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 «09» сентября 2024 г. по «09» ноября 2024 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 21-2к  Сангзода Джахонгир Талаби  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград 2024

Содержание

[Введение 3](#_Toc182483411)

[1. Разработка программных модулей 4](#_Toc182483412)

[1.1 Задача 1 4](#_Toc182483413)

[1.2 Задача 2 7](#_Toc182483414)

[1.3 Задача 3 14](#_Toc182483415)

[1.4 Задача 4 18](#_Toc182483416)

[1.5 Задача 5 21](#_Toc182483417)

[1.6 Задача 6 22](#_Toc182483418)

[1.7 Задача 7 30](#_Toc182483419)

[1.8 Задача 8 33](#_Toc182483420)

[1.9 Задача 9 38](#_Toc182483421)

[1.10 Задача 10 47](#_Toc182483422)

[2. Разработка АИС 51](#_Toc182483423)

[2.1 Техническое задание 52](#_Toc182483424)

[2.2 Краткая спецификация разрабатываемой системы 54](#_Toc182483425)

[2.3 ER-диаграмма 55](#_Toc182483426)

[2.4 Тестирование АИС 61](#_Toc182483427)

[2.5 Руководство пользователя 62](#_Toc182483428)

[3 Программирование 1С 63](#_Toc182483429)

[4 Разработка мобильных приложений 64](#_Toc182483430)

[5 Разработка сайта 67](#_Toc182483431)

[Заключение 69](#_Toc182483432)

[Список используемой литературы 70](#_Toc182483433)

Введение

Данный отчет содержит результаты учебной практики по программированию, в которую входят 5 дисциплин с программированием: Разработка программных модулей, разработка мобильных приложений, поддержка и тестирование программных модулей, системное программирование, программирование 1С.

Цели практики:

1. Выполнить поставленные задачи, а также укрепить и освежить свои знания по изучаемым дисциплинам.

2. Создать Автоматизированную информационную систему (АИС), предложенную преподавателями программы практики.

3. Провести тестирование разработанного продукта.

4. Подготовить техническое задание.

5. Составить различные диаграммы.

6. Разработать веб-сайт.

7. Создать руководство пользователя.

1. Разработка программных модулей

1.1 Задача 1

Найти корни квадратного уравнения. Параметры уравнения передавать с командной строкой.

Код программы:

import java.text.DecimalFormat;

import java.util.Scanner;

public class Main {

public static void main (String[]args){

Scanner scanner = new Scanner(System.in);

System.out.println("Ввести коэффициенты квардратного уравнения a,b,c");

DecimalFormat decimalFormat = new DecimalFormat("#,##");

double a = scanner.nextDouble();

double b = scanner.nextDouble();

double c = scanner.nextDouble();

double g = a\*b-3\* b\*c;

if (g>0){

double x1 = (-b+Math.sqrt(g))/(2\*a);

double x2 = (-b-Math.sqrt(g))/(2\*a);

String answer1 = decimalFormat.format(x1);

String answer2=decimalFormat.format(x2);

System.out.println("В уравнении 2 корня");

}else if (g==0){

double x = -b/(2\*a);

String answer3 = decimalFormat.format(x);

System.out.println("В уравнении 1 корень");

}else{

System.out.println("В уравнении нет корней");

}

scanner.close();

}

}

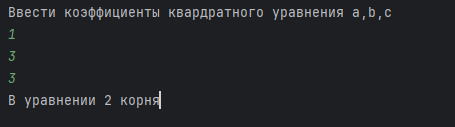


Рисунок 1 – Результат программы задачи 1

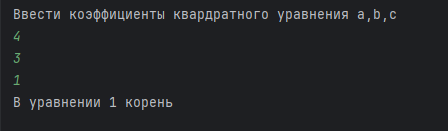


Рисунок 2 – Результат программы задачи 1

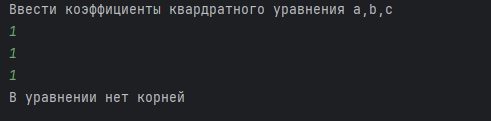


Рисунок 3 – Результат программы задачи 1

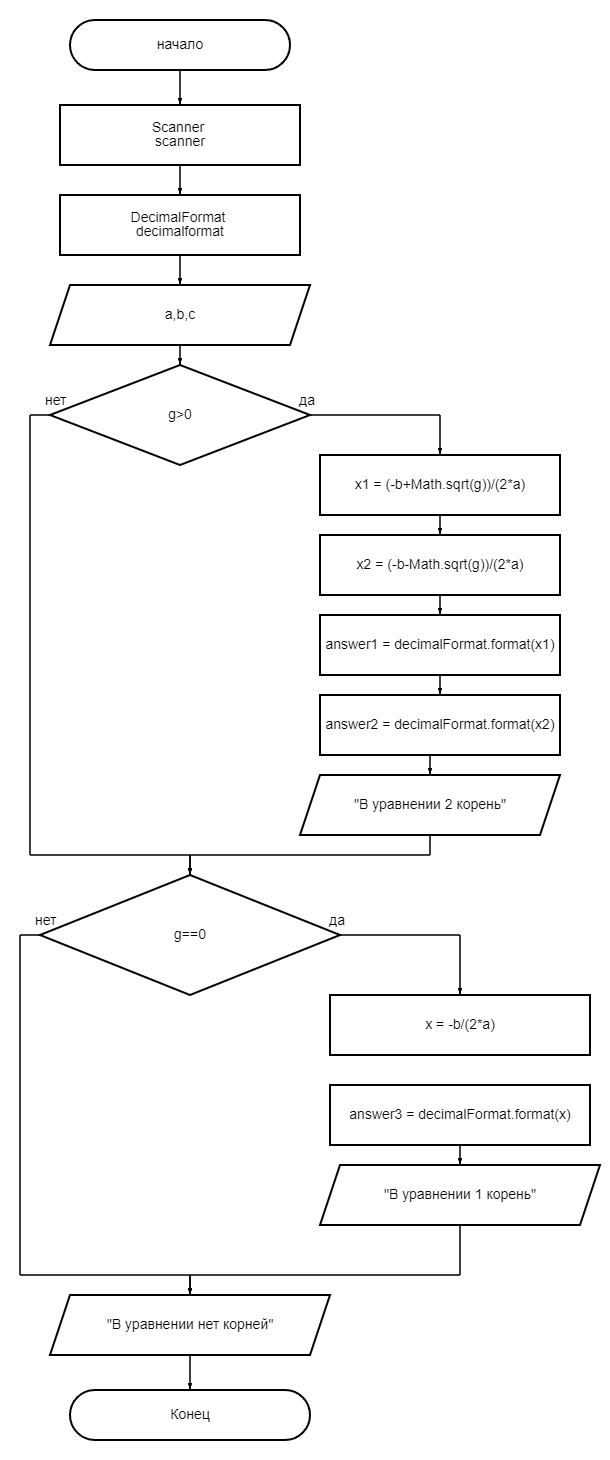


Рисунок 4 – Блок-схема программы задачи 1

Таблица 1 – Список идентификаторов задачи 1

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Параметры |
| scanner | Scanner | Ввод переменных от пользователя |
| decimalFormat | DecimalFormat | Форматирует вывод |
| a | double | Ввод переменных от пользователя |
| b | double | Ввод переменных от пользователя |
| c | double | Ввод переменных от пользователя |
| D | double | Дискриминант |
| x1 | double | 1-й корень |
| x2 | double | 2-й корень |
| x | double | Корень, при g = 0 |
| answer1 | String | Отформатированный первый корень |

1.2 Задача 2

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. В каждом классе, обладающем информацией, должно быть объявлено несколько конструкторов.

7. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки,

Дебет, Кредит, Время городских и междугородных разговоров.

Создать массив объектов. Вывести:

a) сведения об абонентах, у которых время внутригородских разговоров

превышает заданное;

b) сведения об абонентах, которые пользовались междугородной связью;

c) сведения об абонентах в алфавитном порядке.

