Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления

ИТОГОВЫЙ ПРОЕКТ

Разработка черепашьей графики с поддержкой Л-систем
по дисциплине «Системный подход в разработке программного обеспечения»

Выполнил
студент гр.3530902/70201 Т.И. Матченко

Руководитель
доцент, к.т.н. С.А. Нестеров

«___» _____ 2019 г.

Санкт-Петербург
2019

Оглавление

Введение

Цель курсовой работы — создание черепашьей графики с поддержкой Л-систем.

В программе должны присутствовать: работа со строками и числами, пользовательским вводом, массивами или ArrayList, циклы, проверка условий, перегрузка методов, использование различных модификаторов доступа к полям и методам классов, обработка исключений.

Проект включает в себя возможность вводить команды напрямую или преобразовывать аксиому и правила л-системы в команды для черепашки.

1 Проектирование и документирование

1.1 Функциональная часть

Классы программы:

- Main — класс, из которого запускается приложение, а также инициализируется интерфейс программы (приложение 1);
- Action — класс, хранящий в себе идентификатор действия, аргумент действия и функцию, приводящую это действие в исполнение (приложение 2);
- Actions — класс, переводящий символы из аксиомы в действия (приложение 3);
- LSystem — класс, который, используя полученную грамматику, генерирует строку через определенное количество итераций (приложение 4);
- Drawer – класс, содержащий набор методов черепашки (приложение 5);
- CommandBox — класс, генерирующий панель для одной команды в интерфейсе программы, а также анализирующий введенные команды и запускающий черепашку (приложение 6);
- RuleBox — класс, генерирующий панель для одного правила в интерфейсе программы (приложение 7);
- IncorrectInputException – класс для исключения о неверно заполненных полях (приложение 8).

При запуске Main инициализируется меню, состоящее из полей аксиомы, правил (RuleBox-ов), команд черепашки (CommandBox-ов), выбора количества итераций, кнопки преобразования правил и аксиомы в команды (добавляются к панели команд), кнопок очистки панели команд и отрисовки.

Конструкторы CommandBox и RuleBox добавляют созданные HBox-ы в панель (VBox) команд/правил. У обоих есть кнопки для добавления еще одного элемента или удаления текущего.

Благодаря классам LSystem и Actions осуществляется преобразование аксиомы и правил в набор команд для черепашки.

Благодаря классам Actions, Action и Drawer осуществляется запуск и отрисовка изображения.

1.2 Проектирование

Класс Main взаимодействует с классами LSystem, Actions, RuleBox, CommandBox, Drawer. Класс Action взаимодействует с классом Drawer. Класс Actions взаимодействует с классами Action, LSystem, CommandBox. Класс RuleBox взаимодействует с классами LSystem, Actions, Main.

На рисунке 1 представлена диаграмма классов программы.

