**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп'ютерних технологій, автоматики та метрології**

**кафедра "Електронних обчислювальних машин"**


Звіт

з лабораторної роботи №2

дисципліни «Кросплатформні засоби програмування»

на тему: «КЛАСИ ТА ПАКЕТИ»

Варіант 14

***Виконав: студент групи***
*КІ-303*
*Левченко Д.О.*
***Прийняв:***
*Іванов Ю.С.*

Львів – 2025

# ЛАБОРАТОРНА РОБОТА №2
# КЛАСИ ТА ПАКЕТИ

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

## Теоретичний матеріал

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.

Необов’язковий специфікатор доступу public робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім’я якого співпадає з назвою файлу, та безліч класів без специфікатора public.

Ініціалізація посилання на об’єкт класу здійснюється за допомогою оператора new і вказування конструктора, який має збудувати об’єкт. Одержаний в результаті цих операцій об’єкт розташується у області оперативної пам’яті що зветься ”куча”.

При створенні об’єктів дозволяється суміщати оголошення та ініціалізацію об’єктів, а також створювати анонімні об’єкти. Якщо посилання на об’єкт не посилається на жоден об’єкт, то йому слід присвоїти значення null. На відміну від полів-посилань на об’єкти, локальні змінні-посилання на об’єкти не ініціалізуються значенням null при оголошенні. Для них ініціалізацію посилання слід проводити явно.

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.

Конструктори, методи, та поля класу можуть бути відкритими (public), закритими (private) та захищеними (protected), що визначається специфікатором доступу. Специфікатор доступу public робить елемент класу загальнодоступним в межах пакету (набору класів, з яких складається програма). Специфікатор доступу private робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми (включаючи похідні класи). Специфікатор доступу protected робить елемент класу закритим (недоступним) для всіх зовнішніх відносно даного класу елементів програми, проте цей елемент буде загальнодоступним для похідних класів. Якщо будь-який елемент класу не має специфікатора доступу, то цей елемент автоматично стає відкритим та видимим у межах пакету (не плутати з public). Всі елементи класу, що оголошені без використання ключового слова static, належать об’єкту класу. Тобто, кожен об’єкт класу містить власну копію цих елементів класу. Ключове слово static робить поле або метод членом класу, а не об’єкту, тобто вони є спільними для

всіх об’єктів класу. Оскільки клас існує завжди, на відміну від об’єктів, які створюються в процесі роботи програми, то статичні елементи класу доступні.

Пакет – це механізм мови Java, що дозволяє об’єднувати класи в простори імен. Об’єднання класів в пакети дозволяє відділяти класи, що розроблені одними розробниками, від класів, що розроблені іншими розробниками, забезпечуючи тим самим унікальність імен класів в межах програми та усуваючи можливі конфлікти імен класів. Пакети можуть бути вкладеними одні в одних, утворюючи цим самим ієрархії пакетів. Будь-який зв’язок між вкладеними пакетами відсутній. Всі стандартні пакети належать ієрархіям java і javax, наприклад, java.lang, java.util, java.net тощо.

## Завдання (Варіант №14)

1) Написати та налагодити програму на мові Java 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; **ВАРІАНТ ЗАВДАННЯ: Телевізор**
   - 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.

## Виконання завдання

## TelevisionDriver.java

```
package KI303LEVCHENKOLAB2;

import java.io.FileWriter;
import java.io.IOException;

/**
 * Клас Logger забезпечує логування повідомлень у файл.
 * Використовується для запису дій та подій, що відбуваються в програмі.
 */
public class Logger {
    private FileWriter fileWriter;

    /**
     * Конструктор створює об'єкт Logger для запису повідомлень у вказаний файл.
     *
     * @param fileName ім'я файлу для запису логів.
     * @throws IOException якщо виникає помилка при створенні або відкритті файлу.
     */
    public Logger(String fileName) throws IOException {
        fileWriter = new FileWriter(fileName, true);
    }

    /**
     * Метод записує повідомлення у файл логу.
     *
     * @param message повідомлення, яке потрібно записати у файл.
     * @throws IOException якщо виникає помилка при записі у файл.
     */
    public void log(String message) throws IOException {
        if (fileWriter != null) {
            fileWriter.write(message + "\n");
            fileWriter.flush();
        }
    }

    /**
     * Метод закриває файл логу, звільняючи всі ресурси, пов'язані з ним.
     * У разі виникнення помилки при закритті, повідомлення про помилку буде виведено в
консоль.
     */
    public void close() {
        if (fileWriter != null) {
            try {
                fileWriter.close();
            } catch (IOException e) {
                System.err.println("Виникла помилка при закриванні файла: " +
e.getMessage());
            }
        }
    }
}
```

