МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

з дисципліни « Об'єктно-орієнтоване програмування »

*назва дисципліни*

на тему: «Будівлі »

Виконав: студент 1 курсу групи № 621п

напряму підготовки(спеціальності)

6.121 інженерія програмного забезпечення

(шифр і назва ОП)

Вітковська О.С.

(прізвище й ініціали студента)

Керівник: асистент каф. 603

Дем’яненко Владислав Анатолійович

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала:

Кількість балів:

Члени комісії:

(підпис) (прізвище й ініціали)

(підпис) (прізвище й ініціали)

(підпис) (прізвище й ініціали)

Харків – 2022

ЗМІСТ

[Вступ 3](#_Toc102905290)

[Мета роботи 3](#_Toc102905291)

[Загальна постановка задачі 3](#_Toc102905292)

[1 Детальна постановка задачі 5](#_Toc102905293)

[2 Теоретичний вступ 6](#_Toc102905294)

[3 Архітектура проекту 7](#_Toc102905295)

[4 Опис класів будівель і їх списку 8](#_Toc102905296)

[4.1 Клас Будівля – Building 8](#_Toc102905297)

[4.2 Клас Будівля – Office 14](#_Toc102905298)

[4.3 Клас Склад– Warehouse 18](#_Toc102905299)

[4.4 Житовий будинок – Residential 22](#_Toc102905300)

[5 Опис головної програми 26](#_Toc102905301)

[Висновки 27](#_Toc102905302)

[Додаток А Лістинги програми 28](#_Toc102905303)

[Додаток Б. Результат роботи програми 37](#_Toc102905304)

# Вступ

## **Мета роботи**

1. Вивчити за літературними/Інтернет джерелами предметну область, уточнити з викладачем постановку задачі та побудувати її інфологічну модель (математичну, логічну, структурну, Use Case тощо).

2. Реалізувати у курсовому проекті основні засади ОВП, згідно

вибраній мові програмування (C++, C#, Java):

• інкапсуляція та методи доступу (закриті, відкриті, захищені);

• статичний чи динамічний поліморфізм;

• просте (або множинне) успадкування;

• прості, абстрактні та поліморфні класи

3. Використовувати текстові або двійкові файли користувача для запису та читання об'єктів

4. Зробити log-файл (повний протокол) роботи з програмою (для команди)

## **Загальна постановка задачі**

1. Перша частина завдання. Згідно зі своїм варіантом створити абстрактний клас з полями та віртуальними методами. На його основі реалізувати не менше 2-х похідних класів із поліморфними методами. Побудувати попередню UML діаграму. Зробити тестовий приклад та фрагменти пояснювальної записки.

2. Друга частина завдання. Створити клас, що містить масив/список/параметризовану колекцію об'єктів цих класів у динамічної пам'яті. Колекція об'єктів та їх поля мають бути отримані з використанням датчика псевдовипадкових чисел або за допомогою зрозумілого ручного коректного введення з клавіатури (з урахуванням значень полів, прийнятих у предметної галузі). Зробити тестовий приклад та фрагменти пояснювальної записки.

3. Третя частина завдання. Написати демонстраційну програму з використанням динамічного поліморфізму, в якій будуть використовуватися всі методи класів, для чого в режимі діалогу має бути організований доступ до будь-якого елементу колекції з обчисленнями, згідно з варіантом. Побудувати уточнену UML діаграму.

4. Передбачити варіант Debug для візуалізації виклику конструкторів та деструкторів, а також проміжних обчислень (за погодженням з викладачем).

5. Вихідні дані вводяться або вручну з перевіркою коректності введення, або генеруються з використанням датчика псевдовипадкових чисел із записом в файл користувача (за запитом в режимі діалогу). Дані повинні відповідати постановці задачі та мати відповідний тип (Арифметичний, символьний, рядковий).

6. Вихідні дані (поля класів) мають бути видані на дисплей (перегляд повний або частковий) або відредаговані (змінені, додані, видалені) запит у режимі діалогу.

7. Обчислення мають бути коректними, для чого мають бути використані методи опрацювання виняткових ситуацій.

8. Вихідна інформація видається на дисплей або записується в користувача файл з ехо-супроводом на дисплей (на запит у режимі діалогу).