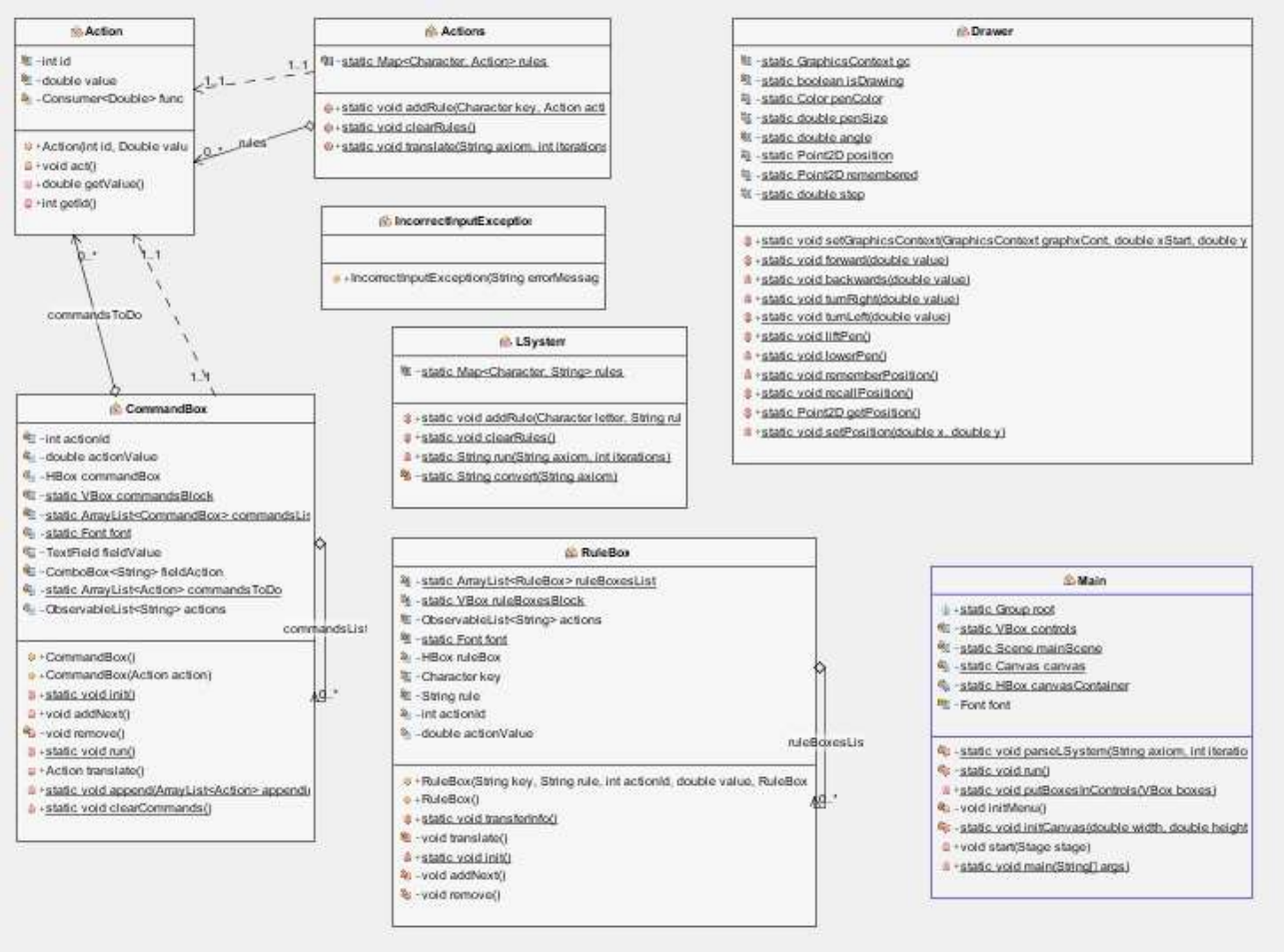

Рисунок 1 - Диаграмма классов приложения

2 Тестирование

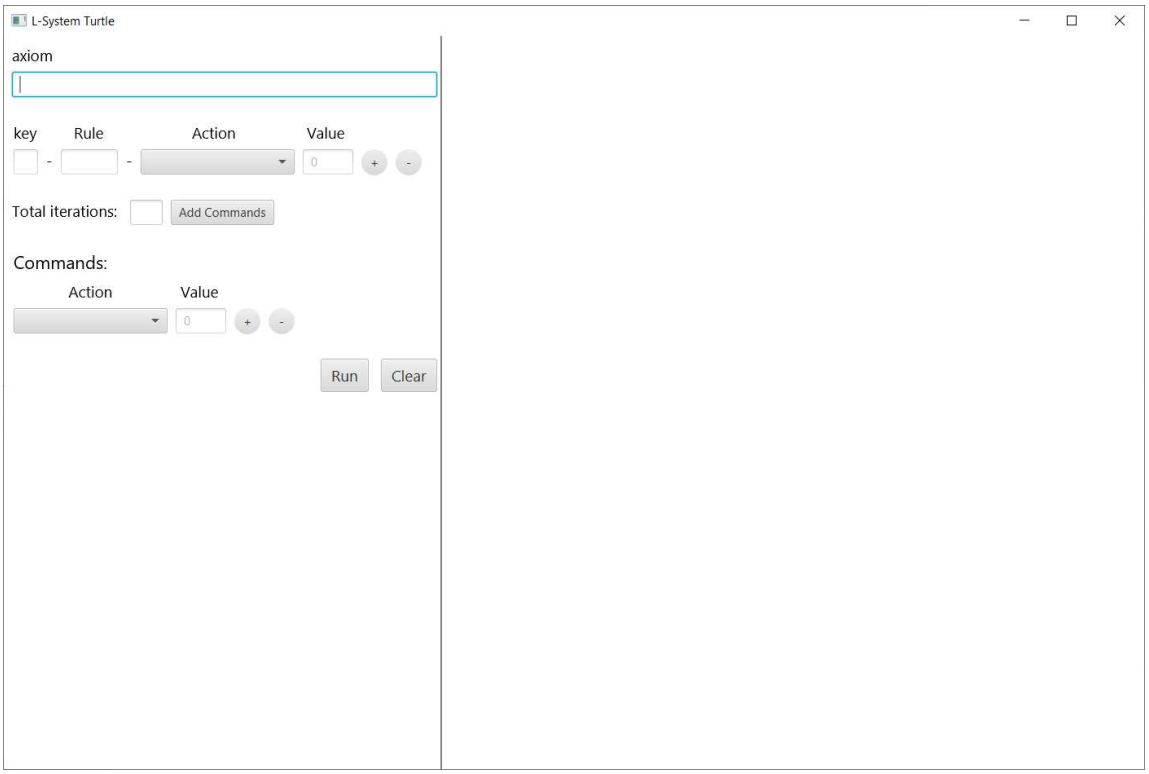

Рисунок 2 – Первоначальный вид окна

Введя аксиому, правила и количество итераций, можно нажать на Add Commands – преобразованные команды появятся в панели команд. Нажав на Run, можно получить изображение:

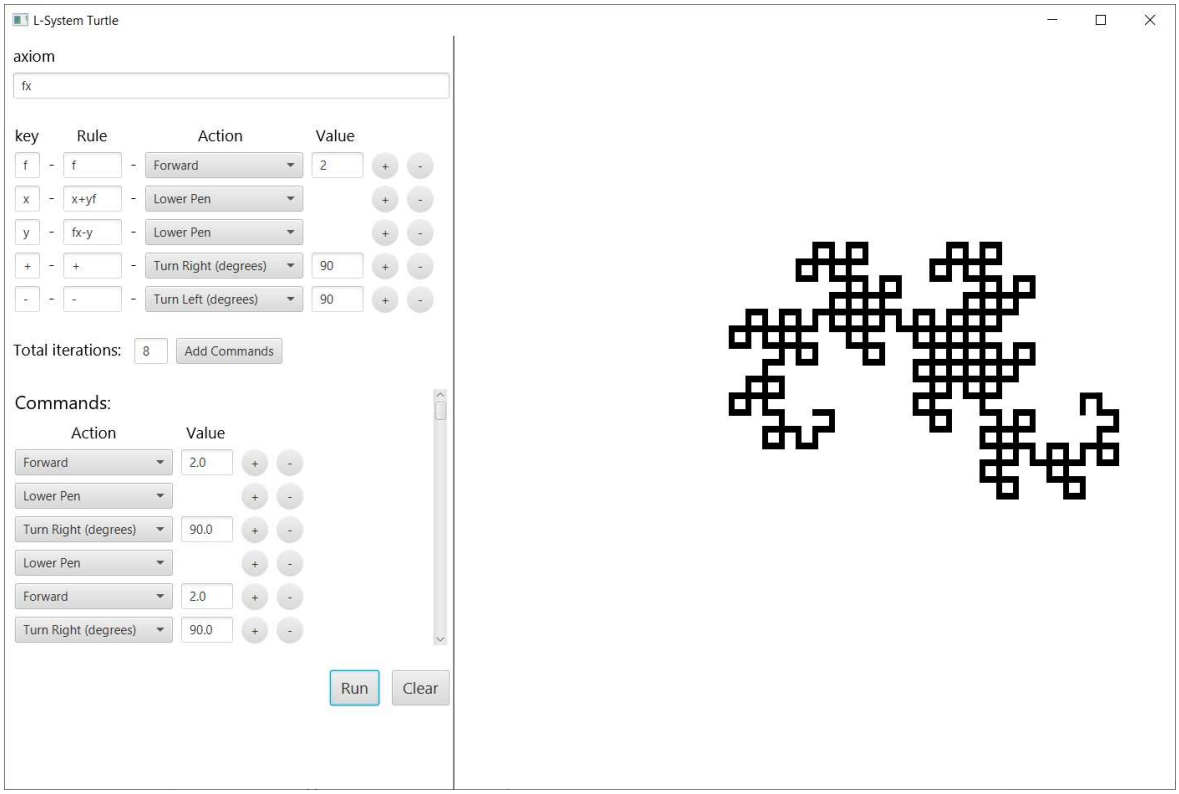

Рисунок 3 – Пример работы программы

Заключение

В ходе выполнения курсовой работы было создано приложение “L-System Turtle”.

Были получены навыки работы со строками и числами, пользовательским вводом, массивами и ArrayList, циклами, проверкой условий, перегрузкой методов, использованием различных модификаторов доступа к полям и методам классов, обработкой исключений.

Программа успешно работает при различных действиях, производимых над ней пользователем.

Список использованной литературы

1. Документация Oracle по JavaFX [Электронный ресурс] – Электрон. документация – 2015 -
Режим доступа: https://docs.oracle.com/javase/8/javafx/api/toc.htm, свободный.

2. Поле TextField [Электронный ресурс] – Электрон. текстовые дан. – 2018 – Режим доступа:
https://www.helloworld.ru/texts/comp/lang/java/java5/vol6/ch7.html, свободный.

3. JavaFX 2.0 основы [Электронный ресурс] – Электрон. текстовые дан. – 2012 - Режим доступа:
http://easy-code.ru/lesson/javafx-2-basic, свободный.

4. Обработка событий мыши в JavaFX[Электронный ресурс] – Электрон. текстовые дан. - 2018 – режим доступа: https:/ru.stackoverflow.com/questions-java-fx, свободный.

5. Списочный массив ArrayList в JavaFX [Электронный ресурс] – Электрон. текстовые дан. – 2009 – Режим
http://java-online.ru/java-arrayList.xhtml, свободный.

Приложения

Приложение 1 (класс Main)

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.control.ScrollPane.ScrollBarPolicy;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Line;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class Main extends Application {

    public static Group root = new Group();
    private static VBox controls;
    private static Scene mainScene;
    private static Canvas canvas; private static HBox canvasContainer;
    Font font = Font.font(19);

    private static void parseLSystem(String axiom, int iterations)
    {
        //очищаем правила
        Actions.clearRules(); LSystem.clearRules();
        //транслейтим рулбоксы и ловим ошибку, если какой-то не заполнен
        //транслейт заполняет правила в лсистеме и действиях
        RuleBox.transferInfo();
        //в действиях бежим по строке, собираем список действий для добавления
        Actions.translate(axiom, iterations);
        //добавляем команды из списка в панель команд
    }
    private static void run()
    {
        //обновляем/ресайзим канвас
        initCanvas(mainScene.getWidth()-537,mainScene.getHeight());
        //транслейтим команды в действия и делаем
        CommandBox.run();
    }
```

