Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Основы Web-программирования»

на тему «Знакомство с ООП»

Выполнили:

студенты группы 22ВП12

Зеляпугин И.А.

Сафронов М.А.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2023

**Название**

ООП.

**Цель работы**

Познакомиться с реализацией объектно-ориентированной парадигмой на языке Java.

**Лабораторное задание**

Реализовать в рамках объектно-ориентированной парадигмы на языке Java класс «недвижимость». Реализовать иерархию минимум из 3 классов. Выполнить анализ предметной области. Разработать программу, демонстрирующую использование возможностей полученной иерархии классов.

Реализовать функции:

1. определить самое высокое здание;
2. определить здания с высотой более 50 м;
3. упорядочить массив по возрастанию высоты;
4. организовать поиск по названию здания, исправление одного из полей и вывод полной информации о здании после редактирования.

**Листинг**

Файл lb3.java

Класс Land

/\*\*  
 \* Класс, представляющий участок земли.  
 \*/  
public class Land {  
 /\*\*  
 \* Адрес участка.  
 \*/  
 protected String address;  
  
 /\*\*  
 \* Площадь участка.  
 \*/  
 protected int area;  
  
 /\*\*  
 \* Расстояние до ближайшего города.  
 \*/  
 protected int distanceToCity;  
  
 /\*\*  
 \* Цена участка.  
 \*/  
 protected int price;  
  
 /\*\*  
 \* Высшая точка на участке.  
 \*/  
 protected int height;  
  
 /\*\*  
 \* Конструктор для создания объекта участка земли.  
 \*  
 \* @param address Адрес участка.  
 \* @param area Площадь участка.  
 \* @param distanceToCity Расстояние до ближайшего города.  
 \* @param price Цена участка.  
 \* @param height Высшая точка на участке.  
 \*/  
 public Land(String address, int area, int distanceToCity, int price, int height) {  
 this.address = address;  
 this.area = area;  
 this.distanceToCity = distanceToCity;  
 this.price = price;  
 this.height = height;  
 }  
  
 /\*\*  
 \* Метод для отображения информации об участке.  
 \*/  
 public void displayInfo() {  
 System.out.print("Участок находится по адресу: " + address + ", стоит: " + price + " денег"  
 + ", занимает площадь: " + area + ", расстояние: " + distanceToCity + ", Высшая точка на участке:" + height);  
 }  
  
 /\*\*  
 \* Метод для редактирования полной информации об участке.  
 \*  
 \* @param address Новый адрес участка.  
 \* @param area Новая площадь участка.  
 \* @param distanceToCity Новое расстояние до ближайшего города.  
 \* @param price Новая цена участка.  
 \* @param height Новая высшая точка на участке.  
 \*/  
 public void editFullInfo(String address, int area, int distanceToCity, int price, int height) {  
 this.address = address;  
 this.area = area;  
 this.distanceToCity = distanceToCity;  
 this.price = price;  
 this.height = height;  
 }  
}

Класс NotResidentialArea

/\*\*  
 \* Класс, представляющий не жилую территорию (например, коммерческую недвижимость) на участке земли.  
 \*/  
public class NotResidentialArea extends Land {  
 /\*\*  
 \* Тип недвижимости на участке (например, офисное здание, магазин и т. д.).  
 \*/  
 protected String propertyType;  
  
 /\*\*  
 \* Владелец не жилой территории.  
 \*/  
 protected String owner;  
  
 /\*\*  
 \* Высота потолка в не жилой территории.  
 \*/  
 protected int heightOfFlor;  
  
 /\*\*  
 \* Конструктор для создания объекта не жилой территории на участке земли.  
 \*  
 \* @param address Адрес участка.  
 \* @param area Площадь участка.  
 \* @param distanceToCity Расстояние до ближайшего города.  
 \* @param price Цена участка.  
 \* @param height Высшая точка на участке.  
 \* @param propertyType Тип недвижимости.  
 \* @param owner Владелец не жилой территории.  
 \* @param heightOfFlor Высота потолка в не жилой территории.  
 \*/  
 public NotResidentialArea(String address, int area, int distanceToCity, int price, int height, String propertyType, String owner, int heightOfFlor) {  
 super(address, area, distanceToCity, price, height);  
 this.propertyType = propertyType;  
 this.owner = owner;  
 this.heightOfFlor = heightOfFlor;  
 }  
  
