

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

## Отчет по лабораторной работе №7

Специальность ПО9(з)

Выполнил  
Д. Н. Кухарев,  
студент группы ПО9

Проверил  
А. А. Крощенко,  
ст. преп. кафедры ИИТ,  
«\_\_k\_\_\_\_\_2024 г.

Брест 2024

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

### Вариант 9

**Задание 1. Создать классы Point и Line. Объявить массив из n объектов класса Point. Для объекта класса Line определить, какие из объектов Point лежат на одной стороне от прямой линии и какие на другой. Реализовать ввод данных для объекта Line и случайное задание данных для объекта Point.**

Выполнение:

Определять с какой стороны Line находятся какие объекты Point будем с помощью процентного соотношения.

### Код программы

**Main.java:**

```
import javafx.application.Application;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.Button;
import javafx.scene.control.TextInputDialog;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
import javafx.scene.Scene;

import java.util.ArrayList;

public class Main extends Application {
    final static public int WIDTH = 800;
    final static public int HEIGHT = 600;
    static ArrayList<Point> points = new ArrayList<>();
    static Pane root;
    static Canvas canvas;
    static GraphicsContext gc;
    @Override
    public void start(Stage primaryStage) throws InterruptedException {
        TextInputDialog xDialog = new TextInputDialog();
        xDialog.setHeaderText("Enter x coordinate:");
        String xResult = xDialog.showAndWait().orElse(WIDTH/2+"");
        int x = Integer.parseInt(xResult);

        TextInputDialog yDialog = new TextInputDialog();
        yDialog.setHeaderText("Enter y coordinate:");
        String yResult = yDialog.showAndWait().orElse(HEIGHT/2+"");
        int y = Integer.parseInt(yResult);

        TextInputDialog rotationDialog = new TextInputDialog();
        rotationDialog.setHeaderText("Enter rotation angle:");
        String rotationResult = rotationDialog.showAndWait().orElse("0");
        int rotation = Integer.parseInt(rotationResult);

        TextInputDialog colorDialog = new TextInputDialog();
        colorDialog.setHeaderText("Enter line color (r,g,b):");
        String colorResult = colorDialog.showAndWait().orElse("0,0,0");
        String[] colors = colorResult.split(",");

        TextInputDialog percentageDialog = new TextInputDialog();
        percentageDialog.setHeaderText("Enter right side dots percentage (0-100):");
```

```

String percentageResult = percentageDialog.showAndWait().orElse("50");
int percentage = Integer.parseInt(percentageResult);

Color color = Color.rgb(Integer.parseInt(colors[0]), Integer.parseInt(colors[1]),
Integer.parseInt(colors[2]));
Line.setX(x);
Line.setY(y);
Line.setRotation(rotation);
Line.setColor(color);
Line.setRightPercentage(percentage);

canvas = new Canvas(800, 600);
gc = canvas.getGraphicsContext2D();

root = new Pane();
Button button = new Button("Draw points");
button.setOnAction(event -> {
    int amount = (int) (Math.random() * 40 + 10);
    for (int i = 0; i < amount; ++i) {
        if(i < Math.round((double) amount * (double) Line.getRightPercentage() / 100)){
            points.add(new Point(true));
        }
        else {
            points.add(new Point(false));
        }
    }
    drawPoints();
});

drawLine();

root.getChildren().add(canvas);
root.getChildren().add(button);
primaryStage.setScene(new Scene(root, WIDTH, HEIGHT));
primaryStage.show();
}
public static void main(String[] args) {
    launch(args);
}
public static void drawLine(){
    gc.save();

    gc.setStroke(Line.getColor());
    gc.setLineWidth(2.0);

    gc.translate(Line.getX(), Line.getY());
    gc.rotate(Line.getRotation());

    gc.strokeLine(-5000, 0, 5000, 0);

    gc.restore();
}
public static void drawPoints(){
    for (int i = 0; i < points.size(); ++i) {

        Circle circle = new Circle(points.get(i).getX(), points.get(i).getY(), points.get(i).getSize());
        circle.setFill(points.get(i).getColor());
        root.getChildren().add(circle);
    }
}
}

