Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «20» ноября 2023 г. по «09» декабря 2023 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 20-1  Якименко Илья Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2023

Содержание

Содержание

[Введение 3](#_Toc152960155)

[1 Решение задач 4](#_Toc152960156)

[1.1 Задача 4](#_Toc152960157)

[1.2. Задача 2 7](#_Toc152960158)

[1.3 Задача 3 12](#_Toc152960159)

[1.4 Задача 4 17](#_Toc152960160)

[1.5 Задача 5 22](#_Toc152960161)

[1.6 Задача 6 25](#_Toc152960162)

[1.7 Задача 7 34](#_Toc152960163)

[1.8 Задача 8 38](#_Toc152960164)

[1.9 Задача 9 41](#_Toc152960165)

[2 Разработка АИС 65](#_Toc152960166)

[2.1 Техническое задание 66](#_Toc152960167)

[2.1.1 Краткая характеристика области применения 66](#_Toc152960168)

[2.1.2 Основания для разработки программного изделия 67](#_Toc152960169)

[2.1.3 Функциональное и эксплуатационное назначение 67](#_Toc152960170)

[2.1.4 Требования к программе 67](#_Toc152960171)

[3 Разработка мобильного приложения 69](#_Toc152960172)

[4 Разработка сайта 71](#_Toc152960173)

[5 Руководство пользователя 73](#_Toc152960174)

[Заключение 74](#_Toc152960175)

[Список использованной литературы 75](#_Toc152960176)

[Приложение А. Руководство пользователя 76](#_Toc152960177)

[Приложение Б. Листинг кода мобильного приложения 78](#_Toc152960178)

# Введение

В современном мире с быстро развивающимися технологиями, где очень быстро меняются подходы, методы и инструменты работы с различной информацией, специалисты в области информационных систем и программирования сталкиваются с постоянной необходимостью совершенствования своих компетенций и навыков. Практика по разработке модулей программного обеспечения для компьютерных систем, проводимая в рамках программы обучения по направлению 09.02.07 "Информационные системы и программирование", предоставляет студентам возможность использования и совершенствования своих умений и познаний, необходимых при работе с созданием и поддерживанием программ, используемых в работе с информацией.

Цели и задачи практики направлены на получение практического опыта при работе с заданиями практики, а также освоение методов проектирования, кодирования и тестирования программных модулей, участвуя в полном цикле разработки. Кроме того, практика дает возможность работать с актуальными технологиями и инструментами, широко используемыми в сфере разработки программного обеспечения в наше время.

Ссылка на GitHub - https://github.com/YakimenkoIlya/PractuceGit

# 1 Решение задач

* 1. Задача

Определить принадлежность некоторого значения k интервалам (n, m], [n, m), (n, m), [n, m].

Таблица 1 – Идентификаторы в задаче 1

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| n | double | Входные данные |
| m | double | Входные данные |
| k | double | Входные данные |

Таблица 2 – Компоненты в задаче 1

|  |  |  |
| --- | --- | --- |
| Наименование компонента | Класс | Назначение |
| scanner | Scanner | Входные данные |

Блок – схема:

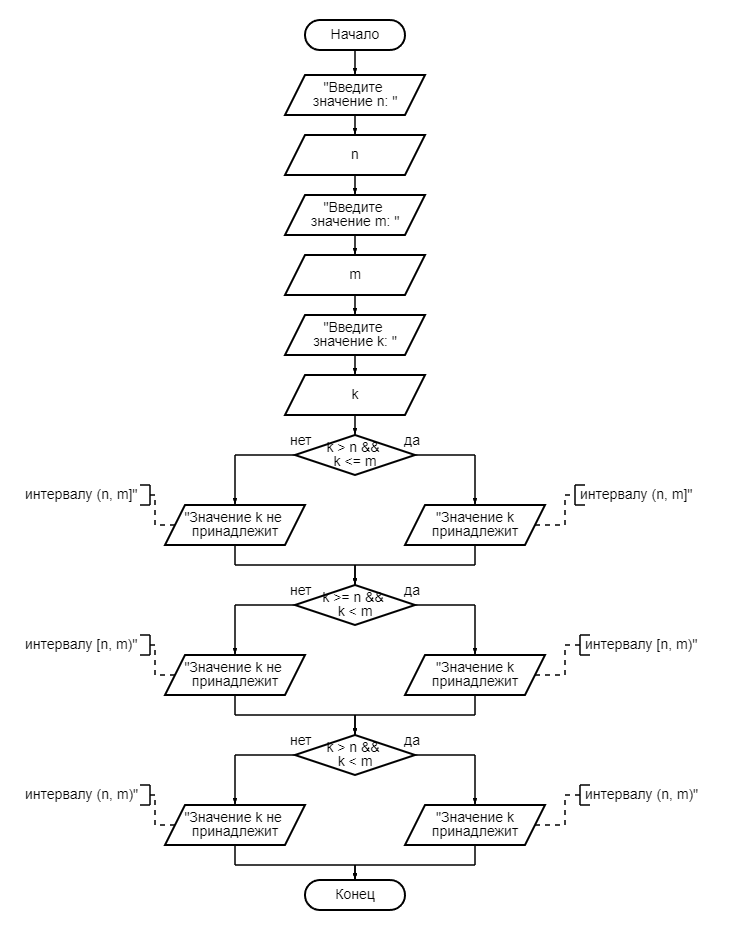


Рисунок 1 – Блок – схема класса в задаче 1

Код программы:

package org.example;

import java.util.Scanner;

public class Ex1\_page50 {

public static void exercise(){

Scanner scanner = new Scanner(System.in);

System.out.print("Введите значение n: ");

double n = scanner.nextDouble();

System.out.print("Введите значение m: ");

double m = scanner.nextDouble();

System.out.print("Введите значение k: ");

double k = scanner.nextDouble();

if (k > n && k <= m) {

System.out.println("Значение k принадлежит интервалу (n, m]");

} else {

System.out.println("Значение k не принадлежит интервалу (n, m]");

}

if (k >= n && k < m) {

System.out.println("Значение k принадлежит интервалу [n, m)");

} else {

System.out.println("Значение k не принадлежит интервалу [n, m)");

}

if (k > n && k < m) {

System.out.println("Значение k принадлежит интервалу (n, m)");

} else {

System.out.println("Значение k не принадлежит интервалу (n, m)");

}

if (k >= n && k <= m) {

System.out.println("Значение k принадлежит интервалу [n, m]");

} else {

System.out.println("Значение k не принадлежит интервалу [n, m]");

}

scanner.close();

}

}

Скриншоты результата:

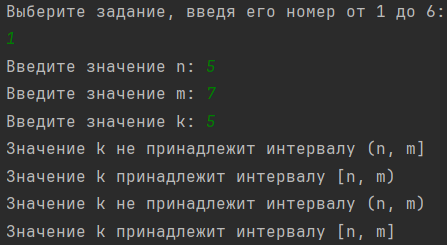


Рисунок 2 – Результат задачи 1

## 1.2. Задача 2

Определить класс Полином с коэффициентами типа Комплексное число.

Объявить массив/список/множество из m полиномов и определить сумму

полиномов массива.

Диаграмма классов:

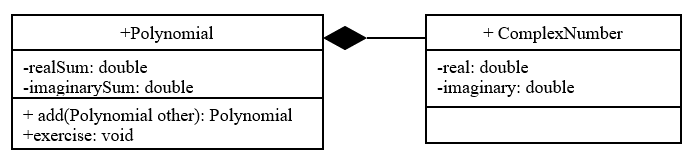


Рисунок 3 – Диаграмма классов в задаче 2

Код программы:

package org.example;

import java.util.ArrayList;

import java.util.List;

public class Polynomial {

private List<ComplexNumber> coefficients;

private static double realSum = 0;

private static double imaginarySum = 0;

public Polynomial(List<ComplexNumber> coefficients) {

this.coefficients = coefficients;

}

public Polynomial() {

this.coefficients = new ArrayList<>();

}

public Polynomial add(Polynomial other) {

int maxSize = Math.max(this.coefficients.size(), other.coefficients.size());

List<ComplexNumber> resultCoefficients = new ArrayList<>();

for (int i = 0; i < maxSize; i++) {

ComplexNumber coef1 = (i < this.coefficients.size()) ? this.coefficients.get(i) : new ComplexNumber(0, 0);

ComplexNumber coef2 = (i < other.coefficients.size()) ? other.coefficients.get(i) : new ComplexNumber(0, 0);

resultCoefficients.add(coef1.add(coef2));

}

return new Polynomial(resultCoefficients);

}

public List<ComplexNumber> getCoefficients() {

return coefficients;

}

public void setCoefficients(List<ComplexNumber> coefficients) {

this.coefficients = coefficients;

}

public static void exercise() {

List<Polynomial> polynomials = new ArrayList<>();

Polynomial polynomial1 = new Polynomial();

Polynomial polynomial2 = new Polynomial();

List<ComplexNumber> coefficients1 = new ArrayList<>();

coefficients1.add(new ComplexNumber(2, 3));

coefficients1.add(new ComplexNumber(-1, 2));

polynomial1.setCoefficients(coefficients1);

List<ComplexNumber> coefficients2 = new ArrayList<>();

coefficients2.add(new ComplexNumber(1, -2));

coefficients2.add(new ComplexNumber(3, 1));

coefficients2.add(new ComplexNumber(0, 4));

polynomial2.setCoefficients(coefficients2);

polynomials.add(polynomial1);

polynomials.add(polynomial2);

Polynomial sum = new Polynomial();

for (Polynomial polynomial : polynomials) {

sum = sum.add(polynomial);

}

List<ComplexNumber> sumCoefficients = sum.getCoefficients();

System.out.println("Полиномы:");

for (ComplexNumber coefficient : sumCoefficients) {

System.out.println(coefficient.getReal() + " + " + coefficient.getImaginary() + "i");

}

System.out.println("\nСумма всех полиномов:");

for (ComplexNumber coefficient : sumCoefficients) {

realSum += coefficient.getReal();

imaginarySum += coefficient.getImaginary();

}

System.out.println(realSum + " + " + imaginarySum + "i");

}

}

class ComplexNumber {

private double real;

private double imaginary;

public ComplexNumber(double real, double imaginary) {

this.real = real;

this.imaginary = imaginary;

}

public double getReal() {

return real;

}

public void setReal(double real) {

this.real = real;

}

public double getImaginary() {

return imaginary;

}

public void setImaginary(double imaginary) {

this.imaginary = imaginary;

}

public ComplexNumber add(ComplexNumber other) {

double newReal = this.real + other.real;

double newImaginary = this.imaginary + other.imaginary;

return new ComplexNumber(newReal, newImaginary);

}

}

Скриншот результата:

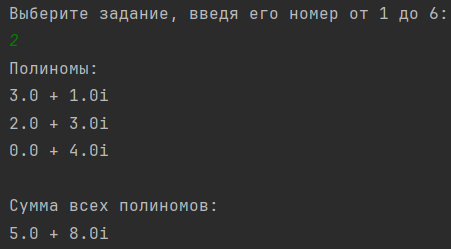


Рисунок 4 – Результат задачи 2

## 1.3 Задача 3

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Наследование применять только в тех заданиях, в которых это логически обосновано. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(),hashCode(), toString().

Создать объект класса Текст, используя классы Предложение, Слово.

Методы: дополнить текст, вывести на консоль текст, заголовок текста

Диаграмма классов:

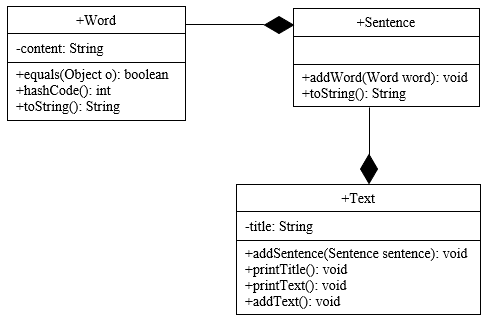


Рисунок 5 – Диаграмма классов в задаче 3

Код программы:

package org.example;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Word {

private String content;

public Word(String content) {

this.content = content;

}

public String getContent() {

return content;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Word word = (Word) o;

return content.equals(word.content);

}

@Override

public int hashCode() {

return content.hashCode();

}

@Override

public String toString() {

return content;

}

}

class Sentence {

private List<Word> words;

public Sentence() {

this.words = new ArrayList<>();

}

public void addWord(Word word) {

words.add(word);

}

@Override

public String toString() {

StringBuilder sentenceString = new StringBuilder();

for (Word word : words) {

sentenceString.append(word.getContent()).append(" ");

}

return sentenceString.toString().trim();

}

}

class Text {

private String title;

private List<Sentence> sentences;

public Text(String title) {

this.title = title;

this.sentences = new ArrayList<>();

}

public void addSentence(Sentence sentence) {

sentences.add(sentence);

}

public void printTitle() {

System.out.println("\n" + title + "\n");

}

public void printText() {

for (Sentence sentence : sentences) {

System.out.println(sentence);

}

}

public static void addText() {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите заголовок текста: ");

String title = scanner.nextLine();

Text userText = new Text(title);

while (true) {

Sentence sentence = new Sentence();

System.out.println("Введите предложение или \"#выход\" для окончания ввода: ");

String input = scanner.nextLine();

if (input.equalsIgnoreCase("#выход")) {

break;

}

String[] words = input.split(" ");

for (String word : words) {

sentence.addWord(new Word(word));

}

userText.addSentence(sentence);

}

System.out.println("Вывод заголовка и текста: ");

userText.printTitle();

userText.printText();

scanner.close();

}

}

Скриншот результата:

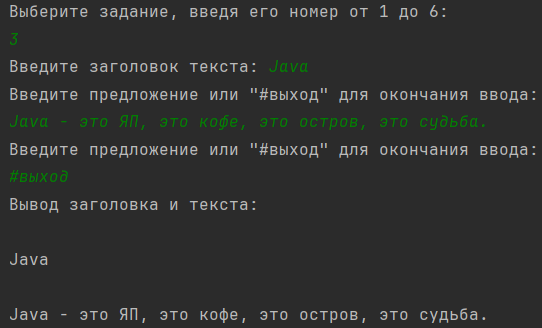


Рисунок 6 – Результат задачи 3

## 1.4 Задача 4

Создать класс Catalog с внутренним классом, с помощью объектов которого

можно хранить информацию об истории выдач книги читателям.