 /\*\*  
 \* Метод для отображения информации о не жилой территории на участке.  
 \*/  
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.out.print(", Тип недвижимости: " + propertyType + ", Владелец: " + owner + ", Высота потолка: " + heightOfFlor);  
 }  
}

класс IndustrialFacilities

/\*\*  
 \* Класс, представляющий промышленные сооружения на не жилой территории.  
 \*/  
public class IndustrialFacilities extends NotResidentialArea {  
 /\*\*  
 \* Название промышленного объекта.  
 \*/  
 protected String name;  
  
 /\*\*  
 \* Количество работников на промышленном объекте.  
 \*/  
 protected int numberOfEmployees;  
  
 /\*\*  
 \* Количество этажей в промышленном объекте.  
 \*/  
 protected int numberOfFloors;  
  
 /\*\*  
 \* Конструктор для создания объекта промышленного объекта в не жилой территории.  
 \*  
 \* @param address Адрес участка.  
 \* @param area Площадь участка.  
 \* @param distanceToCity Расстояние до ближайшего города.  
 \* @param price Цена участка.  
 \* @param height Высшая точка на участке.  
 \* @param propertyType Тип недвижимости на участке.  
 \* @param owner Владелец не жилой территории.  
 \* @param heightOfFlor Высота потолка в не жилой территории.  
 \* @param name Название промышленного объекта.  
 \* @param numberOfEmployees Количество работников на промышленном объекте.  
 \* @param numberOfFloors Количество этажей в промышленном объекте.  
 \*/  
 public IndustrialFacilities(String address, int area, int distanceToCity, int price, int height, String propertyType, String owner, int heightOfFlor, String name, int numberOfEmployees, int numberOfFloors) {  
 super(address, area, distanceToCity, price, height, propertyType, owner, heightOfFlor);  
 this.name = name;  
 this.numberOfEmployees = numberOfEmployees;  
 this.numberOfFloors = numberOfFloors;  
 }  
  
 /\*\*  
 \* Метод для отображения информации о промышленных объектах в не жилой территории.  
 \*/  
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.out.println("Название: " + name + ", Количество работников: " + numberOfEmployees + ", Количество этажей: " + numberOfFloors);  
 }  
}

класс ResidentialLand

/\*\*  
 \* Класс, представляющий жилую территорию на участке земли.  
 \*/  
public class ResidentialLand extends Land {  
 /\*\*  
 \* Тип жилого здания на участке (например, дом, квартира и т. д.).  
 \*/  
 protected String typeOfHouse;  
  
 /\*\*  
 \* Количество комнат в жилом здании.  
 \*/  
 protected int numbersOfRooms;  
  
 /\*\*  
 \* Конструктор для создания объекта жилой территории на участке земли.  
 \*  
 \* @param address Адрес участка.  
 \* @param area Площадь участка.  
 \* @param distanceToCity Расстояние до ближайшего города.  
 \* @param price Цена участка.  
 \* @param height Высшая точка на участке.  
 \* @param typeOfHouse Тип жилого здания.  
 \* @param numbersOfRooms Количество комнат в жилом здании.  
 \*/  
 public ResidentialLand(String address, int area, int distanceToCity, int price, int height, String typeOfHouse, int numbersOfRooms) {  
 super(address, area, distanceToCity, price, height);  
 this.typeOfHouse = typeOfHouse;  
 this.numbersOfRooms = numbersOfRooms;  
 }  
  
 /\*\*  
 \* Метод для отображения информации о жилой территории на участке.  
 \*/  
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.out.print(", тип здания: " + typeOfHouse + ", кол-во комнат: " + numbersOfRooms);  
 }  
}

класс HighRiseBuilding

/\*\*  
 \* Класс, представляющий высотное жилое здание на жилой территории.  
 \*/  
public class HighRiseBuilding extends ResidentialLand {  
 /\*\*  
 \* Количество этажей в высотном здании.  
 \*/  
 protected int floor;  
  
 /\*\*  
 \* Общая жилая площадь в высотном здании.  
 \*/  
 protected int liveSpace;  
  