```

```

class Point {
    private int x;
    private int y;
    private int red, green, blue;
    private double size;
    private boolean right;

    public Point(boolean right) {
        if(right){
            x = (int) (Math.random() * Main.WIDTH);
            y = (int) (Math.random() * Main.HEIGHT);
            while (isPointAboveLine(Line.getX(), Line.getY(), Line.getRotation(), x, y)){
                x = (int) (Math.random() * Main.WIDTH);
                y = (int) (Math.random() * Main.HEIGHT);
            }
        }else {
            x = (int) (Math.random() * Main.WIDTH);
            y = (int) (Math.random() * Main.HEIGHT);
            while (!isPointAboveLine(Line.getX(), Line.getY(), Line.getRotation(), x, y)){
                x = (int) (Math.random() * Main.WIDTH);
                y = (int) (Math.random() * Main.HEIGHT);
            }
        }
        red = (int) (Math.random() * 256);
        green = (int) (Math.random() * 256);
        blue = (int) (Math.random() * 256);
        size = Math.random() * 5 + 2;
        this.right = right;
    }

    public static boolean isPointAboveLine(double centerX, double centerY, double rotationDegrees, double
pointX, double pointY) {
        double rotationRadians = Math.toRadians(rotationDegrees);

        double diffX = pointX - centerX;
        double diffY = pointY - centerY;

        double rotatedDiffX = diffX * Math.cos(-rotationRadians) - diffY * Math.sin(-rotationRadians);
        double rotatedDiffY = diffX * Math.sin(-rotationRadians) + diffY * Math.cos(-rotationRadians);

        return rotatedDiffY > 0;
    }

    public double getSize(){
        return size;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }

    public boolean getPos(){
        return right;
    }

    public Color getColor() {

```

```

        return Color.rgb(red, green, blue);
    }
}

```

### Line.java:

```

public class Line {
    static int x, y, rotation;
    static int rightPercentage;
    static Color color;
    public Line(int x, int y, int rotation, Color color, int rightPercentage){
        this.x = x;
        this.y = y;
        this.rotation = rotation;
        this.color = color;
        this.rightPercentage = rightPercentage;
    }
    static public int getX(){
        return x;
    }
    static public void setX(int newX){
        x = newX;
    }
    static public int getY(){
        return y;
    }
    static public void setY(int newY){
        y = newY;
    }
    static public int getRotation(){
        return rotation;
    }
    static public void setRotation(int newrotation){
        rotation = newrotation;
    }
    static public Color getColor(){
        return color;
    }
    static public void setColor(Color newcolor){
        color = newcolor;
    }
    static public int getRightPercentage(){
        return rightPercentage;
    }
    static public void setRightPercentage(int newrightPercentage){
        rightPercentage = newrightPercentage;
    }
}

```

### Рисунки с результатами работы программы

Confirmation
✕

Enter x coordinate:

Confirmation
✕

Enter y coordinate:

Confirmation

Enter rotation angle: ?

120

OK Cancel

Confirmation

Enter line color (r,g,b): ?

