

Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



## **Звіт**

З лабораторної роботи №2

Варіант – 10

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Виконав: ст. гр. КІ-306

Миценко О. С.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

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

### Завдання (варіант № 10)

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту(10. Будинок). 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 розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання

### Вихідний код програми

Файл House.java

```
package K1306MytsenkoLab2;

import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;

/**
 * The <code>House</code> class represents a house and its operations.
 * It includes functionality for managing the number of floors, addresses,
 * gardens and gives information about house .
 * This class also logs events to a file named "Log.txt".
 *
 * @author Oleksandr Mytsenko
 * @version 1.0
 */

public class House {
    private FileWriter writer; // Поле для зберігання посилання на потік запису в файл

    private String address;
    private int numberOfFloors;
    private boolean hasGarden;
```

```

/**
 * Default constructor for the house.
 *
 */
// Конструктори
public House(){
    address = "No information";
    numberOfFloors = 0;
    hasGarden = false;

}

/**
 * Parameterized constructor for the house.
 *
 * @param address Specifies initial address.
 * @param numberOfFloors The initial number of floors.
 * @param hasGarden Specifies if the garden initially exists.
 */
public House(String address, int numberOfFloors, boolean hasGarden) {
    this.address = address;
    this.numberOfFloors = numberOfFloors;
    this.hasGarden = hasGarden;
}

/**
 * Constructs a house with the specified address and number of floors, defaulting to no garden.
 *
 * @param address The address of the house.
 * @param numberOfFloors The number of floors in the house.
 */
public House(String address, int numberOfFloors) {
    this(address, numberOfFloors, false);
}

/**
 * Constructs a house with the specified address and defaults to one floor and no garden.
 *
 * @param address The address of the house.
 */
public House(String address) {
    this(address, 1);
}

// Методи

// Метод для відкриття файлу для запису
/**
 * Open the file for writing
 */
public void openLogFile() {

```

```

try {
    writer = new FileWriter("log.txt", true);
} catch (IOException e) {
    System.err.println("Помилка при відкритті файлу для запису: " + e.getMessage());
}
}

```

// Метод для закриття файлу після закінчення запису

```

/**
 * Close the file after ending of writing
 */
public void closeLogFile() {
    try {
        if (writer != null) {
            writer.close();
        }
    } catch (IOException e) {
        System.err.println("Помилка при закритті файлу: " + e.getMessage());
    }
}

```

```

/**
 * Display details of the house
 */
public void displayDetails() {
    System.out.println("Адреса будинку: " + address);
    System.out.println("Кількість поверхів: " + numberOfFloors);
    System.out.println("Наявність саду: " + (hasGarden ? "Так" : "Ні"));
}

```

```

/**
 * Log a message to the file.
 *
 * @param message The message to log.
 */
private void logMessage(String message) {
    try (FileWriter writer = new FileWriter("log.txt", true)) {
        LocalDateTime timestamp = LocalDateTime.now();
        writer.write("[ " + timestamp + " ] " + message + " - " + this + "\n");
    } catch (IOException e) {
        System.err.println("Помилка при записі до файлу: " + e.getMessage());
    }
}

```

```

/**
 * Set the number of floors for the house.
 *
 * @param numberOfFloors The number of floors to set.

```

```

*/
public void setNumberOfFloors(int numberOfFloors) {

    this.numberOfFloors = numberOfFloors;
    logMessage("Встановлено кількість поверхів: " + numberOfFloors);
}
/**
 * Set the address for the house.
 *
 * @param address The address to set.
 */
public void setAddress(String address) {
    this.address = address;
    logMessage("Оновлено адресу: " + address);
}
/**
 * Set whether the house has a garden or not.
 *
 * @param hasGarden True if the house has a garden, false otherwise.
 */
public void setHasGarden(boolean hasGarden) {

    this.hasGarden = hasGarden;
    logMessage("Оновлено інформацію про сад.");
}
/**
 * Add a floor to the house.
 */
public void addFloor() {

    numberOfFloors++;
    logMessage("Додано поверх.");
}
/**
 * Remove a floor from the house.
 * If the number of floors is already at the minimum, log a message accordingly.
 */
public void removeFloor() {
    if (numberOfFloors > 0) {
        numberOfFloors--;
        logMessage("Видалено поверх.");
    } else {
        logMessage("Не можна видалити поверх. Кількість поверхів вже мінімальна.");
    }
}
/**
 * Get the number of floors for the house.
 *
 * @return The number of floors.