## Television.java

```java
package KI303LEVCHENKOLAB2;

import java.io.IOException;

/**
 * Клас `Television` представляє функціональність телевізора, який можна вмикати,
вимикати,
 * змінювати канали, гучність, налаштовувати яскравість екрану, а також керувати через
пульт.
 */
public class Television {
    private Screen screen;
    private Speaker speaker;
    private RemoteControl remoteControl;
    private Logger logger;
    private boolean isOn;
    private int currentChannel;
    private int volume;

    /**
     * Конструктор без параметрів, який ініціалізує телевізор зі стандартними
налаштуваннями.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public Television() throws IOException {
        this.screen = new Screen(50);
        this.speaker = new Speaker(100);
        this.remoteControl = new RemoteControl();
        this.isOn = false;
        this.currentChannel = 0;
        this.volume = 0;

        this.logger = new Logger("television_log.txt");
        logger.log(String.format("Телевізор %s створений.", this.toString()));
    }

    /**
     * Конструктор, який приймає параметри для ініціалізації телевізора.
     *
     * @param screen Об'єкт екрану.
     * @param speaker Об'єкт динаміка.
     * @param remoteControl Пульт керування.
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public Television(Screen screen, Speaker speaker, RemoteControl remoteControl) throws
IOException {
        this.screen = screen;
        this.speaker = speaker;
        this.remoteControl = remoteControl;
        this.isOn = false;
        this.currentChannel = 0;
        this.volume = 0;

        this.logger = new Logger("television_log.txt");
        logger.log(String.format("Телевізор %s створений.", this.toString()));
    }

    /**
```

```java
     * Вмикає телевізор.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void turnOn() throws IOException {
        isOn = true;
        logger.log("Телевізор увімкнено");
        System.out.println("Телевізор увімкнено");
    }

    /**
     * Вимикає телевізор.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void turnOff() throws IOException {
        isOn = false;
        logger.log("Телевізор вимкнено");
        System.out.println("Телевізор вимкнено");
    }

    /**
     * Перемикає на наступний канал.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void changeChannelUp() throws IOException {
        if (isOn) {
            currentChannel++;
            logger.log(String.format("Перемкнуто на канал %s", currentChannel));
            System.out.printf("Перемкнуто на канал %s\n", currentChannel);
        }
    }

    /**
     * Перемикає на попередній канал.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void changeChannelDown() throws IOException {
        if (isOn && currentChannel >= 1) {
            currentChannel--;
            logger.log(String.format("Перемкнуто на канал %s", currentChannel));
            System.out.printf("Перемкнуто на канал %s\n", currentChannel);
        }
    }

    /**
     * Збільшує гучність телевізора.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void volumeUp() throws IOException {
        if (isOn && volume < 100) {
            volume++;
            speaker.setVolume(volume);
            logger.log(String.format("Гучність збільшено до %s", volume));
            System.out.printf("Гучність збільшено до %s\n", volume);
        }
    }

    /**
     * Зменшує гучність телевізора.
```

```java
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void volumeDown() throws IOException {
        if (isOn && volume > 0) {
            volume--;
            speaker.setVolume(volume);
            logger.log(String.format("Гучність зменшено до %s", volume));
            System.out.printf("Гучність зменшено до %s\n", volume);
        }
    }

    /**
     * Змінює розмір екрану телевізора.
     *
     * @param size Новий розмір екрану (у дюймах).
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void changeScreenSize(int size) throws IOException {
        screen.setSize(size);
        logger.log(String.format("Розмір екрану змінено на %s дюймів", size));
        System.out.printf("Розмір екрану змінено на %s дюймів\n", size);
    }

    /**
     * Змінює яскравість екрану телевізора.
     *
     * @param brightness Нова яскравість (0-100).
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void adjustBrightness(int brightness) throws IOException {
        if (isOn) {
            screen.setBrightness(brightness);
            logger.log(String.format("Яскравість екрану встановлено на %s", brightness));
            System.out.printf("Яскравість екрану встановлено на %s\n", brightness);
        }
    }

    /**
     * Замінює батарейки в пульті керування.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public void changeRemoteBatteries() throws IOException {
        remoteControl.changeBatteries();
        logger.log("Замінено батарейки в пульті керування");
        System.out.println("Замінено батарейки в пульті керування");
    }

    /**
     * Перевіряє стан телевізора.
     *
     * @return Рядок зі станом телевізора (ввімкнений/вимкнений, канал, гучність).
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public String checkStatus() throws IOException {
        String status = "Телевізор " + (isOn ? "увімкнено" : "вимкнено") +
                ". Канал: " + currentChannel +
                ". Гучність: " + volume;

        logger.log(String.format("Перевірено стан: %s", status));
        System.out.printf("Перевірено стан: %s\n", status);
        return status;
```

```
    }

    /**
     * Закриває логер для збереження даних у файл.
     *
     * @throws IOException якщо виникає помилка під час закриття логера.
     */
    public void closeLogger() throws IOException {
        logger.close();
    }
}
```