```
    public static void putBoxesInControls(VBox boxes)
    {
        //controls.getChildren().add(boxes);
        ScrollPane sp = new ScrollPane(); sp.setContent(boxes); sp.setStyle("-fx-
background: white; -fx-border-color: white;");
        sp.setMaxHeight(310); sp.setPrefWidth(522);
        sp.setHbarPolicy(ScrollBarPolicy.NEVER);
        sp.setVbarPolicy(ScrollBarPolicy.AS_NEEDED);

        controls.getChildren().add(sp);

    }

    private void initMenu()
    {
        Font font = Font.font(19);
        Line border = new Line(537,0,537,900);
        //делаем границу менюшки всегда до конца экрана
        mainScene.heightProperty().addListener((obs, oldValue, newValue) -> bor-
der.setEndY(newValue.doubleValue()));

        TextField axiom = new TextField(); axiom.setPrefSize(380, 20); ax-
iom.setPromptText("Starting state");
        Text textAxiom = new Text("axiom"); textAxiom.setFont(font);

        controls = new VBox(); controls.setLayoutX(10); controls.setLay-
outY(10); controls.setSpacing(28);
        controls.getChildren().add(new VBox(8,textAxiom,axiom));
        RuleBox.init();
        RuleBox rb1 = new RuleBox();

        //#region панель парсинга ЛСистемы
        Text textIterations = new Text("Total iterations: "); textItera-
tions.setFont(font);
        TextField fieldIterations = new TextField(); fieldIterations.setPref-
Width(40);
        fieldIterations.textProperty().addListener( (observa-
ble, oldValue, newValue) ->
        {   //validating double
            if (newValue.matches("[1-
9]\\d*") || newValue.length()==0) fieldIterations.setText(newValue);
            else fieldIterations.setText(oldValue);
        });
        Button btnAddCommands = new Button("Add Commands"); btnAddCom-
mands.setTooltip(new Tooltip("Computes axiom, rules and iterations into com-
mands for the Turtle"));
        btnAddCommands.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                parseLSystem(axiom.getText(),Integer.parseInt(fieldItera-
tions.getText()));
```

```
            }
        });
        HBox iterationAndAdd = new HBox(10,textIterations,fieldIterations,btnAdd-
Commands);
        controls.getChildren().add(iterationAndAdd);
        //#endregion

        //панель команд для черепахи
        CommandBox.init();
        CommandBox cm1 = new CommandBox();

        //панель финальных кнопок
        Button btnRun = new Button("Run"), btnStart = new But-
ton("Place Start"), btnClear = new Button("Clear"), btnAbout = new Button("?");
        btnRun.setFont(font); btnClear.setFont(font); btn-
Start.setFont(font); btnAbout.setFont(font);
        HBox btns = new HBox(15, btnRun,btnClear); btns.setAlignment(Pos.BASE-
LINE_RIGHT);
        btnClear.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                //initCanvas(mainScene.getWidth()-537,mainScene.getHeight());
                CommandBox.clearCommands();
            }
        });
        btnRun.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                run();
            }
        });

        canvasContainer = new HBox(); canvasContainer.setLayoutX(538); //can-
vasContainer.setLayoutY(0);

        controls.getChildren().addAll(btns);
        root.getChildren().addAll(controls,border,canvasContainer);
    }
    private static void initCanvas(double width, double height)
    {
        canvasContainer.getChildren().remove(canvas);
        canvas = new Canvas(width, height);
        canvasContainer.getChildren().add(canvas);
        Drawer.setGraphicsContext(canvas.getGraphicsCon-
text2D(), width/2, height/2);
    }