# 1 Детальна постановка задачі

Створити абстрактний клас Building (будівля) із полями, що задають кількість кімнат, їх розміри, кількість і дані про людей, що там проживають/працюють, а також віртуальними методами виведення цих даних на екран. На його основі реалізувати класи Residential (житлова будівля), Office (офісне) та Warehouse (склад) з можливістю обчислення загальної площі будівлі, площі та периметра заданих кімнат, а також кількості та ПІБ проживаючих/працюючих там людей.

Створити клас District (група), що містить масив/список/параметризовану колекцію об'єктів цих класів у динамічній пам'яті. Передбачити можливість виведення всіх об'єктів списку та вибірки за конкретним об'єктом. Написати демонстраційну програму, в якій використовуватимуться всі методи класів.

Особливості:

* Програма реалізована мовою Java з використанням інтегрованої

середовища розробки IntelliJ IDEA 2021.

* Вихідні дані генеруються з використанням датчика псевдовипадкових

чисел.

- У програмі задіяно механізм умовної компіляції (Debug/Release)

# 2 Теоретичний вступ

2.1. Опис своєї предметної галузі (математичні формули, геометричні малюнки, посади в установі, співвідношення площ будівель, основні характеристики різних транспортних засобів і т.п.).

2.2 Абстрактний клас, його призначення та основні характеристики.

2.3 Поліморфний клас, його призначення та основні характеристики.

2.4 Покажчики на базовий клас. Їхні властивості.

2.5 Колекція об'єктів та їх обробка у динамічній пам'яті.

2.6. Файлове введення-виведення.

2.7. Текстові та двійкові файли (або інші файли, відповідно до мови реалізації курсового проекту).

# 3 Архітектура проекту

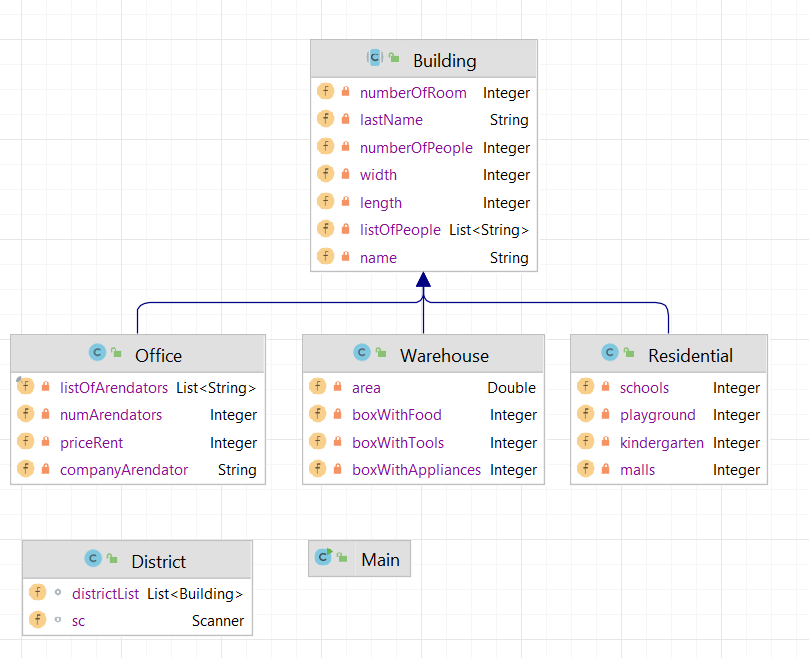


Рисунок 3.1 – Діаграма класів для представлення будівель та їх списку

# 4 Опис класів будівель і їх списку

## 4.1 Клас Будівля – Building

Абстрактний клас з полями, в яких вказується кількість кімнат, їх розміри, кількість і дані про людей, що живуть / там працюють, а також віртуальні методи відображення цих даних на екрані.

public abstract class Building {  
  
 private Integer numberOfRoom;  
 private Integer length;  
 private Integer width;  
  
 private List<String> listOfPeople = new ArrayList<>();  
 private String name;  
 private String lastName;  
 private Integer numberOfPeople;  
  