## Screen.java

```
package KI303LEVCHENKOLAB2;

/**
 * Клас `Screen` представляє екран телевізора з можливістю налаштування розміру та
яскравості.
 */
public class Screen {
    private int size;
    private int brightness;

    /**
     * Конструктор для створення екрану з вказаним розміром.
     *
     * @param size Розмір екрану у дюймах.
     */
    public Screen(int size) {
        this.size = size;
        this.brightness = 50; // За замовчуванням яскравість встановлена на 50
    }

    /**
     * Встановлює новий розмір екрану.
     *
     * @param size Новий розмір екрану у дюймах.
     */
    public void setSize(int size) {
        this.size = size;
    }

    /**
     * Встановлює новий рівень яскравості екрану.
     *
     * @param brightness Нова яскравість екрану (0-100).
     */
    public void setBrightness(int brightness) {
        this.brightness = brightness;
    }
}
```

## Speaker.java

```java
package KI303LEVCHENKOLAB2;

/**
 * Клас `Speaker` представляє динамік телевізора з можливістю налаштування гучності.
 */
public class Speaker {
    private int maxVolume;
    private int currentVolume;

    /**
     * Конструктор для створення динаміка з вказаною максимальною гучністю.
     *
     * @param maxVolume Максимальна гучність динаміка.
     */
    public Speaker(int maxVolume) {
        this.maxVolume = maxVolume;
        this.currentVolume = maxVolume / 2; // За замовчуванням встановлюється половина від максимальної гучності
    }

    /**
     * Встановлює нову гучність для динаміка. Якщо вказана гучність перевищує
     * максимальну, вона автоматично обмежується максимальним значенням.
     *
     * @param volume Нова гучність.
     */
    public void setVolume(int volume) {
        this.currentVolume = Math.min(volume, maxVolume);
    }
}
```

## RemoteControl.java

```java
package KI303LEVCHENKOLAB2;

/**
 * Клас `RemoteControl` представляє пульт керування телевізором з можливістю заміни батарейок.
 */
public class RemoteControl {
    private boolean hasBatteries;

    /**
     * Конструктор для створення пульта керування.
     * За замовчуванням батарейки присутні.
     */
    public RemoteControl() {
        this.hasBatteries = true;
    }

    /**
     * Змінює батарейки у пульті керування.
     * Після виконання методу батарейки вважаються заміненими.
     */
    public void changeBatteries() {
        this.hasBatteries = true;
    }
}
```

