ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и кибербезопасности

Высшая школа программной инженерии

**Курсовая работа**

Разработка графического интерфейса для пользователя на языке JAVA

Выполнил

Студент группы в5130904/10321 Мухин А.Н.

Преподаватель Маслаков А.П.

# 

Санкт-Петербург 2023

# Цели

В процессе выполнения курсовой работы были созданы Java-приложения с текстовым интерфейсом, а также с графическим интерфейсом. Главная цель при выполнении заданий без графического интерфейса заключалась в освоении языка программирования Java, в то время как при разработке приложений с графическим интерфейсом важной задачей было обеспечение выбора задания из списка, ввода необходимых данных и вывода результатов в соответствующих защищенных текстовых областях. Для достижения этих целей использовались средства Java и разнообразные технологии пользовательского интерфейса.

# Технологии

Swing - это набор библиотек пользовательского интерфейса (UI) для Java, предоставляющий компоненты и более высокоуровневые структуры для создания графического интерфейса пользователя (GUI) в приложениях Java. Swing является частью Java Foundation Classes (JFC) и предоставляет широкий спектр компонентов, включая кнопки, текстовые поля, меню, таблицы, панели и другие элементы интерфейса.

Некоторые ключевые особенности технологии Swing:

1. Платформонезависимость: Swing предлагает нативные оформления для различных операционных систем, что позволяет создавать кросс-платформенные интерфейсы на Java.

2. Модель событий и обработчиков: Swing основан на модели событий, которая позволяет действовать на пользовательский ввод или другие события, происходящие в UI.

3. Многофункциональные компоненты: Swing предоставляет широкий выбор компонентов для построения интерфейса, включая сложные компоненты, такие как таблицы и деревья.

4. Гибкость и расширяемость: В Swing существует возможность создания собственных компонентов и элементов управления для создания решений, наиболее соответствующих индивидуальным потребностям.

# Разработка приложения

## Основной класс (Main)

import javax.swing.\*;  
import java.awt.\*;

Main(){…}  
  
 public static void main(String args[]){  
 EventQueue.*invokeLater*(() -> {  
 try {  
 Main frame = new Main();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
}

Данный код служит для вызова главного окна программы, в котором находится интерфейс с выбором заданий.

Класс Main наследуется от JFrame и содержит методы для выполнения каждого задания

## Класс главного окна (MainScreen)

import lab1.Hero;  
import lab1.SwingMovementUI;  
import lab2.MatrixConsoleApplication;  
import lab3.TranslatorConsoleApplication;

public class Main extends JFrame{  
 JMenu menu;  
 JMenuItem a1,a2;

Main(){  
 super("lab\_6");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setBounds(100, 100, 300, 700);  
 JPanel contentPane = new JPanel();  
 setContentPane(contentPane);  
 contentPane.setLayout(null);  
  
 JButton btnLab1 = new JButton("Lab1");  
 btnLab1.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnLab1) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 Hero hero = new Hero();  
  
 SwingMovementUI movementUI = new SwingMovementUI(hero);  
 movementUI.createAndShowUI();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 setVisible(true);  
 });  
 btnLab1.setBounds(27, 20, 250, 100);  
 contentPane.add(btnLab1);  
  
 JButton btnLab2 = new JButton("Lab 2");  
 btnLab2.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnLab2) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 MatrixConsoleApplication frame = new MatrixConsoleApplication();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 setVisible(true);  
 });  
 btnLab2.setBounds(27, 130, 250, 100);  
 contentPane.add(btnLab2);  
  
 JButton btnLab3 = new JButton("Lab 3");  
 btnLab3.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnLab3) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 TranslatorConsoleApplication frame = new TranslatorConsoleApplication();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 setVisible(true);  
 });  
 btnLab3.setBounds(27, 240, 250, 100);  
 contentPane.add(btnLab3);  
  
 JButton btnLab4 = new JButton("Lab 4");  
 btnLab4.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnLab4) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 lab4.Main frame = new lab4.Main();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 setVisible(true);  
 });  
 btnLab4.setBounds(27, 350, 250, 100);  
 contentPane.add(btnLab4);  
  
 JButton btnLab5 = new JButton("Lab 5");  
 btnLab5.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnLab5) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 lab5.Main frame = new lab5.Main();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
 setVisible(true);  
 });  
 btnLab5.setBounds(27, 460, 250, 100);  
 contentPane.add(btnLab5);  
  
 JButton btnExit = new JButton("Exit");  
 btnExit.addActionListener(e -> System.*exit*(0));  
 btnExit.setBounds(202, 570, 75, 30);  
 contentPane.add(btnExit);  
}

Представленный код представляет собой графическое приложение на основе Swing. Он позволяет выбрать одно из 5 заданий из списка.

В данном коде создаётся главное окно размером 300 на 700 с выпадающим списком и мульти-текстовым блоком. Выпадающий список предлагает выбрать одно из заданий на выбор, а текстовый блок служит для вывода информации для пользователя.

Интерфейс представлен на рисунке 1 в Приложениях.

## Задания

### Задание 1

В компьютерной игре герой (класс Hero) может перемещаться между двумя точками (метод move) различными способами: идти пешком, ехать на лошади, лететь и т. п. Реализовать классы, позволяющие пользователю выбирать и менять в ходе выполнения программы способ перемещения героя, используя паттерн “стратегия” (strategy). Продемонстрировать работу реализованных классов.

Интерфейс представлен на рисунке 2 в Приложениях.

GUI

package lab1;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
// Простой графический интерфейс для выбора стратегии перемещения  
public class SwingMovementUI {  
 private Hero hero;  
 private String result;  
  
 public SwingMovementUI(Hero hero) {  
 this.hero = hero;  
 }  
  
 public void createAndShowUI() {  
  
 JFrame frame = new JFrame("Hero Movement");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setLayout(new FlowLayout());  
  
 JComboBox<String> strategyComboBox = new JComboBox<>(new String[]{"Пешком", "На лошади", "Лететь"});  
 JButton moveButton = new JButton("Двигаться");  
 JTextField answer = new JTextField();  
  
 moveButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String selectedStrategy = (String) strategyComboBox.getSelectedItem();  
 switch (selectedStrategy) {  
 case "Пешком":  
 hero.setMovementStrategy(new WalkStrategy());  
 break;  
 case "На лошади":  
 hero.setMovementStrategy(new HorseRideStrategy());  
 break;  
 case "Лететь":  
 hero.setMovementStrategy(new FlyStrategy());  
 break;  
 }  
 result = hero.move();  
 answer.setText(result);  
 }  
 });  
  
  
 frame.add(new JLabel("Выберите способ перемещения:"));  
 frame.add(strategyComboBox);  
 frame.add(moveButton);  
  
 answer.setBounds(20, 60, 200, 200);  
 answer.setColumns(100);  
 answer.setEditable(false);  
 frame.add(answer);  
  
 frame.pack();  
 frame.setVisible(true);  
  
 }  
}

Hero.java

package lab1;  
  