60,0,100

OK Cancel

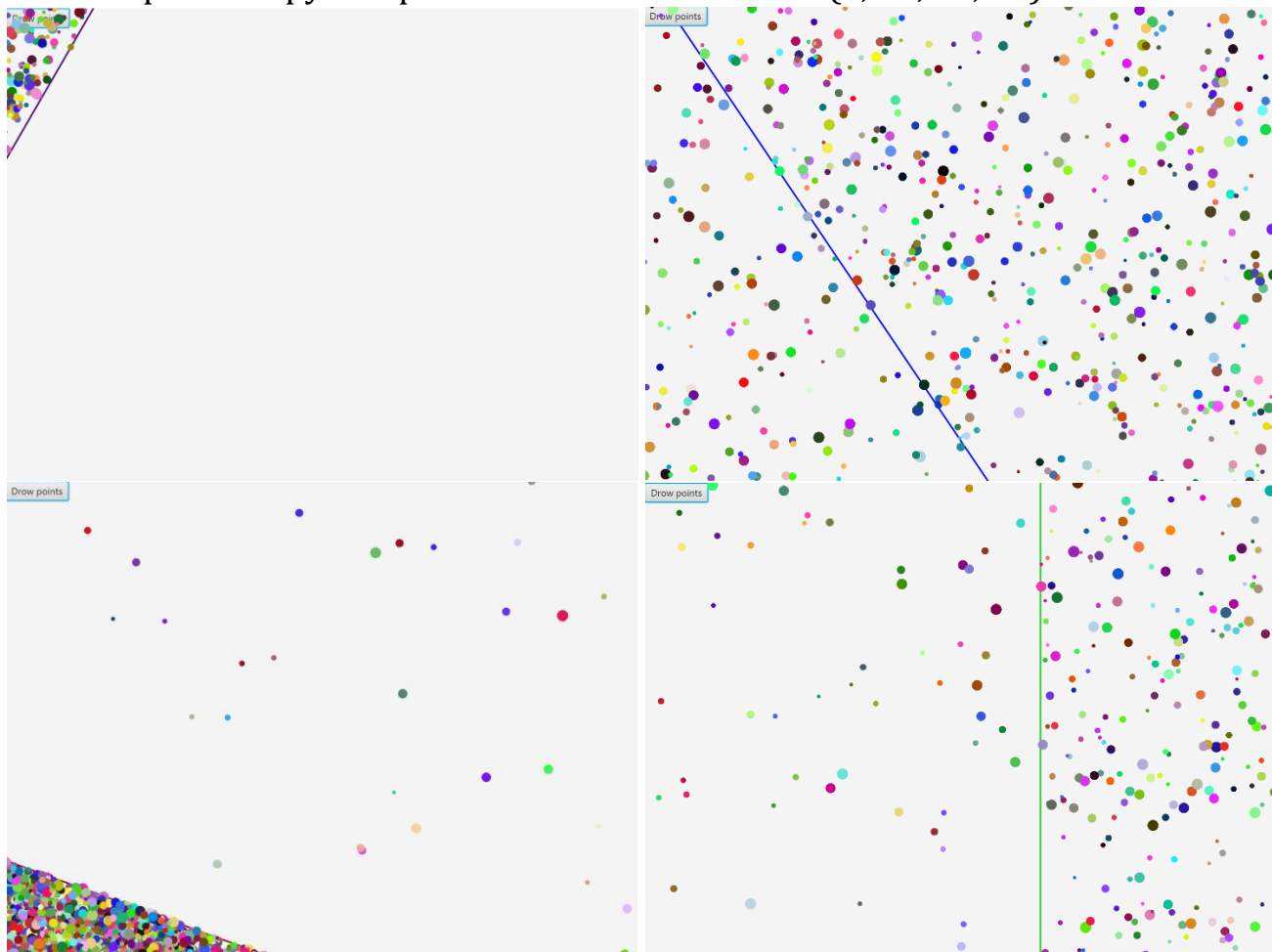
Confirmation

Enter right side dots percentage (0-100): ?

0

OK Cancel

Поэкспериментируем с различными значениями (0, 70, 99, 80)



Кнопка Draw Points добавляет ещё больше случайных точек, при этом сохраняя условия их распределения.

## Задание 2. Реализовать построение фрактала Остров Минковского.

Выполнение:

### Код программы

**Main.java:**

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.TextInputDialog;
import javafx.scene.input.KeyCode;
import javafx.scene.input.MouseButton;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Line;
import javafx.scene.shape.Rectangle;
import javafx.scene.transform.Scale;
import javafx.stage.Stage;
import java.util.Optional;

public class Main extends Application {
    private static final int WIDTH = 900;
    private static final int HEIGHT = 600;
    private static final int SIDE_SIZE = 300;
    private Pane root;
    private double startX, startY;
    private double startTranslateX, startTranslateY;
    static public int generations_amount;

    @Override
    public void start(Stage stage) throws Exception {

        TextInputDialog gDialog = new TextInputDialog();
        gDialog.setHeaderText("Enter amount of generations:");
        String gResult = gDialog.showAndWait().orElse("2");

        generations_amount = Integer.parseInt(gResult);
        TextInputDialog colorDialog = new TextInputDialog();
        colorDialog.setHeaderText("Enter line color (r,g,b):");
        String colorResult = colorDialog.showAndWait().orElse("0,0,0");

