Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных

наименование института

ОТЧЕТ  
к лабораторной работе по дисциплине

Объектно-ориентированное программирование

«Разработка приложения с оконным графическим   
пользовательским интерфейсом»

Наименование темы

Выполнил студент группы ИСМб 19–1 Малиновцев И.А.

шифр Фамилия И.О.

Проверила Маланова Т.В.

Фамилия И.О.

Содержание отчета на 18 стр.

Иркутск 2021 г.

Содержание

[1 Введение 7](#_Toc66096319)

[2 Проектирование графического интерфейса пользователя 8](#_Toc66096320)

[3 Проектирование классов 9](#_Toc66096321)

[3.1 Словесное описание классов 9](#_Toc66096322)

[3.2 Класс Main 9](#_Toc66096323)

[3.3 Класс Controller 9](#_Toc66096324)

[3.4 Класс Shape 10](#_Toc66096325)

[3.5 Класс Сoordinate 10](#_Toc66096326)

[4 Таблица тестов 11](#_Toc66096327)

[5 Результаты тестирования 12](#_Toc66096328)

[6 Исходный код 14](#_Toc66096329)

[6.1 Класс Main 14](#_Toc66096330)

[6.2 Класс Controller 14](#_Toc66096331)

[6.3 Класс Shape 17](#_Toc66096332)

[6.4 Класс Сoordinate 21](#_Toc66096480)

[Список литературы 22](#_Toc66096481)

1 Введение

**Цель работы**

Получение общего представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения.

Получение практического опыта работы с базовыми графическими библиотеками java.

**Варианты индивидуальных заданий**

В качестве индивидуального задания была выбрана реализация графического интерфейса для работы с прямоугольной трапецией.

2 Проектирование графического интерфейса пользователя

Графический интерфейс состоит из одного окна. На нем находятся 5 слайдеров для контроля трапеции, один Canvas для отрисовки фигуры и TableView для вывода координат.

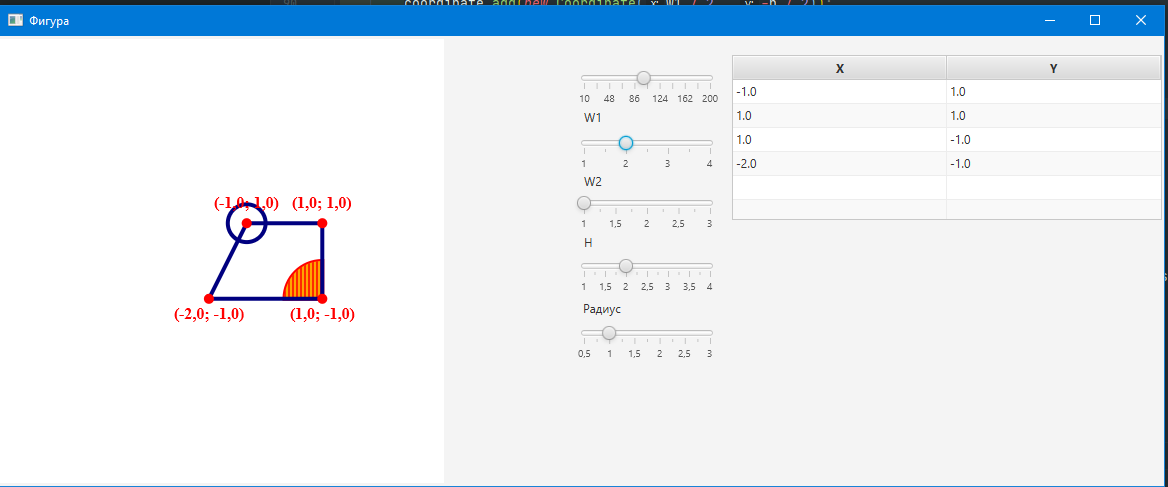


Рисунок 1 – Вид главного окна

3 Проектирование классов

3.1 Словесное описание классов

Программа состоит из 7 классов, один из которых Main – точка входа в программу.

Описание классов:

1. Main – точка входа в программу, запуск графического интерфейса.
2. Controller – логика графического интерфейса окна.
3. Shape – реализация трапеции.
4. Сoordinate – класс, хранящий координаты.