    @Override
    public void start(Stage stage) {

        //setting stage
```

```
        mainScene = new Scene(root, 1400, 900);
        stage.setMinHeight(947); stage.setMinWidth(1418);
        stage.setTitle("L-System Turtle");
        stage.setScene(mainScene);
        stage.show();

        initMenu();

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

Приложение 2 (класс Action)

```
import java.util.function.Consumer;

public class Action {

    private int id =-1;
    private double value = 0;
    private Consumer<Double> func;

    public void act(){func.accept(value);}

    public Action(int id, Double value)
    {
        this.id = id;
        this.value = value;

        switch (id) {
            case 0:
                func = val -> Drawer.forward(val);
                break;
            case 1:
                func = val -> Drawer.backwards(val);
                break;
            case 2:
                func = val -> Drawer.turnRight(val);
                break;
            case 3:
                func = val -> Drawer.turnLeft(val);
                break;
            case 4:
                func = val -> Drawer.liftPen();
                break;
            case 5:
```

```
                func = val -> Drawer.lowerPen();
                break;
            case 6:
                func = val -> Drawer.rememberPosition();
                break;
            case 7:
                func = val -> Drawer.recallPosition();
                break;
            default:
                System.out.println("Error assigning new action");
                break;
        }
    }

    public double getValue() {return value;}
    public int getId() {return id;}
}
```

Приложение 3 (класс Actions)

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

/** Stores rules for translating keys into actions */
public class Actions {

    private static Map<Character,Action> rules = new HashMap<Character,Action>();

    public static void addRule(Character key, Action action) {rules.put(key,ac-
tion);}
    public static void clearRules() {rules.clear();}
    public static void translate(String axiom, int iterations)
    {
        //get res from LSystem
        String resultingStr = LSystem.run(axiom, iterations);
        //parse it into actions
        ArrayList<Action> commandsToAppend = new ArrayList<>();
        for (int i=0; i<resultingStr.length();i++)
        {
            if (rules.containsKey(resultingStr.charAt(i)))
            commandsToAppend.add(rules.get(resultingStr.charAt(i)));
        }
        //put actions in the end of commands
        CommandBox.append(commandsToAppend);
    }
}
```

Приложение 4 (класс LSystem)

```
import java.util.HashMap;
import java.util.Map;

public class LSystem {

    private static Map<Character,String> rules = new HashMap<Character,String>();

    public static void addRule(Character letter, String rule) {rules.put(let-
ter, rule);}
    public static void clearRules() { rules.clear(); }
    public static String run(String axiom, int iterations)
    {
        String result = axiom;
        for (int i=0; i<iterations;i++) result = convert(result);
        System.out.println(result);
        return result;
    }
    private static String convert(String axiom)
    {
        String result = "";
        for (int i = 0; i < axiom.length(); i++)
        {
            if (rules.containsKey(axiom.charAt(i))) result += rules.get(ax-
iom.charAt(i));
            else result+=axiom.charAt(i);
        }
        return result;
    }

}
```

Приложение 5 (класс Drawer)

```
import javafx.scene.paint.Color;
import javafx.geometry.Point2D;
import javafx.scene.canvas.GraphicsContext;

/* Базовые команды:
повернуть направо / повернуть налево (на указанный угол)
переместиться вперёд / переместиться назад (на указанное расстоя-
ние, обычно в условных «шагах», часто равных по длине размеру пикселя экрана)
поднять перо / опустить перо
установить новый цвет пера / установить новую толщину пера
установить новый курс / установить новое место (поворот и перемещение относи-
тельно базовой Декартовой системы координат листа)
стереть всё
показать черепаху / спрятать черепаху
```