Таблица 3 – Компоненты в задаче 4

|  |  |  |
| --- | --- | --- |
| Наименование компонента | Класс | Назначение |
| catalog | Catalog | Вспомогательные данные |
| book | Catalog.Book | Вспомогательные данные |
| history | Catalog.BookHistory | Вспомогательные данные |

Блок – схема:

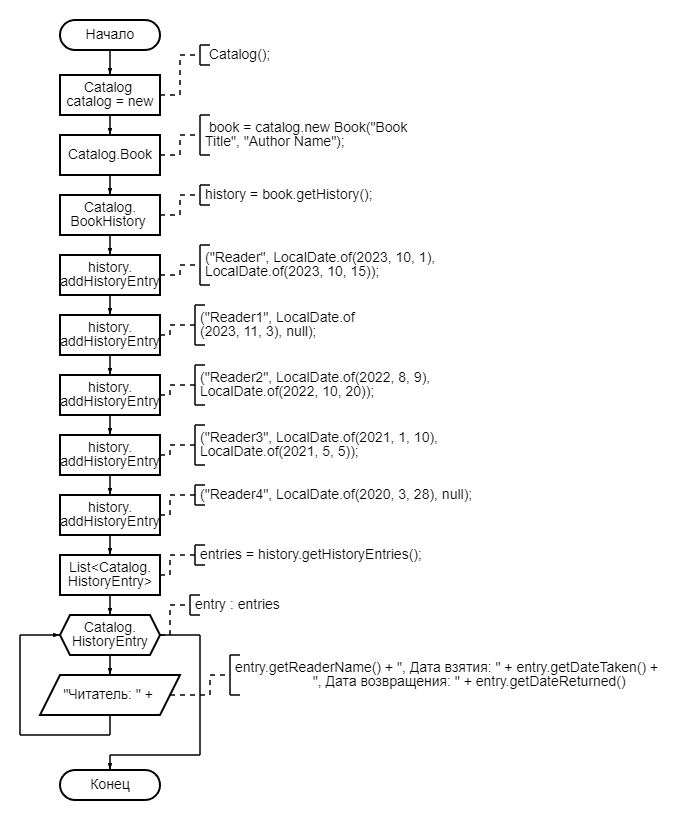


Рисунок 7 – Блок - схема класса Catalog

Код программы:

package org.example;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

public class Catalog {

private List<Book> books;

public Catalog() {

this.books = new ArrayList<>();

}

public void addBook(Book book) {

books.add(book);

}

public List<Book> getBooks() {

return books;

}

public class Book {

private String title;

private String author;

private BookHistory history;

public Book(String title, String author) {

this.title = title;

this.author = author;

this.history = new BookHistory();

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public BookHistory getHistory() {

return history;

}

}

public class BookHistory {

private List<HistoryEntry> historyEntries;

public BookHistory() {

this.historyEntries = new ArrayList<>();

}

public void addHistoryEntry(String readerName, LocalDate dateTaken, LocalDate dateReturned) {

historyEntries.add(new HistoryEntry(readerName, dateTaken, dateReturned));

}

public List<HistoryEntry> getHistoryEntries() {

return historyEntries;

}

}

public class HistoryEntry {

private String readerName;

private LocalDate dateTaken;

private LocalDate dateReturned;

public HistoryEntry(String readerName, LocalDate dateTaken, LocalDate dateReturned) {

this.readerName = readerName;

this.dateTaken = dateTaken;

this.dateReturned = dateReturned;

}

public String getReaderName() {

return readerName;

}

public LocalDate getDateTaken() {

return dateTaken;

}

public LocalDate getDateReturned() {

return dateReturned;

}

}

public static void exercise() {

Catalog catalog = new Catalog();

Catalog.Book book = catalog.new Book("Book Title", "Author Name");

Catalog.BookHistory history = book.getHistory();

history.addHistoryEntry("Reader", LocalDate.of(2023, 10, 1), LocalDate.of(2023, 10, 15));

history.addHistoryEntry("Reader1", LocalDate.of(2023, 11, 3), null);

history.addHistoryEntry("Reader2", LocalDate.of(2022, 8, 9), LocalDate.of(2022, 10, 20));

history.addHistoryEntry("Reader3", LocalDate.of(2021, 1, 10), LocalDate.of(2021, 5, 5));

history.addHistoryEntry("Reader4", LocalDate.of(2020, 3, 28), null);

List<Catalog.HistoryEntry> entries = history.getHistoryEntries();

for (Catalog.HistoryEntry entry : entries) {

System.out.println("Reader: " + entry.getReaderName() + ", Taken: " + entry.getDateTaken() +

", Returned: " + entry.getDateReturned());

}

}

}

Скриншот результата:

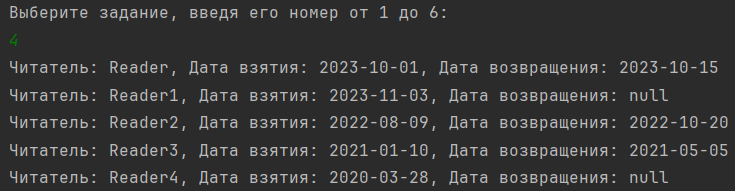


Рисунок 8 – Результат задачи 4

## 1.5 Задача 5

Из текста удалить все лишние пробелы, если они разделяют два различных знака препинания и если рядом с ними находится еще один пробел.

Таблица 4 – Идентификаторы в коде задачи 5

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| text | String | Входные данные |
| cleanedText | String | Выходные данные |

Таблица 5 – Компоненты в задаче 5

|  |  |  |
| --- | --- | --- |
| Наименование компонента | Класс | Назначение |
| scanner | Scanner | Входные данные |

Блок – схема:

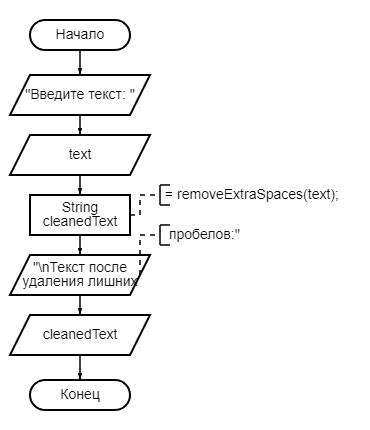


Рисунок 9 – Блок – схема задачи 5

Код программы:

package org.example;

import java.util.Scanner;

public class Ex5\_page194 {

public static void exercise() {

Scanner scanner = new Scanner(System.in);

System.out.println("Введите текст:");

String text = scanner.nextLine();

String cleanedText = removeExtraSpaces(text);

System.out.println("\nТекст после удаления лишних пробелов:");

System.out.println(cleanedText);

scanner.close();

}

public static String removeExtraSpaces(String text) {

return text.replaceAll("(\\p{P})\\s+(\\p{P})", "$1 $2").replaceAll("(\\p{P})\\s+", "$1 ");

}

}

Скриншот результата:

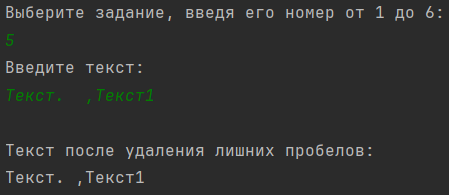


Рисунок 10 – Результат задачи 5

## 1.6 Задача 6

Выполнить задания на основе задачи № 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Таблица 6 – Идентификаторы в задаче 6

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| title | String | Входные данные |
| input | String | Входные данные |

Таблица 7 – Компоненты в задаче 6

|  |  |  |
| --- | --- | --- |
| Наименование компонента | Класс | Назначение |
| userText | TextEx6 | Выходные данные |
| sentence | SentenceEx6 | Вспомогательные данные |
| scanner | Scanner | Входные данные |

Блок – схема:

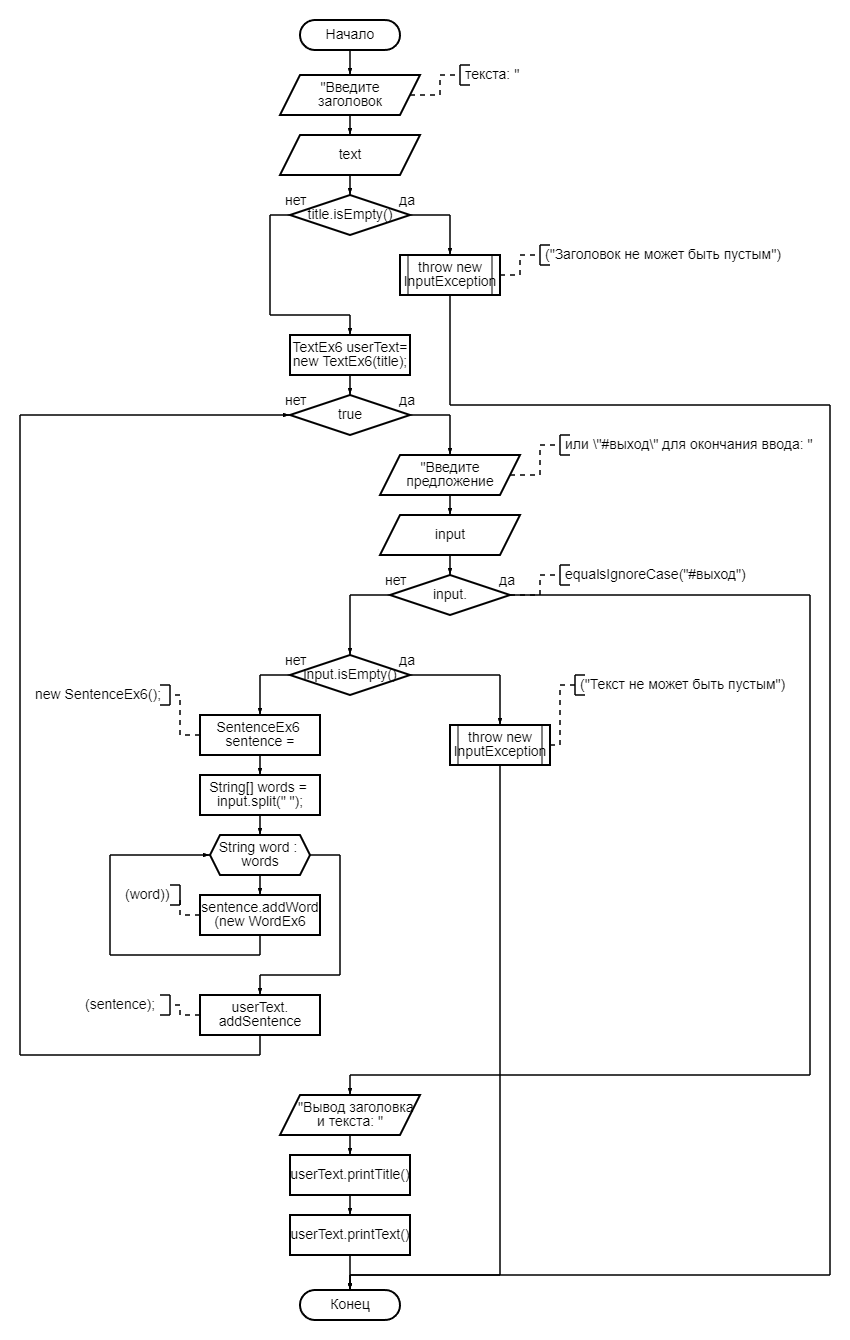


Рисунок 11 – Блок схема работы классов в задаче 6

Код программы:

package org.example;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class WordEx6 {

private String content;

public WordEx6(String content) {

this.content = content;

}

public String getContent() {

return content;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

WordEx6 word = (WordEx6) o;

return content.equals(word.content);

}

@Override

public int hashCode() {

return content.hashCode();

}

@Override

public String toString() {

return content;

}

}

class SentenceEx6 {

private List<WordEx6> words;

public SentenceEx6() {

this.words = new ArrayList<>();

}

public void addWord(WordEx6 word) {

words.add(word);

}

@Override

public String toString() {

StringBuilder sentenceString = new StringBuilder();

for (WordEx6 word : words) {

sentenceString.append(word.getContent()).append(" ");

}

return sentenceString.toString().trim();

}

}

class TextEx6 {

private String title;

private List<SentenceEx6> sentences;

public TextEx6(String title) {

this.title = title;

this.sentences = new ArrayList<>();

}

public void addSentence(SentenceEx6 sentence) {

sentences.add(sentence);

}

public void printTitle() {

System.out.println("\n" + title + "\n");

}

public void printText() {

for (SentenceEx6 sentence : sentences) {

System.out.println(sentence);

}

}

public static void addText() throws InputException {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите заголовок текста: ");

String title = scanner.nextLine();

if (title.isEmpty()) {

throw new InputException("Заголовок не может быть пустым");