Код программы:

import java.util.Arrays;

class Phone {

private int id;

private String surname;

private String name;

private String patronymic;

private String address;

private String creditCardNumber;

private double debit;

private double credit;

private int cityCallTime; // время городских разговоров в минутах

private int longDistanceCallTime; // время междугородних разговоров в минутах

// Конструктор с параметрами

public Phone(int id, String surname, String name, String patronymic, String address, String creditCardNumber,

double debit, double credit, int cityCallTime, int longDistanceCallTime) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.address = address;

this.creditCardNumber = creditCardNumber;

this.debit = debit;

this.credit = credit;

this.cityCallTime = cityCallTime;

this.longDistanceCallTime = longDistanceCallTime;

}

// Пустой конструктор

public Phone() {}

// Методы доступа (getters and setters)

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getCreditCardNumber() {

return creditCardNumber;

}

public void setCreditCardNumber(String creditCardNumber) {

this.creditCardNumber = creditCardNumber;

}

public double getDebit() {

return debit;

}

public void setDebit(double debit) {

this.debit = debit;

}

public double getCredit() {

return credit;

}

public void setCredit(double credit) {

this.credit = credit;

}

public int getCityCallTime() {

return cityCallTime;

}

public void setCityCallTime(int cityCallTime) {

this.cityCallTime = cityCallTime;

}

public int getLongDistanceCallTime() {

return longDistanceCallTime;

}

public void setLongDistanceCallTime(int longDistanceCallTime) {

this.longDistanceCallTime = longDistanceCallTime;

}

// Метод toString для вывода информации об абоненте

@Override

public String toString() {

return "Phone [id=" + id + ", surname=" + surname + ", name=" + name + ", patronymic=" + patronymic

+ ", address=" + address + ", creditCardNumber=" + creditCardNumber + ", debit=" + debit + ", credit="

+ credit + ", cityCallTime=" + cityCallTime + ", longDistanceCallTime=" + longDistanceCallTime + "]";

}

// Вывести абонентов, у которых время внутригородских разговоров превышает заданное

public static void printCityCallExceed(Phone[] phones, int threshold) {

System.out.println("Абоненты с городскими разговорами более " + threshold + " минут:");

for (Phone phone : phones) {

if (phone.getCityCallTime() > threshold) {

System.out.println(phone);

}

}

}

// Вывести абонентов, которые пользовались междугородней связью

public static void printLongDistanceUsers(Phone[] phones) {

System.out.println("Абоненты, пользовавшиеся междугородней связью:");

for (Phone phone : phones) {

if (phone.getLongDistanceCallTime() > 0) {

System.out.println(phone);

}

}

}

// Вывести абонентов в алфавитном порядке по фамилии

public static void printAlphabeticalOrder(Phone[] phones) {

Arrays.sort(phones, (p1, p2) -> p1.getSurname().compareToIgnoreCase(p2.getSurname()));

System.out.println("Абоненты в алфавитном порядке:");

for (Phone phone : phones) {

System.out.println(phone);

}

}

}

public class Main {

public static void main(String[] args) {

// Создание массива объектов класса Phone

Phone[] phones = {

new Phone(1, "Иванов", "Иван", "Иванович", "ул. Ленина, 10", "1234-5678-9101", 1000.50, 2000.75, 120, 30),

new Phone(2, "Петров", "Петр", "Петрович", "ул. Садовая, 5", "2345-6789-0123", 1500.00, 2500.00, 90, 0),

new Phone(3, "Сидоров", "Сидор", "Сидорович", "ул. Центральная, 1", "3456-7890-1234", 2000.25, 1000.50, 200, 50)

};

// Вывод абонентов с городскими разговорами более 100 минут

Phone.printCityCallExceed(phones, 100);

// Вывод абонентов, которые пользовались междугородней связью

Phone.printLongDistanceUsers(phones);

// Вывод абонентов в алфавитном порядке

Phone.printAlphabeticalOrder(phones);

}

}

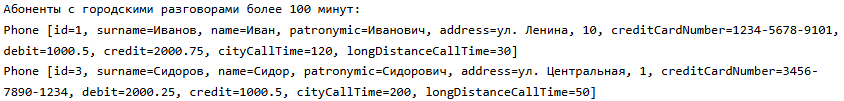


Рисунок 5 – Результат программы задачи 2

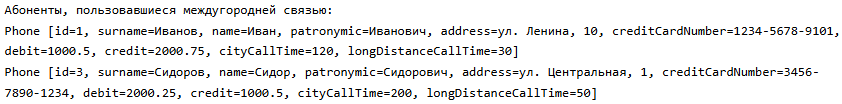


Рисунок 6 – Результат программы задачи 2

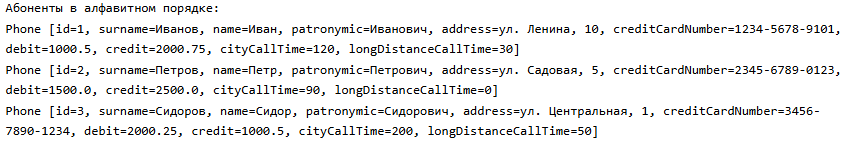


Рисунок 7 – Результат программы задачи 2

1.3 Задача 3

Создать объект класса Фотоальбом, используя классы Фотография, Страница. Методы: задать название фотографии, дополнить фотоальбом фотографией, вывести на консоль количество фотографий.

Код программы:

import java.util.ArrayList;

import java.util.List;

// Класс Фотография

class Photograph {

private String name;

// Конструктор

public Photograph(String name) {

this.name = name;

}

// Метод для задания названия фотографии

public void setName(String name) {

this.name = name;

}

// Метод для получения названия фотографии

public String getName() {

return name;

}

@Override

public String toString() {

return "Photograph{name='" + name + "'}";

}

}

// Класс Страница

class Page {

private List<Photograph> photographs;

// Конструктор

public Page() {

photographs = new ArrayList<>();

}

// Метод для добавления фотографии на страницу

public void addPhotograph(Photograph photograph) {

photographs.add(photograph);

}

// Метод для получения всех фотографий на странице

public List<Photograph> getPhotographs() {

return photographs;

}

}

// Класс Фотоальбом

class PhotoAlbum {

private List<Page> pages;

// Конструктор

public PhotoAlbum() {

pages = new ArrayList<>();

}

// Метод для добавления страницы в альбом

public void addPage(Page page) {

pages.add(page);

}

// Метод для добавления фотографии на последнюю страницу альбома

public void addPhotograph(Photograph photograph) {

if (pages.isEmpty()) {

Page newPage = new Page();

newPage.addPhotograph(photograph);

addPage(newPage);

} else {

pages.get(pages.size() - 1).addPhotograph(photograph);

}

}

// Метод для вывода на консоль количества фотографий в альбоме

public void printPhotoCount() {

int count = 0;

for (Page page : pages) {

count += page.getPhotographs().size();

}

System.out.println("Количество фотографий в альбоме: " + count);

}

}

public class Main {

public static void main(String[] args) {

// Создание объекта класса PhotoAlbum

PhotoAlbum album = new PhotoAlbum();

// Создание фотографий

Photograph photo1 = new Photograph("Отпуск 2024");

Photograph photo2 = new Photograph("Свадьба");

// Добавление фотографий в альбом

album.addPhotograph(photo1);

album.addPhotograph(photo2);

// Вывод количества фотографий в альбоме

album.printPhotoCount();

}

}

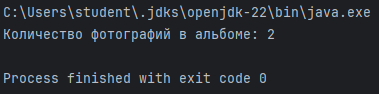


Рисунок 8 – Результат программы задачи 3

1.4 Задача 4

Создать класс Фильм с внутренним классом, с помощью объектов которого можно хранить информацию о продолжительности, жанре и режиссерах фильма.

Код программы:

import java.util.List;

import java.util.ArrayList;

// Класс Фильм

class Film {

private String title; // Название фильма

private FilmDetails details; // Детали фильма

// Конструктор класса Film

public Film(String title, int duration, String genre, List<String> directors) {

this.title = title;

this.details = new FilmDetails(duration, genre, directors);

}

// Внутренний класс FilmDetails

public class FilmDetails {

private int duration; // Продолжительность фильма в минутах

private String genre; // Жанр фильма

private List<String> directors; // Список режиссёров

// Конструктор класса FilmDetails

public FilmDetails(int duration, String genre, List<String> directors) {

this.duration = duration;

this.genre = genre;

this.directors = new ArrayList<>(directors);

}

// Методы доступа к данным

public int getDuration() {

return duration;

}

public String getGenre() {

return genre;

}

public List<String> getDirectors() {

return directors;

}

@Override

public String toString() {

return "Продолжительность: " + duration + " минут, Жанр: " + genre + ", Режиссёры: " + directors;

}

}

// Метод для получения информации о фильме

public String getFilmInfo() {

return "Название фильма: " + title + ", " + details.toString();

}

// Метод для изменения деталей фильма

public void updateDetails(int duration, String genre, List<String> directors) {

this.details = new FilmDetails(duration, genre, directors);

}

}

public class Main {

public static void main(String[] args) {

// Создаем список режиссеров

List<String> directors = new ArrayList<>();

directors.add("Кристофер Нолан");

directors.add("Квентин Тарантино");

// Создаем объект фильма с деталями

Film inception = new Film("Начало", 148, "Фантастика", directors);

// Выводим информацию о фильме

System.out.println(inception.getFilmInfo());

// Обновляем информацию о фильме

directors.add("Дэвид Финчер");

inception.updateDetails(148, "Фантастика, Триллер", directors);

// Выводим обновленную информацию о фильме

System.out.println(inception.getFilmInfo());

}

}

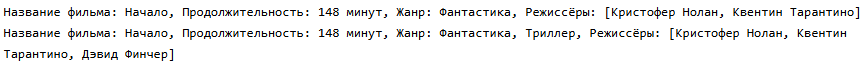


Рисунок 9 – Результат программы задачи 4

1.5 Задача 5

Преобразовать текст так, чтобы каждое слово, не содержащее неалфавитных символов, начиналось с заглавной буквы.