 /\*\*  
 \* Конструктор для создания объекта высотного жилого здания на жилой территории.  
 \*  
 \* @param address Адрес участка.  
 \* @param area Площадь участка.  
 \* @param distanceToCity Расстояние до ближайшего города.  
 \* @param price Цена участка.  
 \* @param height Высшая точка на участке.  
 \* @param typeOfHouse Тип жилого здания.  
 \* @param numbersOfRooms Количество комнат в жилом здании.  
 \* @param floor Количество этажей в высотном здании.  
 \* @param liveSpace Общая жилая площадь в высотном здании.  
 \*/  
 public HighRiseBuilding(String address, int area, int distanceToCity, int price, int height, String typeOfHouse, int numbersOfRooms, int floor, int liveSpace) {  
 super(address, area, distanceToCity, price, height, typeOfHouse, numbersOfRooms);  
 this.floor = floor;  
 this.liveSpace = liveSpace;  
 }  
  
 /\*\*  
 \* Метод для отображения информации о высотном жилом здании на жилой территории.  
 \*/  
 @Override  
 public void displayInfo() {  
 super.displayInfo();  
 System.out.println(", кол-во этажей: " + floor + ", жилая площадь: " + liveSpace);  
 }  
}

класс Consultant

import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
*/\*\*  
 \* Класс Consultant предоставляет функциональность для управления данными об участках земли.  
 \*/*public class Consultant {  
 private List<Land> lands = new ArrayList<>();  
  
 */\*\*  
 \* Добавляет участок земли в список.  
 \*  
 \* @param land Участок земли для добавления.  
 \*/* public void addLand(Land land) {  
 lands.add(land);  
 }  
  
 */\*\*  
 \* Находит самое высокое здание на участке.  
 \*  
 \* @return Самое высокое здание или null, если список пуст.  
 \*/* public Land findTallestBuilding() {  
 if (lands.isEmpty()) {  
 return null;  
 }  
  
 Land tallestBuilding = lands.get(0);  
 for (Land land : lands) {  
 if (land.height > tallestBuilding.height) {  
 tallestBuilding = land;  
 }  
 }  
 return tallestBuilding;  
 }  
  
 */\*\*  
 \* Находит здания, выше 50 метров.  
 \*  
 \* @return Список зданий, выше 50 метров.  
 \*/* public List<Land> findBuildingOver50Meters() {  
 List<Land> buildingsOver50Meters = new ArrayList<>();  
 if (lands.isEmpty()) return buildingsOver50Meters;  
  
 for (Land land : lands) {  
 if (land.height > 50) {  
 buildingsOver50Meters.add(land);  
 }  
 }  
 return buildingsOver50Meters;  
 }  
  
 */\*\*  
 \* Упорядочивает участки земли по высоте.  
 \*/* public void sortByHeight() {  
 if (lands.isEmpty())  
 System.*out*.println("Список пуст.");  
 lands.sort(Comparator.*comparing*(land -> land.height));  
 }  
  
 */\*\*  
 \* Отображает информацию о всех участках земли.  
 \*/* public void displayAllInfo() {  
 if (lands.isEmpty())  
 System.*out*.println("Список пуст.");  
  
 for (Land land : lands) {  
 land.displayInfo();  
 System.*out*.println();  
 }  
 }  
  
 */\*\*  
 \* Находит участок земли по адресу.  
 \*  
 \* @param address Адрес для поиска.  
 \* @return Участок земли с указанным адресом или null, если не найден.  
 \*/* public Land findLandByAddress(String address) {  
 for (Land land : lands) {  
 if (land.address.equalsIgnoreCase(address)) {  
 return land;  
 }  
 }  
 return null;  
 }  
  
 */\*\*  
 \* Редактирует атрибуты участка земли по его адресу.  
 \*  
 \* @param address Адрес участка земли для редактирования.  
 \*/* public void editBuilding(String address) {  
 Land landToEdit = findLandByAddress(address);  
 if (landToEdit == null) {  
 System.out.println("Участка по такому адресу не найдено.");  
 return;  
 }  
  
 System.out.println("Выберите атрибут для редактирования:");  
 System.out.println("1. Адрес");  
 System.out.println("2. Площадь");  
 System.out.println("3. Расстояние до центра города");  
 System.out.println("4. Стоимость");  
 System.out.println("5. Высшая точка на участке");  
  
 switch (landToEdit.getClass().getSimpleName()) {  
 case "ResidentialLand":  
 System.out.println("6. Кол-во комнат");  
 System.out.println("7. Тип здания");  
 break;  
 case "NotResidentialArea":  
 System.out.println("6. Высота потолка");  
 System.out.println("7. Тип недвижимости ");  
 System.out.println("8. Владелец ");  
 break;  
 case "HighRiseBuilding":  
 System.out.println("6. Кол-во комнат");  
 System.out.println("7. Тип здания");  
 System.out.println("8. Кол-во этажей");  
 System.out.println("9. Жилая площадь");  
 break;  
 case "IndustrialFacilities":  
 System.out.println("6. Высота потолка");  
 System.out.println("7. Тип недвижимости ");  
 System.out.println("8. Владелец ");  
 System.out.println("9. Количество работников ");  
 System.out.println("10. Количество этажей ");  
 break;  
 }  
  