}

TextEx6 userText = new TextEx6(title);

while (true) {

System.out.println("Введите предложение или \"#выход\" для окончания ввода: ");

String input = scanner.nextLine();

if (input.equalsIgnoreCase("#выход")) {

break;

}

if (input.isEmpty()) {

throw new InputException("Текст не может быть пустым");

}

SentenceEx6 sentence = new SentenceEx6();

String[] words = input.split(" ");

for (String word : words) {

sentence.addWord(new WordEx6(word));

}

userText.addSentence(sentence);

}

System.out.println("Вывод заголовка и текста: ");

userText.printTitle();

userText.printText();

scanner.close();

}

}

class InputException extends Exception{

public InputException(String message){

super(message);

}

}

Скриншоты результата:

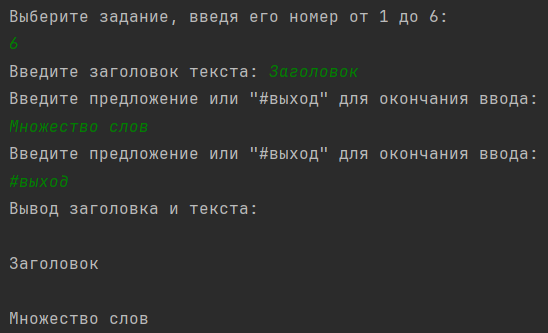


Рисунок 12 – Вывод в консоль при заполнении заголовка и текста

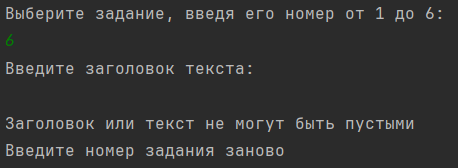


Рисунок 13 – Вывод программы при не заполнении заголовка или текста

Класс Main (Меню программы):

package org.example;

import java.util.InputMismatchException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

System.out.println("Выберите задание, введя его номер от 1 до 6:");

enteringExerciseNumber();

}

private static void enteringExerciseNumber() {

Scanner scanner = new Scanner(System.in);

int exerciseNumber = 1;

try {

exerciseNumber = scanner.nextInt();

}catch (InputMismatchException e){

System.out.println("Некорректный тип данных, попробуйте снова:");

enteringExerciseNumber();

}

try {

switch (exerciseNumber) {

case 1:

Ex1\_page50.exercise();

break;

case 2:

Polynomial.exercise();

break;

case 3:

Text.addText();

break;

case 4:

Catalog.exercise();

break;

case 5:

Ex5\_page194.exercise();

break;

case 6:

TextEx6.addText();

break;

default: {

System.out.println("Число неправильное, попробуйте заново");

enteringExerciseNumber();

}

}

}catch (InputException e){

System.out.println("\nЗаголовок или текст не могут быть пустыми\n" +

"Введите номер задания заново");

enteringExerciseNumber();

}

scanner.close();

}

}

## 1.7 Задача 7

Составить программу получения на экране рисунков, изображенных в табл. 2.

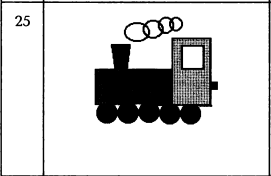


Рисунок 14 – Рисунок по задаче 7

Код программы:

package com.example.practice\_one;

import javafx.scene.Scene;

import javafx.scene.layout.Pane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Ellipse;

import javafx.scene.shape.Polygon;

import javafx.scene.shape.Rectangle;

import javafx.stage.Stage;

public class Page114 {

public static void show(){

Stage stage = new Stage();

stage.setTitle("Паровоз");

Pane root = new Pane();

Scene scene = new Scene(root, 400, 300);

for (int i = 0; i < 5; i++) {

Circle wheel = new Circle(70 + i \* 60, 220, 30);

wheel.setFill(Color.BLACK);

root.getChildren().add(wheel);

}

Rectangle rectangle = new Rectangle(35, 100, 225, 100);

rectangle.setFill(Color.BLACK);

root.getChildren().add(rectangle);

Rectangle rectangle1 = new Rectangle();

rectangle1.setX(250);

rectangle1.setY(10);

rectangle1.setWidth(100);

rectangle1.setHeight(190);

rectangle1.setStroke(Color.BLACK);

rectangle1.setFill(Color.GRAY);

root.getChildren().add(rectangle1);

Rectangle rectangle2 = new Rectangle();

rectangle2.setX(270);

rectangle2.setY(30);

rectangle2.setWidth(60);

rectangle2.setHeight(70);

rectangle2.setFill(Color.WHITE);

rectangle2.setStroke(Color.BLACK);

root.getChildren().add(rectangle2);

Rectangle rectangle3 = new Rectangle();

rectangle3.setX(350);

rectangle3.setY(135);

rectangle3.setWidth(20);

rectangle3.setHeight(20);

rectangle3.setFill(Color.BLACK);

root.getChildren().add(rectangle3);

Ellipse ellipse = new Ellipse();

ellipse.setCenterX(110);

ellipse.setCenterY(35);

ellipse.setRadiusX(19);

ellipse.setRadiusY(12);

ellipse.setFill(Color.TRANSPARENT);

ellipse.setStroke(Color.BLACK);

root.getChildren().add(ellipse);

Ellipse ellipse1 = new Ellipse();

ellipse1.setCenterX(135);

ellipse1.setCenterY(30);

ellipse1.setRadiusX(13);

ellipse1.setRadiusY(11);

ellipse1.setFill(Color.TRANSPARENT);

ellipse1.setStroke(Color.BLACK);

root.getChildren().add(ellipse1);

Ellipse ellipse3 = new Ellipse();

ellipse3.setCenterX(154);

ellipse3.setCenterY(25);

ellipse3.setRadiusX(11);

ellipse3.setRadiusY(10);

ellipse3.setFill(Color.TRANSPARENT);

ellipse3.setStroke(Color.BLACK);

root.getChildren().add(ellipse3);

Ellipse ellipse4 = new Ellipse();

ellipse4.setCenterX(170);

ellipse4.setCenterY(25);

ellipse4.setRadiusX(9);

ellipse4.setRadiusY(9);

ellipse4.setFill(Color.TRANSPARENT);

ellipse4.setStroke(Color.BLACK);

root.getChildren().add(ellipse4);

Polygon polygon = new Polygon();

polygon.getPoints().addAll(70.0, 50.0,

90.0, 200.0,

110.0, 50.0);

polygon.setFill(Color.BLACK);

root.getChildren().add(polygon);

stage.setScene(scene);

stage.show();

}

}

Скриншот результата:

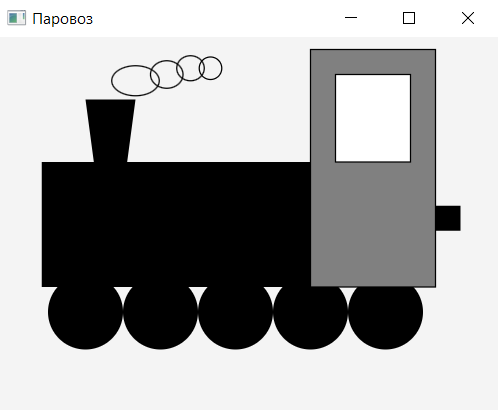


Рисунок 15 – Результат задачи 7

## 1.8 Задача 8

Составить программу для игры в домино. Здесь объектами являются кости домино, а методами — способы выставления той или иной кости.

Код программы:

package com.example.practice\_one.page\_118;

import java.util.ArrayList;

public class Dice {

public int oneValue;

public int twoValue;

public static ArrayList<Dice> diceArrayList = new ArrayList<>();

public Dice(int oneValue, int twoValue){

this.oneValue = oneValue;

this.twoValue = twoValue;

}

public static void createDices(){

diceArrayList.add(new Dice(6, 6));

for (int i = 0; i <= 6; i++){

for (int j = 0; j <= 6; j++){

boolean isDiceExistsAlready = false;

for (Dice dice: diceArrayList) {

if((dice.getOneValue() == i && dice.getTwoValue() == j) ||

(dice.getOneValue() == j && dice.getTwoValue() == i)){

isDiceExistsAlready = true;

}

}

if(!isDiceExistsAlready) {

diceArrayList.add(new Dice(i, j));

}

}

}

}

public int getOneValue(){

return oneValue;

}

public int getTwoValue(){

return twoValue;

}

public void setOneValue(int value){

this.oneValue = value;

}

public void setTwoValue(int value){

this.twoValue = value;

}

}

Скриншоты результата:

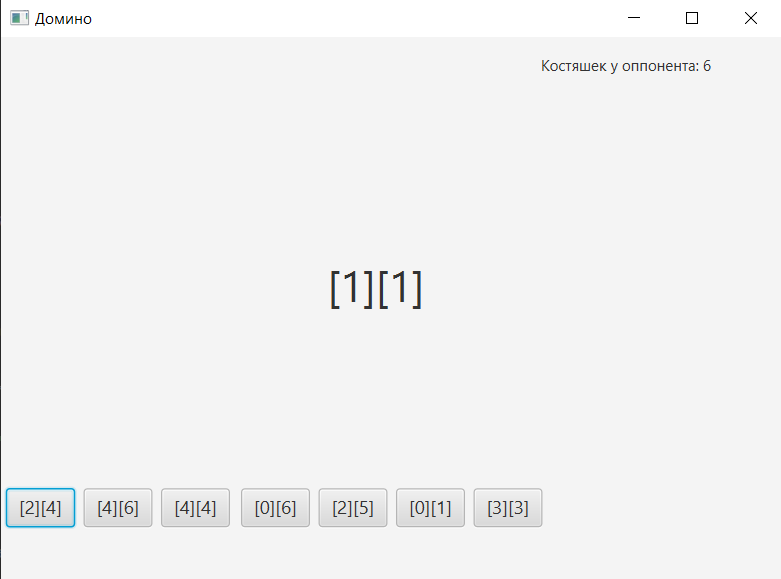


Рисунок 16 – Открытие приложения «Домино»

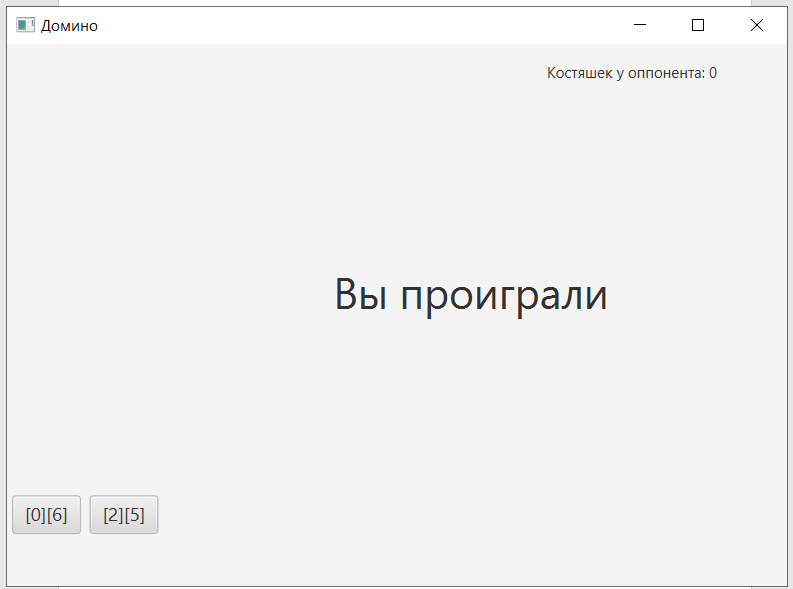


Рисунок 17 – Окно при проигрыше игрока

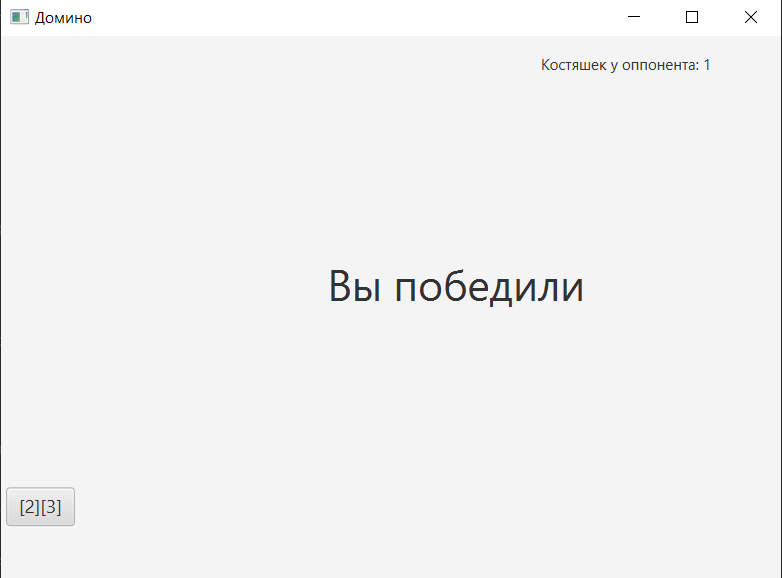


Рисунок 18 – Окно при выигрыше игрока

## 1.9 Задача 9

Итальянская игра «Математико»

Имеется квадратное поле из 25 клеток и набор из 52 карточек, на которых записаны числа от 1 до 13, причем карточки с каждым из этих чисел встречаются по четыре раза.

Разработать программу, которая позволит имитировать игру человека с компьютером: случайным образом извлекается какая-либо из имеющихся карточек и выдается записанное на ней число. Каждый игрок заносит это число в одну из клеток квадрата, и так продолжается до тех пор, пока не будут заполнены все клетки квадрата.

