MATRIX ALMOST EXPERT

OOP project by Zuzanna Ławniczak

WHAT DOES IT DO?

CLASS UI

```
public class UI {
    public void printUI() {

System.out.println("=========================================
======================");
        System.out.println("                    MENU                    ");
        System.out.println("1. Create a new matrix");
        System.out.println("2. Delete a matrix from memory");
        System.out.println("3. Print all stored matrices");
        System.out.println("-----------------------------------");
        System.out.println("4. Add selected matrices");
        System.out.println("5. Subtract selected matrices");
        System.out.println("6. Multiply selected matrices");
        System.out.println("7. Transpose a matrix");
        System.out.println("8. Multiply a matrix by a number");
        System.out.println("-----------------------------------");
        System.out.println("9. Exit");
        System.out.println("-----------------------------------");
        System.out.println("Your choice (input an integer):");
    }
```

CORRESPONDING CLASS MENU

```
@Override
public void showMenu() {
    dialogue.printUI();
    int input = cin.nextInt();
    switch (input) {
        case 1 -> this.createAMatrixMenu();//creating a matrix
        case 2 -> this.deleteAMatrixMenu();//deleting a matrix from the memory
        case 3 -> this.displayMatrices();//printing matrices
        case 4 -> this.addMatrices();//adding two matrices
        case 5 -> this.subtractMatrices();//subtracting matrices
        case 6 -> this.multiplyMatrices();//multiplying matrices
        case 7 -> this.transposeAMatrix();//transposing a matrix
        case 8 -> this.multiplyByANumber();//multiplying matrix by a number
        case 9 -> System.exit(0);//exit
    }
    this.showMenu();
}
```

HOW IS IT IMPLEMENTED?

```
public class Element {
    private double value;

    Element(double initial) {
        //constructor to create an element with initial value
        this.value = initial;
    }

    Element() {
        //constructor to create an element with default initial value
        this.value = 0;
    }

    public Element newValue(double val) {
        return new Element(val);
    }

    public double printValue() {
        return value;
    }
}
```

```
public class Matrix implements Operations {
    private int numberOfRows, numberOfColumns;
    private Element[][] elements;

    Matrix(int m, int n) {
        this.numberOfRows = m;
        this.numberOfColumns = n;
        this.elements = new Element[numberOfRows][numberOfColumns];
    }

    public Matrix(int m, int n, double[][] tab) {
        this.numberOfRows = m;
        this.numberOfColumns = n;
        this.elements = new Element[numberOfRows][numberOfColumns];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                elements[i][j] = new Element(tab[i][j]);
            }
        }
    }
```

CLASS EXAMPLE & CLASS MENU PART 1

```java
package pl.poznan.put.example;
import pl.poznan.put.matrix.Matrix;

public class Example {
    //example matrices that will be added to the memory
    final private double tab[][] = {
            {21.0, 37.0, 5.0},
            {4.0, 5.0, 8.0},
            {2.9, 4.7, 1.9},
    };
    final private Matrix example1 = new Matrix(3, 3, tab);
    private double tab2[][] = {
            {6.0, 9.7, 4.9},
            {3.6, 2.9, 1.0},
            {2.3, 4.5, 5.8},
    };
    private Matrix example2 = new Matrix(3, 3, tab2);

    public Matrix firstExampleMatrix() {
        return example1;
    }

    public Matrix secondExampleMatrix() {
        return example2;
    }
}
```

```java
public class Menu implements MenuFunctions {
    private ArrayList<Matrix> matrices = new ArrayList<>();
    private Scanner cin = new Scanner(System.in);
    private UI dialogue = new UI();

    @Override
    public void insertAMatrix(Matrix m1) {
        //instead of using these 3 lines everywhere, I optimized it and used a method
        matrices.add(m1);
        System.out.println("Your matrix has been added to the memory");
        matrices.get(matrices.size() - 1).show();
    }

    @Override
    public boolean ifMatricesEmpty() {
        //instead of having try...catch block in every method, I check it here
        boolean flag = true;
        try {
            if (matrices.isEmpty()) {
                throw new InaccessibleObjectException("There are no matrices stored in memory!");
            } else flag = false;
        } catch (Exception ex1) {
            System.out.println(ex1.getMessage());
        }
        return flag;
    }
```

CLASS MENU PART (SKIPPED SHOWMENU METHOD AS IT IS SHOWN IN THE SECOND SLIDE)

```java
@Override
public boolean indexOutOfRange(int id) {
    //instead of having try...catch block in every method, I check it here
    //checking if index of an object exists
    boolean flag = true;
    try {
        if (((matrices.size() - 1) < id) || id < 0) {
            throw new IllegalAccessException("Index is out of range!");
        } else flag = false;
    } catch (Exception ex1) {
        System.out.println(ex1.getMessage());
    }
    return flag;
}
```

```java
@Override
public void createAMatrixMenu() {
    try {
        System.out.println("Input number of rows for the new matrix");
        int m = cin.nextInt();
        System.out.println("Input number of columns for the new matrix");
        int n = cin.nextInt();
        if (m <= 0 || n <= 0) {
            throw new IllegalArgumentException("Number of rows and columns of a matrix must be positive!");
        } else {
            double[][] tab2 = new double[m][n];
            System.out.println("Input values of the elements of the matrix. You should use your local notation (in Poland use a comma (,)
):");
            for (int i = 0; i < m; i++) {
                for (int j = 0; j < n; j++) {
                    tab2[i][j] = cin.nextDouble();
                }
            }
            Matrix mat = new Matrix(m, n, tab2);
            insertAMatrix(mat);
        }
    } catch (IllegalArgumentException ex1) {
        System.out.println(ex1.getMessage());
        this.createAMatrixMenu();
    }
}
```

