Object Oriented Programming (Java)

This Java program, when given an integer input between 1 and 10 or two file paths, follows distinct paths. In the case of an integer input, the program generates two random matrices of size N by N, where N is the provided integer. The matrices are then multiplied, and the program displays both the randomly generated matrices and the resulting matrix. If file paths are provided instead, the program anticipates the matrices to be stored in the files, with the size of each matrix specified in the initial line, separated by a space. For instance, for a 2 by 3 matrix, the first line of the file should contain '2 3', followed by the actual matrix on the next lines. The program checks for matrix dimensions and file access errors during execution.

Code:

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import java.io.FileWriter;

import java.io.FileReader;

import java.io.BufferedWriter;

import java.io.BufferedReader;

import java.util.Random;

public class TimesMatrix

{

    public static void main(String[] args)

    {

        Scanner scan = new Scanner(System.in);

        if(args.length == 1 && isNumeric(args[0]))

        {

            int size = Integer.parseInt(args[0]);

            RandomMatrixOperation(size);

        }

        else if(args.length == 2)

        {

            String file1Path = args[0];

            String file2Path = args[1];

            FileMatrixOperations(file1Path, file2Path);

        }

        else

        {

            System.out.println(" ");

            System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

            System.out.println("Enter a integer between 1 - 10 to specify the size of randomly");

            System.out.println("generated matrices for multiplication. Alternatively, input the file path of two files containing matrices.");

            System.out.println("Ensure each file starts with the size of the matrix, for example");

            System.out.println("for a 2 by 3 matrix the first line of the file should contain '2 3' followed by");

            System.out.println("the actual matrix on the next.");

            System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

            if(scan.hasNextInt())

            {

                int size = scan.nextInt();

                RandomMatrixOperation(size);

            }

            else

            {

               String file1Path = scan.next();

               String file2Path = scan.next();

               FileMatrixOperations(file1Path, file2Path);

            }

        }

        scan.close();

    }

    static boolean isNumeric(String str)

    {

        return str.matches("-?\\d+(\\.\\d+)?");

    }

    static void RandomMatrixOperation(int size)

    {

        int[][] mt1 = new int[size][size];

        int[][] mt2 = new int[size][size];

        int[][] mtResult;

        MatrixRandom(mt1,size);

        MatrixRandom(mt2,size);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

        System.out.println("Matrix 1 =\n");

        SeeMatrix(mt1);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

        System.out.println("\nMatrix 2 =\n");

        SeeMatrix(mt2);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

        mtResult = MatrixMultiplyOp(mt1,mt2);

        if(mtResult != null)

        {

            System.out.println("Result Matrix = ");

            SeeMatrix(mtResult);

        }

        else

        {

           System.out.println("ERROR!!");

        }

    }

    static void FileMatrixOperations(String file1Path, String file2Path)

    {

        int[][] mt1;

        int[][] mt2;

        mt1 = ReadMatrixFromF(file1Path);

        mt2 = ReadMatrixFromF(file2Path);

        if(mt1 == null || mt2 == null)

        {

            System.out.println("ERROR!!!");

            return;

        }

        System.out.println("Matrix 1 =");

        SeeMatrix(mt1);

        System.out.println("Matrix 2 =");

        SeeMatrix(mt2);

        if (mt1[0].length != mt2.length)

        {

            System.out.println("Matrix dimensions are incompatiable");

            return;

        }

        int[][] Resultmt = MatrixMultiplyOp(mt1,mt2);

        if(Resultmt != null)

        {

            System.out.println("Result Matrix =");

            SeeMatrix(Resultmt);

            FileMatrixSve(Resultmt,"matrixResult.txt");

        }

        else

        {

            System.out.println ("ERROR!!");

        }

    }

    static void MatrixRandom(int[][] matrix, int size)

    {

        Random rand = new Random();

        for(int i = 0; i< size; i++)

        {

            for(int j =0; j < size; j++)

            {

                matrix[i][j] = rand.nextInt(10);

            }

        }

    }

    static void SeeMatrix(int[][] matrix)

    {

        for(int i = 0; i < matrix.length; i++)

        {

            for(int j = 0; j < matrix[0].length; j++)

            {

                System.out.print(matrix[i][j] + " ");

            }

            System.out.println();

        }

    }

    static int[][] MatrixMultiplyOp(int[][] matrix1, int[][] matrix2)

    {

        int row1 = matrix1.length;

        int col1 = matrix1[0].length;

        int row2 = matrix2.length;

        int col2 = matrix2[0].length;

        if (col1 != row2)

        {

            System.out.println("Matrix dimensions are incompatiple for multiplication");

            return null;

        }

        int[][] Resultmt = new int[row1][col2];

        for(int i = 0; i < row1; i++)

        {

            for(int j = 0; j < col2; j++)

            {

                for(int k = 0; k < col1; k++)

                {

                    Resultmt[i][j] = Resultmt[i][j] + matrix1[i][k]\*matrix2[k][j];

                }

            }

        }

        return Resultmt;

    }

    static int[][] ReadMatrixFromF(String filePath)

    {

        File file = new File(filePath);

        if(!file.exists())

        {

            System.out.println("ERROR!!");

            return null;

        }

        try (Scanner scan = new Scanner(file))

        {

            int rows = scan.nextInt();

            int cols = scan.nextInt();

            int[][] matrix = new int[rows][cols];

            for(int i = 0; i < rows; i++)

            {

                for(int j = 0; j < cols; j++)

                {

                    matrix[i][j] = scan.nextInt();

                }

            }

            return matrix;

        }

        catch(FileNotFoundException e)

        {

            System.out.println("ERROR!!"+ e.getMessage());

            return null;

        }

    }

static void FileMatrixSve(int[][] matrix, String filePath)

{

    try(BufferedWriter writer = new BufferedWriter(new FileWriter(filePath)))

    {

        writer.write(matrix.length + " "+ matrix[0].length);

        writer.newLine();

        for(int[] row : matrix)

        {

            for (int value : row)

            {

                writer.write(value +" ");

            }

            writer.newLine();

        }

        System.err.println("Result save to file " + filePath);

    }

    catch(Exception e)

    {

        System.out.println("ERROR!!!");

    }

}

}