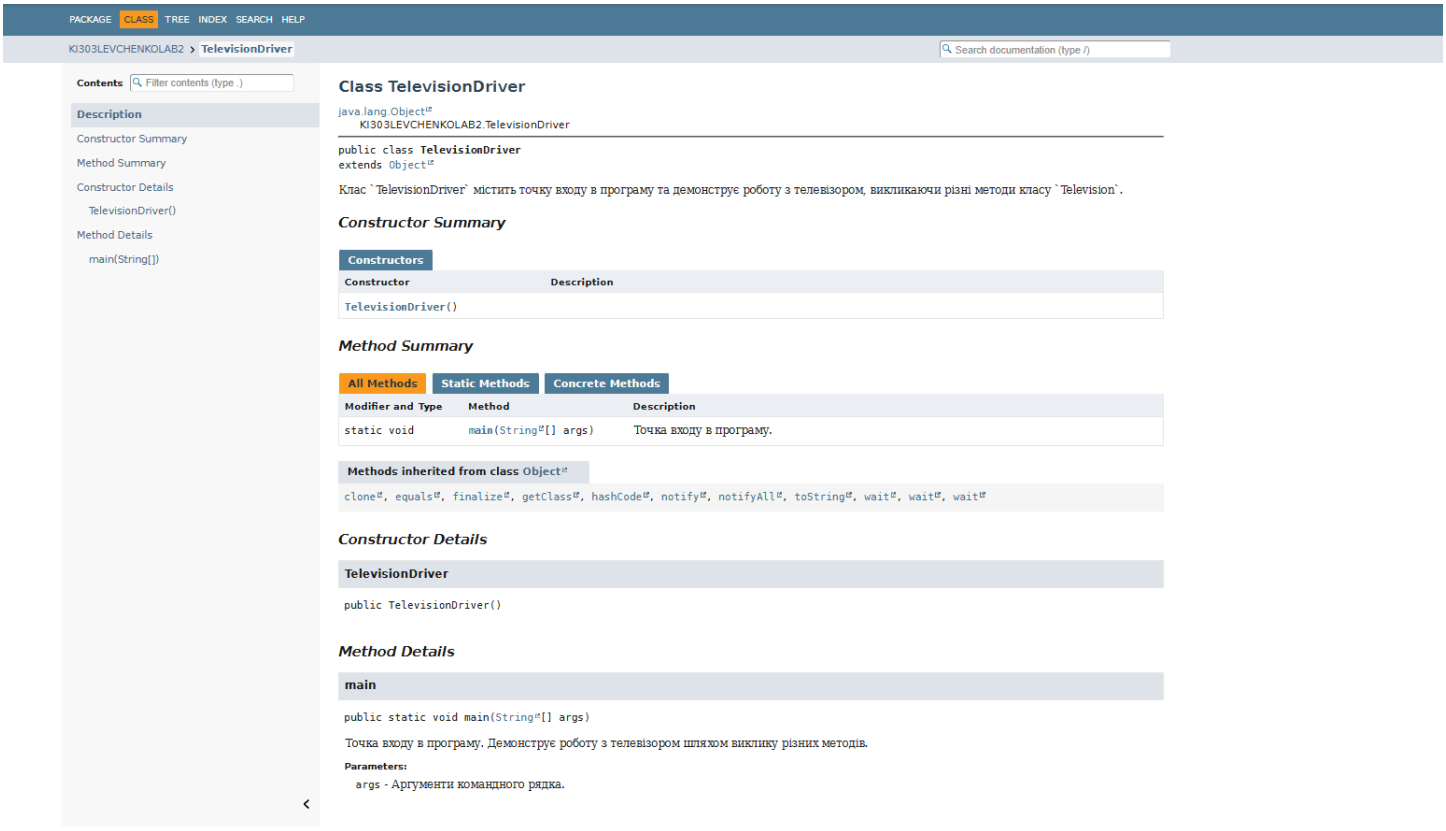
## Logger.java

```java
package KI303LEVCHENKOLAB2;

import java.io.FileWriter;
import java.io.IOException;

/**
 * Клас Logger забезпечує логування повідомлень у файл.
 * Використовується для запису дій та подій, що відбуваються в програмі.
 */
public class Logger {
    private FileWriter fileWriter;

    /**
     * Конструктор створює об'єкт Logger для запису повідомлень у вказаний файл.
     *
     * @param fileName ім'я файлу для запису логів.
     * @throws IOException якщо виникає помилка при створенні або відкритті файлу.
     */
    public Logger(String fileName) throws IOException {
        fileWriter = new FileWriter(fileName, true);
    }

    /**
     * Метод записує повідомлення у файл логу.
     *
     * @param message повідомлення, яке потрібно записати у файл.
     * @throws IOException якщо виникає помилка при записі у файл.
     */
    public void log(String message) throws IOException {
        if (fileWriter != null) {
            fileWriter.write(message + "\n");
            fileWriter.flush();
        }
    }

    /**
     * Метод закриває файл логу, звільняючи всі ресурси, пов'язані з ним.
     * У разі виникнення помилки при закритті, повідомлення про помилку буде виведено в
консоль.
     */
    public void close() {
        if (fileWriter != null) {
            try {
                fileWriter.close();
            } catch (IOException e) {
                System.err.println("Виникла помилка при закриванні файла: " +
e.getMessage());
            }
        }
    }
}
```

```
Телевізор увімкнено
Перемкнуто на канал 1
Перемкнуто на канал 0
Гучність збільшено до 1
Гучність зменшено до 0
Розмір екрану змінено на 61 дюймів
Яскравість екрану встановлено на 64
Замінено батарейки в пульті керування
Перевірено стан: Телевізор увімкнено. Канал: 0. Гучність: 0
Телевізор вимкнено
```