        String[] colors = colorResult.split(",");
        Color lineColor = Color.rgb(Integer.parseInt(colors[0]), Integer.parseInt(colors[1]),
Integer.parseInt(colors[2]));
        root = new Pane();
        Group group = new Group(root);
        Scale scale = new Scale(1, 1);
        root.getTransforms().add(scale);
        Scene scene = new Scene(group, WIDTH, HEIGHT);
        double squareSize = SIDE_SIZE;
        startX = (WIDTH - squareSize) / 2;
        startY = (HEIGHT - squareSize) / 2;
        double endX = startX + squareSize;
        double endY = startY + squareSize;
        drawMinkowskiSquare(startX, startY, endX, endY, generations_amount, lineColor);
        scene.setOnScroll(event -> {
            double scaleFactor = event.getDeltaY() > 0 ? 1.1 : 0.9;
            scale.setX(scale.getX() * scaleFactor);
            scale.setY(scale.getY() * scaleFactor);
            event.consume();
        });
    }
}
```

```

scene.setOnMousePressed(event -> {
    if (event.getButton() == MouseButton.PRIMARY) {
        startTranslateX = event.getSceneX() - root.getTranslateX();
        startTranslateY = event.getSceneY() - root.getTranslateY();
    }
});
scene.setOnMouseDragged(event -> {
    if (event.getButton() == MouseButton.PRIMARY) {
        double offsetX = event.getSceneX() - startTranslateX;
        double offsetY = event.getSceneY() - startTranslateY;
        root.setTranslateX(offsetX);
        root.setTranslateY(offsetY);
    }
});
stage.setScene(scene);
stage.show();
scene.setOnKeyPressed(event -> {
    if (event.getCode() == KeyCode.Q) {
        if (generations_amount < 4) {
            generations_amount++;
            root.getChildren().clear();
            drawMinkowskiSquare(startX, startY, endX, endY, generations_amount, lineColor);
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setHeaderText("Warning");
            alert.setContentText("Application will crush");
            alert.showAndWait();
        }
    } else if (event.getCode() == KeyCode.A) {
        if (generations_amount > 0) {
            generations_amount--;
            root.getChildren().clear();
            drawMinkowskiSquare(startX, startY, endX, endY, generations_amount, lineColor);
        } else {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setHeaderText("Warning");
            alert.setContentText("Cannot decrease generations amount below 0.");
            alert.showAndWait();
        }
    }
});
}

/**
 * Draws a square made of Minkowski curves.
 */
private void drawMinkowskiSquare(double startX, double startY, double endX, double endY, int
generations, Color color) {
    double width = endX - startX;
    double height = endY - startY;

    drawMinkowski(startX, startY, startX + width, startY, generations, color); // Top side
    drawMinkowski(startX + width, startY, startX + width, startY + height, generations, color); // Right side
    drawMinkowski(startX + width, startY + height, startX, startY + height, generations, color); // Bottom
side
    drawMinkowski(startX, startY + height, startX, startY, generations, color); // Left side
}

/**
 * Draws a Minkowski curve between two points.
 */
private void drawMinkowski(double xa, double ya, double xi, double yi, int i, Color color) {
    if (i == 0) {

```



```

        Line line = new Line(xa, ya, xi, yi);
        line.setStroke(color);
        root.getChildren().add(line);
    } else {
        double xb = xa + (xi - xa) * 1 / 4;
        double yb = ya + (yi - ya) * 1 / 4;

        double xe = xa + (xi - xa) * 2 / 4;
        double ye = ya + (yi - ya) * 2 / 4;

        double xh = xa + (xi - xa) * 3 / 4;
        double yh = ya + (yi - ya) * 3 / 4;

        double cos90 = 0;
        double sin90 = -1;
        double xc = xb + (xe - xb) * cos90 - sin90 * (ye - yb);
        double yc = yb + (xe - xb) * sin90 + cos90 * (ye - yb);

        double xd = xc + (xe - xb);
        double yd = yc + (ye - yb);

        sin90 = 1;
        double xf = xe + (xh - xe) * cos90 - sin90 * (yh - ye);
        double yf = ye + (xh - xe) * sin90 + cos90 * (yh - ye);

        double xg = xf + (xh - xe);
        double yg = yf + (yh - ye);


