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

Отчёт
по лабораторной работе №7

Выполнил:
студент группы ПО 9
Ступак Д.Р

Проверил:
Крощенко А. А.

Брест 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования

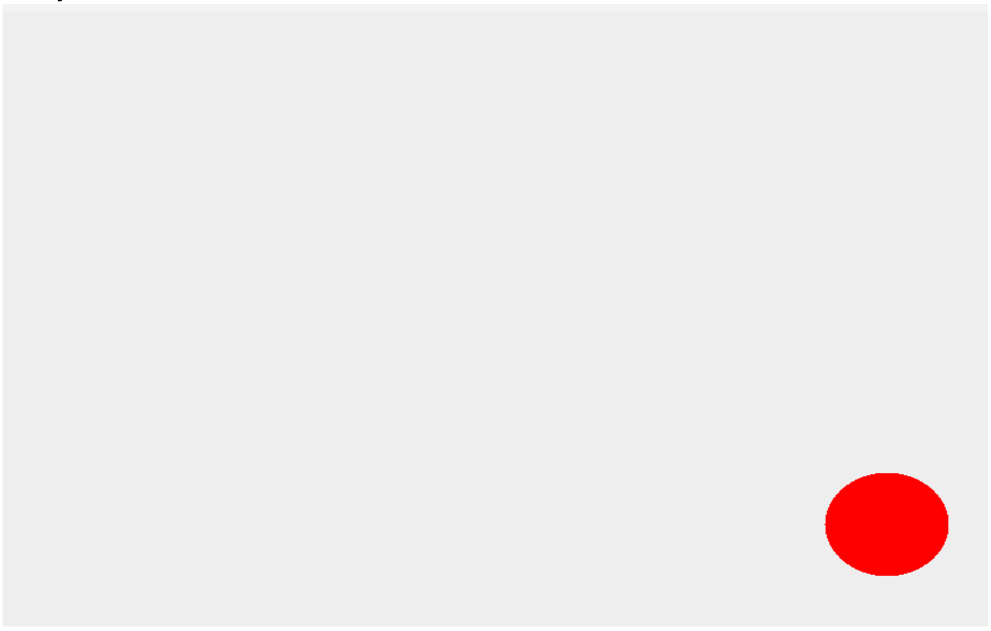
Вариант 7

Задание 1

Задать движение окружности по апплету так, чтобы при касании границы окружность отражалась от нее.

Код программы (файл City.java)

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

public class BouncingCircle extends JPanel implements ActionListener {

    private static final int WIDTH = 800;
    private static final int HEIGHT = 600;
    private static final int CIRCLE_RADIUS = 50;
    private static final int CIRCLE_SPEED = 5;

    private int circleX;
    private int circleY;
    private int circleVelocityX;
    private int circleVelocityY;

    public BouncingCircle() {
        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        circleX = WIDTH / 2;
        circleY = HEIGHT / 2;
        circleVelocityX = CIRCLE_SPEED;
        circleVelocityY = CIRCLE_SPEED;

        Timer timer = new Timer(10, this);
        timer.start();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setColor(Color.RED);
        g.fillOval(circleX  -  CIRCLE_RADIUS,  circleY  -  CIRCLE_RADIUS,  CIRCLE_RADIUS  *  2,
CIRCLE_RADIUS * 2);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        circleX += circleVelocityX;
        circleY += circleVelocityY;

        if (circleX - CIRCLE_RADIUS < 0 || circleX + CIRCLE_RADIUS > WIDTH) {
            circleVelocityX = -circleVelocityX;
        }
        if (circleY - CIRCLE_RADIUS < 0 || circleY + CIRCLE_RADIUS > HEIGHT) {
            circleVelocityY = -circleVelocityY;
        }
        repaint();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setResizable(false);
            frame.add(new BouncingCircle());
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        });
    }
```



Задание 1

Реализовать построение заданного типа фрактала по варианту

Везде, где это необходимо, предусмотреть ввод параметров, влияющих на внешний вид фрактала

Код программы:

```
import javax.swing.JFrame;
import javax.swing.JPanel;

import java.awt.Polygon;
import java.awt.Graphics;

public class KochKurve extends JFrame {

    private static Polygon returnPolygon = new Polygon();

    public KochKurve(int steps) {
        Polygon polygon = new Polygon();

        initializePolygon(polygon);
        subdivide(polygon, steps);
        paintPolygon(returnPolygon);
    }

    private Polygon initializePolygon(Polygon polygon) {
        polygon.addPoint(250, 50);
        polygon.addPoint(400, 350);
        polygon.addPoint(100, 350);
        return polygon;
    }

    public static Polygon subdivide(Polygon polygon, int steps) {
        while (steps > 0) {
            polygon = subdivide(polygon);
            steps--;
        }
        returnPolygon = polygon;
        return returnPolygon;
    }
    static Polygon subdivide(Polygon polygon) {
        Polygon localPolygon = new Polygon();
        int numPoints = polygon.npoints;

        for(int i = 0; i < numPoints; i++) {
            int currentX = polygon.xpoints[i];
            int currentY = polygon.ypoints[i];

            int nextX = polygon.xpoints[(i + 1) % numPoints];
            int nextY = polygon.ypoints[(i + 1) % numPoints];

            Vector2 pointA = new Vector2(currentX, currentY);
            Vector2 pointE = new Vector2(nextX, nextY);

            Vector2 pointB = pointA.scale(2.0 / 3.0).add(pointE.scale(1.0 / 3.0));

            Vector2 pointM = pointA.linearInterpolation(pointE, 0.5);
            Vector2 vectorAE = pointE.subtract(pointA);
            Vector2 vectorN = new Vector2(vectorAE.y, -vectorAE.x).normalized();
            double lengthD = Math.sqrt(1.0 / 12.0)*vectorAE.length();
            Vector2 pointC = pointM.add(vectorN.scale(lengthD));

            Vector2 pointD = pointA.scale(1.0 / 3.0).add(pointE.scale(2.0 / 3.0));

            localPolygon.addPoint((int) pointA.x, (int) pointA.y);
            localPolygon.addPoint((int) pointB.x, (int) pointB.y);
            localPolygon.addPoint((int) pointC.x, (int) pointC.y);
            localPolygon.addPoint((int) pointD.x, (int) pointD.y);
        }
        return localPolygon;
    }

    private void paintPolygon(Polygon returnPolygon) {
        JPanel panel = new JPanel() {
            @Override protected void paintComponent(Graphics g) {
                g.drawPolygon(returnPolygon);
            }
        };
        add(panel);
        pack();

        setSize(500, 500);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }


    public static void main(String[] args) {
        int steps = 5;
        new KochKurve(steps);
    }
}

 class Vector2 {
    public final double x;
    public final double y;

    public Vector2(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double length() {
        return Math.sqrt(x * x + y * y);
    }

    public Vector2 normalized() {
        double len = length();
        return new Vector2(x / len, y / len);
    }

    public Vector2 scale(double scalar) {
        return new Vector2(x * scalar, y * scalar);
    }

    public Vector2 add(Vector2 other) {
        return new Vector2(x + other.x, y + other.y);
    }

    public Vector2 subtract(Vector2 other) {
        return new Vector2(x - other.x, y - other.y);
    }

    public Vector2 linearInterpolation(Vector2 other, double t) {
        return new Vector2((1 - t) * x + t * other.x, (1 - t) * y + t * other.y);
    }
}
```