 public Integer getNumberOfRoom() {}  
  
 public void setNumberOfRoom(Integer numberOfRoom) {}  
  
 public Integer getLength() {}  
  
 public void setLength(Integer length) {}  
  
 public Integer getWidth() {}  
  
 public void setWidth(Integer width) {}  
  
 public String getName() {}  
  
 public void setName(String name) {}  
  
 public String getLastName() {}  
  
 public void setLastName(String lastName) {}  
  
 public Integer getNumberOfPeople() {}  
  
 public void setNumberOfPeople(Integer numberOfPeople) {}

public List<String> getListOfPeople() {}

@Override  
 public String toString() {}  
  
  
 public Integer areaOfRoom() {}  
  
 public Integer perimOfRoom() {}  
  
 public Integer areaOfBuilding() {}

4.1.1 **Поля класу**

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | **Тип** | **Рівень** **доступу** | **Опис** |
| numberOfRoom | Integer | private | Кількість кімнат |
| length | Integer | private | Довжина кімнати |
| width | Integer | private | Висота кімнати |
| listOfPeople | List<String> | private | Список мешканців |
| name | String | private | Ім’я мешканця |
| lastName | String | private | Призвіще мешканця |
| numberOfPeople | Integer | private | Кількість мешканців |

4.1.2 Методи класу

4.1.2.1 public Integer getNumberOfRoom()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

numberOfRoom. Кількість кімнат.

*Примітки*

Нема.

4.1.2.2 public void setNumberOfRoom(Integer numberOfRoom)

*Рівень доступу*

public.

*Параметри*

Integer numberOfRoom. Кількість кімнат.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.3 public Integer getLength()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

length. Довжина кімнати

*Примітки*

Нема.

4.1.2.4 public void setLength (Integer length)

*Рівень доступу*

public.

*Параметри*

Integer length. Довжина кімнати.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.5 public Integer getWidth ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

width. Ширина кімнати.

*Примітки*

Нема.

4.1.2.6 public void setWidth (Integer width)

*Рівень доступу*

public.

*Параметри*

Integer width. Ширина кімнати.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.7 public String getName ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

name. Ім’я мешканця.

*Примітки*

Нема.

4.1.2.8 public void setName (String name)

*Рівень доступу*

public.

*Параметри*

String name. Ім’я мешканця.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.9 public String getLastName ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

lastName. Призвіще мешканця.

*Примітки*

Нема.

4.1.2.10 public void setLastName (String lastName)

*Рівень доступу*

public.

*Параметри*

String lastName. Призвіще мешканця.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.11 public Integer getNumberOfPeople()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

numberOfPeople. Кількість мешканців.

*Примітки*

Нема.

4.1.2.12 public void setNumberOfPeople(Integer numberOfPeople)

*Рівень доступу*

public.

*Параметри*

Integer numberOfPeople. Кількість мешканців.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.1.2.13 public List<String> getListOfPeople()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

listOfPeople. Кількість мешканців.

*Примітки*

Нема.

4.1.2.14 public String toString()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Рядок з інформацією про поля класу.

*Примітки*

Нема.

4.1.2.15 public Integer areaOfRoom()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Площа кімнати.

*Примітки*

Нема.

4.1.2.16 public Integer perimOfRoom()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Периметр кімнати.

*Примітки*

Нема.

4.1.2.17 public Integer areaOfBuilding()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Площа будівлі.

*Примітки*

Нема.

4.1.2.18 public void addPeople()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

## 4.2 Клас Будівля – Office

Клас з можливістю розрахувати загальну площу будівлі, площу і периметр зазначених приміщень, а також кількість і ПІБ людей, що проживають / працюють там.

public class Office extends Building {  
 private String companyArendator;  
 private Integer numArendators;  
 private Integer priceRent;  
  
 private final List<String> listOfArendators = new ArrayList<>();  
  
 public Integer costOfRent() {}  
  
 public Integer getPriceRent() {}  
  
 public void setPriceRent(Integer priceRent) {}  
  
 public String getCompanyArendator() {}  
  
 public void setCompanyArendator(String companyArendator) {}  
  
 public List<String> getListOfArendators() {}  
  
 public Integer getNumArendators() {}  
  
 public void setNumArendators(Integer numArendators) {}  
  
 @Override  
 public String toString() {}

4.2.1 **Поля класу**

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | **Тип** | **Рівень** **доступу** | **Опис** |
| numArendators | Integer | private | Кількість арендаторів |
| priceRent | Integer | private | Ціна ренти |
| companyArendator | String | private | Назва компанії |
| listOfArendators | List<String> | private | Список арендаторів |

4.2.2 Методи класу

4.2.2.1 public Integer getPriceRent()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

priceRent. Ціна ренти.

*Примітки*

Нема.

4.2.2.2 public void setPriceRent(Integer priceRent)

*Рівень доступу*

public.

*Параметри*

Integer priceRent. Ціна ренти.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.2.2.3 public Integer getNumOfArendators()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

priceRent. Кількість арендаторів.