По окончании игры заполнение заполнение клеток поля оценивается определенным количеством очков. Цель игры — разместить числа в клетках таким образом, чтобы набрать наибольшее количество очков.

Разработать для компьютера наиболее оптимальную стратегию заполнения квадрата.

Код программы:

package com.example.practice\_one.page\_132;

import javafx.scene.control.Label;

import java.util.Random;

public class Card {

private int i = new Random().nextInt(0, 3);

private int j = new Random().nextInt(0, 12);

private int[][] cards = {

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13},

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13},

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13},

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}};

private int currentCard = cards[i][j];

public Label label;

public Card(Label label) {

this.label = label;

}

public void cardNew() {

cards[i][j] = -1;

do {

i = new Random().nextInt(0, 4);

j = new Random().nextInt(0, 13);

currentCard = cards[i][j];

}while (currentCard == -1);

}

public void writeCurrentCard() {

label.setText("Текущая карта: " + currentCard);

}

public int getCurrentCard() {

return currentCard;

}

public void setCurrentCard(int currentCard) {

this.currentCard = currentCard;

}

public Label getLabel() {

return label;

}

public void setLabel(Label label) {

this.label = label;

}

}

package com.example.practice\_one.page\_132;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

import java.util.Map;

public class Check {

private Map elemC, elemP;

private int markC = 0, markP = 0;

private GridPane poleC, poleP;

private Card card;

public static String compResult = "";

public static String playerResult = "";

public Check(Map elemC, GridPane poleC, Map elemP, GridPane poleP, Card card) {

this.elemC = elemC;

this.elemP = elemP;

this.poleC = poleC;

this.poleP = poleP;

this.card = card;

}

private int compCheck() {

for (int i = 0; i < 5; i++) {

Integer[] list = new Integer[5];

for (int j = 0; j < 5; j++) {

int b = Integer.parseInt(compLabel(elemC, getID(poleC, i, j)).getText());

list[j] = b;

}

markC += new Scoring().beginScoring(list, false);

}

for (int i = 0; i < 5; i++) {

Integer[] list = new Integer[5];

for (int j = 0; j < 5; j++) {

int b = Integer.parseInt(compLabel(elemC, getID(poleC, j, i)).getText());

list[j] = b;

}

markC += new Scoring().beginScoring(list, false);

}

Integer[] list = new Integer[5];

for (int i = 0; i < 5; i++) {

int b = Integer.parseInt(compLabel(elemC, getID(poleC, i, i)).getText());

list[i] = b;

}

markC += new Scoring().beginScoring(list, true);

Integer[] list2 = new Integer[5];

for (int i = 0; i < 5; i++) {

int b = Integer.parseInt(compLabel(elemC, getID(poleC, i, 4 - i)).getText());

list2[i] = b;

}

markC += new Scoring().beginScoring(list2, true);

return markC;

}

private int playerCheck() {

for (int i = 0; i < 5; i++) {

Integer[] list = new Integer[5];

for (int j = 0; j < 5; j++) {

int b = Integer.parseInt(compLabel(elemP, getID(poleP, i, j)).getText());

list[j] = b;

}

markP += new Scoring().beginScoring(list, false);

}

for (int i = 0; i < 5; i++) {

Integer[] list = new Integer[5];

for (int j = 0; j < 5; j++) {

int b = Integer.parseInt(compLabel(elemP, getID(poleP, j, i)).getText());

list[j] = b;

}

markP += new Scoring().beginScoring(list, false);

}

Integer[] list = new Integer[5];

for (int i = 0; i < 5; i++) {

int b = Integer.parseInt(compLabel(elemP, getID(poleP, i, i)).getText());

list[i] = b;

}

markP += new Scoring().beginScoring(list, true);

Integer[] list2 = new Integer[5];

for (int i = 0; i < 5; i++) {

int b = Integer.parseInt(compLabel(elemP, getID(poleP, i, 4 - i)).getText());

list2[i] = b;

}

markP += new Scoring().beginScoring(list2, true);

return markP;

}

public void getResults() {

compResult = String.valueOf(compCheck());

playerResult = String.valueOf(playerCheck());

}

public String getID(GridPane pole, int i, int j) {

return pole.getChildren().get(i \* 5 + j).getId();

}

private Label compLabel(Map elem, String id) {

return (Label) elem.get(id);

}

}

package com.example.practice\_one.page\_132;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

import java.util.Map;

import java.util.Objects;

import java.util.Random;

public class Computer {

protected GridPane pole;

public Map elem;

Card card;

public Computer(GridPane pole, Map elem, Card card){

this.pole = pole;

this.elem = elem;

this.card = card;

}

public void putCard(Player player){

while (true) {

int i1 = new Random().nextInt(0, 5),

i2 = new Random().nextInt(0, 5);

if (Objects.equals(getCompLabel(getCoordinates(i1, i2)).getText(), "")) {

getCompLabel(getCoordinates(i1,i2)).setText(String.valueOf(card.getCurrentCard()));

card.cardNew();

card.writeCurrentCard();

break;

}

}

boolean bool = false;

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

String b = getCompLabel(getCoordinates(i, j)).getText();

if (Objects.equals(b, "")) {

bool = false;

break;

}else {

bool = true;

}

}if (!bool) break;

}

if (bool){

new Check(elem, pole, player.elements, player.pole, card).getResults();

}

}

private Label getCompLabel(String id){

return (Label) elem.get(id);

}

public String getCoordinates(int i, int j){

return pole.getChildren().get(i\*5+j).getId();

}

}

package com.example.practice\_one.page\_132;

import com.example.practice\_one.HelloApplication;

import javafx.animation.KeyFrame;

import javafx.animation.Timeline;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

import javafx.util.Duration;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

public class Page132 {

@FXML

private GridPane computerGridPane;

@FXML

private GridPane playerGridPane;

@FXML

private Label playerResultLabel;

@FXML

private Label compResultLabel;

@FXML

private Label label1;

@FXML

private Label label10;

@FXML

private Label compLabel10;

@FXML

private Label label11;

@FXML

private Label compLabel11;

@FXML

private Label label12;

@FXML

private Label compLabel12;

@FXML

private Label label13;

@FXML

private Label compLabel13;

@FXML

private Label label14;

@FXML

private Label compLabel14;

@FXML

private Label label15;

@FXML

private Label compLabel15;

@FXML

private Label label16;

@FXML

private Label compLabel16;

@FXML

private Label label17;

@FXML

private Label compLabel17;

@FXML

private Label label18;

@FXML

private Label compLabel18;

@FXML

private Label label19;

@FXML

private Label compLabel19;

@FXML

private Label compLabel1;

@FXML

private Label label2;

@FXML

private Label label20;

@FXML

private Label compLabel20;

@FXML

private Label label21;

@FXML

private Label compLabel21;

@FXML

private Label label22;

@FXML

private Label compLabel22;

@FXML

private Label label23;

@FXML

private Label compLabel23;

@FXML

private Label label24;

@FXML

private Label compLabel24;

@FXML

private Label label25;

@FXML

private Label compLabel25;

@FXML

private Label compLabel2;

@FXML

private Label label3;

@FXML

private Label compLabel3;

@FXML

private Label label4;

@FXML

private Label compLabel4;

@FXML

private Label label5;

@FXML

private Label compLabel5;

@FXML

private Label label6;

@FXML

private Label compLabel6;

@FXML

private Label label7;

@FXML

private Label compLabel7;

@FXML

private Label label8;

@FXML

private Label compLabel8;

@FXML

private Label label9;

@FXML

private Label compLabel9;

@FXML

private Label currentCard;

private Map<String, Label> labelsP = new HashMap<>();

private Map<String, Label> labelsC = new HashMap<>();

public static void show() {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("page132.fxml"));

Stage stage = new Stage();

stage.setTitle("Математико");

try {

stage.setScene(new Scene(fxmlLoader.load()));

} catch (IOException e) {

throw new RuntimeException(e);

}

stage.show();

}

@FXML

void initialize() {

labelsP.put("label1", label1);

labelsP.put("label2", label2);

labelsP.put("label3", label3);

labelsP.put("label4", label4);

labelsP.put("label5", label5);

labelsP.put("label6", label6);

labelsP.put("label7", label7);

labelsP.put("label8", label8);

labelsP.put("label9", label9);

labelsP.put("label10", label10);

labelsP.put("label11", label11);

labelsP.put("label12", label12);

labelsP.put("label13", label13);

labelsP.put("label14", label14);

labelsP.put("label15", label15);

labelsP.put("label16", label16);

labelsP.put("label17", label17);

labelsP.put("label18", label18);

labelsP.put("label19", label19);

labelsP.put("label20", label20);

labelsP.put("label21", label21);

labelsP.put("label22", label22);

labelsP.put("label23", label23);

labelsP.put("label24", label24);

labelsP.put("label25", label25);

labelsC.put("compLabel1", compLabel1);

labelsC.put("compLabel2", compLabel2);

labelsC.put("compLabel3", compLabel3);

labelsC.put("compLabel4", compLabel4);

labelsC.put("compLabel5", compLabel5);

labelsC.put("compLabel6", compLabel6);

labelsC.put("compLabel7", compLabel7);

labelsC.put("compLabel8", compLabel8);

labelsC.put("compLabel9", compLabel9);

labelsC.put("compLabel10", compLabel10);

labelsC.put("compLabel11", compLabel11);

labelsC.put("compLabel12", compLabel12);

labelsC.put("compLabel13", compLabel13);

labelsC.put("compLabel14", compLabel14);

labelsC.put("compLabel15", compLabel15);

labelsC.put("compLabel16", compLabel16);

labelsC.put("compLabel17", compLabel17);

labelsC.put("compLabel18", compLabel18);

labelsC.put("compLabel19", compLabel19);

labelsC.put("compLabel20", compLabel20);

labelsC.put("compLabel21", compLabel21);

labelsC.put("compLabel22", compLabel22);

labelsC.put("compLabel23", compLabel23);

labelsC.put("compLabel24", compLabel24);

labelsC.put("compLabel25", compLabel25);

Card card = new Card(currentCard);

card.writeCurrentCard();

Computer computer = new Computer(this.computerGridPane, labelsC, card);

Player player = new Player(this.playerGridPane, labelsP, card, computer);

player.putCard();

resultObserving();

}

private void resultObserving(){

Timeline timeline = new Timeline();

javafx.util.Duration duration = new Duration(100);

KeyFrame keyFrame = new KeyFrame(duration, event -> {

if(!Check.playerResult.isEmpty() && !Check.compResult.isEmpty()){

compResultLabel.setText("Очки: " + Check.compResult);

playerResultLabel.setText("Очки: " + Check.playerResult);

if(Integer.parseInt(Check.compResult) > Integer.parseInt(Check.playerResult)){

currentCard.setText("Вы проиграли");

}

else{

if(Integer.parseInt(Check.compResult) < Integer.parseInt(Check.playerResult)){

currentCard.setText("Вы выиграли");

}

else currentCard.setText("Ничья");

}

}

}

);

timeline.getKeyFrames().add(keyFrame);

timeline.setCycleCount(Timeline.INDEFINITE);

timeline.play();

}

}

package com.example.practice\_one.page\_132;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

import java.util.Map;

import java.util.Objects;

public class Player {

protected GridPane pole;

public Map elements;

Card card;

Computer computer;

public Player(GridPane gridPane, Map elements, Card card, Computer computer){

this.pole = gridPane;

this.elements = elements;

this.card = card;

this.computer = computer;

}

public void putCard(){

pole.getChildren().forEach(item -> item.setOnMouseClicked(event -> {

if (Objects.equals(getPlayerLabel(item.getId()).getText(), "")) {

getPlayerLabel(item.getId()).setText(String.valueOf(card.getCurrentCard()));

computer.putCard(Player.this);

}

}));

}

private Label getPlayerLabel(String id){

return (Label) elements.get(id);

}

}

package com.example.practice\_one.page\_132;

import java.util.\*;

public class Scoring {

public int beginScoring(Integer[] list, boolean diagonal) {

int scoreOne = 0;

int scoreTwo = diagonal ? 10 : 0;

List<Integer> numbers = Arrays.asList(list);

Set<Integer> set = new HashSet<>();

List<Object> duplicates = new ArrayList<>();

numbers.forEach(n -> {

if (!set.add(n)) {

duplicates.add(n);

}

});

Set<Integer> set2 = new HashSet<>();

List<Object> duplicates2 = new ArrayList<>();

duplicates.forEach(n -> {

if (!set2.add((Integer) n)) {

duplicates2.add(n);

}

});

if (!duplicates2.isEmpty()) {

if (set2.size() != 1) {

if (new ArrayList<>(set).get(0) == 1 && new ArrayList<>(set).get(1) == 13) {

scoreOne += 100 + scoreTwo;

} else {

scoreOne += 80 + scoreTwo;

}

} else if (duplicates2.size() == 2) {

if (Objects.equals(duplicates2.get(0), 1)) {

scoreOne += 200 + scoreTwo;

} else {

scoreOne += 160 + scoreTwo;

}

} else {

scoreOne += 40 + scoreTwo;

}

} else {

if (!duplicates.isEmpty()) {

scoreOne += 10 \* duplicates.size() + scoreTwo;

} else {

List<Integer> numbers2 = new ArrayList<>(set);

boolean bool = true;

for (int i = 0; i < 5; i++) {

if (numbers2.get(i) != i + 1) {

bool = false;

break;