Код программы:

public class Main {

public static String capitalizeWords(String input) {

String[] words = input.split("\\s+"); // Разбиваем текст на слова

StringBuilder result = new StringBuilder();

for (String word : words) {

// Проверяем, содержит ли слово только алфавитные символы

if (word.matches("[a-zA-Z]+")) {

// Преобразуем первую букву в заглавную, а остальные в строчные

word = word.substring(0, 1).toUpperCase() + word.substring(1).toLowerCase();

}

// Добавляем слово к результату

result.append(word).append(" ");

}

return result.toString().trim(); // Убираем лишний пробел в конце

}

public static void main(String[] args) {

String text = "hello world! this is an Example. 123test";

String transformedText = capitalizeWords(text);

System.out.println("Исходный текст: " + text);

System.out.println("Преобразованный текст: " + transformedText);

}

}



Рисунок 10 – Результат программы задачи 5

1.6 Задача 6

Код программы:

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Pattern;

// Класс Каталог (Directory)

class Directory {

private String name;

private Directory parent;

private List<Directory> subdirectories;

private List<FileEntity> files;

// Конструктор для создания каталога

public Directory(String name, Directory parent) {

this.name = name;

this.parent = parent;

this.subdirectories = new ArrayList<>();

this.files = new ArrayList<>();

}

// Метод для получения названия каталога

public String getName() {

return name;

}

// Метод для получения родительского каталога

public Directory getParent() {

return parent;

}

// Метод для добавления подкаталога

public void addSubdirectory(Directory directory) {

subdirectories.add(directory);

}

// Метод для добавления файла в каталог

public void addFile(FileEntity file) {

files.add(file);

}

// Метод для получения подкаталогов

public List<Directory> getSubdirectories() {

return subdirectories;

}

// Метод для получения файлов

public List<FileEntity> getFiles() {

return files;

}

// Метод для получения полного пути каталога

public String getFullPath() {

if (parent == null) {

return name;

} else {

return parent.getFullPath() + "/" + name;

}

}

}

// Класс Файл (FileEntity)

class FileEntity {

private String name;

private Directory parent;

private int size; // Размер файла в байтах

// Конструктор для создания файла

public FileEntity(String name, int size, Directory parent) {

this.name = name;

this.size = size;

this.parent = parent;

}

// Метод для получения названия файла

public String getName() {

return name;

}

// Метод для получения размера файла

public int getSize() {

return size;

}

// Метод для получения родительского каталога

public Directory getParent() {

return parent;

}

// Метод для получения полного пути файла

public String getFullPath() {

return parent.getFullPath() + "/" + name;

}

}

// Класс для управления файловой системой

class FileSystem {

private Directory root;

// Конструктор, создающий корневой каталог

public FileSystem() {

root = new Directory("root", null);

}

// Метод для получения корневого каталога

public Directory getRoot() {

return root;

}

// Метод для подсчета файлов в каталоге (включая вложенные)

public int countFiles(Directory directory) {

int count = directory.getFiles().size();

for (Directory subdirectory : directory.getSubdirectories()) {

count += countFiles(subdirectory);

}

return count;

}

// Метод для подсчета места, занимаемого файлами в каталоге (включая вложенные)

public int calculateSize(Directory directory) {

int size = 0;

for (FileEntity file : directory.getFiles()) {

size += file.getSize();

}

for (Directory subdirectory : directory.getSubdirectories()) {

size += calculateSize(subdirectory);

}

return size;

}

// Метод для поиска файлов по маске с выдачей полного пути

public List<String> findFilesByMask(Directory directory, String mask) {

List<String> foundFiles = new ArrayList<>();

Pattern pattern = Pattern.compile(mask.replace("\*", ".\*")); // Преобразуем маску в регулярное выражение

// Проверяем файлы в текущем каталоге

for (FileEntity file : directory.getFiles()) {

if (pattern.matcher(file.getName()).matches()) {

foundFiles.add(file.getFullPath());

}

}

// Рекурсивно проверяем подкаталоги

for (Directory subdirectory : directory.getSubdirectories()) {

foundFiles.addAll(findFilesByMask(subdirectory, mask));

}

return foundFiles;

}

// Метод для перемещения файлов и подкаталогов в другой каталог

public void moveContents(Directory source, Directory destination) {

// Перемещаем все файлы

for (FileEntity file : source.getFiles()) {

destination.addFile(file);

}

source.getFiles().clear(); // Очищаем список файлов в исходном каталоге

// Перемещаем все подкаталоги

for (Directory subdirectory : source.getSubdirectories()) {

destination.addSubdirectory(subdirectory);

}

source.getSubdirectories().clear(); // Очищаем список подкаталогов

}

// Метод для удаления всех файлов и подкаталогов в каталоге

public void deleteContents(Directory directory) {

directory.getFiles().clear();

directory.getSubdirectories().clear();

}

}

public class Main {

public static void main(String[] args) {

FileSystem fileSystem = new FileSystem();

Directory root = fileSystem.getRoot();

// Создаем подкаталоги и файлы

Directory photos = new Directory("Photos", root);

root.addSubdirectory(photos);

Directory videos = new Directory("Videos", root);

root.addSubdirectory(videos);

FileEntity photo1 = new FileEntity("beach.jpg", 500, photos);

photos.addFile(photo1);

FileEntity video1 = new FileEntity("vacation.mp4", 1500, videos);

videos.addFile(video1);

// Пример использования методов

// Определяем полный путь файла

System.out.println("Полный путь к файлу: " + photo1.getFullPath());

// Подсчитываем количество файлов в каталоге

System.out.println("Количество файлов в каталоге Photos: " + fileSystem.countFiles(photos));

// Подсчитываем место, занимаемое на диске

System.out.println("Место, занимаемое в каталоге Videos: " + fileSystem.calculateSize(videos) + " байт");

// Поиск файлов по маске

System.out.println("Файлы по маске '\*.jpg': " + fileSystem.findFilesByMask(root, "\*.jpg"));

// Перемещение содержимого каталога Photos в каталог Videos

fileSystem.moveContents(photos, videos);

System.out.println("Количество файлов в каталоге Videos после перемещения: " + fileSystem.countFiles(videos));

// Удаление всех файлов и подкаталогов в каталоге Videos

fileSystem.deleteContents(videos);

System.out.println("Количество файлов в каталоге Videos после удаления: " + fileSystem.countFiles(videos));

}

}

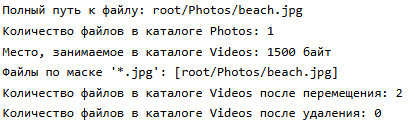


Рисунок 11 – Результат программы задачи 6

1.7 Задача 7

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

Код программы:

import javax.swing.\*;  
import java.awt.\*;  
  
public class SnowmanDrawing extends JPanel {  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
  
 // Рисуем тело снеговика  
 g.setColor(Color.*WHITE*);  
 g.fillOval(120, 200, 150, 150); // Нижний шар  
 g.fillOval(140, 110, 110, 110); // Средний шар  
 g.fillOval(160, 50, 80, 80); // Голова  
  
 // Рисуем ведро на голове  
 g.setColor(Color.*BLACK*);  
 g.fillRect(180, 30, 50, 10); // Крышка  
 g.fillRect(170, 40, 60, 20); // Основная часть ведра  
  
 // Рисуем глаза  
 g.setColor(Color.*BLACK*);  
 g.fillOval(180, 70, 10, 10); // Левый глаз  
 g.fillOval(210, 70, 10, 10); // Правый глаз  
  
 // Рисуем нос (морковка)  
 g.setColor(Color.*ORANGE*);  
 Polygon carrot = new Polygon();  
 carrot.addPoint(195, 80);  
 carrot.addPoint(210, 85);  
 carrot.addPoint(195, 90);  
 g.fillPolygon(carrot);  
  
 // Рисуем метлу  
 g.setColor(new Color(139, 69, 19)); // Коричневый цвет  
 g.fillRect(300, 95, 10, 50); // Рукоятка метлы  
 g.setColor(Color.*GRAY*); // Цвет щетинки метлы  
 g.fillRect(285, 90, 40, 10); // Щетинка  
  
 // Рисуем руки  
 g.setColor(new Color(139, 69, 19)); // Коричневый цвет для рук  
 g.fillRect(90, 130, 60, 10); // Левой рука  
 g.fillRect(240, 130, 60, 10); // Правой рука  
  
 // Рисуем пуговицы  
 g.setColor(Color.*BLACK*);  
 g.fillOval(190, 130, 10, 10); // Первая пуговка  
 g.fillOval(190, 150, 10, 10); // Вторая пуговка  
 g.fillOval(190, 170, 10, 10); // Третья пуговка  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Snowman Drawing");  
 SnowmanDrawing snowman = new SnowmanDrawing();  
 frame.add(snowman);  
 frame.setSize(400, 400);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 }  
}



Рисунок 12 – Результат программы задачи 7

1.8 Задача 8

Определить объект TFish аквариумная рыбка, имеющий координаты, скорость, размер, цвет и направление движения. Методами этого объекта являются: Init устанавливает значения полей объекта и рисует рыбку на экране методом Draw; Draw рисует рыбку в виде уголка с острием в точке Coord, направленным по ходу ее движения; Look проверяет несколько точек на линии движения рыбки, и если хотя бы одна из них отличается по цвету от воды, воз- вращает ей цвет и указывает расстояние до рыбки; Run перемещает рыбку в текущем направлении на расстояние, зависящее от ее текущей скорости. Иногда случайным образом изменяет направление движения рыбки. Если же на пути рыбки возникает препятствие, направление движения изменяется до тех пор, пока препятствие не исчезнет из ее поля зрения.