  
 String attributeChoice = Main.scanner.nextLine().trim();  
  
 switch (attributeChoice) {  
 case "1":  
 System.out.println("Введите новый адрес здания:");  
 String newAddress = Main.checkEmpty("Новый адресс здания не может быть пустым. Попробуйте снова.");  
 landToEdit.editFullInfo(newAddress, landToEdit.area, landToEdit.distanceToCity, landToEdit.price, landToEdit.height);  
 break;  
 case "2":  
 int newArea = Main.checkPos("Введите новую площадь участка:", "Площадь не может быть отрицательной. Попробуйте снова.");  
 landToEdit.editFullInfo(landToEdit.address, newArea, landToEdit.distanceToCity, landToEdit.price, landToEdit.height);  
 break;  
 case "3":  
 int newDist = Main.checkPos("Введите новое расстояние до центра города:", "Расстояние не может быть отрицательным. Попробуйте снова.");  
 landToEdit.editFullInfo(landToEdit.address, landToEdit.area, newDist, landToEdit.price, landToEdit.height);  
 break;  
 case "4":  
 int newPrice = Main.checkPos("Введите цену участка:", "Цена не может быть отрицательным. Попробуйте снова.");  
 landToEdit.editFullInfo(landToEdit.address, landToEdit.area, landToEdit.distanceToCity, newPrice, landToEdit.height);  
 break;  
 case "5":  
 int newHeight = Main.checkPos("Введите самую высокую точку участка:", "Высота не может быть отрицательной. Попробуйте снова.");  
 landToEdit.editFullInfo(landToEdit.address, landToEdit.area, landToEdit.distanceToCity, landToEdit.price, newHeight);  
 break;  
 case "6":  
 int newRooms;  
 int newHeightFlor;  
 switch (landToEdit.getClass().getSimpleName()) {  
 case "ResidentialLand":  
 newRooms = Main.checkPos("Введите новое кол-во комнат:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 ((ResidentialLand) landToEdit).numbersOfRooms=newRooms;  
 break;  
 case "NotResidentialArea":  
 newHeightFlor = Main.checkPos("Введите новую высоту потолков :", "Высота не может быть отрицательной. Попробуйте снова.");  
 ((NotResidentialArea) landToEdit).heightOfFlor=newHeightFlor;  
 break;  
 case "HighRiseBuilding":  
 newRooms = Main.checkPos("Введите новое кол-во комнат:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 ((HighRiseBuilding) landToEdit).numbersOfRooms=newRooms;  
 break;  
 case "IndustrialFacilities":  
 newHeightFlor = Main.checkPos("Введите новую высоту потолков :", "Высота не может быть отрицательной. Попробуйте снова.");  
 ((IndustrialFacilities) landToEdit).heightOfFlor=newHeightFlor;  
 break;  
 }  
 break;  
 case "7":  
 String newTypeHouse;  
 switch (landToEdit.getClass().getSimpleName()) {  
 case "ResidentialLand":  
 System.out.println("Введите новый тип здания:");  
 newTypeHouse = Main.checkEmpty("Новый тип здания не может быть пустым. Попробуйте снова.");  
 ((ResidentialLand) landToEdit).typeOfHouse=newTypeHouse;  
 break;  
 case "NotResidentialArea":  
 System.out.println("Введите новый тип здания:");  
 newTypeHouse = Main.checkEmpty("Новый тип здания не может быть пустым. Попробуйте снова.");  
 ((NotResidentialArea) landToEdit).propertyType=newTypeHouse;  
 break;  
 case "HighRiseBuilding":  
 System.out.println("Введите новый тип здания:");  
 newTypeHouse = Main.checkEmpty("Новый тип здания не может быть пустым. Попробуйте снова.");  
 ((HighRiseBuilding) landToEdit).typeOfHouse=newTypeHouse;  
 break;  
 case "IndustrialFacilities":  
 System.out.println("Введите новый тип здания:");  
 newTypeHouse = Main.checkEmpty("Новый тип здания не может быть пустым. Попробуйте снова.");  
 ((IndustrialFacilities) landToEdit).propertyType=newTypeHouse;  
 break;  
 }  
 break;  
 case "8":  
 String newStr;  
 switch (landToEdit.getClass().getSimpleName()) {  
 case "NotResidentialArea":  
 System.out.println("Введите новый владельца:");  
 newStr = Main.checkEmpty("Новый владелец не может быть пустым. Попробуйте снова.");  
 ((NotResidentialArea) landToEdit).owner=newStr;  
 break;  
 case "HighRiseBuilding":  
 int newFloor = Main.checkPos("Введите новое кол-во этажей:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 ((HighRiseBuilding) landToEdit).floor=newFloor;  
 break;  
 case "IndustrialFacilities":  
 System.out.println("Введите новый владельца:");  
 newStr = Main.checkEmpty("Новый владелец не может быть пустым. Попробуйте снова.");  
 ((IndustrialFacilities) landToEdit).owner=newStr;  
 break;  
 }  
 break;  
 case "9":  
 switch (landToEdit.getClass().getSimpleName()) {  
 case "HighRiseBuilding":  
 int newLiveSpace = Main.checkPos("Введите новую жилую площадь:", "Площадь не может быть отрицательной. Попробуйте снова.");  
 ((HighRiseBuilding) landToEdit).liveSpace=newLiveSpace;  
 break;  
 case "IndustrialFacilities":  
 int newEmploes = Main.checkPos("Введите новое кол-во работников:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 ((IndustrialFacilities) landToEdit).numberOfEmployees=newEmploes;  
 break;  
 }  
 break;  
 case "10":  
 int newFloors = Main.checkPos("Введите новое кол-во этажей:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 ((IndustrialFacilities) landToEdit).numberOfFloors=newFloors;  
 break;  
 default:  
 System.out.println("Неверный выбор атрибута. Пожалуйста, попробуйте снова.");  
 }  
  