}

}

if (bool) scoreOne += 50 + scoreTwo;

boolean bool2 = numbers2.get(0) == 1 &&

numbers2.get(1) == 10 &&

numbers2.get(2) == 11 &&

numbers2.get(3) == 12 &&

numbers2.get(4) == 13;

if (bool2) scoreOne += 150 + scoreTwo;

}

}

return scoreOne;

}

}

Скриншоты результатов:

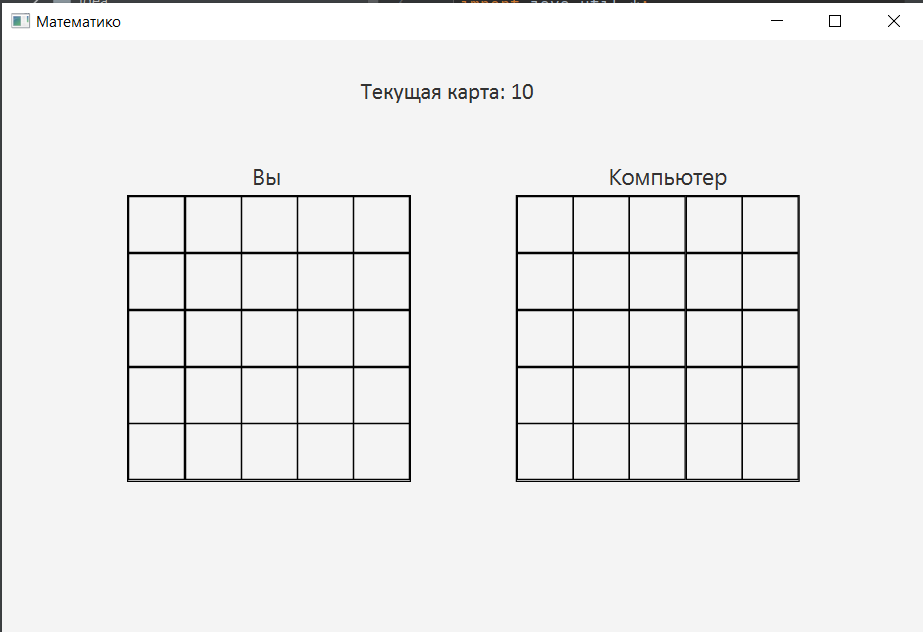


Рисунок 19 – Открытие программы «Математико»

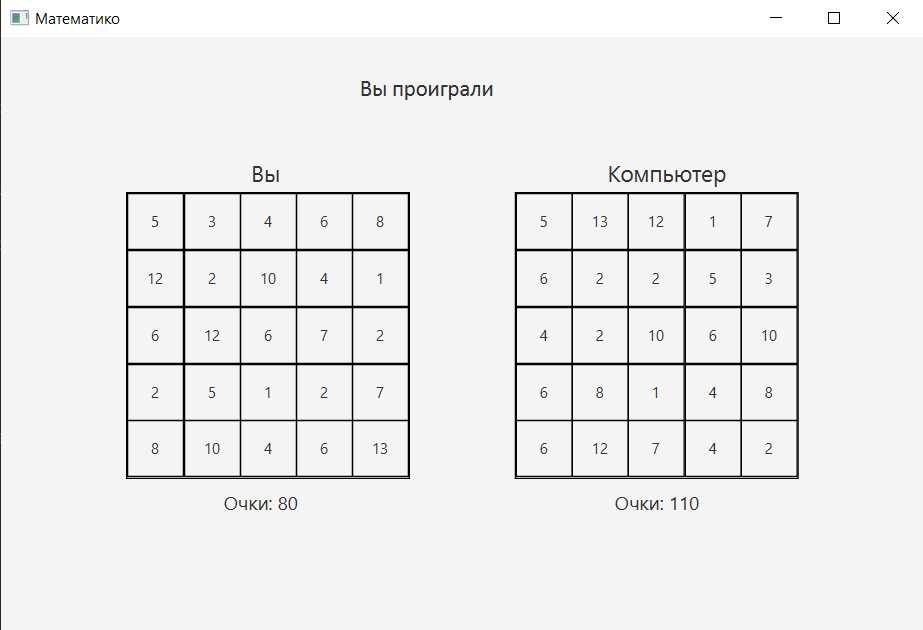


Рисунок 20 – Окно при проигрыше игрока

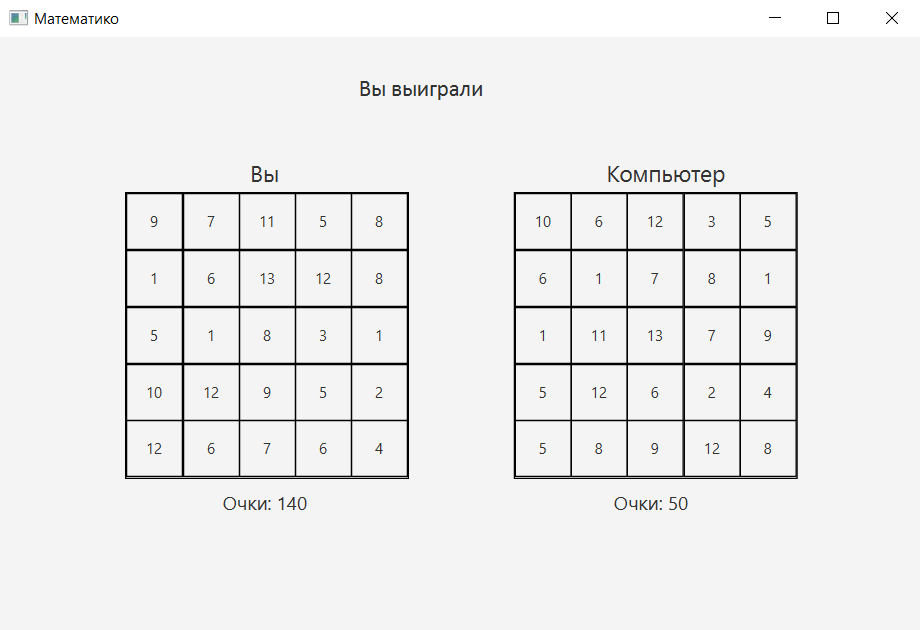


Рисунок 21 – Окно при выигрыше игрока

1. Разработка АИС

В ходе учебной практики была разработана АИС для мерчендайзеров, позволяющая хранить и добавлять информацию о товарах, их поставщиках и торговых точек организации.

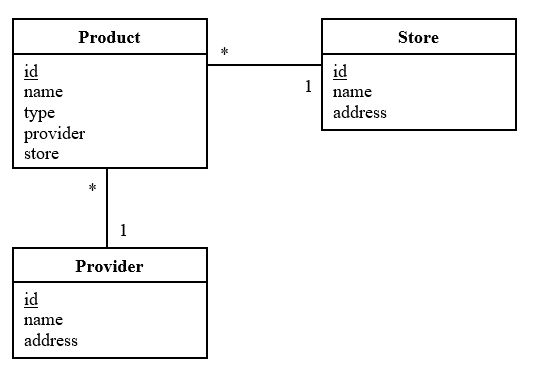


Рисунок 22 – ER – диаграмма сущностей в АИС

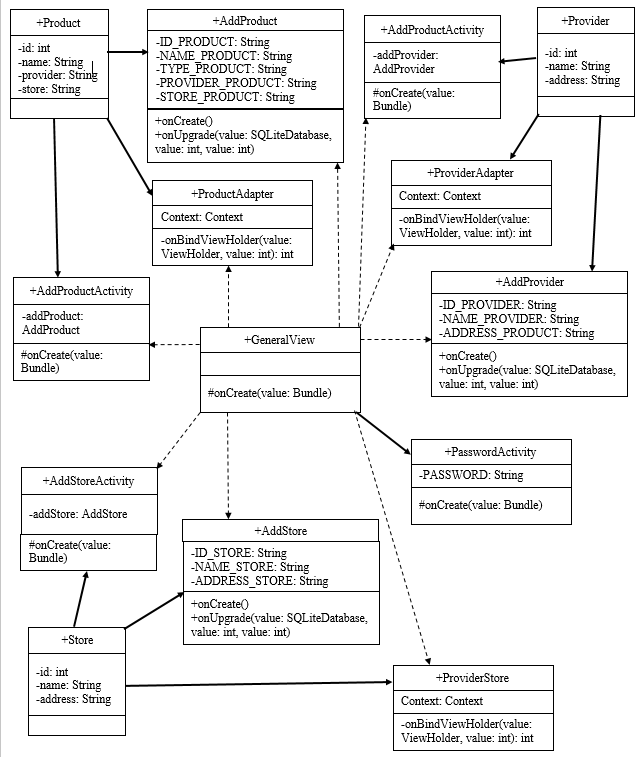


Рисунок 23 – Диаграмма взаимодействия классов в АИС

## 2.1 Техническое задание

2.1.1 Краткая характеристика области применения

Программа предназначена для добавления и хранения информации, необходимой для работы мерчендайзера торговой сети.

Приложение устанавливается через USB-кабель или Wi-Fi – соединение.

### 2.1.2 Основания для разработки программного изделия

Основанием для разработки данного программного продукта является учебная практика.

* + 1. Функциональное и эксплуатационное назначение

Приложение должно содержать следующий функционал:

При запуске приложения будет появляться экран с интерфейсом, предлагающим ввести пароль пользователю. Если он верный, приложение перейдет к таблицам сущностей, которые отражают информацию из встроенной базы данных, если нет, оповестит об этом всплывающим сообщением.

После правильного введения пароля выводятся таблицы со всеми сущностями, которые можно вертикально пролистывать.

Пользователь может добавить элемент в одну из таблиц, выбрав ее в выпадающем списке, появляющемся при нажатии троеточия.

При нажатии на одну из них появляется окно добавления элемента в таблицу. Добавление элемента осуществляется обязательным заполнением всех полей (в противном случае появится сообщение об ошибке) и нажатием кнопки «Ок». В случае успешного добавления нового элемента информация вносится в БД и отображается в соответствующей табли, также появляется соответствующее, подтверждающее успешное выполнение операции, после чего данные в полях для заполнения сотрутся.

Пользователь также может вернуться в представление полей благодаря кнопке «Назад».

* + 1. Требования к программе

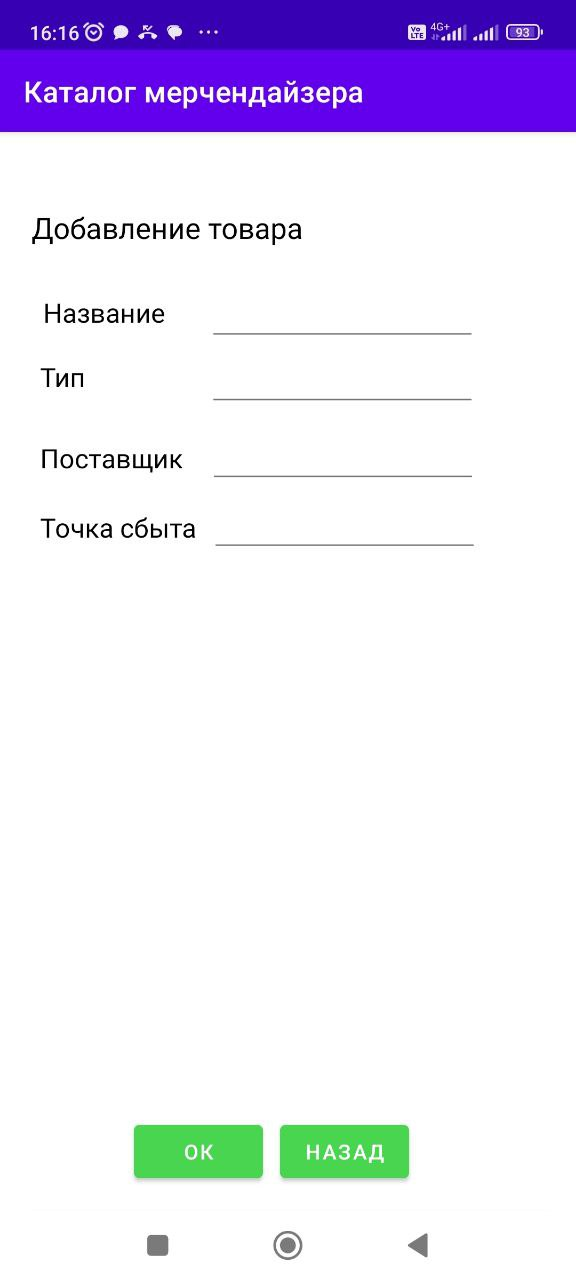
1. Корректный запуск на тестируемом устройстве (Redmi Note 10 Pro).
2. Исправная работа приложения во работы.
   1. Своевременный и корректный отклик на действия пользователя.
   2. Правильное отображение информации в базе данных.
   3. Корректное предупреждение пользователя о неправильных действиях с его стороны.
3. Соответствие между нажатыми кнопками и ожидаемыми от них действиями.

# 3 Разработка мобильного приложения

В качестве среды разработки была выбрана Android Studio ввиду ее популярности в среде мобильной разработки, а в качестве языка – Java, по той же причине.

Для создания и интеграции в приложение базы данных была использована SQLite, так как данная СУБД обладает высокой скоростью чтения и записи информации.

Интерфейс был разработан с помощью XML - встроенного в среду разработки инструмента для работы с визуальной составляющей каждого отдельного окна.



