Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

**Кафедра ЕОМ**



**Звіт**

**Лабораторна робота № 2**

З дисципліни «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

*Виконала: ст. гр. КІ-303*

*Кілик І.Р.*

*Перевірив:*

*доцент кафедри ЕОМ*

*Іванов Ю.С.*

***Львів 2025***

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

1. *ТЕОРЕТИЧНІ ВІДОМОСТІ*

*Класи*

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів. Синтаксис оголошення простого класу в мові Java має наступний вигляд:

[public] class Назва Класу

{

[конструктори]

[методи]

[поля]

}

Метод – функція-член класу, яка призначена маніпулювати станом об’єкту класу. Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом вказування різної кількості параметрів та їх типів методам з однаковими назвами. Синтаксис оголошення методу наступний:

[Специфікатор Доступу] [static] [final] Тип назва Методу([параметри])

[throws класи]

{

[Тіло методу]

[return [значення]];

}

Конструктор – спеціальний метод класу, який не повертає значення, має ім’я класу та призначений для початкової ініціалізації об’єктів класу. Синтаксис оголошення конструктора:

[Специфікатор Доступу] Назва Класу([параметри])

{

Тіло конструктора

}

Конструкторів може бути кілька. Конструктор без параметрів називається конструктором за замовчуванням.

***ЗАВДАННЯ***

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну

область згідно варіанту. Програма має задовольняти наступним вимогам:

• програма має розміщуватися в пакеті Група.Прізвище.Lab2;

• клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові частини предметної області;

• клас має містити кілька конструкторів та мінімум 10 методів;

• для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;

• методи класу мають вести протокол своєї діяльності, що записується у файл;

• розробити механізм коректного завершення роботи з файлом (не надіятися на

метод finalize());

• програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

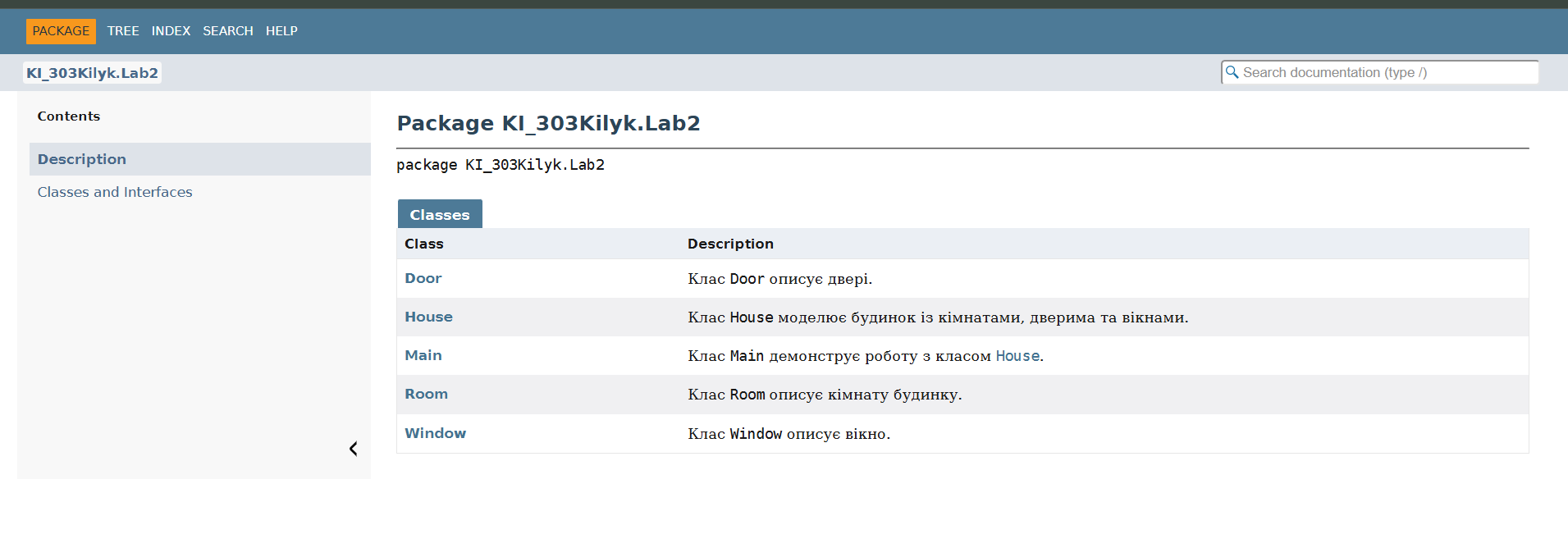
5. Дати відповідь на контрольні запитання.

***Варіант – 10***

***Будинок***

***Висновок:*** Завдяки даній лабораторній роботі, яознайомилась з процесом розробки класів та пакетів мовою Java.

***Згенерована документація:***

******

***Додатки***

package KI\_303Kilyk.Lab2;

/\*\*

 \* Клас {@code Door} описує двері.

