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

Згурський Т.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
House.java:
```
package org.example;

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 provides information about the house.
 * This class also logs events to a file named "Log.txt".
 *
 * @version 1.0
 */
public class House {
    private FileWriter writer; // Field to store a reference to the file writing stream

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

```
// Static variables for tracking parameters
public static int param3 = 0;
public static int param2 = 0;

/**
 * Default constructor for the house.
 */
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;
    param3++;
}

/**
 * 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);
    param2++;
    param3--;
}

/**
```

```
 * 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);
}

// Methods

/**
 * Open the file for writing.
 */
public void openLogFile() {
    try {
        writer = new FileWriter("log.txt", true);
    } catch (IOException e) {
        System.err.println("Error opening the file for writing: " + e.getMessage());
    }
}

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

/**
 * Display details of the house.
 */
public void displayDetails() {
    System.out.println("House address: " + address);
    System.out.println("Number of floors: " + numberOfFloors);
    System.out.println("Has a garden: " + (hasGarden ? "Yes" : "No"));
```

```
}

/**
 * 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("Error writing to the file: " + 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("Set the number of floors to: " + numberOfFloors);
}

/**
 * Set the address for the house.
 *
 * @param address The address to set.
 */
public void setAddress(String address) {
    this.address = address;
    logMessage("Updated address: " + 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("Updated garden information.");
}

/**
 * Add a floor to the house.
 */
public void addFloor() {
    numberOfFloors++;
    logMessage("Added a floor.");
}

/**
 * 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("Removed a floor.");
    } else {
        logMessage("Cannot remove a floor. The number of floors is already at the minimum.");
    }
}

/**
 * Get the number of floors for the house.
 *
 * @return The number of floors.
 */
public int getNumberOfFloors() {
    logMessage("Retrieved information about the number of floors.");
    return numberOfFloors;
}

/**
 * Get the address of the house.
 *
 * @return The address.
 */
```

```
    public String getAddress() {
        logMessage("Retrieved information about the address.");
        return address;
    }

    /**
     * Check if the house has a garden.
     *
     * @return True if the house has a garden, false otherwise.
     */
    public boolean hasGarden() {
        logMessage("Retrieved information about the garden.");
        return hasGarden;
    }

    // Additional methods

}
```

HouseDrive.java:

```
package org.example;

public class HouseDrive {
    public static void main(String[] args) {
        // Виклик методу main1 для виконання першої частини програми
        main1();

        // Створення об'єктів класу House
        House house1 = new House("Вулиця Лінкольна, 123", 3, true);
        House house2 = new House("Вулиця Індепенденс, 456");
        House house3 = new House("Вулиця Індепенденс, 456");

        // Виведення значень статичних змінних класу House
        System.out.print(House.param3 + "   " + House.param2);

        // Коментарі до наступних рядків закоментовані, оскільки вони викликають методи,
які наразі закоментовані

        /*
        // Відкриття лог-файлів для кожного об'єкту
```

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

        // Виведення деталей кожного будинку
        house1.displayDetails();
        house2.displayDetails();
        house3.displayDetails();

        // Зміни деталей першого будинку
        house1.setAddress("Вулиця Нова, 555");
        house1.setNumberOfFloors(4);
        house1.addFloor();
        house1.setHasGarden(false);
        house1.removeFloor();

        // Зміни деталей другого будинку
        house2.setNumberOfFloors(6);
        house2.addFloor();
        house2.setHasGarden(true);
        house2.removeFloor();

        // Зміни деталей третього будинку
        house3.setHasGarden(true);
        house3.addFloor();
```

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 додаваючи 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.