  
 System.out.println("Информация об участке после редактирования:");  
 landToEdit.displayInfo();  
 }  
  
  
 */\*\*  
 \* Выводит меню действий для пользователя.  
 \*/* public void menu() {  
 System.out.println("\nВыберите действие:");  
 System.out.println("add - Добавить участок");  
 System.out.println("show - Показать все участки");  
 System.out.println("find - Найти участок по адресу");  
 System.out.println("edit - Редактировать данные участка");  
 System.out.println("tallest - Найти самое высокое здание на участке");  
 System.out.println("buildover50m - Показать здания выше 50м");  
 System.out.println("sort - Отсортировать здания по высоте");  
 System.out.println("exit - Выход");  
 }  
}

Класс Main

/\*\*  
 \*Задание: реализовать программу имеющую иерархию минимум из 3 классов  
 \* Тема: Недвижимость  
 \* Функции:  
 \* – определить самое высокое здание;  
 \* – определить здания с высотой более 50 м;  
 \* – упорядочить массив по возрастанию высоты;  
 \* – организовать поиск по названию здания, исправление одного из полей и вывод полной информации о здании после редактирования  
 \* @author Зеляпугин Сафронов 22ВП2  
 \* @version 1.4  
 \* @since 9.10.2023  
 \*/  
  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
 public static final Scanner scanner = new Scanner(System.in);  
  
 public static void main(String[] args) {  
 Consultant consultant=new Consultant();  
   
 while (true){  
 consultant.menu();  
 String choose = scanner.nextLine().trim().toLowerCase();  
 switch (choose){  
 case "add":  
 System.out.println("Выберете тип недвижимости:\n1. Жилая недвижимость\n2. Нежилая недвижимость\n3.Многоэтажное здание\n4.Промышленное предприятие");  
 String type = checkEmpty("Некорректно");  
  
 if(!(type.equals("1")||type.equals("2")||type.equals("3")||type.equals("4"))) {  
 System.out.println("Неверный выбор. Пожалуйста, попробуйте снова.");  
 break;  
 }  
  