\*\*\*\*\*\*\*

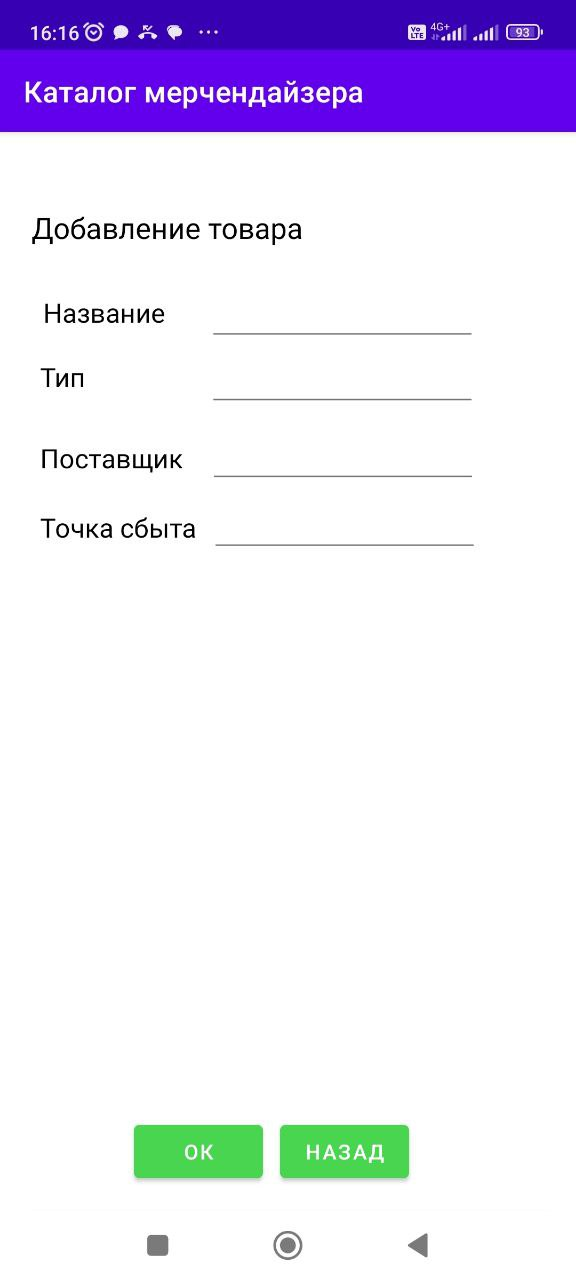
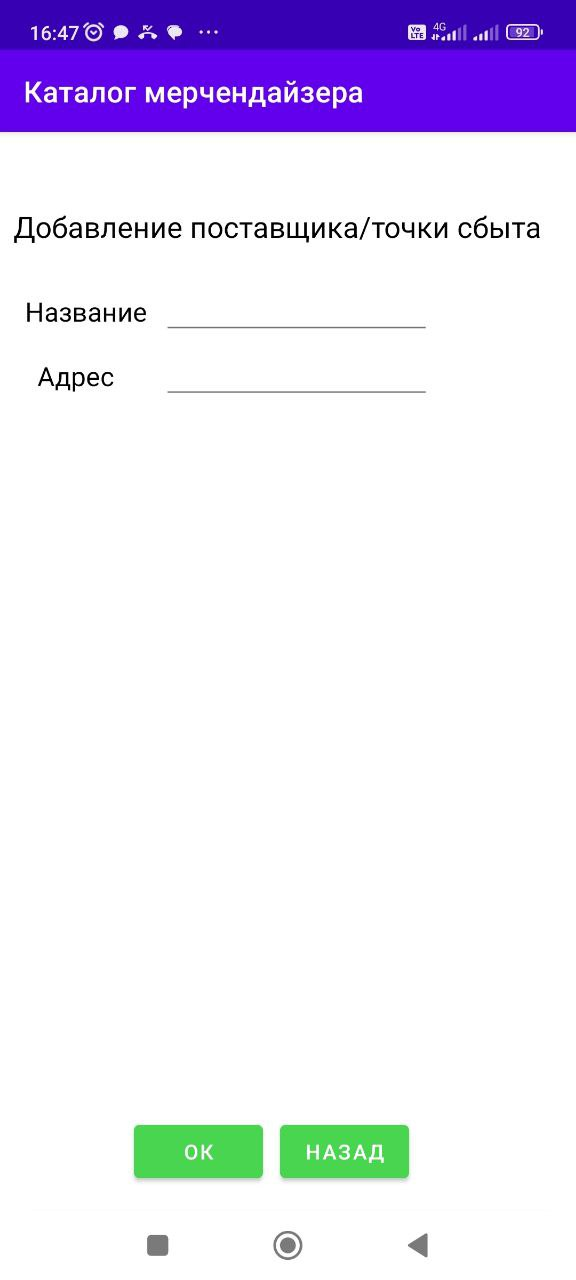


Рисунок 24 – Интерфейс окна добавления товара



\*\*\*\*\*\*\*\*\*

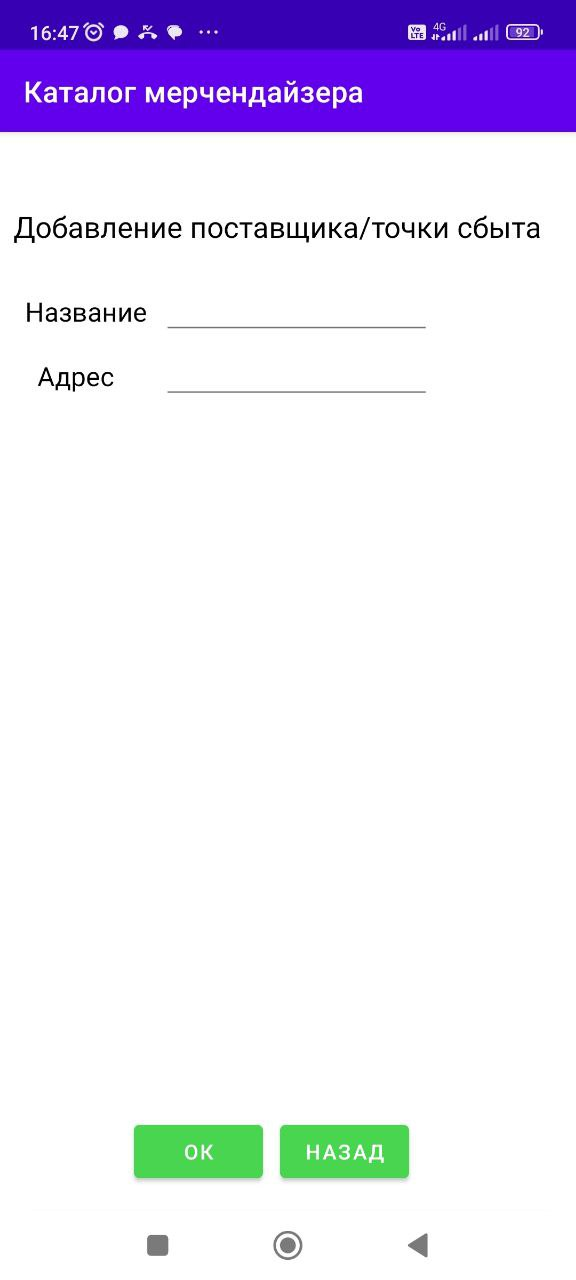


Рисунок 25 – Интерфейс окна добавления товара

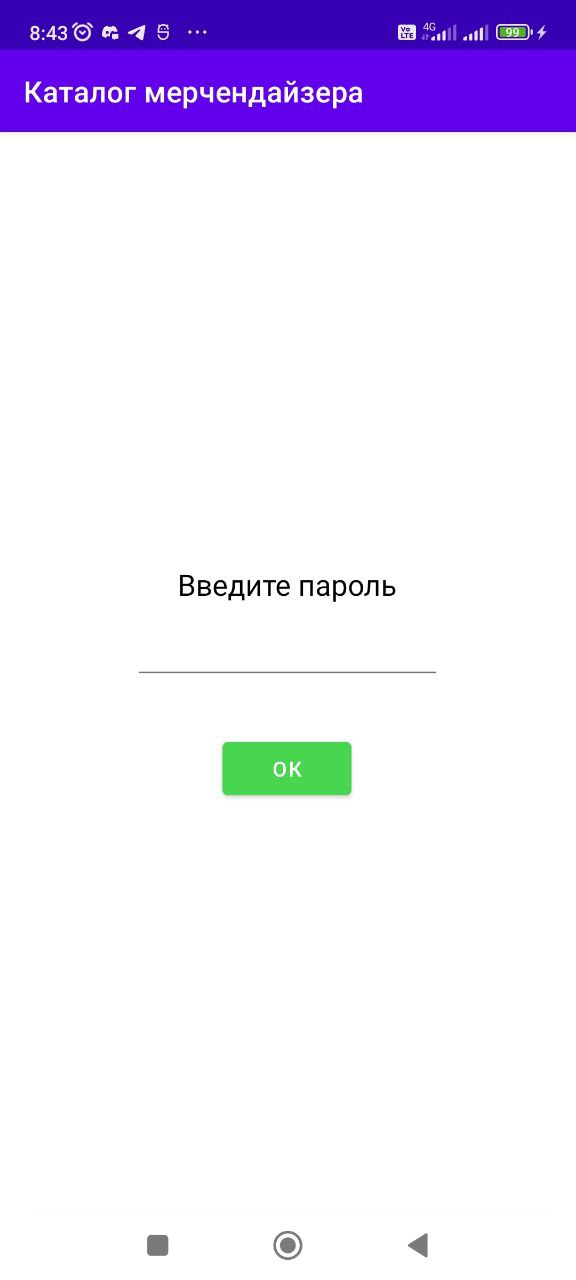


Рисунок 26 – Интерфейс окна ввода пароля

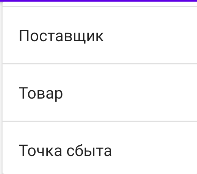


Рисунок 27 – Интерфейс выпадающего списка таблиц

# 4 Разработка сайта

В качестве инструмента для разработки была выбрана CMS WordPress в виду своей простоты в освоении и работе, а также потому данная система является одной из самых популярных в мире в сфере разработки Web-приложений.

Сам сайт представляет собой одностраничный проект, в котором кратко указана информация о созданных во время учебной практики программах, их особенностях о внешнем виде, в основном о мобильном и оконных приложениях.

