МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №7

Выполнил:
А. Н. Марзан,
студент 3 курса
группы ПО-9

Проверил:
А. А. Крощенко,
«29» 04 2024 г.

Брест 2024

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

Вариант 12

Ход работы

Общее задание

Реализовать соответствующие классы, указанные в задании; Организовать ввод параметров для создания объектов (можно использовать файлы); Осуществить визуализацию графических примитивов, решить поставленную задачу

Задание 1

12) Задать составление строки из символов, появляющихся из разных углов апплета и выстраивающихся друг за другом. Процесс должен циклически повторяться.

Задание 2

Реализовать построение заданного типа фрактала по варианту. Везде, где это необходимо, предусмотреть ввод параметров, влияющих на внешний вид фрактала.

12) Кривая Гильберта

Код программы 1:

```
import javafx.animation.TranslateTransition;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.util.Random;

public class Main extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        Random random = new Random();
        primaryStage.setTitle("Letter Animation");
```

```
double width = 1000, height = 1000;

Group root = new Group();
Scene scene = new Scene(root, width, height, Color.WHITE);

char[] letters = {'A', 'N', 'D', 'R', 'E', 'Y', ' ', 'M', 'A', 'R', 'Z', 'A', 'N'};
int startX = 10;
int startY = 30;

for (char letterChar : letters) {
    Text letter = new Text(String.valueOf(letterChar));
    letter.setFont(Font.font("Arial", FontWeight.BOLD, 30));
    letter.setFill(Color.BLACK);
    letter.setX(startX);
    letter.setY(startY);

    root.getChildren().add(letter);

    TranslateTransition transition = new TranslateTransition(Duration.seconds(2), letter);

    switch (random.nextInt(4))
    {
        case 0:
            transition.setFromX(-startX);
            transition.setFromY(height-startY);
            break;
        case 1:
            transition.setFromX(width-startX);
            transition.setFromY(-startY);
            break;
        case 2:
            transition.setFromX(width-startX);
            transition.setFromY(height-startY);
            break;
        default:
            transition.setFromX(-startX);
            transition.setFromY(-startY);
            break;
    }

    transition.setToX(250);
    transition.setToY(250);

    transition.setCycleCount(TranslateTransition.INDEFINITE);
    transition.setAutoReverse(true);
    transition.play();

    startX += 40; // Увеличиваем горизонтальное смещение для следующей буквы
}
```

```
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

Код программы 2:

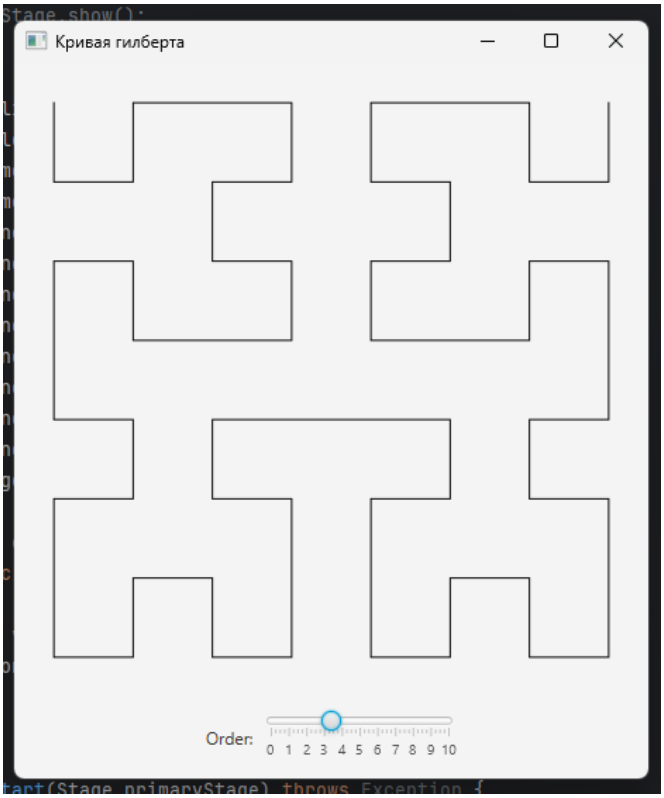
```
import javafx.application.Application;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Polyline;
import javafx.stage.Stage;

public class Main extends Application {
    private static final double SIZE = 450;

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        Pane pane = new HilbertCurvePane();
        Scene scene = new Scene(pane);