*Примітки*

Нема.

4.2.2.4 public void setNumOfArendators(Integer numOfArendators)

*Рівень доступу*

public.

*Параметри*

Integer priceRent. Кількість арендаторів.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.2.2.5 public String getCompanyArendator ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

companyArendator. Назва компанії.

*Примітки*

Нема.

4.2.2.6 public void setCompanyArendator (String companyArendator)

*Рівень доступу*

public.

*Параметри*

String companyArendator. Назва компанії.

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.2.2.7 public List<String> getListOfArendators()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

listOfArendators. Список арендаторів.

*Примітки*

Нема.

4.2.2.8 public String toString()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Рядок з інформацією про поля класу.

*Примітки*

Нема.

4.2.2.9 public Integer getCostRent()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Ціна ренти.

*Примітки*

Нема.

## 4.3 Клас Склад– Warehouse

Клас з можливістю розрахувати загальну площу складу, кількість боксів з іжею, інструментами та приладами.

public class Warehouse extends Building {  
 private Integer boxWithFood;  
 private Integer boxWithTools;  
 private Integer boxWithAppliances;  
 private Double area;  
  
 public boolean hasFreePlace() {}  
  
 public Double getArea() {}  
  
 public void setArea(Double area) {}  
  
 public Integer getBoxWithAppliances() {}  
  
 public void setBoxWithAppliances(Integer boxWithAppliances) {}  
  
 public Integer getBoxWithFood() {}  
  
 public void setBoxWithFood(Integer boxWithFood) {}  
  
 public Integer getBoxWithTools() {}  
  
 public void setBoxWithTools(Integer boxWithTools) {}  
  
 @Override  
 public String toString() {}  
}

4.3.1 **Поля класу**

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | **Тип** | **Рівень** **доступу** | **Опис** |
| boxWithFood | Integer | private | Кількість боксів з іжею |
| boxWithTools | Integer | private | Кількість боксів з інструментами |
| boxWithAppliances | Integer | private | Кількість боксів з приладами |
| area | Double | private | Площа складу |

4.3.2 Методи класу

4.3.2.1 public boolean hasFreePlace()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Boolean FreePlace Є чи немає вільного місця на складі

*Примітки*

Нема.

4.3.2.2 public Double getArea()

*Рівень доступу*

public.

*Параметри*

Нема

*Значення, що повертається*

Double Area. Площа складу

*Примітки*

Нема.

4.3.2.3 public void setArea(Double area)

*Рівень доступу*

public.

*Параметри*

Double area *Нова площа складу*

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.3.2.4 public Integer getBoxWithAppliances()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

BoxWithAppliances Кількість боксів з приладами

*Примітки*

Нема.

4.3.2.5 public void setBoxWithAppliances(Integer boxWithAppliances)

*Рівень доступу*

public.

*Параметри*

Integer boxWithAppliances Нова кількість боксів з прилдами

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.3.2.6 public Integer getBoxWithFood()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

BoxWithFood Кількість боксів з іжею

*Примітки*

Нема.

4.3.2.7 public void setBoxWithFood (Integer boxWithFood)

*Рівень доступу*

public.

*Параметри*

Integer boxWithFood Нова кількість боксів з іжею

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.3.2.8 public Integer getBoxWithTools()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

BoxWithhTools Кількість боксів з інструментами

*Примітки*

Нема.

4.3.2.7 public void setBoxWithhTools (Integer boxWithhTools)

*Рівень доступу*

public.

*Параметри*

Integer boxWithhTools Нова кількість боксів з інструментами

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.2.2.10 public String toString()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Рядок з інформацією про поля класу.

*Примітки*

Нема.

## 4.4 Житовий будинок – Residential

Клас з можливістю розрахувати загальну площу складу, кількість боксів з іжею, інструментами та приладами.

public class Residential extends Building {  
 private Integer playground;  
 private Integer malls;  
 private Integer schools;  
 private Integer kindergarden;  
  
 public Integer getPlayground() {}  
  
 public void setPlayground(Integer playground) {}  
  
 public Integer getMalls() {}  
  
 public void setMalls(Integer malls) {}  
  
 public Integer getSchools() {}  
  
 public void setSchools(Integer schools) {}  
  
 public Integer getKindergarten() {}  
  
 public void setKindergarten(Integer kindergarten) {}  
  
 public Integer costOfRent() {}  
  
 @Override  
 public String toString() {}  
}

4.4.1 Поля класу

|  |  |  |  |
| --- | --- | --- | --- |
| **Поля** | **Тип** | **Рівень** **доступу** | **Опис** |
| playground | Integer | private | Кількість майданчиків |
| malls | Integer | private | Кількість магазинів |
| schools | Integer | private | Кількість шкіл |
| kindergarten | Double | private | Кількість дитсадків |

4.4.2 Методи класу

4.4.2.1 public Double getPlayground ()

*Рівень доступу*

public.