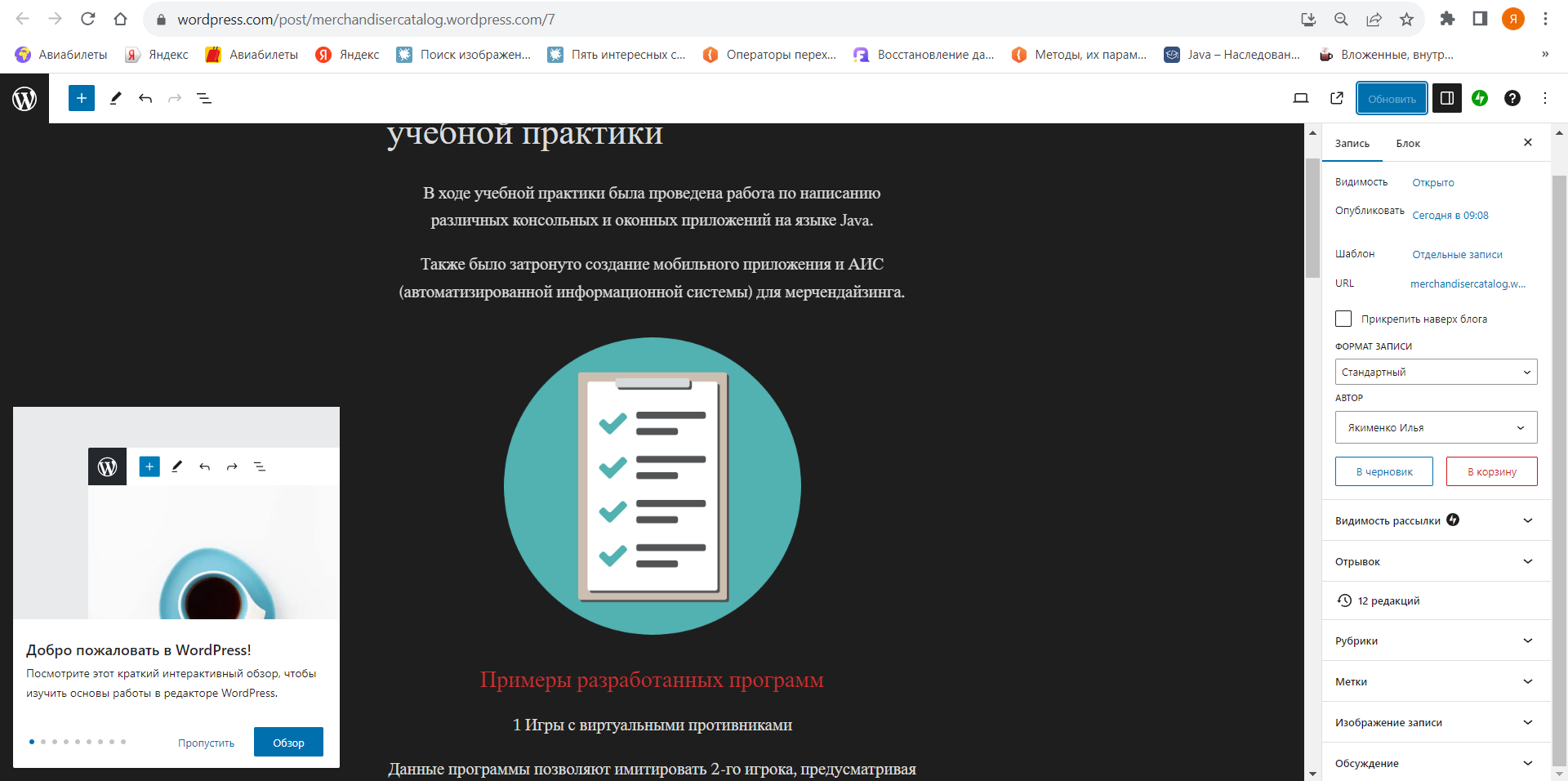


Рисунок 28 – Сайт в конструкторе WordPress

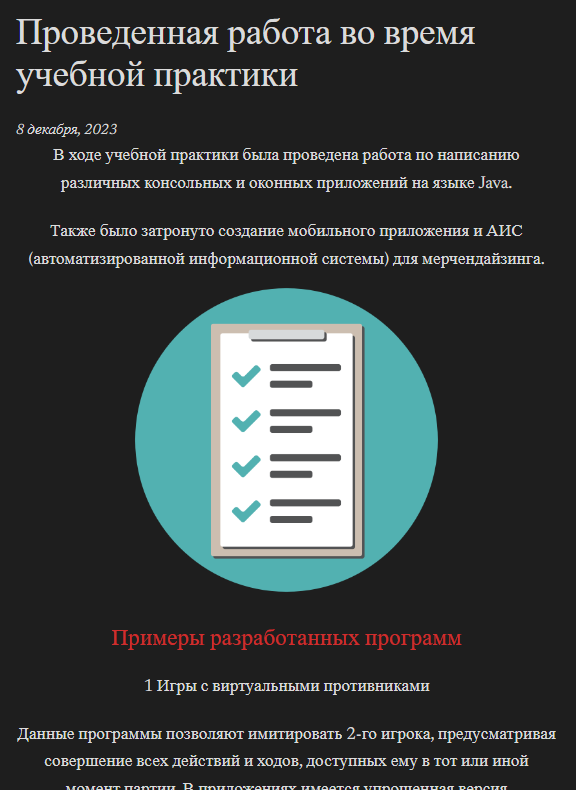


Рисунок 29 – Готовый интерфейс сайта

Ссылка на сайт: https://merchandisercatalog.wordpress.com/

# 5 Руководство пользователя

Руководство пользователя представлено в Приложении А.

# Заключение

За время выполнения учебной практики были разработаны 6 консольных приложений, и 3 оконных. Также было создано мобильное приложение с использованием базы данных и сайт.

В ходе работы были выявлены наиболее подходящие и оптимальные средства и инструменты работы над программами, навыки владения которыми улучшились в результате их использования.

В результате тестирования приложений выяснилось, что они соответствуют своим изначальным требованиям и является корректно работающими программами. Все определенные задачи по их созданию были успешно выполнены.

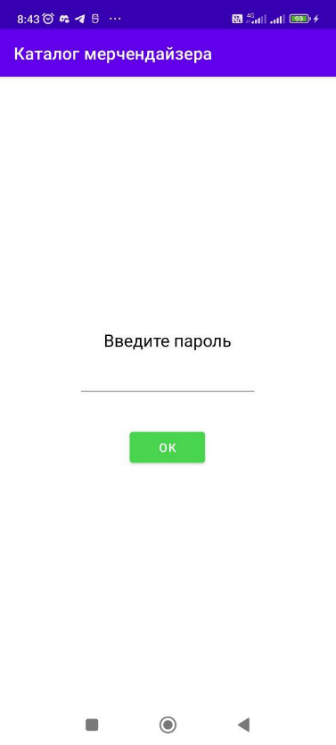
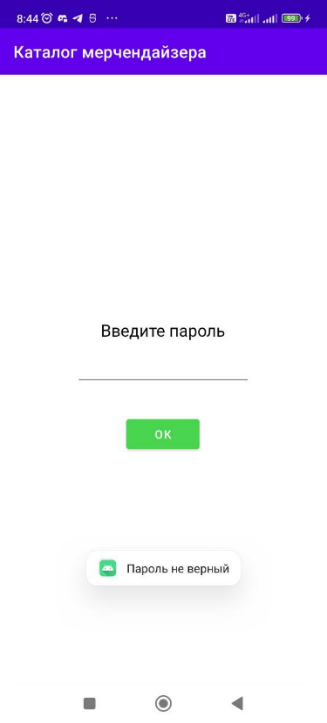
# Список использованной литературы

1. SQLite [Электронный ресурс]. – Режим доступа: - <https://blog.skillfactory.ru/glossary/sqlite/>
2. Android Studio: среда разработки мобильных приложений - [Электронный ресурс]. – Режим доступа: - <https://arduinoplus.ru/android-studio/>
3. IntelliJ IDEA [Электронный ресурс]. – Режим доступа: - <https://blog.skillfactory.ru/glossary/intellij-idea/>
4. Что такое Wordpress: о программе и работе с ней [Электронный ресурс]. – Режим доступа: - <https://help.reg.ru/support/hosting/cms/chto-takoye-wordpress-o-programme-i-rabote-s-ney>
5. Что такое реляционная база данных? [Электронный ресурс]. – Режим доступа: - <https://aws.amazon.com/ru/relational-database/>
6. Сущности и связи: как и для чего системные аналитики создают ER‑диаграммы [Электронный ресурс]. – Режим доступа: - https://practicum.yandex.ru/blog/chto-takoe-er-diagramma/

Приложения

# Приложение А. Руководство пользователя

При входе в приложение необходимо ввести пароль, если он верный, приложение перейдет к таблицам сущностей, если нет, оповестит об этом всплывающим сообщением.



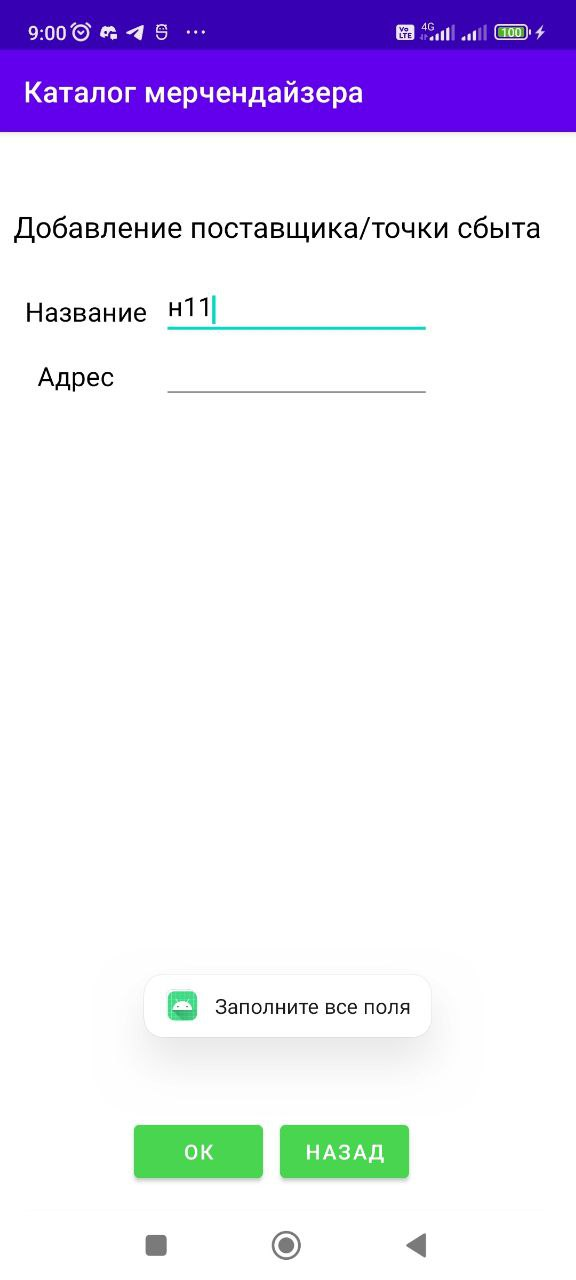
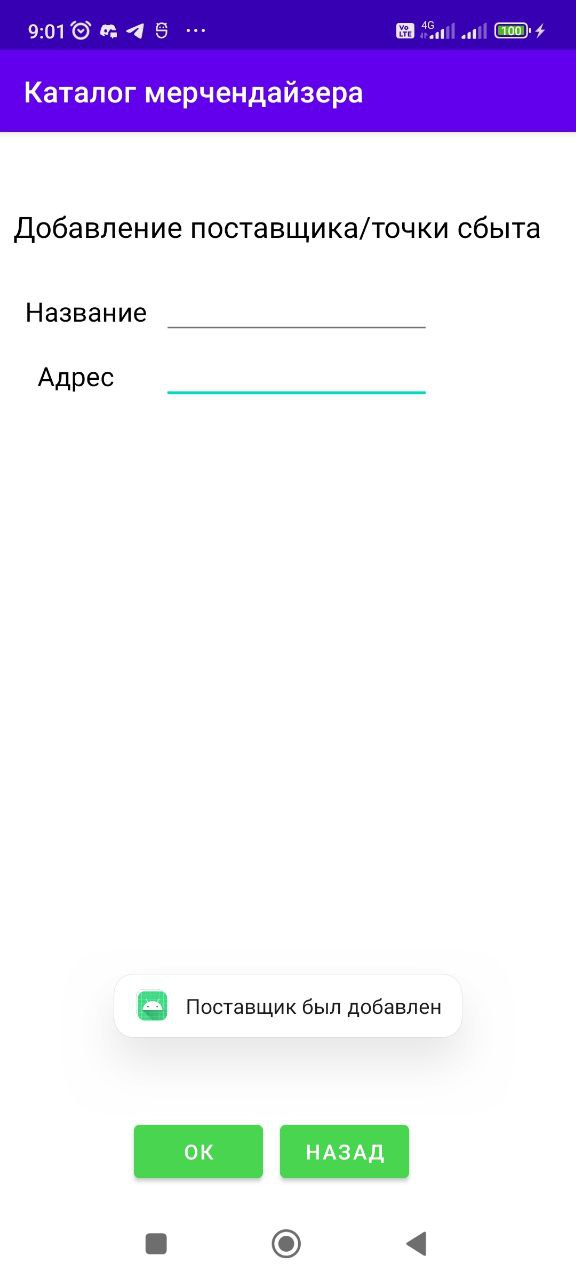
После правильного введения пароля выводятся таблицы со всеми сущностями, которые можно вертикально пролистывать.



Чтобы добавить элемент в одну из таблиц, нужно выбрать ее в выпадающем списке, появляющемся при нажатии троеточия.



При нажатии на одну из них появляется окно добавления элемента в таблицу. Добавление элемента осуществляется обязательным заполнением всех полей (в противном случае появится сообщение об ошибке) и нажатием кнопки «Ок». В случае успешного добавления нового элемента также появляется соответствующее окно, после чего данные в полях для заполнения сотрутся. Вернуться в представление полей можно благодаря кнопке «Назад».

# Приложение Б. Листинг кода мобильного приложения

package com.yakimenko.lessonhomework3fake3;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.DatabaseErrorHandler;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.os.Build;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.annotation.RequiresApi;

import java.util.ArrayList;

public class AddProduct extends SQLiteOpenHelper {

private static final String DB\_NAME = "product4";

private static final int DB\_VERSION = 1;

private static final String TABLE\_NAME = "product";

private static final String ID\_PRODUCT = "id";

private static final String NAME\_PRODUCT = "name";

private static final String TYPE\_PRODUCT = "type";

private static final String PROVIDER\_PRODUCT = "provider";

private static final String STORE\_PRODUCT = "store";

public AddProduct(Context context) {

super(context, DB\_NAME, null, DB\_VERSION);

}

public AddProduct(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public AddProduct(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version, @Nullable DatabaseErrorHandler errorHandler) {

super(context, name, factory, version, errorHandler);

}

@RequiresApi(api = Build.VERSION\_CODES.P)

public AddProduct(@Nullable Context context, @Nullable String name, int version, @NonNull SQLiteDatabase.OpenParams openParams) {

super(context, name, version, openParams);

}

@Override

public void onCreate(SQLiteDatabase db) {

String query = "CREATE TABLE " + TABLE\_NAME + " ("

+ ID\_PRODUCT + " INTEGER PRIMARY KEY AUTOINCREMENT, "

+ NAME\_PRODUCT + " TEXT,"

+ TYPE\_PRODUCT + " TEXT,"

+ PROVIDER\_PRODUCT + " TEXT,"

+ STORE\_PRODUCT + " TEXT)";

db.execSQL(query);

}

public void addProduct(String name, String type, String provider, String store) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(NAME\_PRODUCT, name);

values.put(TYPE\_PRODUCT, type);

values.put(PROVIDER\_PRODUCT, provider);

values.put(STORE\_PRODUCT, store);

db.insert(TABLE\_NAME, null, values);

db.close();

}

public ArrayList<Product> readProducts() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursorProducts = db.rawQuery("SELECT \* FROM " + TABLE\_NAME, null);

ArrayList<Product> productsArrayList = new ArrayList<>();

if (cursorProducts.moveToFirst()) {

do {

productsArrayList.add(new Product(cursorProducts.getString(4), cursorProducts.getString(3),

cursorProducts.getString(2),

cursorProducts.getString(1)));

} while (cursorProducts.moveToNext());

}

cursorProducts.close();

return productsArrayList;

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Intent;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class AddProductActivity extends AppCompatActivity {

private EditText nameEdt, typeEdt, providerEdt, storeEdt;

private Button okBtn, backBtn;

private AddProduct addProduct;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_product);

nameEdt = findViewById(R.id.productNameEditText);

typeEdt = findViewById(R.id.typeEditText);

providerEdt = findViewById(R.id.providerEditText);

storeEdt = findViewById(R.id.storeEditText);

okBtn = findViewById(R.id.okBtn);

backBtn = findViewById(R.id.cancelBtn);

addProduct = new AddProduct(AddProductActivity.this);

okBtn.setOnClickListener(v -> {

String name = nameEdt.getText().toString();