CLASS MENU

```java
@Override
public void displayMatrices() {
    if (!ifMatricesEmpty()) {
        for (int i = 0; i < matrices.size(); i++) {
            System.out.print(i);
            System.out.println(": ");
            matrices.get(i).show();
        }
    }
}

@Override
public void deleteAMatrixMenu() {
    if (!ifMatricesEmpty()) {
        System.out.println("Input an index of a matrix you want to delete!:");
        int id = cin.nextInt();
        if (!indexOutOfRange(id)) {
            matrices.remove(id);
            System.out.println("Your matrix has been deleted.");
        }
    }
}
```

```java
@Override
public void addMatrices() {
    int id1, id2;
    System.out.println("Input the index of the matrix you want to add to:");
    id1 = cin.nextInt();
    System.out.println("Input the index of the matrix you want to add:");
    id2 = cin.nextInt();
    if (!indexOutOfRange(id1) && !indexOutOfRange(id2)) {
        if (matrices.get(id1).numberOfRows() != matrices.get(id2).numberOfRows()) {
            System.out.println("Impossible to add.");
        } else if (matrices.get(id1).numberOfColumns() != matrices.get(id2).numberOfColumns()) {
            System.out.println("Impossible to add.");
        } else {
            Matrix newMatrix = matrices.get(id1).addToThisMatrix(matrices.get(id2));
            insertAMatrix(newMatrix);
        }
    }
}
```

CLASS MENU

```
@Override
public void subtractMatrices() {
    int id1, id2;
    System.out.println("Input the index of the matrix you want to subtract from:");
    id1 = cin.nextInt();
    System.out.println("Input the index of the matrix you want to subtract:");
    id2 = cin.nextInt();
    if (!indexOutOfRange(id1) && !indexOutOfRange(id2)) {
        if (matrices.get(id1).numberOfRows() != matrices.get(id2).numberOfRows()) {
            System.out.println("Impossible to subtract.");
        } else if (matrices.get(id1).numberOfColumns() != matrices.get(id2).numberOfColumns())
{
            System.out.println("Impossible to subtract.");
        } else {
            Matrix newMatrix = matrices.get(id1).subtractFromThisMatrix(matrices.get(id2));
            insertAMatrix(newMatrix);
        }
    }
}
```

```
@Override
public void multiplyMatrices() {
    int id1, id2;
    boolean fl1 = ifMatricesEmpty();
    if (!fl1) {
        System.out.println("Input the index of the matrix you want to multiply");
        id1 = cin.nextInt();
        System.out.println("input the index of the matrix you want to multiply by");
        id2 = cin.nextInt();
        fl1 = indexOutOfRange(id1) || indexOutOfRange(id2);
        if (!fl1) {
            Matrix newMatrix = new Matrix(matrices.get(id1), matrices.get(id2));
            insertAMatrix(newMatrix);
        }
    }
}
```

CLASS MENU

```java
@Override
public void transposeAMatrix() {
    int id1;
    boolean fl1 = ifMatricesEmpty();
    if (!fl1) {
        System.out.println("Input the index of the matrix you want to transpose:");
        id1 = cin.nextInt();
        fl1 = indexOutOfRange(id1);
        if (!fl1) {
            Matrix newMatrix = new Matrix(matrices.get(id1));
            insertAMatrix(newMatrix);
        }
    }
}
```

```java
@Override
public void multiplyByANumber() {
    int id1;
    double number;
    boolean fl1 = ifMatricesEmpty();
    if (!fl1) {
        System.out.println("Input the index of the matrix you want to multiply:");
        id1 = cin.nextInt();
        fl1 = indexOutOfRange(id1);
        if (!fl1) {
            System.out.println("Input the number");
            number = cin.nextDouble();
            Matrix newMatrix = new Matrix(matrices.get(id1), number);
            insertAMatrix(newMatrix);
        }
    }
}
```

THERE ARE TWO INTERFACES

```java
public interface Operations {
    //this interface is implemented by the Matrix class
    //addition, subtraction are methods
    //multiplication, transposition are handled by
constructors in Matrix class
    void show();

    Operations addToThisMatrix(Matrix m1);

    Operations subtractFromThisMatrix(Matrix m2);
}
```

```java
public interface MenuFunctions {
    void insertAMatrix(Matrix m1);

    boolean ifMatricesEmpty();

    boolean indexOutOfRange(int id);//checking if index of an object exists

    void showMenu();

    void createAMatrixMenu();

    void displayMatrices();

    void deleteAMatrixMenu();

    void addMatrices();

    void subtractMatrices();

    void multiplyMatrices();

    void transposeAMatrix();

    void multiplyByANumber();

}
```

PROBLEMS AND HOW I HANDLED THEM

GUI

After a few attempts…

I abandoned it

REDUNDANT ERROR CHECKING IN METHODS

I made new methods that did the same & allowed me to delete about 60 lines of code

PROJECT STRUCTURE

After finishing the project, I was told that the project structure has to be changed & I spent a few hours trying to fix all the errors

How I fixed it: importing packages

DID YOU LEARN ANYTHING NEW?

Fast answer: a lot

I REALIZED:

- It’s incredibly easy to switch to Java after programming in C++ for 7 years
- Looking for an inverse matrix is a lot harder that I thought it be
- JetBrains tools are **the best**

WHAT COULD BE IMPROVED?

Nothing.

WHAT COULD BE IMPROVED?

I’m happy with the result

But If I needed to find sth to improve: actually making a GUI

THANK YOU FOR YOUR ATTENTION

Zuzanna Ławniczak, 151835