```
получить значения текущих координат, угла поворота черепахи, цвета и толщины пера
*/

public class Drawer {

    private static GraphicsContext gc;
    private static boolean isDrawing = true;
    private static Color penColor = Color.BLACK;
    private static double penSize = 7;
    private static double angle = 0; //градусы
    private static Point2D position;
    private static Point2D remembered;
    private static double step = 10; //длина отрезка при forward 1

    public static void setGraphicsContext(GraphicsContext graphxCont, dou-
ble xStart, double yStart)
    {
        gc = graphxCont;
        gc.setFill(penColor);
        gc.setStroke(Color.BLACK);
        gc.setLineWidth(penSize);
        gc.beginPath();
        position = new Point2D(xStart, yStart);
        gc.moveTo(position.getX(), position.getY());
    }

    public static void forward(double value)
    {
        double nextY = position.getY() + Math.sin(angle*Math.PI/180)*step*value;
        //double nextYRight = position.getY() + Math.sin(-an-
gle*Math.PI/180)*step*value;
        double nextX = position.getX() + Math.cos(angle*Math.PI/180)*step*value;
        //double nextXRight = position.getX() + Math.cos(-an-
gle*Math.PI/180)*step*value;
        if (isDrawing) gc.lineTo(nextX, nextY);
        position = new Point2D(nextX, nextY); angle = 0;
        gc.stroke();
    }
    public static void backwards(double value) {forward(-value);}
    public static void turnRight(double value) {angle += value;}
    public static void turnLeft(double value) {angle -= value;}
    public static void liftPen() {isDrawing = false;}
    public static void lowerPen() {isDrawing = true;}
    public static void rememberPosition() {remembered = position;}
    public static void recallPosition() {setPosition(remembered.getX(), remem-
bered.getY());}

    public static Point2D getPosition() {return position;}
    public static void setPosition(double x, double y) {posi-
tion = new Point2D(x, y); gc.moveTo(x, y);}
```

```
}
```

Приложение 6 (класс CommandBox)

```
import java.util.ArrayList;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Circle;
import javafx.scene.text.Font;
import javafx.scene.text.Text;

//TODO: make checkers for when turtle will go step-by-step

public class CommandBox {

    private int actionId = -1; private double actionValue;
    private HBox commandBox; private static VBox commandsBlock; pri-
vate static ArrayList<CommandBox> commandsList = new ArrayList<>();
    private static Font font = Font.font(19);
    private TextField fieldValue; private ComboBox<String> fieldAction;
    private static ArrayList<Action> commandsToDo = new ArrayList<>();
    //#region список доступных действий
    private final ObservableList<String> actions = FXCollections.observableAr-
rayList(
        "Forward",              //0
        "Bacwards",             //1
        "Turn Right (degrees)", //2
        "Turn Left (degrees)",  //3
        "Lift Pen",             //4
        "Lower Pen",            //5
        "Remember Postion",     //6
        "Recall Postion"        //7
    );
    //#endregion

    /** инициализация панели для панелек команд */
    public static void init()
    {
        Text textCommands = new Text("Commands:"); textCom-
mands.setFont(Font.font(22));
        Text textAction = new Text("             Action"); textAc-
tion.setFont(font);
        Text textValue = new Text("Value"); textValue.setFont(font);
```

```
        HBox title = new HBox(83, textAction, textValue);
        commandsBlock = new VBox(8,textCommands,title);
        Main.putBoxesInControls(commandsBlock);
    }
    /** добавляет панельку команды следом за этой */
    public void addNext()
    {
        CommandBox tmp = new CommandBox();
        commandsBlock.getChildren().remove(tmp.commandBox);
        commandsBlock.getChildren().add(commandsBlock.getChildren().in-
dexOf(this.commandBox)+1,tmp.commandBox);
    }
    /** удаляет одну (эту) панельку команды */
    private void remove()
    {
        commandsBlock.getChildren().remove(commandBox);
        commandsList.remove(this);
    }