3.2 Класс Main

Таблица 3.1 – Таблица методов класса Main

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Статический** | **Уровень доступа** | **Аргументы** | **Назначение** |
| main | void | Да | public | String[] args | Точка входа в программу, вызов метода launch(args) |
| start | void | Нет | public | Stage primaryStage | Создание главного окна графического интерфейса |

3.3 Класс Controller

Таблица 3.2 – Таблица методов класса Controller

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Статический** | **Уровень доступа** | **Аргументы** | **Назначение** |
| initialize | void | Нет | private | - | Инициализация обработчиков эвентов нажатия кнопок в окне |
| Render | void | Нет | private | - | Отрисовка фигуры в Canvas |
| Upstate | void | Нет | private |  | Отслеживание изменений и запись в объект фигуры |
| onMouseCanvas | void | Нет | public | MouseEvent mouseEvent | Событие, отслеживающее перемещение мыши по Canvas |
| onMouseClick | void | Нет | public | MouseEvent mouseEvent | Событие, отслеживающее первый клик мыши по Canvas |

3.4 Класс Shape

Таблица 3.3 – Таблица методов класса Shape

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Статический** | **Уровень доступа** | **Аргументы** | **Назначение** |
| setСoordinates | void | нет | public | - | Запись координат фигуры |
| Render | void | нет | public | GraphicsContext cxt, Canvas canvas, Image image | Метод для отрисовки фигуры |
| RenderСoordinates | void | нет | private | GraphicsContext cxt, Affine transform, Сoordinate coor, double zoom, String text | Метод для вывода координат точек на |

3.5 Класс Сoordinate

Таблица 3.4 – Таблица методов класса Сoordinate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Имя** | **Тип** | **Статический** | **Уровень доступа** | **Аргументы** | **Назначение** |
| toString | String | нет | public | - | Вывод информации о классе |

4 Таблица тестов

Таблица 4.1 – таблица тестов

|  |  |  |
| --- | --- | --- |
| **№** | **Действие** | **Результат** |
| 1 | Изменение значения sidZoom | Изменение приближения фигуры |
| 2 | Изменение значения sidH | Изменение высоты |
| 3 | Изменение значения sidW1 | Изменение ширины верхней стороны |
| 4 | Изменение значения sidW2 | Изменение размеров нижней стороны |
| 5 | Изменение значения sidR | Изменение радиуса у окружностей |
| 6 | Клик и перемещение мыши по mainCanvas | Передвижение фигуры |

5 Результаты тестирования

Результаты тестирования представлены на рисунках 3 – 8.