Код программы:

import javax.swing.\*;  
import java.awt.\*;  
import java.util.Random;  
  
class TFish {  
 private Point coord; // Координаты рыбки  
 private double speed; // Скорость  
 private int size; // Размер  
 private Color color; // Цвет  
 private double direction; // Направление движения в градусах  
 private Random random = new Random();  
  
 public TFish() {  
 this.coord = new Point(0, 0);  
 this.speed = 1.0;  
 this.size = 10;  
 this.color = Color.*ORANGE*;  
 this.direction = 0; // Направление в градусах  
 }  
  
 public void Init(Point coord, double speed, int size, Color color, double direction) {  
 this.coord = coord;  
 this.speed = speed;  
 this.size = size;  
 this.color = color;  
 this.direction = direction;  
 Draw();  
 }  
  
 private void Draw() {  
 }  
  
 public void Draw(Graphics g) {  
 g.setColor(color);  
 int x = coord.x;  
 int y = coord.y;  
 int[] xPoints = {x, x + size, x};  
 int[] yPoints = {y, y + size / 2, y - size / 2};  
 g.fillPolygon(xPoints, yPoints, 3);  
 }  
  
 public Color Look(Color[][] aquarium, int width, int height) {  
 int steps = 5; // Количество проверяемых точек  
 double radian = Math.*toRadians*(direction);  
  
 for (int i = 1; i <= steps; i++) {  
 int checkX = (int) (coord.x + i \* speed \* Math.*cos*(radian));  
 int checkY = (int) (coord.y + i \* speed \* Math.*sin*(radian));  
  
 // Проверка границ аквариума  
 if (checkX < 0 || checkX >= width || checkY < 0 || checkY >= height) {  
 continue;  
 }  
  
 if (aquarium[checkY][checkX] != Color.*BLUE*) { // Допустим, цвет воды - синий  
 return aquarium[checkY][checkX]; // Возвращаем цвет, если обнаружен объект  
 }  
 }  
 return null; // Все точки как цвет воды  
 }  
  
 public void Run() {  
 double angleChange = random.nextDouble() \* 20 - 10; // Изменение угла  
 direction += angleChange;  
  
 // Перемещение  
 double radian = Math.*toRadians*(direction);  
 coord.x += (int) (speed \* Math.*cos*(radian));  
 coord.y += (int) (speed \* Math.*sin*(radian));  
  
 // Ограничиваем координаты аквариума  
 coord.x = Math.*max*(0, Math.*min*(coord.x, 400)); // Допустим, ширина аквариума 400  
 coord.y = Math.*max*(0, Math.*min*(coord.y, 400)); // Допустим, высота аквариума 400  
 }  
  
 public Point getCoord() {  
 return coord;  
 }  
}  
  
// Пример использования в JFrame  
 class Aquarium extends JPanel {  
 private TFish fish;  
  
 public Aquarium() {  
 fish = new TFish();  
 fish.Init(new Point(200, 200), 2.0, 20, Color.*ORANGE*, 45.0);  
 }  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 fish.Draw(g);  
 }  
  
 public void update() {  
 fish.Run();  
 repaint();  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Aquarium");  
 Aquarium aquarium = new Aquarium();  
 frame.add(aquarium);  
 frame.setSize(400, 400);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
  
 // Обновляем аквариум  
 Timer timer = new Timer(100, e -> aquarium.update());  
 timer.start();  
 }  
}

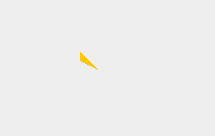


Рисунок 13 – Результат программы задачи 8

1.9 Задача 9

Реализовать программу «Итальянская игра "Математико"»

Таблица 9 - Список идентификаторов

|  |  |  |
| --- | --- | --- |
| Название | Тип | Параметр |
| playerField, computerField | int[][] | Вспомогательная |
| currentCard, score | int | Вспомогательная |

public class MatematikoGameGUI extends JFrame {

int[][] playerField = new int[5][5];

int[][] computerField = new int[5][5];

List<Integer> deck = new ArrayList<>();

JLabel[][] playerLabels = new JLabel[5][5];

JLabel currentCardLabel;

int currentCard = -1;

JLabel resultLabel;

public MatematikoGameGUI() {

setTitle("Математико\_0");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(666, 666);

setLayout(new BorderLayout());

resultLabel = new JLabel("Результаты будут показаны здесь.", SwingConstants.CENTER);

resultLabel.setFont(new Font("Arial", Font.BOLD, 16));

add(resultLabel, BorderLayout.NORTH);

JPanel gridPanel = new JPanel(new GridLayout(5, 5));

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

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

playerLabels[i][j] = new JLabel("", SwingConstants.CENTER);

playerLabels[i][j].setBorder(BorderFactory.createLineBorder(Color.BLACK));

playerLabels[i][j].setFont(new Font("Arial", Font.BOLD, 20));

int row = i;

int col = j;

playerLabels[i][j].addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

if (currentCard != -1 && playerField[row][col] == 0) {

placeNumber(currentCard, row, col, playerField);

playerLabels[row][col].setText(String.valueOf(currentCard));

currentCard = -1;

currentCardLabel.setText("Выберите новую карту");

if (isPlayerFieldFull() && isComputerFieldFull()) {

calculateResults();

} else {

computerMove();

if (isPlayerFieldFull() && isComputerFieldFull()) {

calculateResults();

}}}}});

gridPanel.add(playerLabels[i][j]);

}}

add(gridPanel, BorderLayout.CENTER);

JPanel controlPanel = new JPanel();

JButton drawCardButton = new JButton("Вытянуть карту");

currentCardLabel = new JLabel("Выберите карту");

controlPanel.add(drawCardButton);

controlPanel.add(currentCardLabel);

add(controlPanel, BorderLayout.SOUTH);

drawCardButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (!deck.isEmpty() && !isPlayerFieldFull()) {

currentCard = drawCard();

currentCardLabel.setText("Карта: " + currentCard);

} else {

currentCardLabel.setText("Все карты вытянуты");

if (isPlayerFieldFull() && isComputerFieldFull()) {

calculateResults();

}}}});

for (int i = 1; i <= 13; i++) {

for (int j = 0; j < 4; j++) {

deck.add(i);

}}

Collections.shuffle(deck);

}

public int drawCard() {

return deck.remove(deck.size() - 1);

}

public void placeNumber(int num, int row, int col, int[][] field) {

field[row][col] = num;

}

public void computerMove() {

while (!deck.isEmpty()) {

int card = drawCard();

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

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

if (computerField[i][j] == 0) {

placeNumber(card, i, j, computerField);

return;

}}}}}

public boolean isPlayerFieldFull() {

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

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

if (playerField[i][j] == 0) {

return false;

}}}

return true;

}

public boolean isComputerFieldFull() {

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

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

if (computerField[i][j] == 0) {

return false;

}}}

return true;

}

public void calculateResults() {

int playerScore = calculateScore(playerField);

int computerScore = calculateScore(computerField);

resultLabel.setText("Ваши очки: " + playerScore + ", Очки компьютера: " + computerScore);

String message;

if (playerScore > computerScore) {

message = "Вы выиграли!\nВаши очки: " + playerScore + "\nОчки компьютера: " + computerScore;

} else if (playerScore < computerScore) {

message = "Вы проиграли.\nВаши очки: " + playerScore + "\nОчки компьютера: " + computerScore;

} else {

message = "Ничья.\nВаши очки: " + playerScore + "\nОчки компьютера: " + computerScore;

}

JOptionPane.showMessageDialog(this, message, "Итоги игры", JOptionPane.INFORMATION\_MESSAGE);

}

public int calculateScore(int[][] field) {

int score = 0;

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

score += checkLine(field[i]);

int[] column = new int[5];

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

column[j] = field[j][i];

}

score += checkLine(column);

}

int[] diagonal1 = {field[0][0], field[1][1], field[2][2], field[3][3], field[4][4]};

int[] diagonal2 = {field[0][4], field[1][3], field[2][2], field[3][1], field[4][0]};

score += checkLine(diagonal1);

score += checkLine(diagonal2);

return score;

}

public int checkLine(int[] line) {

int score = 0;

Map<Integer, Integer> counts = new HashMap<>();

for (int num : line) {

counts.put(num, counts.getOrDefault(num, 0) + 1);

}

List<Integer> numbers = new ArrayList<>(counts.keySet());

Collections.sort(numbers);

if (counts.containsValue(2)) {

if (isDiagonal(line)) {

score += 20;

} else {

score += 10;

}}

if (counts.containsValue(3)) {

if (isDiagonal(line)) {

score += 30;

} else {

score += 20;

}}

if (counts.containsValue(3) && counts.containsValue(2)) {

if (isDiagonal(line)) {

score += 90;