*Параметри*

Нема

*Значення, що повертається*

Playground Кількість майданчиків

*Примітки*

Нема.

4.4.2.2 public void set Playground (Integer playground)

*Рівень доступу*

public.

*Параметри*

Integer playground Нова кількість майданчиків

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.4.2.3 public Integer getMalls ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Malls Кількість магазинів

*Примітки*

Нема.

4.4.2.4 public void setMalls (Integer malls)

*Рівень доступу*

public.

*Параметри*

Integer Malls Нова кількість магазинів

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.4.2.5 public Integer getSchools ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Schools Кількість шкіл

*Примітки*

Нема.

4.4.2.6 public void setSchools (Integer Schools)

*Рівень доступу*

public.

*Параметри*

Integer Schools Нова кількість шкіл

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.4.2.7 public Integer getBoxWithKindergarten ()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Kindergarten Кількість дитсадків

*Примітки*

Нема.

4.3.2.7 public void Kindergarten (Integer Kindergarten)

*Рівень доступу*

public.

*Параметри*

Integer Kindergarten Нова кількість дитсадків

*Значення, що повертається*

Нема.

*Примітки*

Нема.

4.2.2.10 public String toString()

*Рівень доступу*

public.

*Параметри*

Нема.

*Значення, що повертається*

Рядок з інформацією про поля класу.

*Примітки*

Нема.

# 5 Опис головної програми

**Class District**

5.1 Створення колекції будівель

public void create()

Функція реалізує створення колекції будівель

public void add(Building building)

Функція реалізує додавання будівлі в колекцію

5.2 Виведення інформації о будівлях в колекції

public List<Buildings> show()

Функція реалізує виведення інформації о колекції

public List<Buildings> getOffice()

Функція реалізує виведення інформації о Office

public List<Buildings> getResidential()

Функція реалізує виведення інформації о Residential

public List<Buildings> getWarehouse()

Функція реалізує виведення інформації о Warehouse

5.3 Видалення колекції будівель

public void remove(Building building)

Функція реалізує видалення будівлі з коллекції

# Висновки

# Додаток А Лістинги програми

***Файл Building.java***

public abstract class Building {  
  
 private Integer numberOfRoom;  
 private Integer length;  
 private Integer width;  
  
 private List<String> listOfPeople = new ArrayList<>();  
 private String name;  
 private String lastName;  
 private Integer numberOfPeople;  
  
 public Integer getNumberOfRoom() {  
 return numberOfRoom;  
 }  
  
 public void setNumberOfRoom(Integer numberOfRoom) {  
 this.numberOfRoom = numberOfRoom;  
 }  
  
 public Integer getLength() {  
 return length;  
 }  
  
 public void setLength(Integer length) {  
 this.length = length;  
 }  
  
 public Integer getWidth() {  
 return width;  
 }  
  
 public void setWidth(Integer width) {  
 this.width = width;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public Integer getNumberOfPeople() {  
 return numberOfPeople;  
 }  
  
 public void setNumberOfPeople(Integer numberOfPeople) {  
 this.numberOfPeople = numberOfPeople;  
 }  
  
 public List<String> getListOfPeople() {  
 return listOfPeople;  
 }  
 @Override  
 public String toString() {  
 return "Building{" +  
 ", numberOfRoom=" + numberOfRoom +  
 ", length=" + length +  
 ", width=" + width +  
 ", name='" + name + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", numberOfPeople=" + numberOfPeople +  
 '}';  
 }  
  
  
 public Integer areaOfRoom() {  
 Integer area = length \* width;  
 return area;  
 }  
  
 public Integer perimOfRoom() {  
 Integer perim = length + length + width + width;  
 return perim;  
 }  
  
 public Integer areaOfBuilding() {  
 Integer area = numberOfRoom \* areaOfRoom();  
 return area;  
 }  
  
 public void addPeople(String name, String lastName) {  
 listOfPeople.add(name + " " + lastName);  
 }  
}