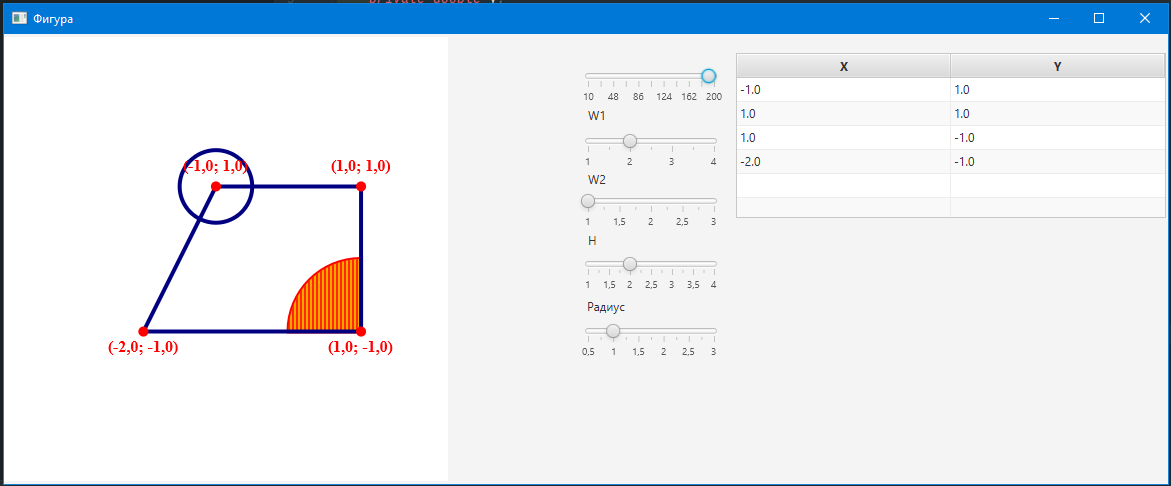


Рисунок 2 – Результат выполнения теста 1

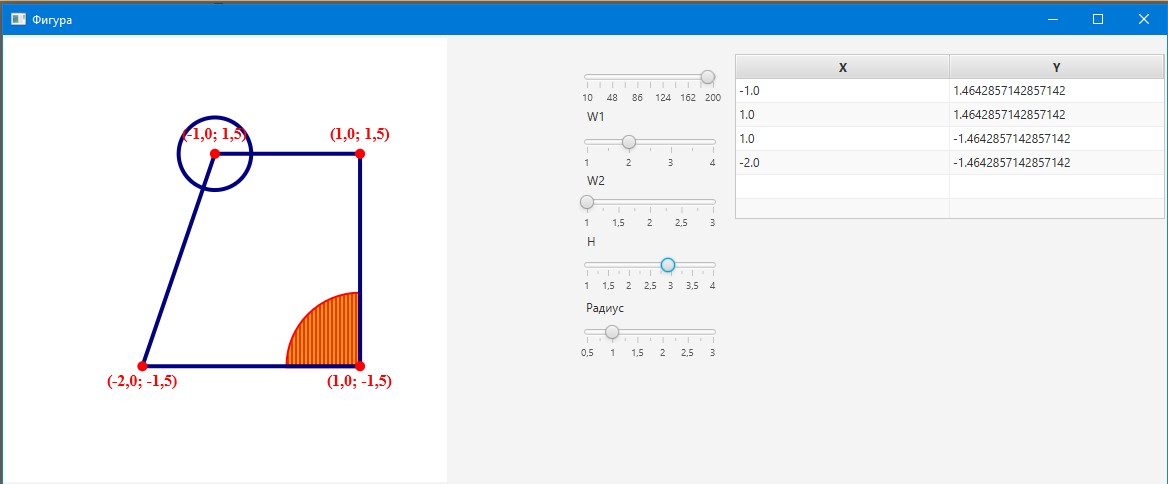


Рисунок 3 – Результат выполнения теста 2

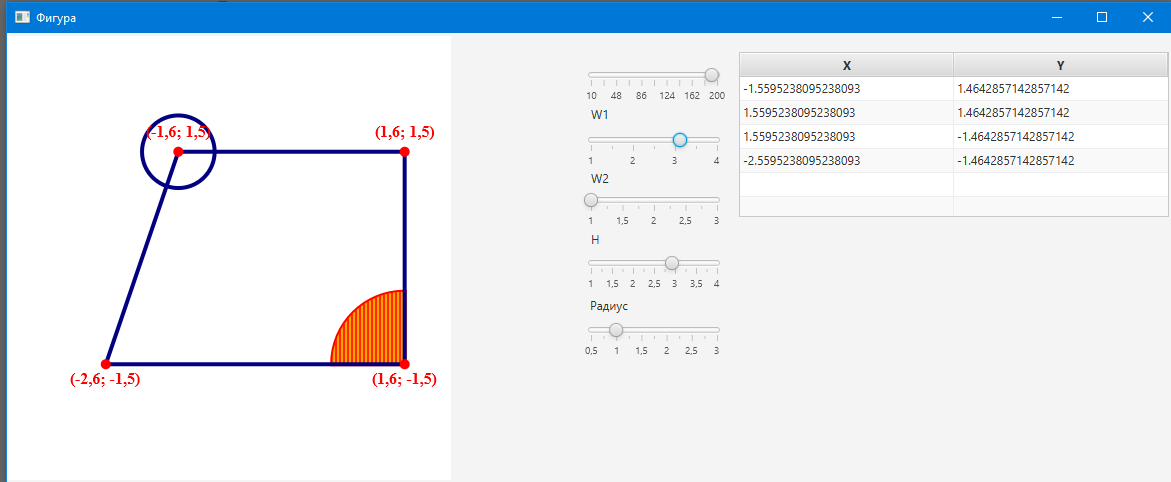


Рисунок 4 – Результат выполнения теста 3

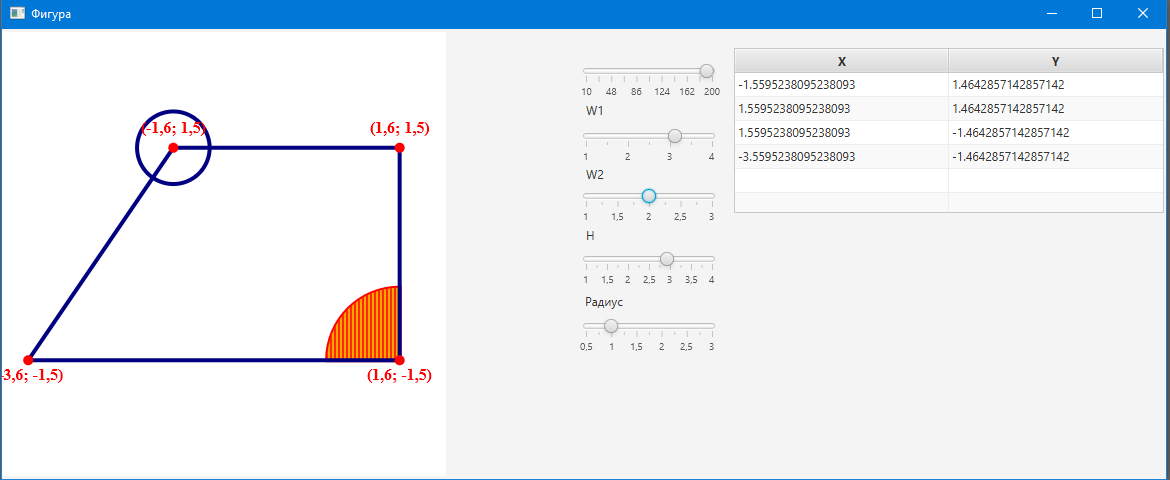


Рисунок 5 – Результат выполнения теста 4

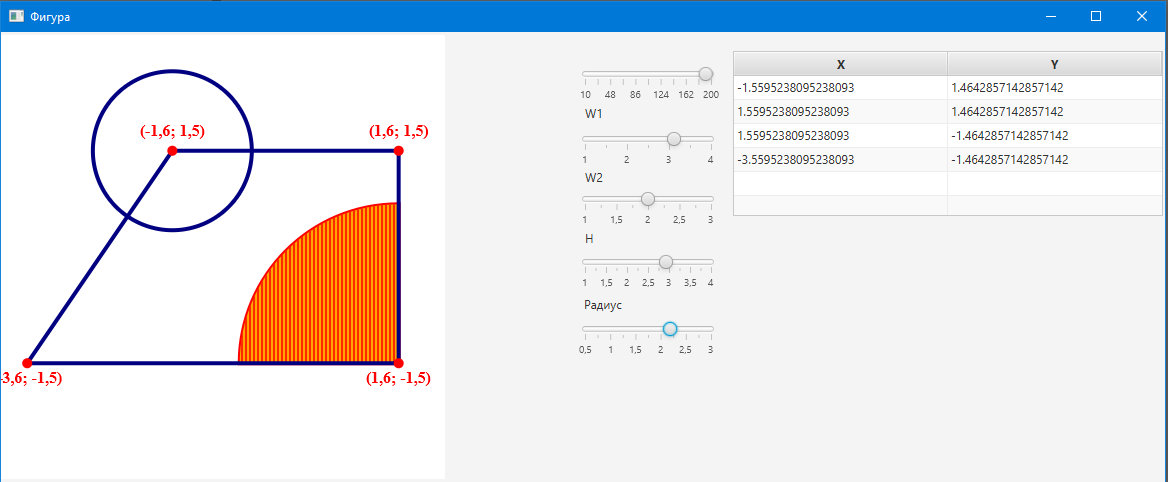


Рисунок 6 – Результат выполнения теста 5

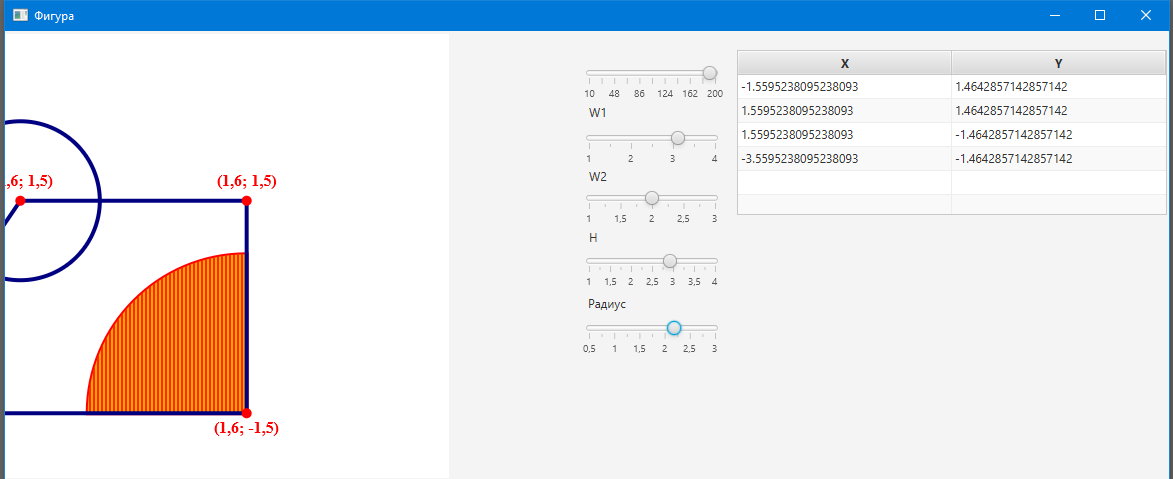


Рисунок 7 – Результат выполнения теста 6

6 Исходный код

6.1 Класс Main

public class Main extends Application {

@Override

public void start(Stage primaryStage) throws Exception{

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

primaryStage.setTitle("Фигура");

primaryStage.setScene(new Scene(root));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

6.2 Класс Controller

public class Controller implements Initializable {

public Slider sidZoom;

public TableView<Сoordinate> tblOutput;

public TableColumn<Сoordinate, Double> columnX;

public TableColumn<Сoordinate, Double> columnY;

@FXML

Canvas mainCanvas;

@FXML

Slider sidW1;

@FXML

Slider sidW2;

@FXML

Slider sidH;

@FXML

Slider sidR;

Shape shape;

GraphicsContext cxt;

private ImagePattern imagePattern;

private Image image;

private double offsetX = 0;