        drawMinkowski(xa, ya, xb, yb, i - 1, color);
        drawMinkowski(xb, yb, xc, yc, i - 1, color);
        drawMinkowski(xc, yc, xd, yd, i - 1, color);
        drawMinkowski(xd, yd, xe, ye, i - 1, color);
        drawMinkowski(xe, ye, xf, yf, i - 1, color);
        drawMinkowski(xf, yf, xg, yg, i - 1, color);
        drawMinkowski(xg, yg, xh, yh, i - 1, color);
        drawMinkowski(xh, yh, xi, yi, i - 1, color);
    }
}
}

```

## Рисунки с результатами работы программы

Confirmation


✕

Enter amount of generations: 

OK Cancel

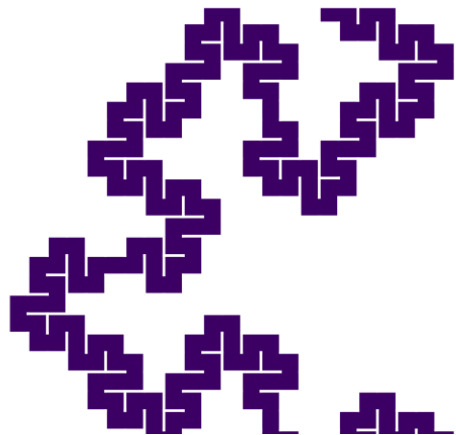
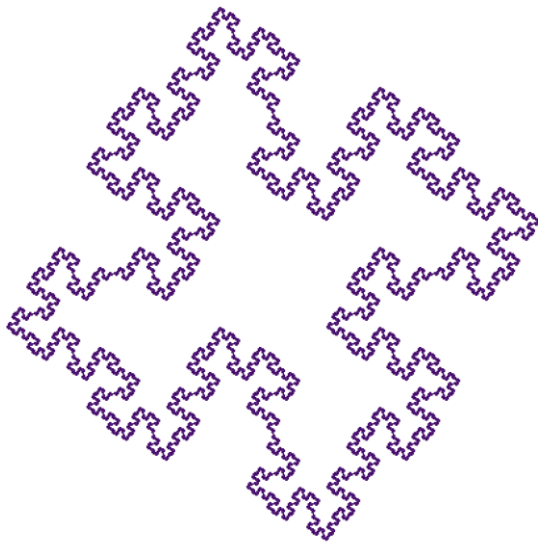
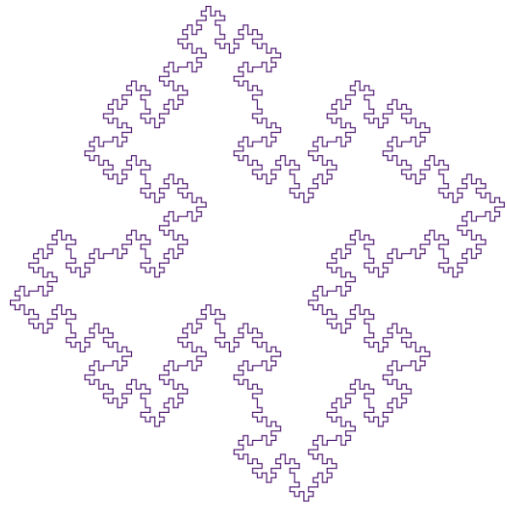
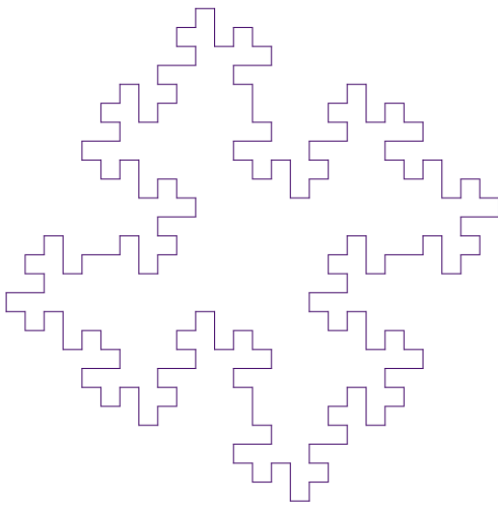