**Рис 1.1 (Знімок результату з television_log.txt)**

# Фрагменти документації до коду



PACKAGE CLASS TREE INDEX SEARCH HELP

KI303LEVCHENKOLAB2 › TelevisionDriver

Search documentation (type /)

Contents Filter contents (type .)

- Description
- Constructor Summary
- Method Summary
- Constructor Details
  - TelevisionDriver()
- Method Details
  - main(String[])

## Class TelevisionDriver

java.lang.Object
  KI303LEVCHENKOLAB2.TelevisionDriver

public class **TelevisionDriver**
extends Object

Клас `TelevisionDriver` містить точку входу в програму та демонструє роботу з телевізором, викликаючи різні методи класу `Television`.

### Constructor Summary

**Constructors**

| Constructor | Description |
|---|---|
| TelevisionDriver() | |

### Method Summary

**All Methods** | **Static Methods** | **Concrete Methods**

| Modifier and Type | Method | Description |
|---|---|---|
| static void | main(String[] args) | Точка входу в програму. |

**Methods inherited from class Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Details

**TelevisionDriver**

```
public TelevisionDriver()
```

### Method Details

**main**

```
public static void main(String[] args)
```

Точка входу в програму. Демонструє роботу з телевізором шляхом виклику різних методів.

**Parameters:**

`args` - Аргументи командного рядка.

**Відповіді на контрольні питання**

1) Синтаксис визначення класу. [public] class Name {}
2) Синтаксис визначення методу. [public] returnType nameMethod(param){}
3) Синтаксис оголошення поля. [public] тип fieldname;
4) Як оголосити та ініціалізувати константне поле? [public] static final [int ]CONSTANT_NAME = 100;
5) Які є способи ініціалізації полів? Пряма ініціалізація при оголошенні, Ініціалізація в конструкторі, Ініціалізація через блок ініціалізації або статичний блок.
6) Синтаксис визначення конструктора. public ClassName (param){}
7) Синтаксис оголошення пакету. package com.example.myapp;
8) Як підключити до програми класи, що визначені в зовнішніх пакетах? import com.example.otherpackage.ClassName;
9) В чому суть статичного імпорту пакетів? Дозволяє імпортувати статичні поля або методи класу, щоб використовувати їх без вказування імені класу.
10) Які вимоги ставляться до файлів і каталогів при використанні пакетів? Ім'я пакету має відповідати структурі каталогів. Файли класів мають бути розміщені в каталогах, що відповідають іменам пакетів.

**Висновок:** Я ознайомився з процесом розробки класів та пакетів мовою Java.