    /** читает все панельки команд и собирает из них список действий */
    public static void run()
    {
        commandsToDo.clear();
        for (CommandBox box : commandsList) {
            try {commandsToDo.add(box.translate());}
            catch (IncorrectInputException e)
            {
                System.out.println("В командах где-то "+e.getMessage());
                commandsToDo.clear();
                break;
            }
        }
        for (Action action : commandsToDo) {action.act();}
    }
    /** читает панельку команды и возвращает Action */
    public Action translate() throws IncorrectInputException
    {
        if (this.actionId == -1) throw new IncorrectInputExcep-
tion("отсутствует значение действия");
        return new Action(actionId, actionValue);
    }

    /** добавляет в список команд на интерфейсе данный список команд */
    public static void append(ArrayList<Action> appending) {for (Action ac-
tion : appending) {CommandBox tmp = new CommandBox(action);}}
    /** очистить весь список панелек команд */
    public static void clearCommands()
    {
        for (CommandBox box : commandsList) {commandsBlock.getChildren().re-
move(box.commandBox);}
        commandsList.clear();
```

```
        commandsList.add(new CommandBox());
    }

    /** генерирует одну панельку команды */
    public CommandBox()
    {
        //поле для значения. прячется, если выбрано действие, где не нужно значе-
ние
        fieldValue = new Text-
Field(); fieldValue.setPrefSize(62, 20); fieldValue.setPromptText("0");
        fieldValue.textProperty().addListener( (observable, oldValue, newValue) -
>
        {   //validating double, setting value
            if ( newValue.matches("[+-
]?\\d+\\.?(\\d+)?") || newValue.length()==0) fieldValue.setText(newValue);
            else fieldValue.setText(oldValue);

            actionValue = (newValue.length()==0 || newValue.equals("-
")) ? 0 : Double.parseDouble(newValue);
        });

        // drop-down list
        fieldAction = new ComboBox<>(actions); fieldAction.setVisibleRow-
Count(4); fieldAction.setPrefWidth(189);
        fieldAction.valueProperty().addListener((obs, oldValue, newValue) -
> { //hiding or showing value field, setting id
            this.actionId = fieldAction.getSelectionModel().getSelectedIndex();
            if (fieldAction.getSelectionModel().getSelectedIn-
dex() < 4) fieldValue.setVisible(true);
            else fieldValue.setVisible(false);
        });

        // add button
        Button btnAdd = new Button("+"); int r = 16; btnAdd.setShape(new Cir-
cle(r));
        btnAdd.setStyle("-fx-background-radius: 5em; " + "-fx-min-
width:  32px; " + "-fx-min-height: 32px; " + "-fx-max-width:  32px; " + "-fx-max-
height: 32px; " + "-fx-background-color: -fx-body-color;" + "-fx-background-in-
sets: 0px; " + "-fx-padding: 0px;");
        btnAdd.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                addNext();
            }
        });
        // remove button
        Button btnRemove = new Button("-"); btnRemove.setShape(new Circle(r));
        btnRemove.setStyle("-fx-background-radius: 5em; " + "-fx-min-
width:  32px; " + "-fx-min-height: 32px; " + "-fx-max-width:  32px; " + "-fx-max-
height: 32px; " + "-fx-background-color: -fx-body-color;" + "-fx-background-in-
sets: 0px; " + "-fx-padding: 0px;");
```

```
        btnRemove.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                if (commandsList.size()>1) remove();
            }
        });

        commandBox = new HBox(fieldAction, fieldValue, btnAdd, btnRemove);
        commandBox.setSpacing(10);
        commandsBlock.getChildren().add(commandBox);
        commandsList.add(this);

    }
    /** генерирует панельку команды и заполняет ее информацией */
    public CommandBox(Action action)
    {
        this();
        this.actionId = action.getId();
        this.actionValue = action.getValue();
        fieldValue.setText(actionValue+""); fieldAction.getSelectionModel().se-
lect(actionId);
    }

}
```

Приложение 7 (класс IncorrectInputException)

```
public class IncorrectInputException extends Exception {
    public IncorrectInputException(String errorMessage) {
        super(errorMessage);
    }
}
```