 \*/

public class Door {

    private String type;

    private boolean isOpen;

    public Door(String type) {

        this.type = type;

        this.isOpen = false;

    }

    public void open() { isOpen = true; }

    public void close() { isOpen = false; }

    public String getType() { return type; }

}

package KI\_303Kilyk.Lab2;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDateTime;

/\*\*

 \* Клас {@code House} моделює будинок із кімнатами, дверима та вікнами.

 \* Веде журнал дій у файл та дублює повідомлення в консоль.

 \*/

public class House {

    private Room room;

    private Door door;

    private Window window;

    private FileWriter logWriter;

    /\*\*

     \* Конструктор за замовчуванням.

     \*/

    public House() throws IOException {

        this(new Room("Вітальня", 20), new Door("дерев’яні"), new Window("пластикові"));

    }

    /\*\*

     \* Повний конструктор.

     \* @param room кімната

     \* @param door двері

     \* @param window вікно

     \*/

    public House(Room room, Door door, Window window) throws IOException {

        this.room = room;

        this.door = door;

        this.window = window;

        this.logWriter = new FileWriter("house\_log.txt", true);

        log("Створено будинок");

    }

    // ---------- Методи роботи з дверима ----------

    public void openDoor() throws IOException {

        door.open();

        log("Двері відкрито");

    }

    public void closeDoor() throws IOException {

        door.close();

        log("Двері зачинено");

    }

    // ---------- Методи роботи з вікном ----------

    public void openWindow() throws IOException {

        window.open();

        log("Вікно відкрите");

    }

    public void closeWindow() throws IOException {

        window.close();

        log("Вікно закрите");

    }

    // ---------- Методи роботи з кімнатою ----------

    public void cleanRoom() throws IOException {

        log("Кімнату \"" + room.getName() + "\" прибрано");

    }

    public void paintWalls(String color) throws IOException {

        room.setColor(color);

        log("Стіни кімнати пофарбовано в " + color);

    }

    public void enlargeRoom(double extraArea) throws IOException {

        double newArea = room.getArea() + extraArea;

        log("Площа кімнати збільшена на " + extraArea + " м². Нова площа: " + newArea);

    }

    public void changeRoomName(String newName) throws IOException {

        log("Назву кімнати змінено з \"" + room.getName() + "\" на \"" + newName + "\"");

    }

    // ---------- Інші методи ----------

    public void describe() throws IOException {

        String info = "Будинок: " + room.getName() + " (" + room.getArea() + " м², колір стін: " + room.getColor() +

                "), двері: " + door.getType() + ", вікна: " + window.getType();

        log(info);

    }

    /\*\*

     \* Метод логування: пише повідомлення у файл і дублює в консоль.

     \* @param message повідомлення

     \*/

    public void log(String message) throws IOException {

        String logMessage = LocalDateTime.now() + " : " + message;

        logWriter.write(logMessage + "\n");

        logWriter.flush(); // обов’язкове збереження у файл

        System.out.println(logMessage); // дублювання в консоль

    }

    /\*\*

     \* Коректно завершує роботу з файлом журналу.

     \*/

    public void closeLog() {

        try {

            if (logWriter != null) {

                log("Закриття журналу");

                logWriter.close();

            }

        } catch (IOException e) {

            System.err.println("Помилка при закритті файлу логів: " + e.getMessage());

        }

    }

}

package KI\_303Kilyk.Lab2;

/\*\*

 \* Клас {@code Main} демонструє роботу з класом {@link House}.

 \*/

public class Main {

    public static void main(String[] args) {

        try {

            House house = new House();

            house.describe();

            house.openDoor();

            house.openWindow();

            house.paintWalls("блакитний");

            house.cleanRoom();

            house.enlargeRoom(5);

            house.changeRoomName("Спальня");

            house.closeDoor();

            house.closeWindow();

            house.closeLog();

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

package KI\_303Kilyk.Lab2;

/\*\*

 \* Клас {@code Room} описує кімнату будинку.

 \*/

public class Room {

    private String name;

    private double area;

    private String color;

    public Room(String name, double area) {

        this.name = name;

        this.area = area;

        this.color = "білий";

    }

    public String getName() { return name; }

    public double getArea() { return area; }

    public void setColor(String color) { this.color = color; }

    public String getColor() { return color; }

}

package KI\_303Kilyk.Lab2;

/\*\*

 \* Клас {@code Window} описує вікно.

 \*/

public class Window {

    private String type;

    private boolean isOpen;

    public Window(String type) {

        this.type = type;

        this.isOpen = false;

    }

    public void open() { isOpen = true; }

    public void close() { isOpen = false; }

    public String getType() { return type; }

}