// Класс для представления героя  
public class Hero {  
 private MovementStrategy movementStrategy;  
  
 public void setMovementStrategy(MovementStrategy strategy) {  
 this.movementStrategy = strategy;  
 }  
  
 public String move() {  
 String result = "";  
 if (movementStrategy != null) {  
 result = movementStrategy.move();  
 } else {  
 result = "Герой не может двигаться";  
 }  
 return result;  
 }  
  
}

FlyStrateg.java

class FlyStrategy implements MovementStrategy {  
 @Override  
 public String move() {  
 return "Герою стало мало лошади поэтому он спер вертолет на котором полетл в светлое будущие";  
 }  
}

FlyStrategy.java

package TasksCode.task1.strategies;

import constants.MoveType;

public class FlyStrategy implements MoveStrategy {

@Override

public int move() {

return MoveType.*FLY*.getValue();

}

@Override

public String toString() {

return "орёл";

}

}

HeroMovementApp.java

public class HeroMovementApp {  
 public static void main(String[] args) {  
 // Создаем героя  
 Hero hero = new Hero();  
  
 // Создаем графический интерфейс  
 SwingMovementUI movementUI = new SwingMovementUI(hero);  
 movementUI.createAndShowUI();  
 }  
}

HorseRideStrategy.java

class HorseRideStrategy implements MovementStrategy {  
 @Override  
 public String move() {  
 return "Герою стало лень идти он украл лошадь у цыган и поскакал";  
 }  
}

WalkStrategy.java

class WalkStrategy implements MovementStrategy {  
 @Override  
 public String move() {  
 return "Герой пошуршал на своих двоих";  
 }  
}

### Задание 2

Написать приложение, которое:

a. Считывает из текстового файл размерность матрицы N\*N.

b. Создаёт и заполняет матрицу случайными числами от -N до N.

c. Последовательно поворачивает матрицу на 90, 180 и 270 градусов против часовой стрелки и делит каждый элемент на сумму соседних. d. Каждую из трёх получившихся матриц вывести в общий файл

Требования к обработке исключительных ситуаций:

a. контролировать состояние потоков ввода/вывода (отсутствие записи в файле, недопустимые значения, etc);

b. генерировать и обрабатывать исключение при некорректных математических операциях;

c. логировать исключение при нехватке памяти;

d. реализовать собственные классы исключений для случаев

● деление на 0

● файл не существует/нет доступа к файлу

● N > 1\_000\_000

Интерфейс представлен на рисунке 3 в Приложениях.

GUI

package lab2;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.io.File;  
import java.io.IOException;  
  
public class MatrixConsoleApplication extends JFrame {  
 private String pathToInputFile = "";  
 private String exception;  
 private final boolean append = true; //Параметр для очистки выходного файла, чтобы не чистить руками  
  
 private final String pathToOutputFile = "./src/lab2/matrixOutput.txt";  
  
 public MatrixConsoleApplication() {  
 super("lab\_2");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setBounds(100, 100, 1000, 1000);  
 JPanel contentPane = new JPanel();  
 setContentPane(contentPane);  
 contentPane.setLayout(null);  
  
 JLabel pathToInputFileLabel = new JLabel("Please specify the location of the input data file:");  
 pathToInputFileLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 pathToInputFileLabel.setBounds(27, 16, 400, 23);  
 contentPane.add(pathToInputFileLabel);  
  
 JTextField pathToInputFileField = new JTextField();  
 pathToInputFileField.setBounds(27, 39, 500, 23);  
 pathToInputFileField.setColumns(500);  
 pathToInputFileField.setEditable(false);  
 contentPane.add(pathToInputFileField);  
  
 JButton btnSelectInputFile = new JButton("Select a path to input file");  
 btnSelectInputFile.setBounds(527, 39, 100, 23);  
 contentPane.add(btnSelectInputFile);  
  
 JFileChooser pathInputFileChooser = new JFileChooser("./src/lab2/");  
 btnSelectInputFile.addActionListener(e -> {  
 // Определение режима - только каталог  
 pathInputFileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 int result = pathInputFileChooser.showOpenDialog(MatrixConsoleApplication.this);  
 // Если директория выбрана, покажем ее в сообщении  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
 pathToInputFileField.setText(String.*valueOf*(pathInputFileChooser.getSelectedFile()));  
 pathToInputFile = pathToInputFileField.getText();  
 }  
 });  
  
 JLabel calculationResultsLabel = new JLabel("Calculation results:");  
 calculationResultsLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 calculationResultsLabel.setBounds(27, 120, 400, 23);  
 contentPane.add(calculationResultsLabel);  
  
 JLabel exceptionLabel = new JLabel("Exceptions when the program is running:");  
 exceptionLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 exceptionLabel.setBounds(27, 150, 400, 23);  
 contentPane.add(exceptionLabel);  
  
 JTextField exceptionField = new JTextField();  
 exceptionField.setBounds(27, 173, 500, 23);  
 exceptionField.setColumns(500);  
 exceptionField.setEditable(false);  
 contentPane.add(exceptionField);  
  
 JLabel pathToOutputFileLabel = new JLabel("Path of the output file:");  
 pathToOutputFileLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 pathToOutputFileLabel.setBounds(27, 200, 400, 23);  
 contentPane.add(pathToOutputFileLabel);  
  
 JTextField pathToOutputFileField = new JTextField();  
 pathToOutputFileField.setBounds(27, 223, 500, 23);  
 pathToOutputFileField.setColumns(500);  
 pathToOutputFileField.setEditable(false);  
 contentPane.add(pathToOutputFileField);  
  
 JLabel originalMatrixLabel = new JLabel("The original matrix:");  
 originalMatrixLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 originalMatrixLabel.setBounds(727, 46, 450, 23);  
 contentPane.add(originalMatrixLabel);  
  
 JTextArea originalMatrixField = new JTextArea();  
 originalMatrixField.setBounds(727, 69, 138, 138);  
 originalMatrixField.setColumns(500);  
 originalMatrixField.setEditable(false);  
 contentPane.add(originalMatrixField);  
  
 JLabel rotatedMatrix90Label = new JLabel("The matrix is rotated by 90 degrees:");  
 rotatedMatrix90Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix90Label.setBounds(27, 246, 300, 23);  
 contentPane.add(rotatedMatrix90Label);  
  
 JTextArea rotatedMatrix90Field = new JTextArea();  
 rotatedMatrix90Field.setBounds(27, 269, 138, 138);  
 rotatedMatrix90Field.setColumns(500);  
 rotatedMatrix90Field.setEditable(false);  
 contentPane.add(rotatedMatrix90Field);  
  
 JLabel rotatedMatrix90DevidedLabel = new JLabel("A matrix rotated by 90 degrees and with elements divided by the sum of adjacent elements");  
 rotatedMatrix90DevidedLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix90DevidedLabel.setBounds(300, 246, 600, 23);  
 contentPane.add(rotatedMatrix90DevidedLabel);  
  