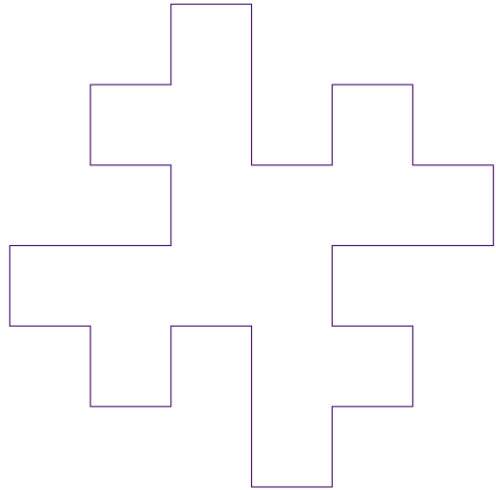
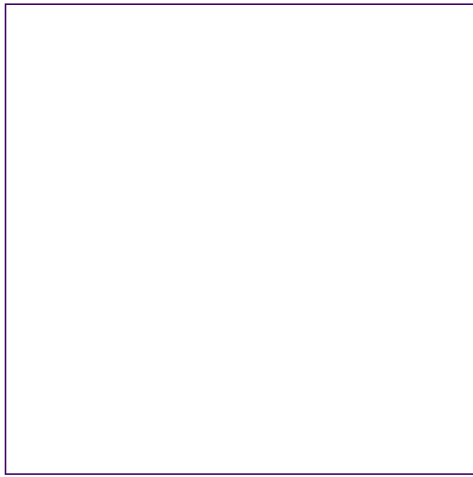
Confirmation

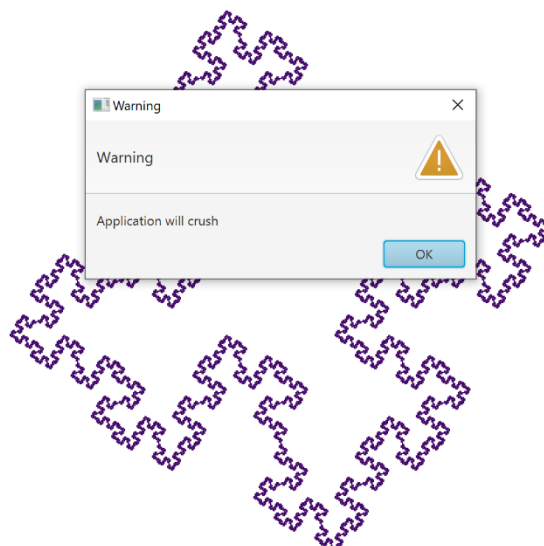
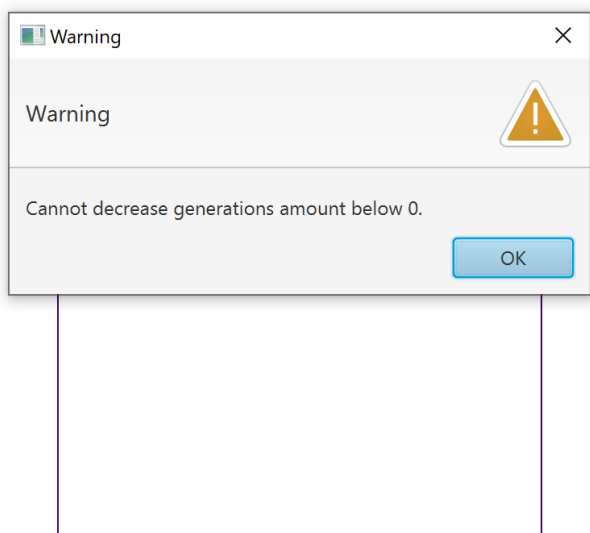
✕

Enter line color (r,g,b): 

OK Cancel

Регулировать количество генераций можно клавишами q и a для увеличения и уменьшения соответственно. Размеры не задаются так как остров можно приближать и перетаскивать.





При попытке выйти за нижнюю или верхнюю границу числа генераций вылезет предупреждение. Обойти верхнюю границу можно при запуске но это может повлечь вылет программы и делается на мощных компьютерах либо на свой страх и риск.

**Вывод:** освоил возможности языка программирования Java в построении графических приложений.