***Файл District.java***

public class District {  
  
 List<Building> districtList = new ArrayList<>();  
 Scanner sc=new Scanner(System.*in*);  
 public void create(){  
  
 int i = 3;  
 int j=2;  
 while (i != 0) {  
 System.*out*.println();  
 System.*out*.println("Please choose some building to add");  
 System.*out*.println("If you want to add Office, enter 1");  
 System.*out*.println("If you want to add Residential, enter 2");  
 System.*out*.println("If you want to add Warehouse, enter 3");  
 System.*out*.println("If don't want to add, enter 0");  
 System.*out*.println();  
 i = sc.nextInt();  
 switch (i) {  
 case 1 -> {  
  
 Office office =new Office();  
 System.*out*.println("If you want to fill fields in random way, enter 1");  
 System.*out*.println("If you want to fill fields from file, enter 2");  
 j=sc.nextInt();  
 if(j==1) {  
 randomOffice(office);  
 }else if(j==2){  
 fileOffice(office);  
 }  
 System.*out*.println("Office was added");  
 }  
 case 2 -> {  
  
 Residential residential = new Residential();  
 System.*out*.println("If you want to fill fields in random way, enter 1");  
 System.*out*.println("If you want to fill fields from file, enter 2");  
 j=sc.nextInt();  
 if(j==1) {  
 randomResidential(residential);  
 }else if(j==2) {  
 fileResidential(residential);  
  
 }  
 System.*out*.println("Residential was added");  
 }  
 case 3 -> {  
  
 Warehouse warehouse = new Warehouse();  
 System.*out*.println("If you want to fill fields in random way, enter 1 ");  
 System.*out*.println("If you want to fill fields from file, enter 2 ");  
 j=sc.nextInt();  
 if(j==1) {  
 randomWarehouse(warehouse);  
 }else if(j==2) {  
 fileWarehouse(warehouse);  
 }  
 System.*out*.println("Warehouse was added");  
 }  
 default -> System.*out*.println();  
 }  
 }  
 }  
 public void delete(){  
 districtList.clear();  
 }  
 public void add(Building building){  
 districtList.add(building);  
 }  
  
 public void remove(Building building){  
 System.*out*.println();  
 System.*out*.println("Please, choose name of building to delete");  
  
 districtList.remove(building);  
 }  
 public List<Building> getOffice(){  
 return districtList.stream().filter(p -> p.getClass().equals(Office.class)).collect(Collectors.*toList*());  
 }  
 public List<Building> getResidential(){  
 return districtList.stream().filter(p -> p.getClass().equals(Residential.class)).collect(Collectors.*toList*());  
 }  
 public List<Building> getWarehouse(){  
 return districtList.stream().filter(p -> p.getClass().equals(Warehouse.class)).collect(Collectors.*toList*());  
 }  
 public List<Building> show(){  
 return districtList;  
 }  
  
 public void randomOffice(Office office){  
 String [] arrendators={"Sony Pictures","Apple","NIX","Epam","A-Level","Samsung","Asus","HP","Space-X","LG"};  
 Random random = new Random();  
 int n = random.nextInt(0,arrendators.length);  
 office.setCompanyArendator(arrendators[n]);  
 n=random.nextInt(1,10);  
 office.setNumArendators(n);  
 n=random.nextInt(300,700);  
 office.setPriceRent(n);  
 n=random.nextInt(200,900);  
 office.setNumberOfRoom(n);  
 n=random.nextInt(40,200);  
 office.setLength(n);  
 n=random.nextInt(40,200);  
 office.setWidth(n);  
 add(office);  
  
 }  
  
 public void randomResidential(Residential residential){  
 Random random = new Random();  
 int n = random.nextInt(200,900);  
 residential.setNumberOfRoom(n);  
 n = random.nextInt(2,13);  
 residential.setKindergarten(n);  
 n = random.nextInt(1,6);  
 residential.setMalls(n);  
 n = random.nextInt(2,9);  
 residential.setPlayground(n);  
 n = random.nextInt(1,3);  
 residential.setSchools(n);  
 n=random.nextInt(40,200);  
 residential.setLength(n);  
 n=random.nextInt(40,200);  
 residential.setWidth(n);  
 add(residential);  
 }  
  