 JTextArea rotatedMatrix90DevidedField = new JTextArea();  
 rotatedMatrix90DevidedField.setBounds(300, 269, 138, 138);  
 rotatedMatrix90DevidedField.setColumns(500);  
 rotatedMatrix90DevidedField.setEditable(false);  
 contentPane.add(rotatedMatrix90DevidedField);  
  
 JLabel rotatedMatrix180Label = new JLabel("The matrix is rotated by 180 degrees:");  
 rotatedMatrix180Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix180Label.setBounds(27, 410, 300, 23);  
 contentPane.add(rotatedMatrix180Label);  
  
 JTextArea rotatedMatrix180Field = new JTextArea();  
 rotatedMatrix180Field.setBounds(27, 430, 138, 138);  
 rotatedMatrix180Field.setColumns(500);  
 rotatedMatrix180Field.setEditable(false);  
 contentPane.add(rotatedMatrix180Field);  
  
 JLabel rotatedMatrix180DevidedLabel = new JLabel("A matrix rotated by 180 degrees and with elements divided by the sum of adjacent elements");  
 rotatedMatrix180DevidedLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix180DevidedLabel.setBounds(300, 410, 600, 23);  
 contentPane.add(rotatedMatrix180DevidedLabel);  
  
 JTextArea rotatedMatrix180DevidedField = new JTextArea();  
 rotatedMatrix180DevidedField.setBounds(300, 430, 138, 138);  
 rotatedMatrix180DevidedField.setColumns(500);  
 rotatedMatrix180DevidedField.setEditable(false);  
 contentPane.add(rotatedMatrix180DevidedField);  
  
 JLabel rotatedMatrix270Label = new JLabel("The matrix is rotated by 270 degrees:");  
 rotatedMatrix270Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix270Label.setBounds(27, 568, 300, 23);  
 contentPane.add(rotatedMatrix270Label);  
  
 JTextArea rotatedMatrix270Field = new JTextArea();  
 rotatedMatrix270Field.setBounds(27, 591, 138, 138);  
 rotatedMatrix270Field.setColumns(500);  
 rotatedMatrix270Field.setEditable(false);  
 contentPane.add(rotatedMatrix270Field);  
  
 JLabel rotatedMatrix270DevidedLabel = new JLabel("A matrix rotated by 270 degrees and with elements divided by the sum of adjacent elements");  
 rotatedMatrix270DevidedLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 rotatedMatrix270DevidedLabel.setBounds(300, 568, 600, 23);  
 contentPane.add(rotatedMatrix270DevidedLabel);  
  
 JTextArea rotatedMatrix270DevidedField = new JTextArea();  
 rotatedMatrix270DevidedField.setBounds(300, 591, 138, 138);  
 rotatedMatrix270DevidedField.setColumns(500);  
 rotatedMatrix270DevidedField.setEditable(false);  
 contentPane.add(rotatedMatrix270DevidedField);  
  
 JButton btnCalculate = new JButton("Calculate");  
 btnCalculate.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnCalculate) {  
 pathToOutputFileField.setText("");  
 originalMatrixField.setText("");  
 rotatedMatrix90Field.setText("");  
 rotatedMatrix90DevidedField.setText("");  
 rotatedMatrix180Field.setText("");  
 rotatedMatrix180DevidedField.setText("");  
 rotatedMatrix270Field.setText("");  
 rotatedMatrix270DevidedField.setText("");  
  
 MatrixManipulationApp workingOnMatrix = new MatrixManipulationApp();  
 try {  
 String basePath = "src/lab2"; // Adjust this base path according to your project structure  
 int size = workingOnMatrix.*readMatrixSize*(basePath + File.*separator* + "matrix\_size.txt");  
  
 try {  
 workingOnMatrix.*checkMatrixSize*(size);  
 } catch (IllegalArgumentException e) {  
 e.printStackTrace();  
 System.*out*.println(e.getMessage());  
 return;  
 }  
  
  
 int[][] matrix = workingOnMatrix.*createRandomMatrix*(size);  
 workingOnMatrix.*writeMatrixToFile*(pathToOutputFile, matrix);  
  
 pathToOutputFileField.setText(pathToOutputFile);  
 originalMatrixField.setText(workingOnMatrix.printMatrix(matrix, pathToOutputFile, "Original matrix\n"));  
  
 int[][] rotatedMatrix90 = workingOnMatrix.*rotateMatrix*(matrix, 90);  
 int[][] rotatedMatrix180 = workingOnMatrix.*rotateMatrix*(matrix, 180);  
 int[][] rotatedMatrix270 = MatrixManipulationApp.*rotateMatrix*(matrix, 270);  
  
 int[][] dividedMatrix90 = workingOnMatrix.*divideMatrix*(rotatedMatrix90);  
 int[][] dividedMatrix180 = workingOnMatrix.*divideMatrix*(rotatedMatrix180);  
 int[][] dividedMatrix270 = workingOnMatrix.*divideMatrix*(rotatedMatrix270);  
  
 rotatedMatrix90Field.setText(workingOnMatrix.printMatrix(rotatedMatrix90, pathToOutputFile, "Matrix rotated 90 degrees counterclockwise\n"));  
 rotatedMatrix90DevidedField.setText(workingOnMatrix.printMatrix(dividedMatrix90, pathToOutputFile, "A matrix rotated by 90 degrees and with elements divided by the sum of adjacent elements\n"));  
  
 workingOnMatrix.*writeMatrixToFile*(pathToOutputFile, dividedMatrix90);  
  
 rotatedMatrix180Field.setText(workingOnMatrix.printMatrix(rotatedMatrix180, pathToOutputFile, "Matrix rotated 180 degrees counterclockwise\n"));  
 rotatedMatrix180DevidedField.setText(workingOnMatrix.printMatrix(dividedMatrix180, pathToOutputFile, "A matrix rotated by 180 degrees and with elements divided by the sum of adjacent elements\n"));  
  
 workingOnMatrix.*writeMatrixToFile*(pathToOutputFile, dividedMatrix180);  
  
 rotatedMatrix270Field.setText(workingOnMatrix.printMatrix(rotatedMatrix270, pathToOutputFile, "Matrix rotated 270 degrees counterclockwise\n"));  
 rotatedMatrix270DevidedField.setText(workingOnMatrix.printMatrix(dividedMatrix270, pathToOutputFile, "A matrix rotated by 270 degrees and with elements divided by the sum of adjacent elements\n"));  
  
 workingOnMatrix.*writeMatrixToFile*(pathToOutputFile, dividedMatrix270);  
  
 System.*out*.println("Результаты записаны в файлы result90.txt, result180.txt и result270.txt");  
  
 exception = "";  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 exception = "Произошла ошибка при чтении/записи файла";  
 } catch (NumberFormatException e) {  
 e.printStackTrace();  
 exception = "Произошла ошибка, введены некорректные исходные данные";  
 } catch (ArithmeticException e) {  
 e.printStackTrace();  
 exception = "Произошло деление на ноль во время обработки матрицы";  
 }  
  