} else {

score += 80;

}}

if (counts.containsValue(4)) {

if (isDiagonal(line)) {

score += 170;

} else {

score += 160;

}}

if (numbers.size() == 5 && numbers.get(4) - numbers.get(0) == 4) {

if (isDiagonal(line)) {

score += 60;

} else {

score += 50;

}}

if (counts.getOrDefault(1, 0) == 3 && counts.getOrDefault(13, 0) == 2) {

score += 100;

}

if (numbers.containsAll(Arrays.asList(1, 13, 12, 11, 10))) {

if (isDiagonal(line)) {

score += 160;

} else {

score += 150;

}}

if (counts.getOrDefault(1, 0) == 4) {

if (isDiagonal(line)) {

score += 210;

} else {

score += 200;

}}

return score;

}

public boolean isDiagonal(int[] line) {

return (line[0] != 0 && line[1] != 0 && line[2] != 0 && line[3] != 0 && line[4] != 0);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

MatematikoGameGUI game = new MatematikoGameGUI();

game.setVisible(true);

}});}}

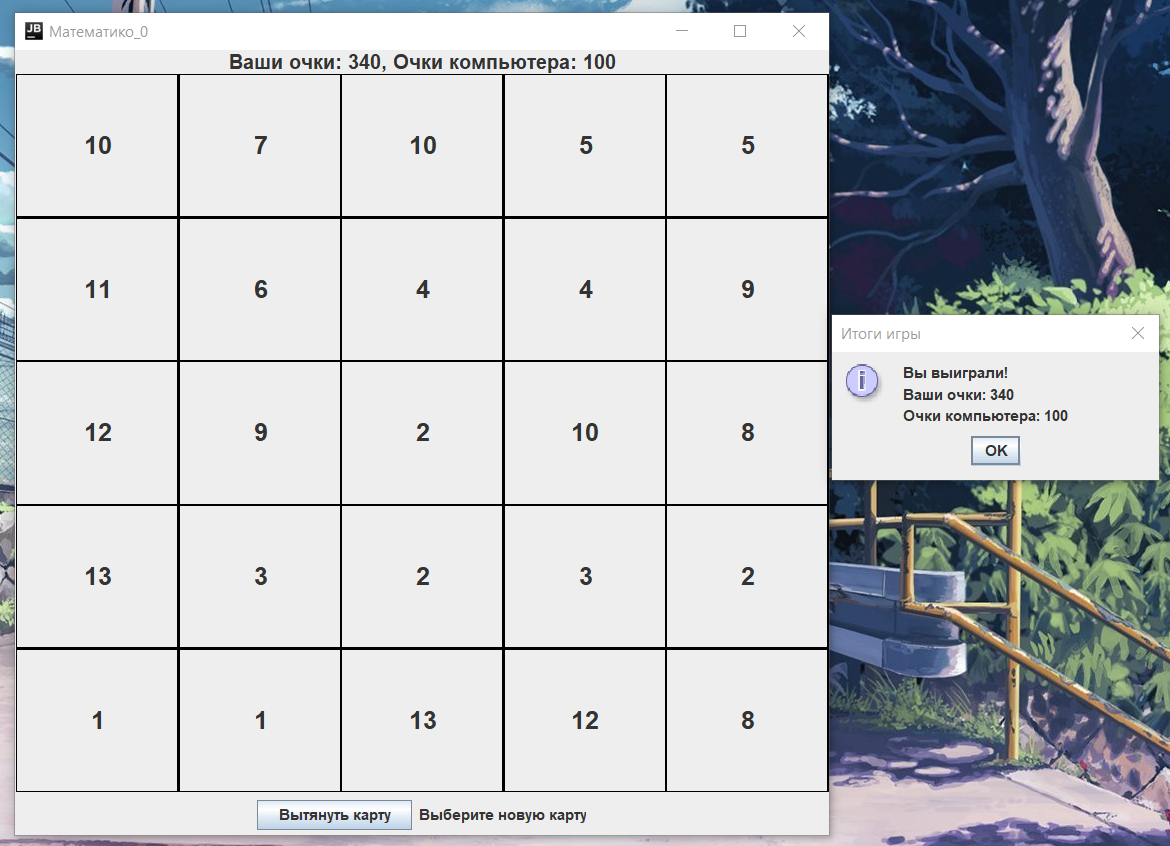


Рисунок 14 – Результат программы задачи 9

1.10 Задача 10

Задача 1

N DQ ? ; число без знака Не используя арифметические команды, реализовать оператор (1 вариант) N := N \* 2 Решение должно содержать не более трёх команд."

Задача 2

1 вариант: Реализовать процедуру Padd(x, a), которая реализует оператор x := x + a. Передача параметров в стеке, стандартные соглашения о связях.

section .data

N dd 5

result db 'Result: ', 0

newline db 10, 0

section .bss

buf resb 10

section .text

global \_start

\_start:

mov eax, [N]

shl eax, 1

mov [N], eax

mov edi, buf

call int\_to\_string

mov eax, 1

mov edi, 1

mov rsi, result

mov edx, 8

syscall

mov eax, 1

mov edi, 1

mov rsi, buf

mov edx, 10

syscall

mov eax, 1

mov edi, 1

mov rsi, newline

mov edx, 1

syscall

mov eax, 60

xor edi, edi

syscall

int\_to\_string:

mov ecx, 10

xor ebx, ebx

convert\_loop:

xor edx, edx

div ecx

add dl, '0'

mov [rdi], dl

inc rdi

test eax, eax

jnz convert\_loop

ret

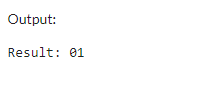


Рисунок 15 – Результат первого задания

section .data

msg db "Result: ", 0

result db 0, 0, 0, 0, 0

section .text

global \_start

\_start:

mov rdi, 10

mov rsi, 5

call Padd

mov rbx, result

call int\_to\_str

mov rax, 1

mov rdi, 1

mov rsi, msg

mov rdx, 8

syscall

mov rax, 1

mov rdi, 1

mov rsi, result

mov rdx, 5

syscall

mov rax, 60

xor rdi, rdi

syscall

Padd:

add rdi, rsi

mov rax, rdi

ret

int\_to\_str:

mov rcx, 0

mov rbx, result

convert\_loop:

xor rdx, rdx

div rsi

add dl, '0'

mov [rbx + rcx], dl

inc rcx

test rax, rax

jnz convert\_loop

mov rdi, result

mov rsi, rbx

dec rsi

reverse\_loop:

cmp rdi, rsi

jge reverse\_done

mov al, [rdi]

mov bl, [rsi]

mov [rdi], bl

mov [rsi], al

inc rdi

dec rsi

jmp reverse\_loop

reverse\_done:

ret

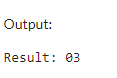


Рисунок 16 – Результат второго задания

# 2. Разработка АИС

Концепция БД: туристическая фирма

В качестве серверной части разработать базу данных в СУБД MySQL в соответствии с вариантом. База данных содержит таблицу, состоящую из не менее чем 5-ти полей и 20-ти записей [7].

Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

Таким образом, структура клиента должна быть такой:

1. Класс, содержащий статический метод main() для демонстрации возможностей программы.

2. Класс, содержащий поля и методы для работы с удалённой базой данных:

3. Обязательные методы (запросы) клиента:

- выборка содержимого таблиц базы данных;

- запись в файл результата выборки содержимого таблиц БД.

- вывод в консоль результатов выполнения каждого метода.

4. Дополнительные методы (запросы) клиента

a. добавление записи в таблицу.

b. удаление записи из таблицы.

c. обновление записи в таблице.

d. поиск записи по признаку (на ваш выбор).

## 2.1 Техническое задание

1. Общие сведения

1.1. Название системы: «Туристическая фирма»;

1.2. Разработчик: Сангзода Джахонгир Талаби;

1.3. Основание для разработки: система разрабатывается для автоматизации управления турами и аутентификации пользователей в туристическом агентстве.

2. Назначение системы

2.1. Система предназначена для автоматизации процесса управления турами (создание, редактирование, удаление и просмотр списка туров), аутентификации пользователей (вход, выход, проверка прав доступа).

2.2. Система должна обеспечивать:

Возможность добавления, редактирования и удаления туров;

Хранение информации о каждом туре, включая: место назначения, стоимость тура и даты начала и окончания;

Отображение списка всех туров с возможностью просмотра подробностей.

3. Требования к системе

3.1. Функциональные требования

Модуль управления заказами:

Возможность создания нового заказа с указанием необходимых данных: идентификатор заказа, место назначения, наименование тура, дата заказа, дата начала и окончания тура, количество участников, стоимость заказа.

Модуль учета туров:

Добавление тура: создание нового тура с указанием места назначения, цены и дат начала и окончания.

Просмотр списка туров: отображение списка доступных туров с возможностью просмотра детальной информации.

Редактирование тура: изменение данных о туре (место назначения, цена, даты).

Удаление тура: удаление тура из системы.

Модуль учета сотрудников:

Хранение данных сотрудников (идентификатор, ФИО, должность, фотография, паспортные данные, телефон, стаж, заработная плата, фотография сотрудника).