 public void randomWarehouse(Warehouse warehouse){  
 Random random = new Random();  
 int n = random.nextInt(200,900);  
 warehouse.setNumberOfRoom(n);  
 n = random.nextInt(2,11);  
 warehouse.setBoxWithAppliances(n);  
 n = random.nextInt(100,300);  
 warehouse.setArea((double)n);  
 n = random.nextInt(1,7);  
 warehouse.setBoxWithFood(n);  
 n = random.nextInt(1,10);  
 warehouse.setBoxWithTools(n);  
 n=random.nextInt(40,200);  
 warehouse.setLength(n);  
 n=random.nextInt(40,200);  
 warehouse.setWidth(n);  
 add(warehouse);  
 }  
  
 public void fileOffice(Office office){  
 String [] arrendators={"Sony Pictures","Apple","NIX","Epam","A-Level","Samsung","Asus","HP","Space-X","LG"};  
 try (FileReader fileReader = new FileReader("test\_project/src/main/java/ua/com/khai/office.txt")){  
 BufferedReader bufferedReader = new BufferedReader(fileReader);  
 String c;  
 List<Integer> listNum=new ArrayList<Integer>();  
 while((c = bufferedReader.readLine()) != null){  
 listNum.add(Integer.*valueOf*(c));  
 }  
 office.setCompanyArendator(arrendators[listNum.get(0)]);  
 office.setNumArendators(listNum.get(1));  
 office.setPriceRent(listNum.get(2));  
 office.setNumberOfRoom(listNum.get(3));  
 office.setLength(listNum.get(4));  
 office.setWidth(listNum.get(5));  
 add(office);  
 } catch (FileNotFoundException e) {  
 System.*out*.println(e.getMessage());  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public void fileResidential(Residential residential){  
 try (FileReader fileReader = new FileReader("test\_project/src/main/java/ua/com/khai/residential.txt")){  
 BufferedReader bufferedReader = new BufferedReader(fileReader);  
 String c;  
 List<Integer> listNum=new ArrayList<Integer>();  
 while((c = bufferedReader.readLine()) != null){  
 listNum.add(Integer.*valueOf*(c));  
 }  
 residential.setNumberOfRoom(listNum.get(0));  
 residential.setKindergarten(listNum.get(1));  
 residential.setMalls(listNum.get(2));  
 residential.setPlayground(listNum.get(3));  
 residential.setSchools(listNum.get(4));  
 residential.setLength(listNum.get(5));  
 residential.setWidth(listNum.get(6));  
 add(residential);  
 } catch (FileNotFoundException e) {  
 System.*out*.println(e.getMessage());  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
  
 public void fileWarehouse(Warehouse warehouse){  
 try (FileReader fileReader = new FileReader("test\_project/src/main/java/ua/com/khai/warehouse.txt")){  
 BufferedReader bufferedReader = new BufferedReader(fileReader);  
 String c;  
 List<Integer> listNum=new ArrayList<Integer>();  
 while((c = bufferedReader.readLine()) != null){  
 listNum.add(Integer.*valueOf*(c));  
 }  
 warehouse.setNumberOfRoom(listNum.get(0));  
 warehouse.setBoxWithAppliances(listNum.get(1));  
 warehouse.setArea((double)listNum.get(2));  
 warehouse.setBoxWithFood(listNum.get(3));  
 warehouse.setBoxWithTools(listNum.get(4));  
 warehouse.setLength(listNum.get(5));  
 warehouse.setWidth(listNum.get(6));  
 add(warehouse);  
  
  
 } catch (FileNotFoundException e) {  
 System.*out*.println(e.getMessage());  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }

***Файл Main.java***

public class Main {  
 public static void main(String[] args) {  
 District district = new District();  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("Hello, user!!");  
 while (true) {  
 System.*out*.println("If you want to create district, enter 1");  
 System.*out*.println("If you want to delete district, enter 2");  
 System.*out*.println("If you want to show district, enter 3");  
 System.*out*.println("If you want to show Office buildings in district, enter 4");  
 System.*out*.println("If you want to show Residential buildings in district, enter 5");  
 System.*out*.println("If you want to show Warehouse buildings in district, enter 6");  
 System.*out*.println("If you want to exit, enter 0");  
 int i = sc.nextInt();  
 switch (i) {  
 case 1 -> district.create();  
 case 2 -> district.delete();  
 case 3 -> System.*out*.println(district.show());  
 case 4 -> System.*out*.println(district.getOffice());  
 case 5 -> System.*out*.println(district.getResidential());  
 case 6 -> System.*out*.println(district.getWarehouse());  
 case 0 -> System.*exit*(0);  
 }  
 }  
 }  
}