 }  
 exceptionField.setText(exception);  
  
 setVisible(true);  
 });  
 btnCalculate.setBounds(27, 80, 600, 40);  
 contentPane.add(btnCalculate);  
  
 JButton btnExit = new JButton("Exit");  
 btnExit.addActionListener((ActionEvent e) -> {  
 JComponent comp = (JComponent) e.getSource();  
 Window win = SwingUtilities.*getWindowAncestor*(comp);  
 win.dispose();  
 });  
 btnExit.setBounds(727, 706, 75, 23);  
 contentPane.add(btnExit);  
 }  
  
 public static void main(String[] args) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 MatrixConsoleApplication frame = new MatrixConsoleApplication();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
}

MatrixManipulationApp.java

package lab2;  
  
import java.io.\*;  
import java.util.Random;  
  
public class MatrixManipulationApp {  
 private static final int *MAX\_MATRIX\_SIZE* = 1000000;  
  
 public static void main(String[] args) {  
 try {  
 String basePath = "src/lab2"; // Adjust this base path according to your project structure  
 int size = *readMatrixSize*(basePath + File.*separator* + "matrix\_size.txt");  
  
 try {  
 *checkMatrixSize*(size);  
 } catch (IllegalArgumentException e) {  
 e.printStackTrace();  
 System.*out*.println(e.getMessage());  
 return;  
 }  
  
 int[][] matrix = *createRandomMatrix*(size);  
 *writeMatrixToFile*(basePath + File.*separator* + "original\_matrix.txt", matrix);  
  
 int[][] rotatedMatrix90 = *rotateMatrix*(matrix, 90);  
 int[][] rotatedMatrix180 = *rotateMatrix*(matrix, 180);  
 int[][] rotatedMatrix270 = *rotateMatrix*(matrix, 270);  
  
 int[][] dividedMatrix90 = *divideMatrix*(rotatedMatrix90);  
 int[][] dividedMatrix180 = *divideMatrix*(rotatedMatrix180);  
 int[][] dividedMatrix270 = *divideMatrix*(rotatedMatrix270);  
  
 *writeMatrixToFile*(basePath + File.*separator* + "result90.txt", dividedMatrix90);  
 *writeMatrixToFile*(basePath + File.*separator* + "result180.txt", dividedMatrix180);  
 *writeMatrixToFile*(basePath + File.*separator* + "result270.txt", dividedMatrix270);  
  
 System.*out*.println("Результаты записаны в файлы result90.txt, result180.txt и result270.txt");  
 } catch (IOException e) {  
 e.printStackTrace();  
 System.*out*.println("Произошла ошибка при чтении/записи файла");  
 } catch (NumberFormatException e) {  
 e.printStackTrace();  
 System.*out*.println("Произошла ошибка, введены некорректные исходные данные");  
 } catch (ArithmeticException e) {  
 e.printStackTrace();  
 System.*out*.println("Произошло деление на ноль во время обработки матрицы");  
 }  
  
  
 }  
  
 public static void checkMatrixSize(int size) {  
 if (size > *MAX\_MATRIX\_SIZE*) {  
 throw new IllegalArgumentException("Размер матрицы не может превышать " + *MAX\_MATRIX\_SIZE*);  
 }  
 }  
  
 public static int readMatrixSize(String filePath) throws IOException {  
 try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  
 String line = reader.readLine();  
 return Integer.*parseInt*(line);  
 }  
 }  
  
 public static int[][] createRandomMatrix(int size) {  
 if (size > *MAX\_MATRIX\_SIZE*) {  
 throw new IllegalArgumentException("Размер матрицы не может превышать " + *MAX\_MATRIX\_SIZE*);  
 }  
  
 int[][] matrix = new int[size][size];  
 Random random = new Random();  
  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 matrix[i][j] = random.nextInt(2 \* size + 1) - size;  
 }  
 }  
  
 return matrix;  
 }  
  
 public static void writeMatrixToFile(String fileName, int[][] matrix) throws IOException {  
 try (FileWriter writer = new FileWriter(fileName)) {  
 for (int[] row : matrix) {  
 for (int value : row) {  
 writer.write(value + "\t");  
 }  
 writer.write("\n");  
 }  
 writer.write("####################\n");  
 }  
 }  
 public String printMatrix(int[][] matrix, String path, String outputText) {  
 String result = "";  
 try (FileWriter writer = new FileWriter(path, true)) {  
 writer.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 writer.write(outputText);  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix.length; j++) {  
 result = result + matrix[i][j] + " ";  
 writer.write(matrix[i][j] + " ");  
 }  
 result = result + "\n";  
 writer.write("\n");  
 }  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 return result;  
 }  
 public static int[][] rotateMatrix(int[][] matrix, int degrees) {  
 int size = matrix.length;  
 int[][] rotatedMatrix = new int[size][size];  
  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 if (degrees == 90) {  
 rotatedMatrix[i][j] = matrix[size - 1 - j][i];  
 } else if (degrees == 180) {  
 rotatedMatrix[i][j] = matrix[size - 1 - i][size - 1 - j];  
 } else if (degrees == 270) {  
 rotatedMatrix[i][j] = matrix[j][size - 1 - i];  
 }  
 }  
 }  
  
 return rotatedMatrix;  
 }  
  
 public static int[][] divideMatrix(int[][] matrix) {  
 int size = matrix.length;  
 int[][] dividedMatrix = new int[size][size];  
  
 for (int i = 0; i < size; i++) {  
 for (int j = 0; j < size; j++) {  
 int sum = 0;  
 boolean divideByZero = false;  
  
 for (int k = i - 1; k <= i + 1; k++) {  
 for (int l = j - 1; l <= j + 1; l++) {  
 if (k >= 0 && k < size && l >= 0 && l < size && (k != i || l != j)) {  
 sum += matrix[k][l];  
 if (matrix[k][l] == 0) {  
 divideByZero = true;  
 }  
 }  
 }  
 }  
  
 if (divideByZero) {  
 throw new ArithmeticException("Деление на ноль в позиции (" + i + ", " + j + ")");  
 } else {  
 dividedMatrix[i][j] = sum == 0 ? 0 : matrix[i][j] / sum;  
 }  
 }  
 }  
  
 return dividedMatrix;  
 }  
}

### Задание 3

Реализовать программу-переводчик.

• При запуске программы выполняется чтение словаря в следующем формате:

- слово или выражение | перевод

• Затем читается входной файл и выполняется перевод

• Перевод осуществляется по следующим правилам:

- регистр букв игнорируется

- если искомого слова нет в словаре – выводится без перевода

- если есть несколько подходящих вариантов, выбирается вариант максимальной длиной левой части

• Результат перевода выводится в консоль

Интерфейс представлен на рисунке 4 в Приложениях.