            primaryStage.setTitle("Кривая гилберта");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private class HilbertCurvePane extends BorderPane {
        private final int UP = 0;
        private final int LEFT = 1;
```

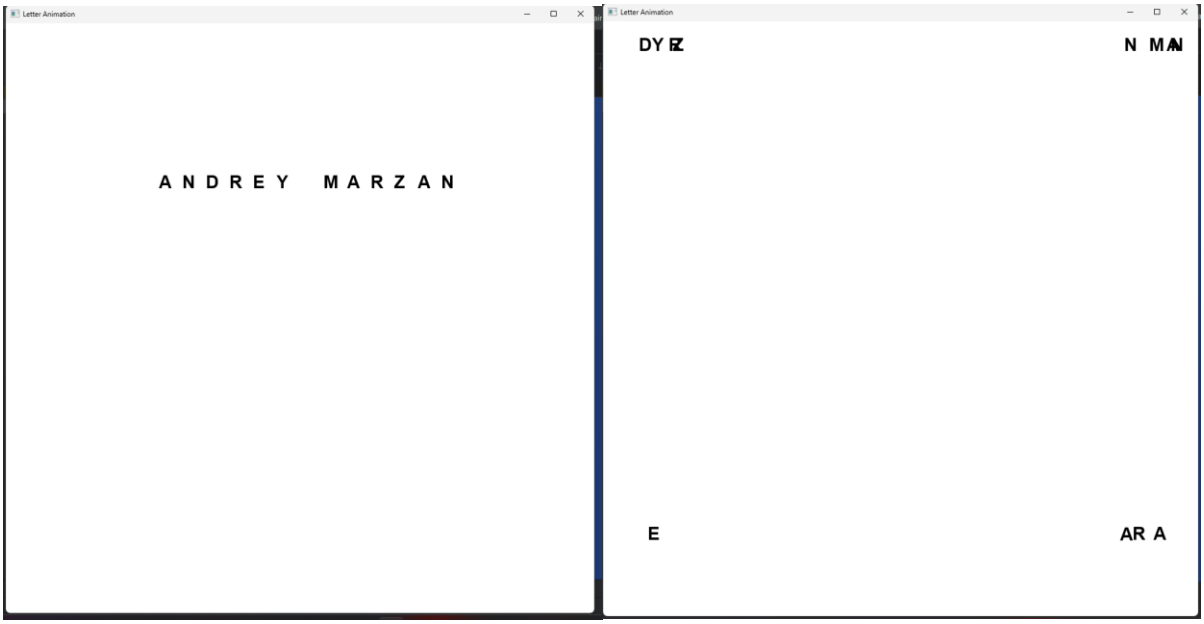
```
    private final int DOWN = 2;
    private final int RIGHT = 3;
    private final Polyline polyline;
    private Pane pane;
    private Slider orderSlider;
    private int direction;

    public HilbertCurvePane() {
        setCenter(createCanvas());
        setBottom(createControlBar());
        polyline = new Polyline();
        pane.getChildren().add(polyline);
    }

    private Node createCanvas() {
        pane = new Pane();
        pane.setPrefSize(SIZE, SIZE);
        return pane;
    }

    private Node createControlBar() {
        Label label = new Label("Order:");
        orderSlider = new Slider(0, 10, 0); // Минимальное значение,
максимальное значение, начальное значение
        orderSlider.setShowTickLabels(true);
        orderSlider.setShowTickMarks(true);
        orderSlider.setMajorTickUnit(1);
        orderSlider.setBlockIncrement(1);
        orderSlider.valueProperty().addListener((observable, oldValue,
newValue) -> drawHilbertCurve(newValue.intValue()));
        HBox hBox = new HBox(5, label, orderSlider);
        hBox.setPadding(new Insets(10));
        hBox.setAlignment(Pos.CENTER);
        return hBox;
    }

    private void drawHilbertCurve(int order) {
        polyline.getPoints().clear();
        direction = UP;
        drawHilbertCurve(order, 0, 0, pane.getWidth(), pane.getHeight());
```

```
    }

    private void drawHilbertCurve(int order, double x, double y, double width,
double height) {
        ObservableList<Double> points = polyline.getPoints();
        if (order == 0) {
            points.addAll(width / 2 + x, height / 2 + y);
        } else {
            double halfWidth = width / 2;
            double halfHeight = height / 2;
            if (direction == UP) {
                direction = LEFT;
                drawHilbertCurve(order - 1, x, y, halfWidth, halfHeight);
                direction = UP;
                drawHilbertCurve(order - 1, x, y + halfHeight, halfWidth,
halfHeight);
                direction = UP;
                drawHilbertCurve(order - 1, x + halfWidth, y + halfHeight,
halfWidth, halfHeight);
                direction = RIGHT;
                drawHilbertCurve(order - 1, x + halfWidth, y, halfWidth,
halfHeight);
            } else if (direction == RIGHT) {
                direction = DOWN;
                drawHilbertCurve(order - 1, x + halfWidth, y + halfHeight,
halfWidth, halfHeight);
                direction = RIGHT;
                drawHilbertCurve(order - 1, x, y + halfHeight, halfWidth,
halfHeight);
                direction = RIGHT;
                drawHilbertCurve(order - 1, x, y, halfWidth, halfHeight);
                direction = UP;
                drawHilbertCurve(order - 1, x + halfWidth, y, halfWidth,
halfHeight);
            } else if (direction == DOWN) {
                direction = RIGHT;
                drawHilbertCurve(order - 1, x + halfWidth, y + halfHeight,
halfWidth, halfHeight);
                direction = DOWN;
```

```
                drawHilbertCurve(order - 1, x + halfWidth, y, halfWidth,
halfHeight);
                direction = DOWN;
                drawHilbertCurve(order - 1, x, y, halfWidth, halfHeight);
                direction = LEFT;
                drawHilbertCurve(order - 1, x, y + halfHeight, halfWidth,
halfHeight);
            } else if (direction == LEFT) {
                direction = UP;
                drawHilbertCurve(order - 1, x, y, halfWidth, halfHeight);
                direction = LEFT;
                drawHilbertCurve(order - 1, x + halfWidth, y, halfWidth,
halfHeight);
                direction = LEFT;
                drawHilbertCurve(order - 1, x + halfWidth, y + halfHeight,
halfWidth, halfHeight);
                direction = DOWN;
                drawHilbertCurve(order - 1, x, y + halfHeight, halfWidth,
halfHeight);
            }
        }
    }
  }
}
```

Вывод программы:





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