МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
**КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

# Отчёт
# по лабораторной работе №5

Выполнил:
студент группы ПО-9
Качаловский Данил Сергеевич

Проверил:
Крощенко А. А.

Брест 2024

**Цель работы**: освоить возможности языка программирования Java в построении графических приложений

**Вариант 7**

**Задание 1**

Изобразить в окне приложения (апплета) отрезок, вращающийся в плоскости фрейма вокруг точки, движущейся по отрезку.

**Код программы**

```
package Lab7_1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Lab7_1 extends JPanel implements ActionListener {

    private static final int FRAME_WIDTH = 600;
    private static final int FRAME_HEIGHT = 400;
    private static final int POINT_RADIUS = 5;
    private static final int LINE_LENGTH = 200;
    private static final int LINE_Y = FRAME_HEIGHT / 2;

    private Timer timer;
    private boolean moveRight = true;
    private double angle;
    private double centerX = FRAME_WIDTH/2;
    private double centerY = FRAME_HEIGHT/2;
    private double pointX=200;
    private double pointX2=300;

    public Lab7_1() {
        timer = new Timer(16, this);
        timer.start();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        g.setColor(Color.RED);
        int pointY = (int) centerY - POINT_RADIUS / 2;
        g.fillOval((int) pointX - POINT_RADIUS / 2, pointY, POINT_RADIUS,
POINT_RADIUS);

        double tempPointX = pointX + LINE_LENGTH / 2 * Math.cos(angle);
        double tempPointY = centerY + LINE_LENGTH / 2 * Math.sin(angle);

        double lineStartX = tempPointX + LINE_LENGTH / 4 *
Math.cos(angle+Math.PI/2);
        double lineStartY = tempPointY + LINE_LENGTH / 4 *
Math.sin(angle+Math.PI/2);
        double lineEndX = tempPointX - LINE_LENGTH / 4 *
Math.cos(angle+Math.PI/2);
        double lineEndY = tempPointY - LINE_LENGTH / 4 *
```

```
Math.sin(angle+Math.PI/2);

        g.setColor(Color.BLACK);
        g.drawLine((int) lineStartX, (int) lineStartY, (int) lineEndX, (int)
lineEndY);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (moveRight) {
            pointX += 1;
            if (pointX >= centerX + LINE_LENGTH/2) {
                moveRight = false;
            }
        } else {
            pointX -= 1;
            if (pointX <= centerX - LINE_LENGTH/2) {
                moveRight = true;
            }
        }
        angle += 0.05;
        repaint();
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(FRAME_WIDTH, FRAME_HEIGHT);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Moving Point Applet");
        JPanel applet = new Lab7_1();
        frame.add(applet);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }
}
```

**Задание 2**

Нарисовать снежинку Коха.

**Код программы**

```
package Lab7_1;

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

public class Lab7_2 extends JPanel {

    private static final int FRAME_WIDTH = 600;
    private static final int FRAME_HEIGHT = 600;

    private void drawKochSnowflake(Graphics g, int x1, int y1, int x5, int
y5, int level) {
```

```
        if (level == 0) {
            g.drawLine(x1, y1, x5, y5);
        } else {
            int deltaX = x5 - x1;
            int deltaY = y5 - y1;

            int x2 = x1 + deltaX / 3;
            int y2 = y1 + deltaY / 3;

            int x3 = (int) (0.5 * (x1 + x5) + Math.sqrt(3) * (y1 - y5) / 6);
            int y3 = (int) (0.5 * (y1 + y5) + Math.sqrt(3) * (x5 - x1) / 6);

            int x4 = x1 + 2 * deltaX / 3;
            int y4 = y1 + 2 * deltaY / 3;

            drawKochSnowflake(g, x1, y1, x2, y2, level - 1);
            drawKochSnowflake(g, x2, y2, x3, y3, level - 1);
            drawKochSnowflake(g, x3, y3, x4, y4, level - 1);
            drawKochSnowflake(g, x4, y4, x5, y5, level - 1);
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.BLACK);

        int x1 = FRAME_WIDTH / 2;
        int y1 = 100;
        int x2 = x1 + (int) (FRAME_HEIGHT / 2 * Math.cos(Math.PI / 3));
        int y2 = y1 + (int) (FRAME_HEIGHT / 2 * Math.sin(Math.PI / 3));
        int x3 = x1 - (int) (FRAME_HEIGHT / 2 * Math.cos(Math.PI / 3));
        int y3 = y1 + (int) (FRAME_HEIGHT / 2 * Math.sin(Math.PI / 3));
        drawKochSnowflake(g, x2, y2, x1, y1, 3);
        drawKochSnowflake(g, x3, y3, x2, y2, 3);
        drawKochSnowflake(g, x1, y1, x3, y3, 3);

    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Koch Snowflake");
        JPanel panel = new Lab7_2();
        frame.add(panel);
        frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```