 System.out.println("Адрес");  
 String adres = checkEmpty("Адрес не может быть пустым. Попробуйте снова.");  
 int area = checkPos("Введите площадь участка:", "Площадь не может быть отрицательной. Попробуйте снова.");  
 int distance = checkPos("Введите расстояние до центра города:", "Расстояние не может быть отрицательным. Попробуйте снова.");  
 int price = checkPos("Введите цену участка:", "Цена не может быть отрицательной. Попробуйте снова.");  
 int height = checkPos("Введите самую высокую точку участка:", "Высота не может быть отрицательной. Попробуйте снова.");  
  
 int rooms;  
 String typeHouse;  
 int heightFloor;  
 String owner;  
 switch (type){  
 case "1":  
 rooms = checkPos("Введите кол-во комнат:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 System.out.println("Введите тип здания:");  
 typeHouse = checkEmpty("Тип не может быть пустым. Попробуйте снова.");  
 consultant.addLand(new ResidentialLand(adres, area, distance, price, height, typeHouse, rooms));  
 System.out.println("Жилая недвижимость была успешно добавлена");  
 break;  
 case "2":  
 heightFloor = checkPos("Введите высоту потолка:", "Высота не может быть отрицательной. Попробуйте снова.");  
 System.out.println("Введите тип недвижимости:");  
 typeHouse = checkEmpty("Тип не может быть пустым. Попробуйте снова.");  
 System.out.println("Введите имя владельца:");  
 owner = checkEmpty("Имя не может быть пустым. Попробуйте снова.");  
 consultant.addLand(new NotResidentialArea(adres, area, distance, price, height, typeHouse, owner, heightFloor));  
 System.out.println("Нежилая территория была успешно добавлена");  
 break;  
 case "3":  
 rooms = checkPos("Введите кол-во комнат:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 System.out.println("Введите тип здания:");  
 typeHouse = checkEmpty("Тип не может быть пустым. Попробуйте снова.");  
 int floors = checkPos("Введите кол-во этажей:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 int liveSpace = checkPos("Введите жилую площадь:", "Площадь не может быть отрицательной. Попробуйте снова.");  
 consultant.addLand(new HighRiseBuilding(adres, area, distance, price, height, typeHouse, rooms, floors, liveSpace));  
 System.out.println("Высотное жилое здание было успешно добавлено");  
 break;  
 case "4":  
 heightFloor = checkPos("Введите высоту потолка:", "Высота не может быть отрицательной. Попробуйте снова.");  
 System.out.println("Введите тип недвижимости:");  
 typeHouse = checkEmpty("Тип не может быть пустым. Попробуйте снова.");  
 System.out.println("Введите имя владельца:");  
 owner = checkEmpty("Имя не может быть пустым. Попробуйте снова.");  
 System.out.println("Введите название предприятия:");  
 String name = checkEmpty("Название не может быть пустым. Попробуйте снова.");  
 int empoyes = checkPos("Введите кол-во рабочих:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 int floor = checkPos("Введите кол-во этажей:", "Кол-во не может быть отрицательной. Попробуйте снова.");  
 consultant.addLand(new IndustrialFacilities(adres, area, distance, price, height, typeHouse, owner, heightFloor, name, empoyes, floor));  
 System.out.println("Промышленное сооружение было успешно добавлено");  
 break;  
 default:  
 System.out.println("Неверный выбор. Пожалуйста, попробуйте снова.");  
 }  
 break;  
 case "show":  
 consultant.displayAllInfo();  
 break;  
 case "find":  
 System.out.println("Введите адрес участка для поиска:");  
 adres = checkEmpty("Адрес не может быть пустым. Попробуйте снова.");  
  
 Land found = consultant.findLandByAddress(adres);  
 if (found != null) {  
 found.displayInfo();  
 } else {  
 System.out.println("Участок по такому адресу не найден.");  
 }  
 break;  
 case "edit":  
 System.out.println("Введите адрес участка для редактирования:");  
 adres = checkEmpty("Адрес не может быть пустым. Попробуйте снова.");  
 consultant.editBuilding(adres);  
 break;  
 case "tallest":  
 Land tall = consultant.findTallestBuilding();  
 if (tall != null) {  
 System.out.println("Самое высокое здание: ");  
 tall.displayInfo();  
 } else {  
 System.out.println("Список пуст.");  
 }  
 break;  
 case "buildover50m":  
 List<Land> lands = consultant.findBuildingOver50Meters();  
 if (!lands.isEmpty()) {  
 System.out.println("Участки где высота больше 50м:");  
 for (Land land : lands) {  
 land.displayInfo();  
 }  
 } else {  
 System.out.println("Нет участков с высотой больше 50м");  
 }  
 break;  
 case "sort":  
 consultant.sortByHeight();  
 break;  
 case "exit":  
 // Выход  
 return;  
 default:  
 System.out.println("Неверная команда. Пожалуйста, попробуйте снова.");  
  