***Файл Office.java***

public class Office extends Building {  
 private String companyArendator;  
 private Integer numArendators = 0;  
 private Integer priceRent = 0;  
  
 private final List<String> listOfArendators = new ArrayList<>();  
  
 public Integer getPriceRent() {  
 return priceRent;  
 }  
  
 public void setPriceRent(Integer priceRent) {  
 this.priceRent = priceRent;  
 }  
  
 public String getCompanyArendator() {  
 return companyArendator;  
 }  
  
 public void setCompanyArendator(String companyArendator) {  
 this.companyArendator = companyArendator;  
 listOfArendators.add(companyArendator);  
 numArendators++;  
 }  
  
 public List<String> getListOfArendators() {  
 return listOfArendators;  
 }  
  
 public Integer getNumArendators() {  
 return numArendators;  
 }  
  
 public void setNumArendators(Integer numArendators) {  
 this.numArendators = numArendators;  
 }  
  
  
 public Integer costOfRent() {  
 return getPriceRent() / numArendators;  
 }  
  
 @Override  
 public String toString() {  
 return "Office{" +  
 "companyArendator='" + companyArendator + '\'' +  
 ", numArendators=" + numArendators +  
 ", costOfRent=" + costOfRent() +  
 ", listOfArendators=" + listOfArendators +  
 '}';  
 }  
}

***Файл Residential.java***

public class Residential extends Building {  
 private Integer playground;  
 private Integer malls;  
 private Integer schools;  
 private Integer kindergarten;  
  
 public Integer getPlayground() {  
 return playground;  
 }  
  
 public void setPlayground(Integer playground) {  
 this.playground = playground;  
 }  
  
 public Integer getMalls() {  
 return malls;  
 }  
  
 public void setMalls(Integer malls) {  
 this.malls = malls;  
 }  
  
 public Integer getSchools() {  
 return schools;  
 }  
  
 public void setSchools(Integer schools) {  
 this.schools = schools;  
 }  
  
 public Integer getKindergarten() {  
 return kindergarten;  
 }  
  
 public void setKindergarten(Integer kindergarten) {  
 this.kindergarten = kindergarten;  
 }  
  
 public Integer costOfRent() {  
 return areaOfRoom() \* 10 + playground \* 3 + kindergarten \* 4 + schools \* 5 + malls \* 2;  
 }  
  
 @Override  
 public String toString() {  
 return super.toString() + "Residential{" +  
 "playground=" + playground +  
 ", malls=" + malls +  
 ", schools=" + schools +  
 ", kindergarten=" + kindergarten +  
 '}';  
 }  
}

***Файл Warehouse.java***

public class Warehouse extends Building {  
 private Integer boxWithFood;  
 private Integer boxWithTools;  
 private Integer boxWithAppliances;  
 private Double area;  
  
 public boolean hasFreePlace() {  
 Double areaBoxWithFood = 21.8;  
 Double areaBoxWithTools = 72.3;  
 Double areaBoxWithAppliances = 56.5;  
 return (getArea() - (areaBoxWithAppliances + areaBoxWithTools + areaBoxWithFood)) > 0;  
 }  
  
 public Double getArea() {  
 return area;  
 }  
  
 public void setArea(Double area) {  
 this.area = area;  
 }  
  
 public boolean isEmpty() {  
 return boxWithFood == 0 && boxWithAppliances == 0 && boxWithTools == 0;  
 }  
  
 public Integer getBoxWithAppliances() {  
 return boxWithAppliances;  
 }  
  
 public void setBoxWithAppliances(Integer boxWithAppliances) {  
 this.boxWithAppliances = boxWithAppliances;  
 }  
  
 public Integer getBoxWithFood() {  
 return boxWithFood;  
 }  
  
 public void setBoxWithFood(Integer boxWithFood) {  
 this.boxWithFood = boxWithFood;  
 }  
  
 public Integer getBoxWithTools() {  
 return boxWithTools;  
 }  
  
 public void setBoxWithTools(Integer boxWithTools) {  
 this.boxWithTools = boxWithTools;  
 }  
  
 @Override  
 public String toString() {  
 return "Warehouse{" +  
 "boxWithFood=" + boxWithFood +  
 ", boxWithTools=" + boxWithTools +  
 ", boxWithAppliances=" + boxWithAppliances +  
 ", area=" + area +  
 '}';  
 }  
}

# Додаток Б. Результат роботи програми

