Чернівецький національний університет імені Юрія Федьковича

Інститут фізико-технічних та комп’ютерних наук

Кафедра Математичних проблем управління і кібернетики

Лабораторна робота № 6

“Тема: присвячена побудові додатків з використанням

графічного інтерфейсу бібліотек java.awt та javax.swing. .”

з дисципліни “ Java”

Варіант № 18

Виконав:

ст. гр. 241 Фрасинюк О. Б.

Прийняв:

Викладач Лазорик В.В.

Чернівці – 2024

https://github.com/VLazorykOOP/jlab6-Oleh-Frasyniuk

**Завдання 1**

Зобразити у вікні програми відрізок, що обертається в площині фрейму навколо точки, що рухається по відрізку.

**Код:**import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.Line2D;

public class RotatingSegment extends JPanel {

    private double angle = 0;

    private double t = 0; // Параметр для руху точки по відрізку

    // Оновлення стану при кожному кроці анімації

    public void update() {

        t += 0.01;

        if (t > 1) {

            t = 0; // Повернення точки на початок відрізка

        }

        angle += Math.PI / 50; // Кут обертання

        repaint();

    }

    // відрізок, що обертається

    @Override

    protected void paintComponent(Graphics g) {

        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g;

        g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

        //Розміри вікна

        int width = getWidth();

        int height = getHeight();

        //Центр координат

        int centerX = width / 2;

        int centerY = height / 2;

        //Початкові координати відрізка

        double startX = -100;

        double startY = 0;

        double endX = 100;

        double endY = 0;

        //Кут обертання

        double cosAngle = Math.cos(angle);

        double sinAngle = Math.sin(angle);

        //Обертання відрізка

        double rotatedEndX = cosAngle \* endX - sinAngle \* endY;

        double rotatedEndY = sinAngle \* endX + cosAngle \* endY;

        //Рухаюча точка

        double movingPointX = startX + t \* (endX - startX);

        double movingPointY = startY + t \* (endY - startY);

        //Малюємо відрізок

        g2d.setColor(Color.BLUE);

        g2d.draw(new Line2D.Double(centerX + movingPointX, centerY + movingPointY, centerX + rotatedEndX, centerY + rotatedEndY));

        //Малюємо точку, що рухається

        g2d.setColor(Color.RED);

        g2d.fillOval(centerX + (int) movingPointX - 3, centerY + (int) movingPointY - 3, 6, 6);

    }

    public static void main(String[] args) {

        JFrame frame = new JFrame("Rotating Segment");

        RotatingSegment panel = new RotatingSegment();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(600, 600);

        frame.add(panel);

        frame.setVisible(true);

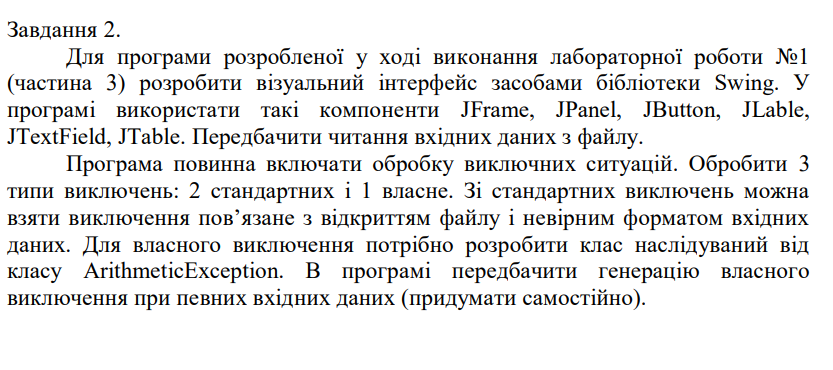
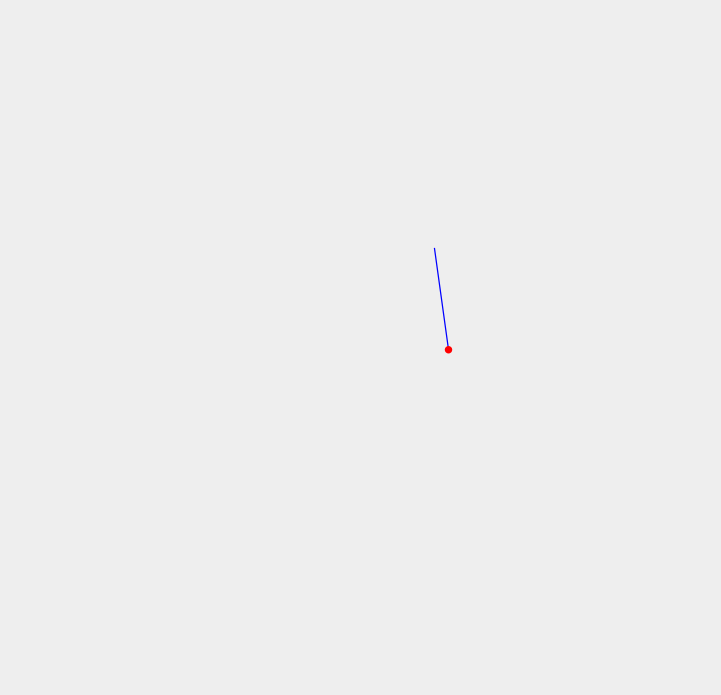
        //Анімація

        new Timer(20, e -> panel.update()).start();