```

```

*/
public int getNumberOfFloors() {
    logMessage("Дана інформація про кількість поверхів.");
    return numberOfFloors;
}
/**
 * Get the address of the house.
 *
 * @return The address.
 */
public String getAddress() {
    logMessage("Дана інформація про адресу.");
    return address;
}
/**
 * Check if the house has a garden.
 *
 * @return True if the house has a garden, false otherwise.
 */
public boolean hasGarden() {
    logMessage("Дана інформація про наявність саду.");
    return hasGarden;
}

// Додаткові методи

}

```

### Файл HouseDrive.java

```

package KI306MytsenkoLab2;

public class HouseDrive {
    public static void main(String[] args) {
        House house1 = new House("Вулиця Лінкольна, 123", 3, true);
        House house2 = new House("Вулиця Індепенденс, 456");
        House house3 = new House("Вулиця Кеннеді, 789", 2);

        house1.openLogFile();
        house2.openLogFile();
        house3.openLogFile();

        house1.displayDetails();
        house2.displayDetails();
        house3.displayDetails();

        house1.setAddress("Вулиця Нова, 555");
        house1.setNumberOfFloors(4);
    }
}

```

```

house1.addFloor();
house1.setHasGarden(false);
house1.removeFloor();

house2.setNumberOfFloors(6);
house2.addFloor();
house2.setHasGarden(true);
house2.removeFloor();

house3.setHasGarden(true);
house3.addFloor();

house1.displayDetails();
house2.displayDetails();
house3.displayDetails();

house1.closeLogFile();
house2.closeLogFile();
house3.closeLogFile();
}
}

```

## Результат виконання програми

Log.txt:

```

1  [2023-09-22T22:24:33.848248200] Оновлено адресу: Вулиця Нова, 555 - KI306MytsenkoLab2.House@7cd84586
2  [2023-09-22T22:24:33.857224400] Встановлено кількість поверхів: 4 - KI306MytsenkoLab2.House@7cd84586
3  [2023-09-22T22:24:33.858223100] Додано поверх. - KI306MytsenkoLab2.House@7cd84586
4  [2023-09-22T22:24:33.858223100] Оновлено інформацію про сад. - KI306MytsenkoLab2.House@7cd84586
5  [2023-09-22T22:24:33.858223100] Видалено поверх. - KI306MytsenkoLab2.House@7cd84586
6  [2023-09-22T22:24:33.858223100] Встановлено кількість поверхів: 6 - KI306MytsenkoLab2.House@7106e68e
7  [2023-09-22T22:24:33.858223100] Додано поверх. - KI306MytsenkoLab2.House@7106e68e
8  [2023-09-22T22:24:33.859219800] Оновлено інформацію про сад. - KI306MytsenkoLab2.House@7106e68e
9  [2023-09-22T22:24:33.859219800] Видалено поверх. - KI306MytsenkoLab2.House@7106e68e
10 [2023-09-22T22:24:33.859219800] Оновлено інформацію про сад. - KI306MytsenkoLab2.House@7eda2dbb
11 [2023-09-22T22:24:33.859219800] Додано поверх. - KI306MytsenkoLab2.House@7eda2dbb

```

## Фрагмент згенерованої документації

Internet Explorer restricted this webpage from running scripts or ActiveX controls. [Allow blocked content](#)

Activate Windows  
Go to Settings to activate Windows.

18°C Mostly cloudy

10:34 PM  
9/22/2023

## Відповіді на контрольні запитання

1. Синтаксис визначення класу.

- `public class ClassName { // Class members (fields, methods, constructors) }`

2. Синтаксис визначення методу.

- `public returnType methodName(parameters) { // Method body }`

3. Синтаксис оголошення поля.

- `accessModifier dataType fieldName;`

4. Як оголосити та ініціалізувати константне поле?

- `public static final dataType CONSTANT_NAME = initial_value;`

5. Які є способи ініціалізації полів?

- Явна ініціалізація при оголошенні поля.

- Ініціалізація у конструкторі класу.

- Ініціалізація у блоку ініціалізації (конструкторі, статичному або звичайному).

6. Синтаксис визначення конструктора.

- `public ClassName(parameters) { // Constructor body }`

7. Синтаксис оголошення пакету.

- `package packageName.subpackage;`

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

- Вказати повне ім'я класу перед використанням (наприклад, `java.util.Date today = new java.util.Date();`).

- Використовувати оператор `import` для підключення класів з інших пакетів, щоб уникнути повторення повного імені класу.

9. В чому суть статичного імпорту пакетів?

- Статичний імпорт дозволяє підключити статичні методи і поля класів без повного імені класу.

- Завдяки статичному імпорту, можна використовувати статичні члени класу, не додаючи перед ними ім'я класу.

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

- Назви пакетів повинні відповідати структурі каталогів.

- Назви загальнодоступних класів повинні співпадати з назвами файлів, де вони розміщені.

- Після компіляції ієрархія каталогів проекту повинна відповідати ієрархії пакетів.

- Для компіляції та запуску програми слід використовувати шляхи до файлів та пакетів.

**Висновок:** в даній лабораторній роботі ознайомитися з процесом розробки класів та пакетів мовою Java.