  
  
 }  
 }  
 }  
 /\*\*  
 \* Проверяет, что введенная строка не пуста.  
 \*  
 \* @param errorMessage Сообщение об ошибке, если строка пуста.  
 \* @return Введенная непустая строка.  
 \*/  
 public static String checkEmpty(String errorMessage) {  
 String input = "";  
 while (input.isEmpty()) {  
  
 input = scanner.nextLine().trim();  
 if (input.isEmpty()) {  
 System.out.println(errorMessage);  
 }  
 }  
 return input;  
 }  
 /\*\*  
 \* Проверяет, что введенное целое число неотрицательно.  
 \*  
 \* @param promptMessage Сообщение для ввода числа.  
 \* @param errorMessage Сообщение об ошибке, если введено отрицательное число.  
 \* @return Неотрицательное целое число.  
 \*/  
 public static int checkPos(String promptMessage, String errorMessage) {  
 int value = -1;  
 while (value < 0) {  
 System.out.println(promptMessage);  
 try {  
 value = Integer.parseInt(scanner.nextLine());  
 if (value < 0) {  
 System.out.println(errorMessage);  
 }else  
 break;  
 }  
 catch (NumberFormatException e) {System.out.println("Пожалуйста, введите корректное число.");}  
 }  
 return value;  
 }  
}

**Пояснительный текст к программе**

Основные переменные, классы, их методы и атрибуты данной программы можно увидеть на диаграмме классов на рисунке 1. Описание методов классов и полей можно увидеть в файлах с кодом.

Диаграмма классов предоставлена на рисунке 1.