private double offsetY = 0;

@Override

public void initialize(URL url, ResourceBundle resourceBundle) {

shape = new Shape();

cxt = mainCanvas.getGraphicsContext2D();

image = new Image("pattern1.png");

//imagePattern = new ImagePattern(image, 0, 0, 4, 1, true);

Upstate();

Render();

sidW1.valueProperty().addListener((ov, old\_val, new\_val) -> {

Upstate();

Render();

});

sidW2.valueProperty().addListener((ov, old\_val, new\_val) -> {

Upstate();

Render();

});

sidH.valueProperty().addListener((ov, old\_val, new\_val) -> {

Upstate();

Render();

});

sidR.valueProperty().addListener((ov, old\_val, new\_val) -> {

Upstate();

Render();

});

sidZoom.valueProperty().addListener((ov, old\_val, new\_val) -> {

Upstate();

Render();

});

}

void Render(){

cxt.setFill(Color.WHITE);

cxt.fillRect(0,0,mainCanvas.getHeight(), mainCanvas.getHeight());

shape.Render(cxt, mainCanvas, image);

}

void Upstate(){

if(sidH.getValue() < sidR.getValue()) {

sidR.setValue(sidH.getValue());

}

if(sidW1.getValue() + sidW2.getValue() < sidR.getValue()){

sidR.setValue(sidW1.getValue() + sidW2.getValue());

}

shape.setW1(sidW1.getValue());

shape.setH(sidH.getValue());

shape.setR(sidR.getValue());

shape.setW2(sidW2.getValue());

shape.setZoom(sidZoom.getValue());

shape.setOffsetX(offsetX);