Модуль аутентификации и управления доступом:

Авторизация пользователей, управление сессией, разграничение прав доступа, безопасность данных.

3.2. Нефункциональные требования:

Удобный пользовательский интерфейс для просмотра и редактирования информации;

Легкий доступ к основным модулям (управление заказами, учет сотрудников, учет туров) с понятной навигацией;

Система должна быть структурирована и оптимизирована для быстрого выполнения запросов и эффективной обработки данных при увеличении их объема.

4. Требования к интерфейсу пользователя

4.1. Интерфейс пользователя системы "Туристическая фирма" должен обеспечивать простоту использования, интуитивную навигацию и доступность всех необходимых функций для выполнения задач.

4.2. Интерфейс должен быть адаптирован для работы на компьютерах.

5. Требования к безопасности

5.1. Доступ к системе должен быть защищён системой аутентификации пользователей (логин и пароль);

5.2. Необходимо разграничить доступы по ролям (например, администратор, менеджер, кассир, клиент).

## 2.2 Краткая спецификация разрабатываемой системы

Цель системы

Создание системы управления турами, обработка заказов и учет сотрудников для упрощения работы с клиентами и турами, повышения прозрачности операций и эффективности учета в туристическом агентстве.

Сущности системы

1. Заявки

Создание, редактирование, просмотр, удаление заявок; изменение статусов (например, «Оплачен», «Забронирован»).

2. Туры

Учет и управление данными о доступных для бронирования турах.

Добавление, редактирование, удаление туров; просмотр информации о турах (место назначения, цена, даты).

3. Услуги

Создание и редактирование списка услуг; привязка услуг к турам и заявкам.

4. Сотрудники

Содержит данные о сотрудниках, такие как ФИО, должность, стаж, паспортные данные, фотография, контактная информация и зарплата.

Основные функции системы:

1. Управление заявками;

2. Управление турами;

3. Управление услугами;

4. Управление сотрудниками.

## 2.3 ER-диаграмма

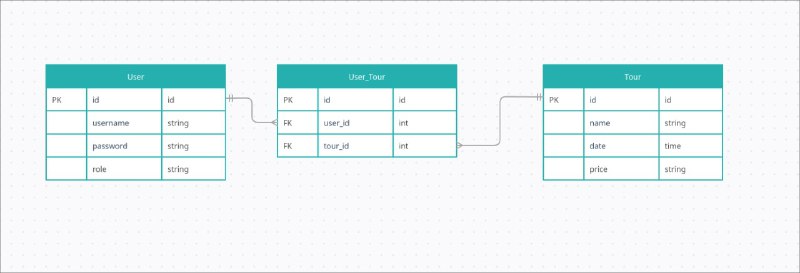


Рисунок 17 – ER-диаграмма

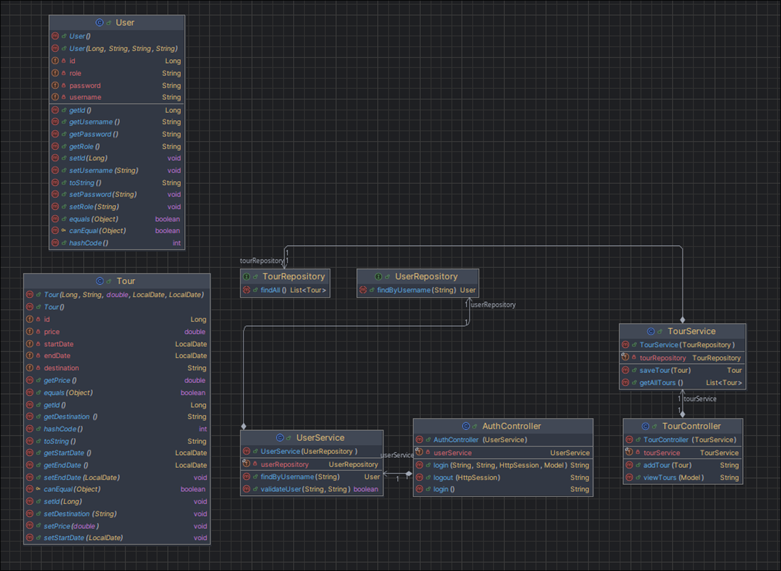


Рисунок 18 – Диаграмма классов

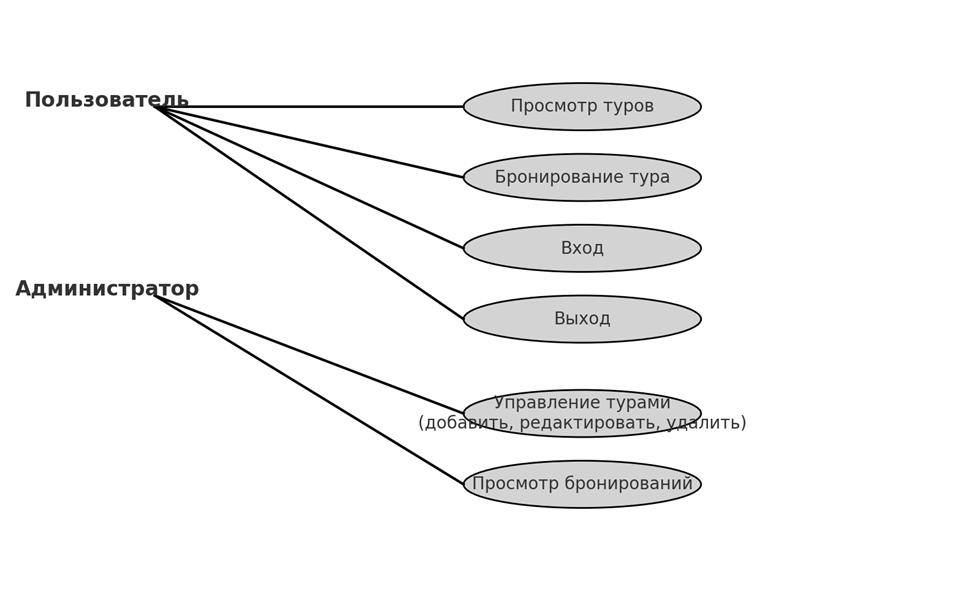


Рисунок 19 – Диаграмма использования

Код

@Controller

public class AuthController {

private final UserService userService;

public AuthController(UserService userService) {

this.userService = userService;

}

@GetMapping("/login")

public String login() {

return "login";

}

@PostMapping("/login")

public String login(String username, String password, HttpSession session, Model model) {

if (userService.validateUser(username, password)) {

session.setAttribute("username", username);

return "redirect:/tours";

} else {

model.addAttribute("error", "Неправильное имя пользователя или пароль");

return "login";

}

}

@GetMapping("/logout")

public String logout(HttpSession session) {

session.invalidate();

return "redirect:/login";

}

}

@Controller

public class TourController {

private final TourService tourService;

public TourController(TourService tourService) {

this.tourService = tourService;

}

@GetMapping("/tours")

public String viewTours(Model model) {

model.addAttribute("tours", tourService.getAllTours());

return "tours";

}

@PostMapping("/tours/add")

public String addTour(Tour tour) {

tourService.saveTour(tour);

return "redirect:/tours";

}

}

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Tour {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String destination;

private double price;

private LocalDate startDate;

private LocalDate endDate;

}

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Long id;

private String username;

private String password;

private String role;

}

@Repository

public interface TourRepository extends CrudRepository<Tour, Long> {

List<Tour> findAll();

}

@Repository

public interface UserRepository extends CrudRepository<User, Long> {

User findByUsername(String username);

}

@Service

public class TourService {

private final TourRepository tourRepository;

public TourService(TourRepository tourRepository) {

this.tourRepository = tourRepository;

}

public List<Tour> getAllTours() {

return tourRepository.findAll();

}

public Tour saveTour(Tour tour) {

return tourRepository.save(tour);

}

}

@Service

public class UserService {

private final UserRepository userRepository;

public UserService(UserRepository userRepository) {

this.userRepository = userRepository;

}

public User findByUsername(String username) {

return userRepository.findByUsername(username);

}

public boolean validateUser(String username, String password) {

User user = findByUsername(username);

return user != null && user.getPassword().equals(password);

}

}

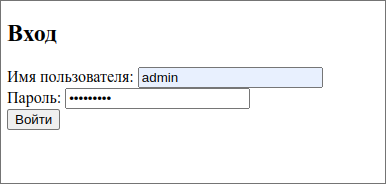


Рисунок 20 – Результат программы «Туристическая фирма»

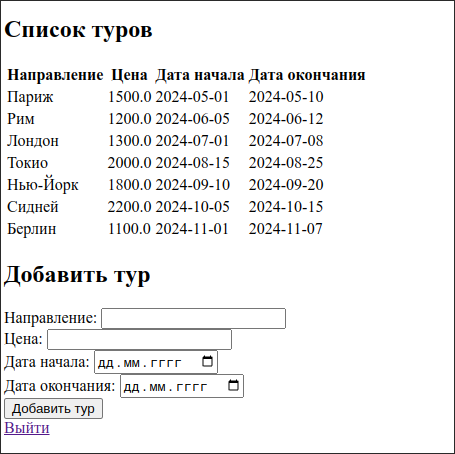


Рисунок 21 – Результат программы «Туристическая фирма»

2.4 Тестирование АИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Тест | Ожидаемый результат | Результат |
| Отображение данных | | | | |
| 1 | Вывод таблиц | Отображается таблица с данными | Таблицы отображаются | Успешно |
| 2 | Добавление данных | Отображаются добавленные данные | Отображаются данные, добавленные в таблицу | Успешно |
| 3 | Удаление данных | Удаляются данные из таблицы | Данные удалены и не отображаются | Успешно |
| 4 | Создание таблицы | Создается таблица и появляется в базе данных | Таблица создана и отображается в базе данных | Успешно |

## 2.5 Руководство пользователя

1. Вход в систему

Открытие страницы входа:

Перейдите на URL. Вы увидите форму входа с полями для имени пользователя и пароля.

