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

Лабораторная работа №7

По дисциплине: «Современные платформы программирования»
Тема: «Графические приложения в Java»

Выполнил:
Студент 3 курса
Группы ПО-8
Бувин Д.А.
Проверил:
А. А. Крощенко

Брест, 2024

Лабораторная работа №7
Вариант 3

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

Задание №1:

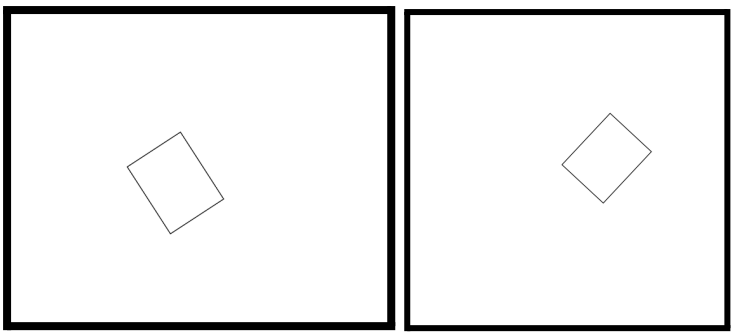
Изобразить четырехугольник, вращающийся в плоскости апплета вокруг своего центра тяжести.

Код программы:

```
import javafx.animation.AnimationTimer;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Task_1 extends Application {
  private static final int WIDTH = 800;
  private static final int HEIGHT = 600;
  private static final Color RED = Color.RED;
  private static final Color BLACK = Color.BLACK;

  private double angle = 0;
  private double centerX = WIDTH / 2;
  private double centerY = HEIGHT / 2;
  private double width = 100;
  private double height = 80;

  @Override
  public void start(Stage primaryStage) {
    Canvas canvas = new Canvas(WIDTH, HEIGHT);
    GraphicsContext gc = canvas.getGraphicsContext2D();

    Group root = new Group();
    root.getChildren().add(canvas);

    Scene scene = new Scene(root, WIDTH, HEIGHT);
```

```
        primaryStage.setScene(scene);
        primaryStage.setTitle("Rotating Quadrilateral");
        primaryStage.show();

        new AnimationTimer() {
            @Override
            public void handle(long now) {
                gc.clearRect(0, 0, WIDTH, HEIGHT);
                drawRotatedQuadrilateral(gc);
                angle += 1;
            }
        }.start();
    }

    private void drawRotatedQuadrilateral(GraphicsContext gc) {
        gc.setFill(RED);
        gc.setStroke(BLACK);

        double[] xPoints = {centerX, centerX + width, centerX + width, centerX};
        double[] yPoints = {centerY, centerY, centerY + height, centerY + height};

        for (int i = 0; i < xPoints.length; i++) {
            double newX = centerX + (xPoints[i] - centerX) * Math.cos(Math.toRadians(angle))
                    - (yPoints[i] - centerY) * Math.sin(Math.toRadians(angle));
            double newY = centerY + (xPoints[i] - centerX) * Math.sin(Math.toRadians(angle))
                    + (yPoints[i] - centerY) * Math.cos(Math.toRadians(angle));
            xPoints[i] = newX;
            yPoints[i] = newY;
        }

        gc.strokePolygon(xPoints, yPoints, 4);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Результат работы:



Задание №2:

Реализовать построение заданного типа фрактала по варианту

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

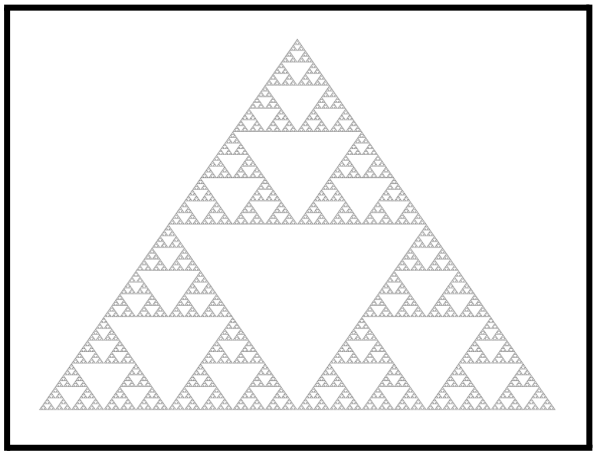
Треугольная салфетка Серпинского

Код программы:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.layout.BorderPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class Task_2 extends Application {
  private static final int WIDTH = 800;
  private static final int HEIGHT = 600;

  private int depth = 10;

  @Override
  public void start(Stage primaryStage) {
     BorderPane root = new BorderPane();
     Canvas canvas = new Canvas(WIDTH, HEIGHT);
     root.setCenter(canvas);

     GraphicsContext gc = canvas.getGraphicsContext2D();
     gc.setFill(Color.WHITE);
     gc.fillRect(0, 0, WIDTH, HEIGHT);

      drawSierpinskiTriangle(gc, 50, HEIGHT - 50, WIDTH - 50, HEIGHT - 50, WIDTH / 2,
50, depth);
```

```
        primaryStage.setTitle("Sierpinski Triangle");
        primaryStage.setScene(new Scene(root, WIDTH, HEIGHT));
        primaryStage.show();
    }

    private void drawSierpinskiTriangle(GraphicsContext gc, double x1, double y1, double x2,
double y2, double x3, double y3, int depth) {
        if (depth == 0) {
            gc.setFill(Color.BLACK);
            gc.fillPolygon(new double[]{x1, x2, x3}, new double[]{y1, y2, y3}, 3);
            return;
        }

        double mid1X = (x1 + x2) / 2;
        double mid1Y = (y1 + y2) / 2;
        double mid2X = (x2 + x3) / 2;
        double mid2Y = (y2 + y3) / 2;
        double mid3X = (x1 + x3) / 2;
        double mid3Y = (y1 + y3) / 2;

        drawSierpinskiTriangle(gc, x1, y1, mid1X, mid1Y, mid3X, mid3Y, depth - 1);
        drawSierpinskiTriangle(gc, mid1X, mid1Y, x2, y2, mid2X, mid2Y, depth - 1);
        drawSierpinskiTriangle(gc, mid3X, mid3Y, mid2X, mid2Y, x3, y3, depth - 1);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Результат работы:



Вывод: По итогу выполнения лабораторной работы, я освоил возможности языка программирования Java в построении графических приложений.