    }

}

**Результат:**



**Код:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.util.\*;

import javax.swing.table.DefaultTableModel;

// Власне виключення

// Власне виключення

class MatrixZeroException extends ArithmeticException {

    private int rowIndex; // Додано для зберігання індексу рядка

    // Конструктор, що приймає повідомлення та індекс рядка

    public MatrixZeroException(String message, int rowIndex) {

        super(message);

        this.rowIndex = rowIndex;

    }

    // Метод для отримання індексу рядка

    public int getRowIndex() {

        return rowIndex;

    }

}

public class MatrixApp {

    private JFrame frame;

    private JTextField matrixSizeField;

    private JTable matrixTable;

    private JButton loadButton, calculateButton;

    private JLabel resultLabel;

    public MatrixApp() {

        frame = new JFrame("Lab 1.3: Matrix Unique Check");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(600, 500);

        frame.setLayout(new BorderLayout());

        // Панель для вводу розміру матриці

        JPanel inputPanel = new JPanel();

        inputPanel.setLayout(new FlowLayout());

        inputPanel.add(new JLabel("Enter matrix size (n ≤ 20):"));

        matrixSizeField = new JTextField(5);

        inputPanel.add(matrixSizeField);

        loadButton = new JButton("Load Matrix");

        inputPanel.add(loadButton);

        calculateButton = new JButton("Calculate");

        inputPanel.add(calculateButton);

        frame.add(inputPanel, BorderLayout.NORTH);

        // Панель для таблиці матриці

        matrixTable = new JTable();

        JScrollPane tableScroll = new JScrollPane(matrixTable);

        frame.add(tableScroll, BorderLayout.CENTER);

        // Панель для результатів

        JPanel resultPanel = new JPanel();

        resultLabel = new JLabel("Result: ");

        resultPanel.add(resultLabel);

        frame.add(resultPanel, BorderLayout.SOUTH);

        // Дії при натисканні кнопок

        loadButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                loadMatrixFromFile();

            }

        });

        calculateButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                calculateUniqueRows();

            }

        });

        frame.setVisible(true);

    }

    // Метод для завантаження матриці з файлу

    private void loadMatrixFromFile() {

        try {

            int n = Integer.parseInt(matrixSizeField.getText());

            if (n <= 0 || n > 20) {

                throw new IllegalArgumentException("Matrix size must be between 1 and 20.");

            }

            // Читання матриці з файлу

            JFileChooser fileChooser = new JFileChooser();

            int result = fileChooser.showOpenDialog(frame);

            if (result != JFileChooser.APPROVE\_OPTION) return;

            File file = fileChooser.getSelectedFile();

            Scanner scanner = new Scanner(file);

            int[][] matrix = new int[n][n];

            for (int i = 0; i < n; i++) {

                for (int j = 0; j < n; j++) {

                    if (!scanner.hasNextInt()) {

                        throw new InputMismatchException("Invalid format in the file.");

                    }

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

                }

            }

            scanner.close();

            // Оновлення таблиці

            DefaultTableModel model = new DefaultTableModel(n, n);

            matrixTable.setModel(model);

            for (int i = 0; i < n; i++) {

                for (int j = 0; j < n; j++) {

                    matrixTable.setValueAt(matrix[i][j], i, j);

                }

            }

        } catch (FileNotFoundException ex) {

            showError("File not found.");

        } catch (InputMismatchException ex) {

            showError("Invalid input format.");

        } catch (IllegalArgumentException ex) {

            showError(ex.getMessage());

        } catch (Exception ex) {

            showError("An unexpected error occurred: " + ex.getMessage());

        }

    }

    // Метод для перевірки унікальності рядків та викидання власного виключення

    private void calculateUniqueRows() {

        try {

            int n = Integer.parseInt(matrixSizeField.getText());

            int[][] matrix = new int[n][n];

            for (int i = 0; i < n; i++) {

                for (int j = 0; j < n; j++) {

                    matrix[i][j] = (int) matrixTable.getValueAt(i, j);

                }

            }

            boolean[] result = new boolean[n];

            for (int i = 0; i < n; i++) {

                HashSet<Integer> set = new HashSet<>();

                for (int j = 0; j < n; j++) {

                    if (matrix[i][j] == 0) {

                        throw new MatrixZeroException("Matrix contains zero in row " + (i + 1), i);

                    }

                    set.add(matrix[i][j]);

                }

                result[i] = set.size() == n;

            }

            // Показ результатів

            StringBuilder resultText = new StringBuilder("Unique rows: ");

            for (int i = 0; i < n; i++) {

                resultText.append("Y[").append(i).append("] = ").append(result[i]).append(" ");

            }

            resultLabel.setText(resultText.toString());

        } catch (MatrixZeroException ex) {

            showError(ex.getMessage());

        } catch (NumberFormatException ex) {

            showError("Invalid matrix size.");

        } catch (Exception ex) {

            showError("An error occurred: " + ex.getMessage());

        }

    }

    // Метод для відображення помилок

    private void showError(String message) {

        JOptionPane.showMessageDialog(frame, message, "Error", JOptionPane.ERROR\_MESSAGE);

    }

    public static void main(String[] args) {

        new MatrixApp();

    }

}

**Результат:**