shape.setOffsetY(offsetY);

shape.setСoordinates();

columnX.setCellValueFactory(new PropertyValueFactory<Сoordinate, Double>("x"));

columnY.setCellValueFactory(new PropertyValueFactory<Сoordinate, Double>("y"));

tblOutput.setItems(shape.getCoordinate());

}

public void onMouseCanvas(MouseEvent mouseEvent) {

offsetX += mouseEvent.getX() - pressedX;

offsetY += mouseEvent.getY() - pressedY;

pressedX = mouseEvent.getX();

pressedY = mouseEvent.getY();

Upstate();

Render();

}

double pressedX;

double pressedY;

public void onMouseClick(MouseEvent mouseEvent) {

pressedX = mouseEvent.getX();

pressedY = mouseEvent.getY();

}

}

6.3 Класс Shape

public class Shape {

private double w1;

private double w2;

private double h;

private double r;

private double zoom;

private double offsetX = 0;

private double offsetY = 0;

private ObservableList<Сoordinate> coordinate;

public ObservableList<Сoordinate> getCoordinate() {

return coordinate;

}

public void setOffsetX(double offsetX) {

this.offsetX = offsetX;

}

public double getOffsetY() {

return offsetY;

}

public void setOffsetY(double offsetY) {

this.offsetY = offsetY;

}

public double getZoom() {

return zoom;

}

public void setZoom(double zoom) {

this.zoom = zoom;

}

public double getW1() {

return w1;