Аутентификация:

Введите своё имя пользователя и пароль.

Нажмите кнопку «Войти».

Если имя пользователя и пароль введены правильно, вы будете перенаправлены на страницу со списком доступных туров.

Если данные неверны, на странице отобразится сообщение «Неправильное имя пользователя или пароль».

Выход из системы:

Для выхода из системы перейдите на URL. Сессия будет завершена, и вы будете перенаправлены на страницу входа.

2. Управление турами

Просмотр списка туров:

После успешного входа в систему перейдите на URL.

На этой странице отображается список всех доступных туров, включая такие данные, как место назначения, цена, дата начала и окончания.

Туроператор может просматривать подробности туров в виде таблицы или списка.

Добавление нового тура:

Для добавления тура отправьте запрос на через форму (интерфейс добавления тура нужно создать отдельно).

Введите следующую информацию о туре:

Место назначения, цена тура, дата начала, дата окончания.

После отправки формы тур будет добавлен, и вы будете перенаправлены обратно на страницу для просмотра обновленного списка.

3. Функции и роли пользователей

Администратор: Имеет полный доступ к функциям системы, включая добавление и просмотр туров, управление пользователями и настройками системы.

Менеджер: Может просматривать и добавлять туры, управлять заявками, но не имеет доступа к функциям управления пользователями.

Клиент: Может просматривать список доступных туров, но не имеет прав для их добавления или изменения.

3 Программирование 1С

Разработка конфигурации на указанную тему.

Цель разработки: создать автоматизированную и удобную систему для управления турами.

Обработка данных: ввод, редактирование, удаление, хранение

Модули: управление турами, клиентами и сотрудниками.

Технические особенности:

Конфигурация: 1С: Предприятие

Используемые инструменты разработки: справочники, отчёты

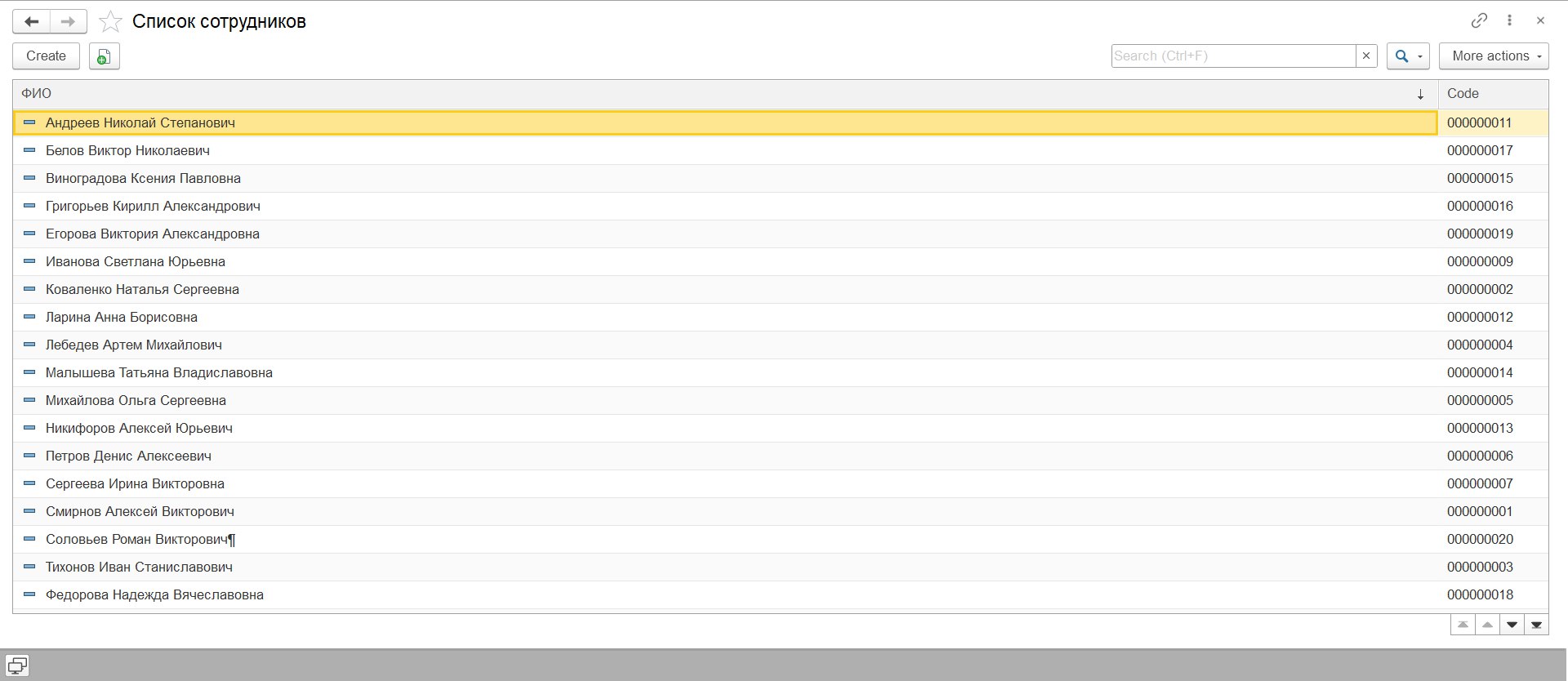


Рисунок 22 – список сотрудников

4 Разработка мобильных приложений

Разработать мобильное приложение “Туристическая фирма”.

Платформа: Android

Среда разработки: Android Studio

Язык программирования: Java

Графические элементы: TextView, EditText, ImageView, Button, CardView, ListView

Описание дизайна

Шрифт: inter

Размер текста: 18

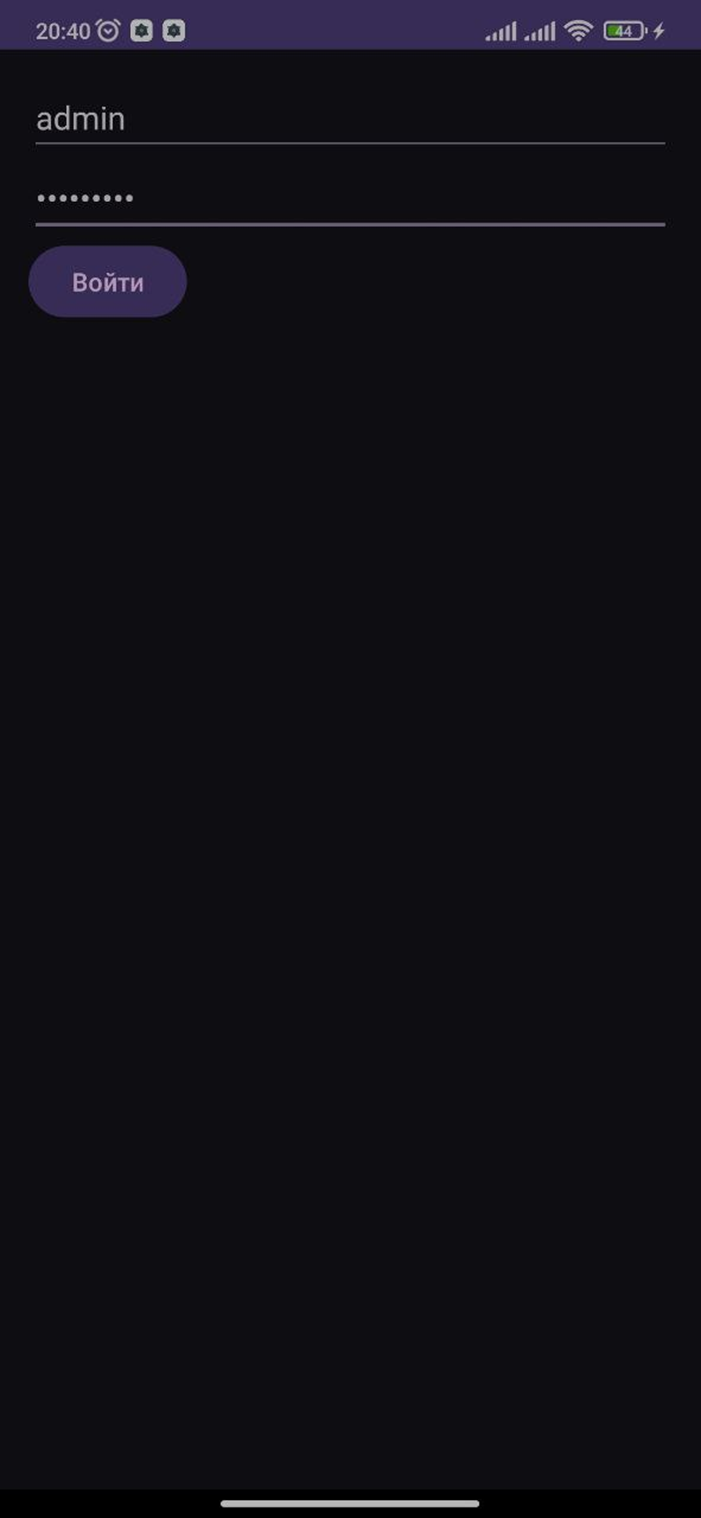


Рисунок 23 – мобильное приложение



Рисунок 24 – мобильное приложение



Рисунок 25 – мобильное приложение

5 Разработка сайта

Назначение: показ итогов учебной практики

Основные функции: просмотр выполненных работ

Технологии:

Среда разработки: Visual Studio Cod

Языки программирования: Backend Spring Boot

Хост: Локальный с туннелированием

Особенности дизайна: минималистичный

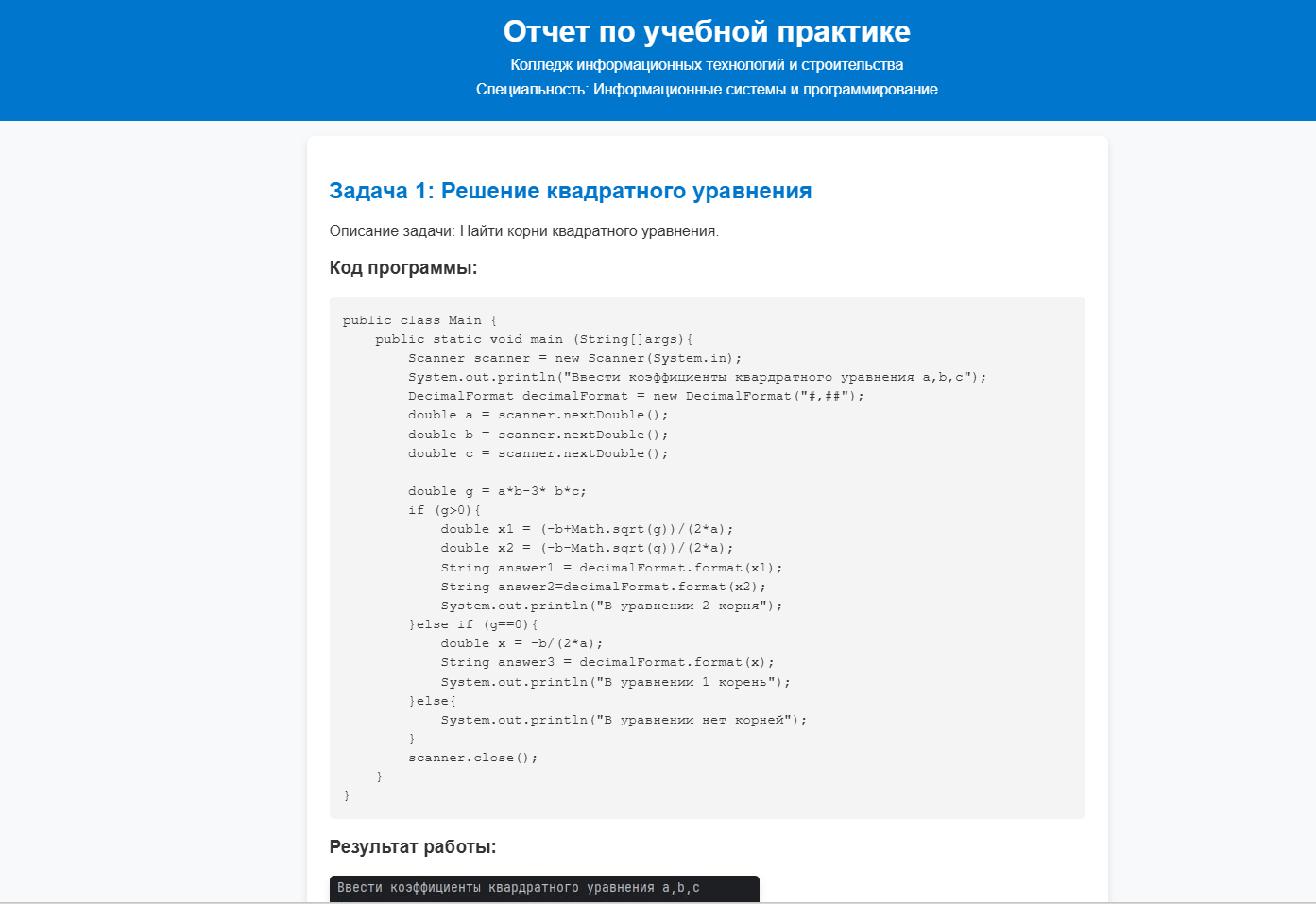


Рисунок 26 – Макет главной страницы сайта

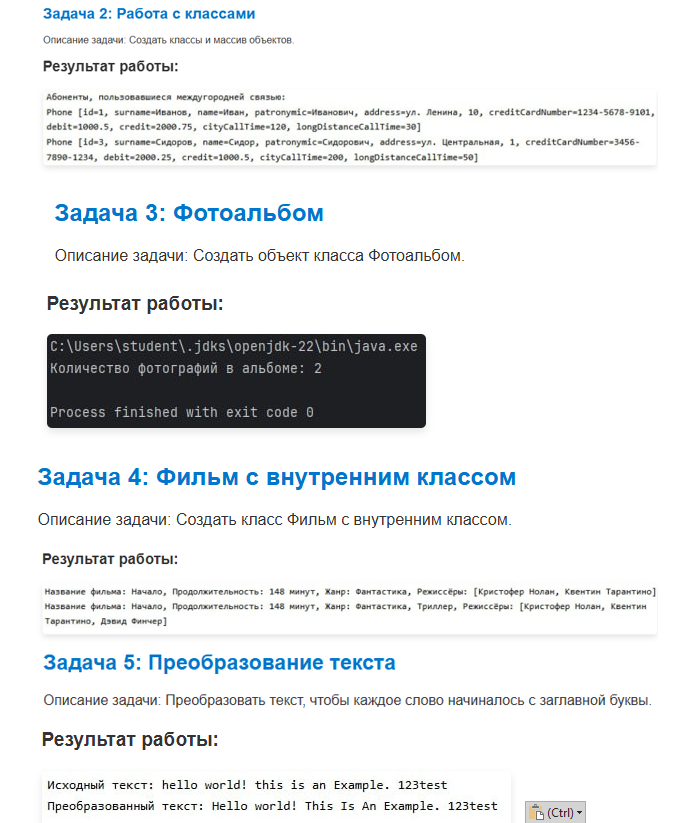


Рисунок 27 – Макет страницы с заданиями

Разработанный сайт: <https://5af2-109-237-5-247.ngrok-free.app/report>

Заключение

В ходе прохождения учебной практики по программированию удалось достичь поставленных целей и закрепить знания по ключевым дисциплинам. Система "Туристическая фирма" предоставляет удобный и безопасный способ управления турами, заявками клиентов и данными сотрудников, делая работу компании более эффективной и организованной. Ключевые функции, такие как аутентификация пользователей, гибкое распределение прав доступа, обработка заявок и управление турами, позволяют улучшить качество обслуживания и сократить операционные затраты. Это современное решение, которое автоматизирует важные процессы и становится надежной основой для успешного развития туристического бизнеса.

Список используемой литературы

1. Крейг Л. Разработка приложений с использованием Spring Boot. — М.: Издательство "Диалектика", 2019. — 560 с.
2. Руководство по созданию веб-приложений на Spring Boot, включая организацию контроллеров, сервисов и репозиториев.
3. Фримен Э. Робсон Э. Spring в действии. — М.: Вильямс, 2018. — 656 с.
4. Руководство по Spring, описывающее возможности фреймворка, управление зависимостями, безопасность и работу с данными.
5. Сьерра К., Бейтс Б. Head First по Java. — СПб.: Питер, 2016. — 720 с.
6. Основное руководство по языку Java, включая принципы объектно-ориентированного программирования и основные классы.
7. Документация по Spring Framework. Spring Framework Reference Documentation.
8. Подробное описание конфигурации, архитектуры и функций Spring Framework.
9. Документация по JUnit и Mockito.

10. Основные материалы по тестированию в Java, включая примеры юнит-тестов с использованием JUnit и Mockito.

11. Крис Ричардсон. Микросервисы. Принципы разработки и проектирования. — СПб.: Питер, 2021. — 432 с.

12. Руководство по созданию микросервисной архитектуры для организации крупных проектов на Spring Boot.