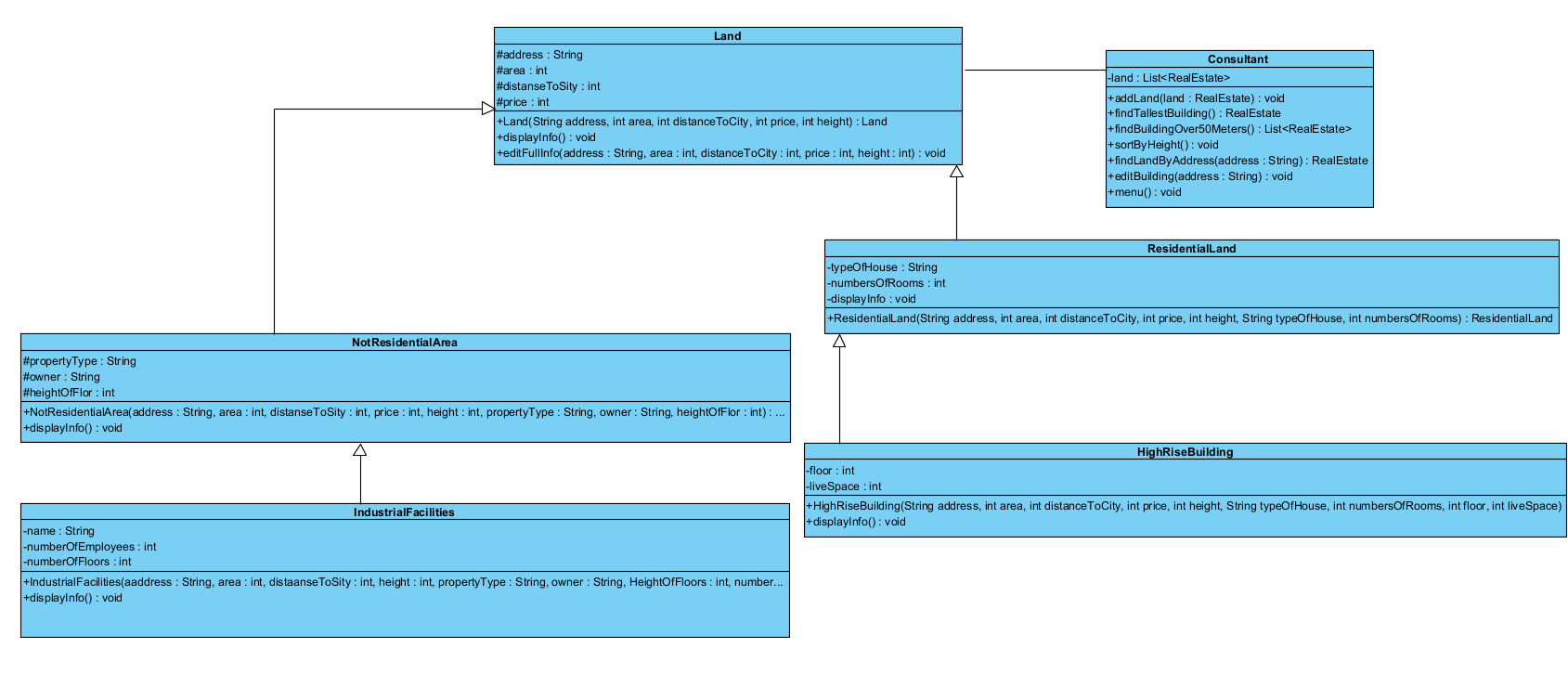


Рисунок 1 ­– Диаграмма классов

**Результаты работы программы**

Результаты работы программы показаны на рисунках 2 - 4.

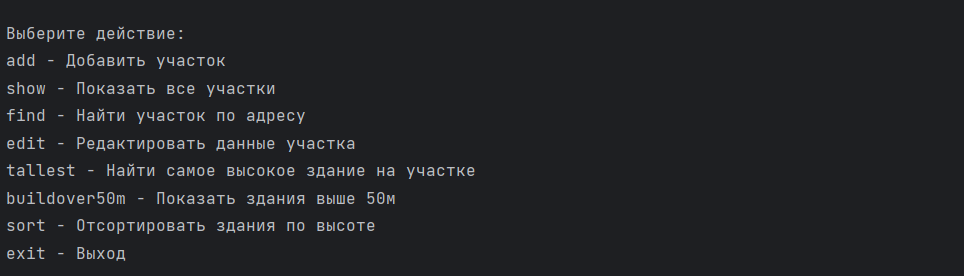


Рисунок 2 – меню для выбора действия

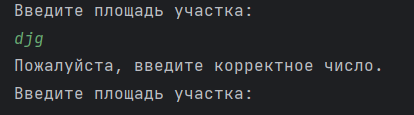


Рисунок 3 – некорректно введенная площадь

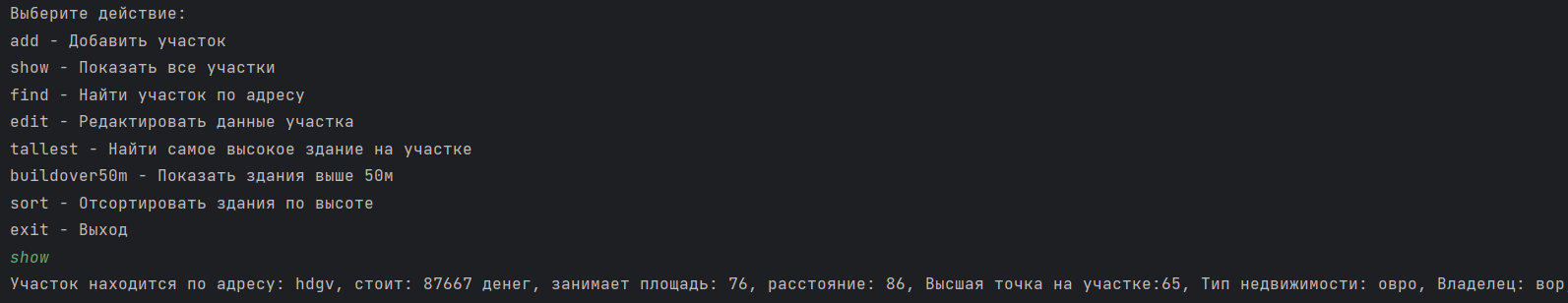


Рисунок 4 – вывод информации о добавленном здании

**Выводы**

В результате работы была разработана иерархия классов на языке Java, представляющая предметную область «Недвижимость». В соответствии с

условиями задания, иерархия включает в себя не менее трёх классов, каждый из которых обладает минимум 5 полями и 3 методами.

Был учтен функционал, позволяющий:

* определить самое высокое здание;
* определить здания с высотой более 50 м;
* упорядочить массив по возрастанию высоты;
* организовать поиск по названию здания, исправление одного из полей и вывод полной информации о здании после редактирования.

Таким образом, решение задачи позволило глубже изучить и закрепить принципы объектно-ориентированного программирования, а также успешно применить их на практике в рамках конкретной предметной области.