String type = typeEdt.getText().toString();

String provider = providerEdt.getText().toString();

String store = storeEdt.getText().toString();

if (name.isEmpty() || type.isEmpty() || provider.isEmpty() || store.isEmpty()) {

Toast.makeText(AddProductActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

addProduct.addProduct(store, provider, type, name);

Toast.makeText(AddProductActivity.this, "Товар был добавлен", Toast.LENGTH\_SHORT).show();

typeEdt.setText("");

nameEdt.setText("");

providerEdt.setText("");

storeEdt.setText("");

});

backBtn.setOnClickListener(v -> {

Intent i = new Intent(AddProductActivity.this, GeneralView.class);

startActivity(i);

});

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.DatabaseErrorHandler;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.os.Build;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.annotation.RequiresApi;

import java.util.ArrayList;

public class AddProvider extends SQLiteOpenHelper {

private static final String DB\_NAME = "provider4";

private static final int DB\_VERSION = 1;

private static final String TABLE\_NAME = "provider";

private static final String ID\_PROVIDER = "id";

private static final String NAME\_PROVIDER = "name";

private static final String ADDRESS\_PROVIDER = "address";

public AddProvider(Context context) {

super(context, DB\_NAME, null, DB\_VERSION);

}

public AddProvider(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public AddProvider(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version, @Nullable DatabaseErrorHandler errorHandler) {

super(context, name, factory, version, errorHandler);

}

@RequiresApi(api = Build.VERSION\_CODES.P)

public AddProvider(@Nullable Context context, @Nullable String name, int version, @NonNull SQLiteDatabase.OpenParams openParams) {

super(context, name, version, openParams);

}

@Override

public void onCreate(SQLiteDatabase db) {

String query = "CREATE TABLE " + TABLE\_NAME + " ("

+ ID\_PROVIDER + " INTEGER PRIMARY KEY AUTOINCREMENT, "

+ NAME\_PROVIDER + " TEXT,"

+ ADDRESS\_PROVIDER + " TEXT)";

db.execSQL(query);

}

public void addProvider(String name, String address) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(NAME\_PROVIDER, name);

values.put(ADDRESS\_PROVIDER, address);

db.insert(TABLE\_NAME, null, values);

db.close();

}

public ArrayList<Provider> readProviders() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursorProviders = db.rawQuery("SELECT \* FROM " + TABLE\_NAME, null);

ArrayList<Provider> providerArrayList = new ArrayList<>();

if (cursorProviders.moveToFirst()) {

do {

providerArrayList.add(new Provider(cursorProviders.getString(2),

cursorProviders.getString(1)));

} while (cursorProviders.moveToNext());

}

cursorProviders.close();

return providerArrayList;

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Intent;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class AddProviderActivity extends AppCompatActivity {

private EditText nameEdt, addressEdt;

private Button okBtn, backBtn;

private AddProvider addProvider;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_provider);

nameEdt = findViewById(R.id.nameEditText);

addressEdt = findViewById(R.id.addressEditText);

okBtn = findViewById(R.id.okBtn);

backBtn = findViewById(R.id.cancelBtn);

addProvider = new AddProvider(AddProviderActivity.this);

okBtn.setOnClickListener(v -> {

String providerName = nameEdt.getText().toString();

String providerAddress = addressEdt.getText().toString();

if (providerName.isEmpty() || providerAddress.isEmpty()) {

Toast.makeText(AddProviderActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

addProvider.addProvider(providerAddress, providerName);

Toast.makeText(AddProviderActivity.this, "Поставщик был добавлен", Toast.LENGTH\_SHORT).show();

addressEdt.setText("");

nameEdt.setText("");

});

backBtn.setOnClickListener(v -> {

Intent i = new Intent(AddProviderActivity.this, GeneralView.class);

startActivity(i);

});

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.DatabaseErrorHandler;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

import android.os.Build;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.annotation.RequiresApi;

import java.util.ArrayList;

public class AddStore extends SQLiteOpenHelper {

private static final String DB\_NAME = "store4";

private static final int DB\_VERSION = 1;

private static final String TABLE\_NAME = "store";

private static final String ID\_STORE = "id";

private static final String NAME\_STORE = "name";

private static final String ADDRESS\_STORE = "address";

public AddStore(Context context) {

super(context, DB\_NAME, null, DB\_VERSION);

}

public AddStore(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {

super(context, name, factory, version);

}

public AddStore(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version, @Nullable DatabaseErrorHandler errorHandler) {

super(context, name, factory, version, errorHandler);

}

@RequiresApi(api = Build.VERSION\_CODES.P)

public AddStore(@Nullable Context context, @Nullable String name, int version, @NonNull SQLiteDatabase.OpenParams openParams) {

super(context, name, version, openParams);

}

@Override

public void onCreate(SQLiteDatabase db) {

String query = "CREATE TABLE " + TABLE\_NAME + " ("

+ ID\_STORE + " INTEGER PRIMARY KEY AUTOINCREMENT, "

+ NAME\_STORE + " TEXT,"

+ ADDRESS\_STORE + " TEXT)";

db.execSQL(query);

}

public void addStore(String name, String address) {

SQLiteDatabase db = this.getWritableDatabase();

ContentValues values = new ContentValues();

values.put(NAME\_STORE, name);

values.put(ADDRESS\_STORE, address);

db.insert(TABLE\_NAME, null, values);

db.close();

}

public ArrayList<Store> readStores() {

SQLiteDatabase db = this.getReadableDatabase();

Cursor cursorStores = db.rawQuery("SELECT \* FROM " + TABLE\_NAME, null);

ArrayList<Store> storeArrayList = new ArrayList<>();

if (cursorStores.moveToFirst()) {

do {

storeArrayList.add(new Store(cursorStores.getString(2),

cursorStores.getString(1)));

} while (cursorStores.moveToNext());

}

cursorStores.close();

return storeArrayList;

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);

onCreate(db);

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Intent;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class AddStoreActivity extends AppCompatActivity {

private EditText nameEdt, addressEdt;

private Button okBtn, backBtn;

private AddStore addStore;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_add\_provider);

nameEdt = findViewById(R.id.nameEditText);

addressEdt = findViewById(R.id.addressEditText);

okBtn = findViewById(R.id.okBtn);

backBtn = findViewById(R.id.cancelBtn);

addStore = new AddStore(AddStoreActivity.this);

okBtn.setOnClickListener(v -> {

String storeName = nameEdt.getText().toString();

String storeAddress = addressEdt.getText().toString();

if (storeName.isEmpty() || storeAddress.isEmpty()) {

Toast.makeText(AddStoreActivity.this, "Заполните все поля", Toast.LENGTH\_SHORT).show();

return;

}

addStore.addStore(storeAddress, storeName);

Toast.makeText(AddStoreActivity.this, "Точка сбыта была добавлена", Toast.LENGTH\_SHORT).show();

addressEdt.setText("");

nameEdt.setText("");

});

backBtn.setOnClickListener(v -> {

Intent i = new Intent(AddStoreActivity.this, GeneralView.class);

startActivity(i);

});

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Intent;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class GeneralView extends AppCompatActivity {

private ArrayList<Provider> providerArrayList;

private AddProvider addProvider;

private ProviderAdapter providerAdapter;

private RecyclerView providersRV;

private ArrayList<Product> productArrayList;

private AddProduct addProduct;

private ProductAdapter productAdapter;

private RecyclerView productsRV;

private ArrayList<Store> storeArrayList;

private AddStore addStore;

private StoreAdapter storeAdapter;

private RecyclerView storesRV;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.provider\_view);

providerArrayList = new ArrayList<>();

addProvider = new AddProvider(GeneralView.this);

providerArrayList = addProvider.readProviders();

providerAdapter = new ProviderAdapter(providerArrayList, GeneralView.this);

providersRV = findViewById(R.id.idRVProvider);

LinearLayoutManager linearLayoutManager = new LinearLayoutManager(GeneralView.this, RecyclerView.VERTICAL, false);

providersRV.setLayoutManager(linearLayoutManager);

providersRV.setAdapter(providerAdapter);

productArrayList = new ArrayList<>();

addProduct = new AddProduct(GeneralView.this);

productArrayList = addProduct.readProducts();

productAdapter = new ProductAdapter(productArrayList, GeneralView.this);

productsRV = findViewById(R.id.idRVProduct);

LinearLayoutManager linearLayoutManager1 = new LinearLayoutManager(GeneralView.this, RecyclerView.VERTICAL, false);

productsRV.setLayoutManager(linearLayoutManager1);

productsRV.setAdapter(productAdapter);

storeArrayList = new ArrayList<>();

addStore = new AddStore(GeneralView.this);

storeArrayList = addStore.readStores();

storeAdapter = new StoreAdapter(storeArrayList, GeneralView.this);

storesRV = findViewById(R.id.idRVStore);

LinearLayoutManager linearLayoutManager2 = new LinearLayoutManager(GeneralView.this, RecyclerView.VERTICAL, false);

storesRV.setLayoutManager(linearLayoutManager2);

storesRV.setAdapter(storeAdapter);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main\_menu, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

switch (id) {

case R.id.provider: {

Intent i = new Intent(GeneralView.this, AddProviderActivity.class);

startActivity(i);

}

return true;

case R.id.product: {

Intent i = new Intent(GeneralView.this, AddProductActivity.class);

startActivity(i);

}

return true;

case R.id.store: {

Intent i = new Intent(GeneralView.this, AddStoreActivity.class);

startActivity(i);

}

return true;

default:

return super.onOptionsItemSelected(item);

}

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Intent;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class PasswordActivity extends AppCompatActivity {

private EditText passwordEdt;

private Button okBtn;

private final String PASSWORD = "123";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_pass\_word);

passwordEdt = findViewById(R.id.editTextTextPassword);

okBtn = findViewById(R.id.okPasswordBtn);

okBtn.setOnClickListener(v -> {

if(passwordEdt.getText().toString().equals(PASSWORD)){

Intent i = new Intent(PasswordActivity.this, GeneralView.class);

startActivity(i);

}

else {

Toast.makeText(PasswordActivity.this, "Пароль не верный", Toast.LENGTH\_SHORT).show();

}

});

}

}

package com.yakimenko.lessonhomework3fake3;

public class Product {

private int id;

private String name;

private String type;

private String provider;

private String store;

public Product(String name, String type, String provider, String store) {

this.name = name;

this.type = type;

this.provider = provider;

this.store = store;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getProvider() {

return provider;

}

public void setProvider(String provider) {

this.provider = provider;

}

public String getStore() {

return store;

}

public void setStore(String store) {

this.store = store;

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ViewHolder> {

private ArrayList<Product> productArrayList;

private Context context;

public ProductAdapter(ArrayList<Product> products, Context context) {

this.productArrayList = products;

this.context = context;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.product\_item, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Product product = productArrayList.get(position);

holder.name.setText("Название: " + product.getName());

holder.type.setText("Тип: " + product.getType());

holder.provider.setText("Поставщик: " + product.getProvider());

holder.store.setText("Точка сбыта: " + product.getStore());

}

@Override

public int getItemCount() {

return productArrayList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder {

private TextView name, type, provider, store;

public ViewHolder(@NonNull View itemView) {

super(itemView);

name = itemView.findViewById(R.id.productName);

type = itemView.findViewById(R.id.type);

provider = itemView.findViewById(R.id.provider);

store = itemView.findViewById(R.id.store);

}

}

}

package com.yakimenko.lessonhomework3fake3;

public class Provider {

private int id;

private String name;

private String address;

public Provider(String name, String address) {

this.name = name;

this.address = address;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class ProviderAdapter extends RecyclerView.Adapter<ProviderAdapter.ViewHolder> {

private ArrayList<Provider> providerArrayList;

private Context context;

public ProviderAdapter(ArrayList<Provider> providers, Context context) {

this.providerArrayList = providers;

this.context = context;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.provider\_item, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Provider provider = providerArrayList.get(position);

holder.name.setText("Название: " + provider.getName());

holder.address.setText("Адрес: " + provider.getAddress());

}

@Override

public int getItemCount() {

return providerArrayList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder {

private TextView name, address;

public ViewHolder(@NonNull View itemView) {

super(itemView);

name = itemView.findViewById(R.id.name);

address = itemView.findViewById(R.id.address);

}

}

}

package com.yakimenko.lessonhomework3fake3;

public class Store {

private int id;

private String name;

private String address;

public Store(String name, String address) {

this.name = name;

this.address = address;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

package com.yakimenko.lessonhomework3fake3;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class StoreAdapter extends RecyclerView.Adapter<StoreAdapter.ViewHolder> {

private ArrayList<Store> storeArrayList;

private Context context;

public StoreAdapter(ArrayList<Store> stores, Context context) {

this.storeArrayList = stores;

this.context = context;

}

@NonNull

@Override

public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.provider\_item, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull ViewHolder holder, int position) {

Store store = storeArrayList.get(position);

holder.name.setText("Название: " + store.getName());

holder.address.setText("Адрес: " + store.getAddress());

}

@Override

public int getItemCount() {

return storeArrayList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder {

private TextView name, address;

public ViewHolder(@NonNull View itemView) {

super(itemView);

name = itemView.findViewById(R.id.name);

address = itemView.findViewById(R.id.address);

}

}

}