GUI

package lab3;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.Map;  
  
public class TranslatorConsoleApplication extends JFrame {  
 private String pathToInputDictionaryFile = "";  
 private String pathToInputTextFile = "";  
  
 private String exception = "";  
  
 public TranslatorConsoleApplication() {  
 super("lab\_3");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setBounds(100, 100, 1000, 1000);  
 JPanel contentPane = new JPanel();  
 setContentPane(contentPane);  
 contentPane.setLayout(null);  
  
 JLabel pathToInputDictionaryFileLabel = new JLabel("Please specify the location of the input dictionary file:");  
 pathToInputDictionaryFileLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 pathToInputDictionaryFileLabel.setBounds(27, 16, 400, 23);  
 contentPane.add(pathToInputDictionaryFileLabel);  
  
 JTextField pathToInputDictionaryFileField = new JTextField();  
 pathToInputDictionaryFileField.setBounds(27, 39, 500, 23);  
 pathToInputDictionaryFileField.setColumns(500);  
 pathToInputDictionaryFileField.setEditable(false);  
 contentPane.add(pathToInputDictionaryFileField);  
  
 JButton btnSelectInputDictionaryFile = new JButton("Select a path to input file");  
 btnSelectInputDictionaryFile.setBounds(527, 39, 100, 23);  
 contentPane.add(btnSelectInputDictionaryFile);  
  
 JFileChooser pathInputDictionaryFileChooser = new JFileChooser("./src/lab3/");  
 btnSelectInputDictionaryFile.addActionListener(e -> {  
 pathInputDictionaryFileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 int result = pathInputDictionaryFileChooser.showOpenDialog(TranslatorConsoleApplication.this);  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
 pathToInputDictionaryFileField.setText(String.*valueOf*(pathInputDictionaryFileChooser.getSelectedFile()));  
 pathToInputDictionaryFile = pathToInputDictionaryFileField.getText();  
 }  
 });  
  
 JLabel pathToInputTextFileLabel = new JLabel("Please specify the location of the input text for translation file:");  
 pathToInputTextFileLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 pathToInputTextFileLabel.setBounds(27, 78, 400, 23);  
 contentPane.add(pathToInputTextFileLabel);  
  
 JTextField pathToInputTextFileField = new JTextField();  
 pathToInputTextFileField.setBounds(27, 101, 500, 23);  
 pathToInputTextFileField.setColumns(500);  
 pathToInputTextFileField.setEditable(false);  
 contentPane.add(pathToInputTextFileField);  
  
 JButton btnSelectInputTextFile = new JButton("Select a path to input file");  
 btnSelectInputTextFile.setBounds(527, 101, 100, 23);  
 contentPane.add(btnSelectInputTextFile);  
  
 JFileChooser pathInputTextFileChooser = new JFileChooser("./src/lab3/");  
 btnSelectInputTextFile.addActionListener(e -> {  
 pathInputTextFileChooser.setFileSelectionMode(JFileChooser.*FILES\_ONLY*);  
 int result = pathInputTextFileChooser.showOpenDialog(TranslatorConsoleApplication.this);  
 if (result == JFileChooser.*APPROVE\_OPTION*) {  
 pathToInputTextFileField.setText(String.*valueOf*(pathInputTextFileChooser.getSelectedFile()));  
 pathToInputTextFile = pathToInputTextFileField.getText();  
 }  
 });  
  
 JLabel calculationResultsLabel = new JLabel("Translation results:");  
 calculationResultsLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 calculationResultsLabel.setBounds(27, 180, 400, 23);  
 contentPane.add(calculationResultsLabel);  
  
 JLabel exceptionLabel = new JLabel("Exceptions when the program is running:");  
 exceptionLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 exceptionLabel.setBounds(27, 203, 400, 23);  
 contentPane.add(exceptionLabel);  
  
 JTextField exceptionField = new JTextField();  
 exceptionField.setBounds(27, 226, 500, 23);  
 exceptionField.setColumns(500);  
 exceptionField.setEditable(false);  
 contentPane.add(exceptionField);  
  
 JLabel translatedTextLabel = new JLabel("Translated text:");  
 translatedTextLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 translatedTextLabel.setBounds(27, 250, 400, 23);  
 contentPane.add(translatedTextLabel);  
  
 JTextArea translatedTextField = new JTextArea();  
 translatedTextField.setBounds(27, 273, 600, 150);  
 translatedTextField.setColumns(500);  
 translatedTextField.setEditable(false);  
 contentPane.add(translatedTextField);  
  
 JButton btnTranslate = new JButton("Translate");  
 btnTranslate.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnTranslate) {  
 try {  
  
 Map<String, String> dictionary = *readDictionary*(pathToInputDictionaryFile);  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(pathToInputTextFile))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String translation = *translateText*(line, dictionary);  
 translatedTextField.setText(translation);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 exception = "Произошла ошибка при чтении/записи файла";  
 exceptionField.setText(exception);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 exception = "Произошла ошибка при чтении словаря";  
 exceptionField.setText(exception);  
 }  
  
 }  
 setVisible(true);  
 });  
 btnTranslate.setBounds(27, 140, 600, 40);  
 contentPane.add(btnTranslate);  
  
 JButton btnExit = new JButton("Exit");  
 btnExit.addActionListener((ActionEvent e) -> {  
 JComponent comp = (JComponent) e.getSource();  
 Window win = SwingUtilities.*getWindowAncestor*(comp);  
 win.dispose();  
 });  
 btnExit.setBounds(552, 425, 75, 23);  
 contentPane.add(btnExit);  
 }  
 private static Map<String, String> readDictionary(String filePath) throws IOException {  
 Map<String, String> dictionary = new HashMap<>();  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split("\\|");  
  
 if (parts.length == 2) {  
 String word = parts[0].trim().toLowerCase();  
 String translation = parts[1].trim();  
 dictionary.put(word, translation);  
 }  
 }  
 }  
  
 return dictionary;  
 }  
 private static String translateText(String text, Map<String, String> dictionary) {  
 String[] words = text.split("\\s+");  
 StringBuilder translation = new StringBuilder();  
  
 for (String word : words) {  
 word = word.toLowerCase();  
 String translatedWord = dictionary.get(word);  
  
 if (translatedWord != null) {  
 translation.append(translatedWord).append(" ");  
 } else {  
 translation.append(word).append(" ");  
 }  
 }  
  
 return translation.toString().trim();  
 }  
  
 public static void main(String[] args) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 TranslatorConsoleApplication frame = new TranslatorConsoleApplication();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
}

### Задание 4

Написать аннотацию с целочисленным параметром. Создать класс, содержащий публичные, защищённые и приватные методы (2–3 каждого вида), аннотировать любые из них. Вызвать из другого класса все аннотированные защищённые и приватные методы столько раз, сколько указано в параметре аннотации.

Интерфейс представлен на рисунке 5 в Приложениях.

