrix) {
    int rowCount = matrix.length;
    int colCount = matrix[0].length;
    int previousNonZeroCol = -1;

    for (int i = 0; i < rowCount; i++) {
        boolean allZeros = true;
        int firstNonZeroCol = -1;

        for (int j = 0; j < colCount; j++) {
            if (matrix[i][j] != 0) {
                allZeros = false;
                if (firstNonZeroCol == -1)
                    firstNonZeroCol = j;
            }
        }

        if (allZeros && i < rowCount - 1)
            return false;

        if (firstNonZeroCol != -1 && firstNonZeroCol <= previousNonZeroCol)
            return false;
```

```
            previousNonZeroCol = firstNonZeroCol;
        }

        return true;
    }

    public static void printMatrix(int[][] matrix) {
        int rows = matrix.length;
        int cols = matrix[0].length;

        for (int[] ints : matrix) {
            for (int j = 0; j < cols; j++) {
                System.out.print(ints[j] + " ");
            }
            System.out.println();
        }
    }
}
```