Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

З лабораторної роботи №3

З дисципліни: «Кросплатформенні засоби  
програмування»

Варіант 10

Виконав: ст. Гр. KI-301

Ларіонов А.О

Прийняв:

Майдан М.В.

**Львів 2023**

**Мета:** ознайомитися з процесом розробки класів та пакетів мовою Java.

**Завдання:** Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету. Автоматично згенерувати документацію до розробленого пакету. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації. Дати відповідь на контрольні запитання.

**Лістинг main.py:**

from abc import ABC, abstractmethod

class House(ABC):

    """

    Клас, який представляє будинок.

    """

    def \_\_init\_\_(self, address="", floors=0, citizens=None, facade="Звичайний фасад"):

        """

        Конструктор класу House.

        :param address: Адреса будинку.

        :param floors: Кількість поверхів у будинку.

        :param citizens: Список жильців у будинку.

        :param facade: Тип фасаду будинку.

        """

        self.address = address

        self.floors = floors

        self.citizens = citizens if citizens is not None else []

        self.facade = facade

    def build\_house(self, address, floors):

        """

        Метод, який створює будинок.

        :param address: Адреса будинку.

        :param floors: Кількість поверхів у будинку.

        """

        self.address = address

        self.floors = floors

        print(f"Будинок на адресі {self.address} з {self.floors} поверхами був успішно збудований.")

    def remove\_house(self):

        """

        Метод, який знищує будинок.

        """

        print(f"Будинок на адресі {self.address} був знищений.")

        self.address = ""

        self.floors = 0

        self.citizens = []

    def add\_floor(self):

        """

        Метод, який додає поверх до будинку.

        """

        self.floors += 1

        print(f"Додано новий поверх. Загальна кількість поверхів: {self.floors}")

    def remove\_floor(self):

        """

        Метод, який деінсталює поверх будинку.

        """

        if self.floors > 1:

            self.floors -= 1

            print(f"Останній поверх був видалений. Загальна кількість поверхів: {self.floors}")

        else:

            print("Будинок не може існувати без поверхів")

    def add\_citizen(self, citizen\_name):

        """

        Метод, який поселяє жильця на поверх.

        :param citizen\_name: Ім'я жильця.

        """

        self.citizens.append(citizen\_name)

        print(f"{citizen\_name} був поселений у будинку.")

    def remove\_citizen(self, citizen\_name):

        """

        Метод, який виселяє жильця.

        :param citizen\_name: Ім'я жильця.

        """

        if citizen\_name in self.citizens:

            self.citizens.remove(citizen\_name)

            print(f"{citizen\_name} був виселений з будинку.")

        else:

            print(f"{citizen\_name} не знайдений у будинку.")

    def give\_address(self):

        """

        Метод, який повертає адресу будинку.

        :return: Адреса будинку.

        """

        return self.address

    def remove\_all\_citizens(self):

        """

        Метод, який виселяє всіх жильців.

        """

        self.citizens.clear()

        print("Всі жильці були виселені.")

    def show\_info(self):

        """

        Метод, який показує інформацію про будинок.

        """

        print(f"Адреса будинку: {self.address}")

        print(f"Кількість поверхів: {self.floors}")

        print(f"Список жильців: {', '.join(self.citizens)}")

        print(f"Фасад будинку: {self.facade}")

    def design\_house(self, new\_facade):

        """

        Метод, який змінює фасад будинку.

        :param new\_facade: Новий тип фасаду будинку.

        """

        self.facade = new\_facade

        print(f"Фасад будинку був змінений на: {self.facade}")

    @abstractmethod

    def office\_functionality(self):

        """

        Абстрактний метод, який потрібно реалізувати в підкласі Office.

        """

        pass

class Office(House):

    def office\_functionality(self, employee\_count):

        """

        Реалізація методу з абстрактного класу House.

        """

        print(f"Додаткові функціональності офісу. Кількість працівників: {employee\_count}")

my\_office = Office()

my\_office.build\_house("123 Main St", 5)

my\_office.add\_citizen("John")

my\_office.show\_info()

my\_office.office\_functionality(20)

**ClassHouse**from abc import ABC, abstractmethod

class House(ABC):

    """

    Клас, який представляє будинок.

    """

    def \_\_init\_\_(self, address="", floors=0, citizens=None, facade="Звичайний фасад"):

        """

        Конструктор класу House.

        :param address: Адреса будинку.

        :param floors: Кількість поверхів у будинку.

        :param citizens: Список жильців у будинку.

        :param facade: Тип фасаду будинку.

        """

        self.address = address

        self.floors = floors

        self.citizens = citizens if citizens is not None else []

        self.facade = facade

    def build\_house(self, address, floors):

        """

        Метод, який створює будинок.

        :param address: Адреса будинку.

        :param floors: Кількість поверхів у будинку.

        """

        self.address = address

        self.floors = floors

        print(f"Будинок на адресі {self.address} з {self.floors} поверхами був успішно збудований.")

    def remove\_house(self):

        """

        Метод, який знищує будинок.

        """

        print(f"Будинок на адресі {self.address} був знищений.")

        self.address = ""

        self.floors = 0

        self.citizens = []

    def add\_floor(self):

        """

        Метод, який додає поверх до будинку.

        """

        self.floors += 1

        print(f"Додано новий поверх. Загальна кількість поверхів: {self.floors}")

    def remove\_floor(self):

        """

        Метод, який деінсталює поверх будинку.

        """

        if self.floors > 1:

            self.floors -= 1

            print(f"Останній поверх був видалений. Загальна кількість поверхів: {self.floors}")

        else:

            print("Будинок не може існувати без поверхів")

    def add\_citizen(self, citizen\_name):

        """

        Метод, який поселяє жильця на поверх.

        :param citizen\_name: Ім'я жильця.

        """

        self.citizens.append(citizen\_name)

        print(f"{citizen\_name} був поселений у будинку.")

    def remove\_citizen(self, citizen\_name):

        """

        Метод, який виселяє жильця.

        :param citizen\_name: Ім'я жильця.

        """

        if citizen\_name in self.citizens:

            self.citizens.remove(citizen\_name)

            print(f"{citizen\_name} був виселений з будинку.")

        else:

            print(f"{citizen\_name} не знайдений у будинку.")

    def give\_address(self):

        """

        Метод, який повертає адресу будинку.

        :return: Адреса будинку.

        """

        return self.address

    def remove\_all\_citizens(self):

        """

        Метод, який виселяє всіх жильців.

        """

        self.citizens.clear()

        print("Всі жильці були виселені.")

    def show\_info(self):

        """

        Метод, який показує інформацію про будинок.

        """

        print(f"Адреса будинку: {self.address}")

        print(f"Кількість поверхів: {self.floors}")

        print(f"Список жильців: {', '.join(self.citizens)}")

        print(f"Фасад будинку: {self.facade}")

    def design\_house(self, new\_facade):

        """

        Метод, який змінює фасад будинку.

        :param new\_facade: Новий тип фасаду будинку.

        """

        self.facade = new\_facade

        print(f"Фасад будинку був змінений на: {self.facade}")

    @abstractmethod

    def office\_functionality(self):

        """

        Абстрактний метод, який потрібно реалізувати в підкласі Office.

        """

        pass

**ClassOffice**

class Office(House):

    def office\_functionality(self, employee\_count):

        """

        Реалізація методу з абстрактного класу House.

        """

        print(f"Додаткові функціональності офісу. Кількість працівників: {employee\_count}")

**Висновок:** ознайомився з процесом розробки класів.