GUI

package lab4;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.lang.reflect.InvocationTargetException;  
import java.lang.reflect.Method;  
  
public class Main extends JFrame {  
 private String exception = "";  
  
  
 public Main() {  
 super("lab\_4");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setBounds(100, 100, 700, 550);  
 JPanel contentPane = new JPanel();  
 setContentPane(contentPane);  
 contentPane.setLayout(null);  
  
 JLabel ResultsLabel = new JLabel("Results:");  
 ResultsLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 ResultsLabel.setBounds(27, 40, 400, 23);  
 contentPane.add(ResultsLabel);  
  
 JLabel exceptionLabel = new JLabel("Exceptions when the program is running:");  
 exceptionLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 exceptionLabel.setBounds(27, 63, 400, 23);  
 contentPane.add(exceptionLabel);  
  
 JTextField exceptionField = new JTextField();  
 exceptionField.setBounds(27, 86, 500, 23);  
 exceptionField.setColumns(500);  
 exceptionField.setEditable(false);  
 contentPane.add(exceptionField);  
  
 JLabel translatedTextLabel = new JLabel("Translated text:");  
 translatedTextLabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 translatedTextLabel.setBounds(27, 110, 400, 23);  
 contentPane.add(translatedTextLabel);  
  
 JTextArea methodTextField = new JTextArea();  
 methodTextField.setBounds(27, 133, 600, 300);  
 methodTextField.setColumns(500);  
 methodTextField.setEditable(false);  
 contentPane.add(methodTextField);  
  
 JButton btnGetMethod = new JButton("Get methods");  
 btnGetMethod.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnGetMethod) {  
 for (Method method : Methods.class.getDeclaredMethods()) {  
 if (method.isAnnotationPresent(IntegerAnnotation.class)) {  
 IntegerAnnotation annotation = method.getAnnotation(IntegerAnnotation.class);  
 method.setAccessible(true);  
 for (int j = 0; j < annotation.count(); j++) {  
 try {  
 if (method.getModifiers() == 2 || method.getModifiers() == 4) {  
 method.invoke(new Methods(methodTextField));  
 }  
 } catch (IllegalAccessException | InvocationTargetException e) {  
 exceptionField.setText("IllegalAccessException | InvocationTargetException");  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
 setVisible(true);  
 });  
 btnGetMethod.setBounds(27, 20, 600, 40);  
 contentPane.add(btnGetMethod);  
  
 JButton btnExit = new JButton("Exit");  
 btnExit.addActionListener((ActionEvent e) -> {  
 JComponent comp = (JComponent) e.getSource();  
 Window win = SwingUtilities.*getWindowAncestor*(comp);  
 win.dispose();  
 });  
 btnExit.setBounds(552, 450, 75, 23);  
 contentPane.add(btnExit);  
 }  
 public static void main(String[] args) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 Main frame = new Main();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
}

IntegerAnnotation.java

package lab4;  
  
import java.lang.annotation.ElementType;  
import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
import java.lang.annotation.Target;  
  
@Target(ElementType.*METHOD*)  
@Retention(RetentionPolicy.*RUNTIME*)  
public @interface IntegerAnnotation {  
 int count() default 0;  
}

Methods.java

package lab4;  
  
import javax.swing.\*;  
  
public class Methods {  
 private final JTextArea jTextArea;  
  
 public Methods(JTextArea jTextArea) {  
 this.jTextArea = jTextArea;  
 }  
  
 @IntegerAnnotation(count = 1)  
 public void getFirstPublicMethod(){  
 System.*out*.println("I first public method");  
 }  
  
 @IntegerAnnotation(count = 2)  
 public void getSecondPublicMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I second public method");  
 System.*out*.println("I second public method");  
 }  
  
 @IntegerAnnotation(count = 3)  
 public void getThirdPublicMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I third public method");  
 System.*out*.println("I third public method");  
 }  
  
 @IntegerAnnotation(count = 1)  
 private void getFirstPrivateMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I first private method");  
 System.*out*.println("I first private method");  
 }  
  
 @IntegerAnnotation(count = 2)  
 private void getSecondPrivateMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I second private method");  
 System.*out*.println("I second private method");  
 }  
  
 @IntegerAnnotation(count = 3)  
 private void getThirdPrivateMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I third private method");  
 System.*out*.println("I third private method");  
 }  
  
 @IntegerAnnotation(count = 1)  
 protected void getFirstProtectedMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I first protected method");  
 System.*out*.println("I first protected method");  
 }  
  
 @IntegerAnnotation(count = 2)  
 protected void getSecondProtectedMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I second protected method");  
 System.*out*.println("I second protected method");  
 }  
  
 @IntegerAnnotation(count = 3)  
 protected void getThirdProtectedMethod(){  
 jTextArea.setText(jTextArea.getText() + '\n' + "I third protected method");  
 System.*out*.println("I third protected method");  
 }  
  
}

### Задание 5

С использованием только Stream API реализовать следующие методы:

● метод, возвращающий среднее значение списка целых чисел;

● метод, приводящий все строки в списке в верхний регистр и добавляющий к ним префикс «\_new\_»;

● метод, возвращающий список квадратов всех встречающихся только один раз элементов списка;

● метод, принимающий на вход коллекцию строк и возвращающий все строки, начинающиеся с заданной буквы, отсортированные по алфавиту;

● метод, принимающий на вход коллекцию и возвращающий её последний элемент или кидающий исключение, если коллекция пуста;

● метод, принимающий на вход массив целых чисел, возвращающий сумму чётных чисел или 0, если чётных чисел нет;

● метод, преобразовывающий все строки в списке в Map, где первый символ – ключ, оставшиеся – значение;

Интерфейс представлен на рисунке 6 в Приложениях.

GUI

Main.java

package lab5;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.util.List;  
import java.util.\*;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class Main extends JFrame{  
 List<Integer> list1;  
 List<String> list2;  
 List<Integer> list3;  
 List<String> list4;  
 List<Integer> list5;  
 int[] a6;  
 List<String> list7;  
  
 public Main() {  
 super("lab\_5");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 setBounds(100, 100, 700, 600);  
 JPanel contentPane = new JPanel();  
 setContentPane(contentPane);  
 contentPane.setLayout(null);  
  
 //Task 1  
  
 JLabel task1Label = new JLabel("Task 1:");  
 task1Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task1Label.setBounds(27, 60, 250, 23);  
 contentPane.add(task1Label);  
  
 JLabel task1InputDatalabel = new JLabel("Input data:");  
 task1InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task1InputDatalabel.setBounds(27, 83, 250, 23);  
 contentPane.add(task1InputDatalabel);  
  
 JTextField task1InputDataField = new JTextField();  
 task1InputDataField.setBounds(27, 106, 250, 23);  
 task1InputDataField.setColumns(500);  
 task1InputDataField.setEditable(false);  
 contentPane.add(task1InputDataField);  
  
 JLabel task1OutputDatalabel = new JLabel("Output data:");  
 task1OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task1OutputDatalabel.setBounds(27, 129, 250, 23);  
 contentPane.add(task1OutputDatalabel);  
  
 JTextField task1OutputDataField = new JTextField();  
 task1OutputDataField.setBounds(27, 152, 250, 23);  
 task1OutputDataField.setColumns(500);  
 task1OutputDataField.setEditable(false);  
 contentPane.add(task1OutputDataField);  
  
 //Task 2  
 JLabel task2Label = new JLabel("Task 2:");  
 task2Label.setBounds(327, 60, 250, 23);  
 task2Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 contentPane.add(task2Label);  
  