}

public void setW1(double w1) {

this.w1 = w1;

}

public double getH() {

return h;

}

public void setH(double h) {

this.h = h;

}

public double getW2() {

return w2;

}

public void setW2(double w2) {

this.w2 = w2;

}

public double getR() {

return r;

}

public void setR(double r) {

this.r = r;

}

public void setСoordinates(){

coordinate = FXCollections.observableArrayList();

coordinate.add(new Сoordinate(-w1 / 2, h / 2));

coordinate.add(new Сoordinate(w1 / 2, h / 2));

coordinate.add(new Сoordinate(w1 / 2, -h / 2));

coordinate.add(new Сoordinate(-w1 / 2 - w2, -h / 2));

}

void Render(GraphicsContext cxt, Canvas canvas, Image image) {

setСoordinates();

//Получить dpi экрана

double dpi = Toolkit.getDefaultToolkit().getScreenResolution();

double zoom = dpi / 2.54 \* getZoom() / 100;

cxt.save(); //Сохранить состояние матрицы

//Матрица преобразования

Affine transform = cxt.getTransform();

//Выровнять по центру

transform.appendTranslation(canvas.getWidth() /2 + offsetX, canvas.getHeight() / 2 + offsetY);

//Увеличили масштаб

transform.appendScale(zoom, -zoom);

//Сохранить изменения

cxt.setTransform(transform);

//Сузить линию

cxt.setLineWidth(4. / zoom);

cxt.setStroke(Color.RED);

cxt.strokeArc(

w1 / 2 - r,

-h / 2 - r,

r \* 2,

r \* 2,

-90,

-90,

ArcType.ROUND

);

Stream.of(

Color.ORANGE,

new ImagePattern(image, 0, 0, 4. / zoom, 1. / zoom, false)

).forEach(paint ->{

cxt.setFill(paint);

cxt.fillArc(

w1 / 2 - r,

-h / 2 - r,

r \* 2,

r \* 2,

-90,

-90,

ArcType.ROUND

);

});

cxt.setFill(new ImagePattern(image, 0, 0, 4. / zoom, 1. / zoom, false));

cxt.fillArc(

w1 / 2 - r,

-h / 2 - r,

r \* 2,

r \* 2,

-90,

-90,

ArcType.ROUND

);

cxt.setStroke(Color.NAVY);

cxt.strokePolygon(

new double[]{-w1 / 2, w1 / 2, w1 / 2, -w1 / 2 - w2},

new double[]{h / 2, h / 2, -h / 2, -h / 2},

4);

cxt.strokeOval(

-w1 /2 - r / 2,

h/2 - r / 2,

r,

r

);

//Текст

for (var coor: coordinate) {

RenderСoordinates(cxt, transform, coor, zoom, String.format("(%.1f; %.1f)", coor.getX(), coor.getY()));

}

cxt.restore(); //Вызвать обратно

}

private void RenderСoordinates(GraphicsContext cxt, Affine transform, Сoordinate coor, double zoom, String text){

cxt.save();

transform = cxt.getTransform();

double x = coor.getX();

double y = coor.getY();

transform.appendTranslation(x, y); //Сдвинуть к координатам рисовку

transform.appendScale(1. / zoom, 1. / -zoom); //Заскуйлить в нужный формат

cxt.setTransform(transform);

cxt.setFill(Color.RED);

cxt.setFont(Font.font("Times New Roman", FontWeight.BOLD, 16));

cxt.setTextAlign(TextAlignment.CENTER);

if(y > 0){

cxt.fillText(text, 0, -15);

}

else {

cxt.fillText(text, 0, 20);

}

cxt.fillOval(-5, -5, 10, 10);

cxt.restore();

}

}

6.4 Класс Сoordinate

public class Сoordinate {

private double x;

private double y;

public Сoordinate(double x, double y) {

this.x = x;

this.y = y;

}

public double getX() {

return x;

}

public void setX(double x) {

this.x = x;

}

public double getY() {

return y;

}

public void setY(double y) {

this.y = y;

}

@Override

public String toString() {

return "(" +

"x=" + x +

"; y=" + y +

')';

}

}

Список литературы

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: http://developer.alexanderklimov.ru/android/java/java.php

2. API Specification for the Java 7 SE. [официальный сайт] URL: http://docs.oracle.com/javase/7/docs/api/

3. The Java Tutorials. SE [электронный ресурс], URL: http://docs.oracle.com/javase/tutorial/index.html

4. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010.

5. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014.