 JLabel task2InputDatalabel = new JLabel("Input data:");  
 task2InputDatalabel.setBounds(327, 83, 250, 23);  
 task2InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 contentPane.add(task2InputDatalabel);  
  
 JTextField task2InputDataField = new JTextField();  
 task2InputDataField.setBounds(327, 106, 250, 23);  
 task2InputDataField.setColumns(500);  
 task2InputDataField.setEditable(false);  
 contentPane.add(task2InputDataField);  
  
 JLabel task2OutputDatalabel = new JLabel("Output data:");  
 task2OutputDatalabel.setBounds(327, 129, 250, 23);  
 task2OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 contentPane.add(task2OutputDatalabel);  
  
 JTextField task2OutputDataField = new JTextField();  
 task2OutputDataField.setBounds(327, 152, 250, 23);  
 task2OutputDataField.setColumns(500);  
 task2OutputDataField.setEditable(false);  
 contentPane.add(task2OutputDataField);  
  
  
 //Task 3  
 JLabel task3Label = new JLabel("Task 3:");  
 task3Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task3Label.setBounds(27, 176, 250, 23);  
 contentPane.add(task3Label);  
  
 JLabel task3InputDatalabel = new JLabel("Input data:");  
 task3InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task3InputDatalabel.setBounds(27, 199, 250, 23);  
 contentPane.add(task3InputDatalabel);  
  
 JTextField task3InputDataField = new JTextField();  
 task3InputDataField.setBounds(27, 222, 250, 23);  
 task3InputDataField.setColumns(500);  
 task3InputDataField.setEditable(false);  
 contentPane.add(task3InputDataField);  
  
 JLabel task3OutputDatalabel = new JLabel("Output data:");  
 task3OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task3OutputDatalabel.setBounds(27, 245, 250, 23);  
 contentPane.add(task3OutputDatalabel);  
  
 JTextField task3OutputDataField = new JTextField();  
 task3OutputDataField.setBounds(27, 268, 250, 23);  
 task3OutputDataField.setColumns(500);  
 task3OutputDataField.setEditable(false);  
 contentPane.add(task3OutputDataField);  
  
 //Task 4  
 JLabel task4Label = new JLabel("Task 4:");  
 task4Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task4Label.setBounds(327, 176, 250, 23);  
 contentPane.add(task4Label);  
  
 JLabel task4InputDatalabel = new JLabel("Input data:");  
 task4InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task4InputDatalabel.setBounds(327, 199, 250, 23);  
 contentPane.add(task4InputDatalabel);  
  
 JTextField task4InputDataField = new JTextField();  
 task4InputDataField.setBounds(327, 222, 250, 23);  
 task4InputDataField.setColumns(500);  
 task4InputDataField.setEditable(false);  
 contentPane.add(task4InputDataField);  
  
 JLabel task4OutputDatalabel = new JLabel("Output data:");  
 task4OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task4OutputDatalabel.setBounds(327, 245, 250, 23);  
 contentPane.add(task4OutputDatalabel);  
  
 JTextField task4OutputDataField = new JTextField();  
 task4OutputDataField.setBounds(327, 268, 250, 23);  
 task4OutputDataField.setColumns(500);  
 task4OutputDataField.setEditable(false);  
 contentPane.add(task4OutputDataField);  
  
 //Task 5  
 JLabel task5Label = new JLabel("Task 5:");  
 task5Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task5Label.setBounds(27, 291, 250, 23);  
 contentPane.add(task5Label);  
  
 JLabel task5InputDatalabel = new JLabel("Input data:");  
 task5InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task5InputDatalabel.setBounds(27, 314, 250, 23);  
 contentPane.add(task5InputDatalabel);  
  
 JTextField task5InputDataField = new JTextField();  
 task5InputDataField.setBounds(27, 337, 250, 23);  
 task5InputDataField.setColumns(500);  
 task5InputDataField.setEditable(false);  
 contentPane.add(task5InputDataField);  
  
 JLabel task5OutputDatalabel = new JLabel("Output data:");  
 task5OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task5OutputDatalabel.setBounds(27, 360, 250, 23);  
 contentPane.add(task5OutputDatalabel);  
  
 JTextField task5OutputDataField = new JTextField();  
 task5OutputDataField.setBounds(27, 383, 250, 23);  
 task5OutputDataField.setColumns(500);  
 task5OutputDataField.setEditable(false);  
 contentPane.add(task5OutputDataField);  
  
 //Task 6  
 JLabel task6Label = new JLabel("Task 6:");  
 task6Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task6Label.setBounds(327, 291, 250, 23);  
 contentPane.add(task6Label);  
  
 JLabel task6InputDatalabel = new JLabel("Input data:");  
 task6InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task6InputDatalabel.setBounds(327, 314, 250, 23);  
 contentPane.add(task6InputDatalabel);  
  
 JTextField task6InputDataField = new JTextField();  
 task6InputDataField.setBounds(327, 337, 250, 23);  
 task6InputDataField.setColumns(500);  
 task6InputDataField.setEditable(false);  
 contentPane.add(task6InputDataField);  
  
 JLabel task6OutputDatalabel = new JLabel("Output data:");  
 task6OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task6OutputDatalabel.setBounds(327, 360, 250, 23);  
 contentPane.add(task6OutputDatalabel);  
  
 JTextField task6OutputDataField = new JTextField();  
 task6OutputDataField.setBounds(327, 383, 250, 23);  
 task6OutputDataField.setColumns(500);  
 task6OutputDataField.setEditable(false);  
 contentPane.add(task6OutputDataField);  
  
  
 //Task 7  
 JLabel task7Label = new JLabel("Task 7:");  
 task7Label.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task7Label.setBounds(27, 406, 250, 23);  
 contentPane.add(task7Label);  
  
 JLabel task7InputDatalabel = new JLabel("Input data:");  
 task7InputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task7InputDatalabel.setBounds(27, 429, 250, 23);  
 contentPane.add(task7InputDatalabel);  
  
 JTextField task7InputDataField = new JTextField();  
 task7InputDataField.setBounds(27, 452, 250, 23);  
 task7InputDataField.setColumns(500);  
 task7InputDataField.setEditable(false);  
 contentPane.add(task7InputDataField);  
  
 JLabel task7OutputDatalabel = new JLabel("Output data:");  
 task7OutputDatalabel.setFont(new Font("Tahoma", Font.*PLAIN*, 14));  
 task7OutputDatalabel.setBounds(27, 475, 250, 23);  
 contentPane.add(task7OutputDatalabel);  
  
 JTextField task7OutputDataField = new JTextField();  
 task7OutputDataField.setBounds(27, 498, 250, 23);  
 task7OutputDataField.setColumns(500);  
 task7OutputDataField.setEditable(false);  
 contentPane.add(task7OutputDataField);  
  
 JButton btnGetInputData = new JButton("Get input data");  
 btnGetInputData.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnGetInputData) {  
 list1 = *getIntegerList*();  
 list2 = *getStringList*();  
 list3 = *getIntegerList*();  
 list4 = *getStringList*();  
 list5 = *getIntegerList*();  
 a6 = *getIntegerArray*();  
 list7 = *getStringList*();  
  
 task1InputDataField.setText(list1.toString());  
 task2InputDataField.setText(list2.toString());  
 task3InputDataField.setText(list3.toString());  
 task4InputDataField.setText(list4.toString());  
 task5InputDataField.setText(list5.toString());  
 task6InputDataField.setText(Arrays.*toString*(a6));  
 task7InputDataField.setText(list7.toString());  
 }  
 setVisible(true);  
 });  
  
 btnGetInputData.setBounds(27, 20, 600, 40);  
 contentPane.add(btnGetInputData);  
  
 JButton btnGetResult = new JButton("Get result");  
 btnGetResult.addActionListener(arg0 -> {  
 if(arg0.getSource() == btnGetResult) {  
 task1OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*calculateAverage*(list1)));  
 task2OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*transformAndAddPrefix*(list2)));  
 task3OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*uniqueSquares*(list3)));  
 task4OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*filterAndSortByLetter*(list4, 'a')));  
 try {  
 task5OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*getLastElementOrThrowException*(list5)));  
 } catch (NoSuchElementException e) {  
 task5OutputDataField.setText("Exception");  
 }  
  
 task6OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*sumEvenNumbers*(a6)));  
  
 try {  
 task7OutputDataField.setText(String.*valueOf*(StreamMethodsExample.*convertToMap*(list7)));  
 } catch (IllegalStateException e) {  
 task7OutputDataField.setText("Duplicate key v (attempted merging values");  
 }  
 }  
 setVisible(true);  
 });  
 btnGetResult.setBounds(327, 406, 250, 115);  
 contentPane.add(btnGetResult);  
  
 JButton btnExit = new JButton("Exit");  
 btnExit.addActionListener((ActionEvent e) -> {  
 JComponent comp = (JComponent) e.getSource();  
 Window win = SwingUtilities.*getWindowAncestor*(comp);  
 win.dispose();  
 });  
 btnExit.setBounds(502, 525, 75, 23);  
 contentPane.add(btnExit);  
 }  
 public static void main(String[] args) {  
 EventQueue.*invokeLater*(() -> {  
 try {  
 Main frame = new Main();  
 frame.setVisible(true);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 });  
 }  
  
 private static List<Integer> getIntegerList() {  
 int n = 5;  
 List<Integer> result = new ArrayList<>(n);  
 for (int i = 0; i < n; i++) {  
 result.add((int) ((Math.*random*()\*100)/10));  
 }  
 return result;  
 }  
  
 private static int[] getIntegerArray() {  
 int n = 5;  
 int[] result = new int[n];  
 for (int i = 0; i < n; i++) {  
 result[i] = (int) ((Math.*random*()\*100)/10);  
 }  
 return result;  
 }  
  
 private static List<String> getStringList() {  
 return Stream.*generate*(() -> {  
 int length = new Random().nextInt(10) + 1;  
 StringBuilder sb = new StringBuilder();  
 new Random().ints(length, 'a', 'z' + 1)  
 .forEach(randomChar -> sb.append((char) randomChar));  
 return sb.toString();  
 })  
 .limit(5)  
 .collect(Collectors.*toList*());  
 }  
}

StreamMethodsExample.java

package lab5;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
public class StreamMethodsExample {  
  
 public static double calculateAverage(List<Integer> numbers) {  
 return numbers.stream()  
 .mapToDouble(Integer::doubleValue)  
 .average()  
 .orElse(0.0);  
 }  
  
 public static List<String> transformAndAddPrefix(List<String> strings) {  
 return strings.stream()  
 .map(s -> "\_new\_" + s.toUpperCase())  
 .collect(Collectors.*toList*());  
 }  
  
 public static List<Integer> uniqueSquares(List<Integer> numbers) {  
 Map<Integer, Long> frequencyMap = numbers.stream()  
 .collect(Collectors.*groupingBy*(i -> i, Collectors.*counting*()));  
 return numbers.stream()  
 .filter(n -> frequencyMap.get(n) == 1)  
 .map(n -> n \* n)  
 .collect(Collectors.*toList*());  
 }  
  
 public static List<String> filterAndSortByLetter(List<String> strings, char letter) {  
 return strings.stream()  
 .filter(s -> s.startsWith(String.*valueOf*(letter)))  
 .sorted()  
 .collect(Collectors.*toList*());  
 }  
  
 public static <T> T getLastElementOrThrowException(Collection<T> collection) {  
 return collection.stream()  
 .reduce((first, second) -> second)  
 .orElseThrow(() -> new NoSuchElementException("Collection is empty"));  
 }  
  
 public static int sumEvenNumbers(int[] numbers) {  
 return Arrays.*stream*(numbers)  
 .filter(n -> n % 2 == 0)  
 .sum();  
 }  
  
 public static Map<Character, String> convertToMap(List<String> strings) {  
 return strings.stream()  
 .collect(Collectors.*toMap*(s -> s.charAt(0), s -> s.substring(1), (existing, replacement) -> existing));  
 }  
  
 public static void main(String[] args) {  
 List<Integer> numbers = Arrays.*asList*(1, 2, 3, 4, 5);  
 List<String> strings = Arrays.*asList*("apple", "banana", "cherry");  
  
 // Примеры использования методов:  
 double average = *calculateAverage*(numbers);  
 System.*out*.println("Average: " + average);  
  
 List<String> transformedStrings = *transformAndAddPrefix*(strings);  
 System.*out*.println("Transformed Strings: " + transformedStrings);  
  
 List<Integer> uniqueSquares = *uniqueSquares*(numbers);  
 System.*out*.println("Unique Squares: " + uniqueSquares);  
  
 List<String> filteredAndSortedStrings = *filterAndSortByLetter*(strings, 'b');  
 System.*out*.println("Filtered and Sorted Strings: " + filteredAndSortedStrings);  
  
 List<String> emptyList = Collections.*emptyList*();  
 try {  
 String lastElement = *getLastElementOrThrowException*(emptyList);  
 System.*out*.println("Last Element: " + lastElement);  
 } catch (NoSuchElementException e) {  
 System.*out*.println("Exception: " + e.getMessage());  
 }  
  
 int[] integerArray = {1, 2, 3, 4, 5};  
 int sumOfEvenNumbers = *sumEvenNumbers*(integerArray);  
 System.*out*.println("Sum of Even Numbers: " + sumOfEvenNumbers);  
  
 Map<Character, String> stringMap = *convertToMap*(strings);  
 System.*out*.println("String Map: " + stringMap);  
 }  
}

# 

# Приложения

Изображение выглядит как снимок экрана, текст, программное обеспечение, дизайн

Автоматически созданное описание

Рис. 1

Изображение выглядит как снимок экрана, текст, программное обеспечение, компьютер

Автоматически созданное описание

Рис. 2

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 3

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 4

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис. 5

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рис. 6