МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних технологій, автоматики та метрології

кафедра "Електронних обчислювальних машин"

Звіт

з лабораторної роботи №3

дисципліни «Кросплатформні засоби програмування»

на тему: «Спадкування та інтерфейси»

Варіант 14

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

Львів – 2025

ЛАБОРАТОРНА РОБОТА №3
СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Теоретичний матеріал

Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови C++ не існує. На відміну від C++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class після якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі.

При наслідувані у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних. Перевизначення методу полягає у визначені у підкласі методу з сигнатурою методу суперкласу. При виклику такого методу з-під об’єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку.

Використання ключового слова super у конструкторах підкласів має дещо інший сенс, ніж у методах. Тут воно застосовується для виклику конструктора суперкласу. Виклик конструктора суперкласу має бути першим оператором конструктора підкласу. Конкретний конструктор, який необхідно викликати, вибирається по переданим параметрам. Явний виклик конструктора суперкласу часто є необхідним, оскільки підкласи не мають доступу до приватних полів суперкласів. Тож ініціалізація їх полів значеннями відмінними від значень за замовчуванням без явного виклику відповідного конструктора суперкласу є неможливою. Якщо виклик конструктора суперкласу не вказаний явно у підкласі або суперклас не має конструкторів, тоді автоматично викликається конструктор за замовчуванням суперкласу.

Механізм поліморфізму забезпечує можливість присвоєння об’єктним змінним суперкласу об’єктів похідних класів та звертання з-під цих змінних до перевизначених у підкласі членів суперкласу. У Java всі об’єктні змінні є поліморфними. Поліморфізм

реалізується за допомогою механізму динамічного (пізнього) зв’язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми.

Завдання (Варіант №14)

1) Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2) Автоматично згенерувати документацію до розробленого пакету.
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

TunerTelevisionDriver.java

```
package KI303LEVCHENKOLAB3;

import java.io.IOException;

/**
 * Клас `TelevisionDriver` містить точку входу в програму та демонструє роботу з
телевізором,
 * викликаючи різні методи класу `Television`.
 */
public class TunerTelevisionDriver {
    /**
     * Точка входу в програму. Демонструє роботу з телевізором шляхом виклику різних
методів.
     *
     * @param args Аргументи командного рядка.
     */
    public static void main(String[] args) {
        try {
            TunerTelevision tv = new TunerTelevision(
                    new Screen(55),
                    new Speaker(100),
```

```
                    new RemoteControl()
            );

            tv.turnOn();
            tv.scanChannels();
            tv.changeChannelUp();
            tv.addFavoriteChannel(5);
            tv.addFavoriteChannel(10);
            tv.volumeUp();
            tv.adjustBrightness(75);
            tv.removeFavoriteChannel(5);
            tv.checkStatus();
            tv.turnOff();

            tv.closeLogger();
        } catch (IOException e) {
            // Обробка помилок, що виникають під час запису в файл
            throw new RuntimeException("Сталася помилка при записі в файл: " +
e.getMessage());
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
}
```

TunerTelevision.java

```
package KI303LEVCHENKOLAB3;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * Клас TunerTelevision розширює Television та реалізує інтерфейс TunerCapable,
 * що дозволяє сканувати канали та керувати улюбленими каналами.
 */
public class TunerTelevision extends Television implements TunerCapable {
    private List<Integer> favoriteChannels;

    /**
     * Конструктор для створення телевізора з тюнером.
     *
     * @param screen Екран телевізора
     * @param speaker Динамік телевізора
     * @param remoteControl Пульт керування
     * @throws IOException якщо виникає помилка при ініціалізації логера
     */
    public TunerTelevision(Screen screen, Speaker speaker, RemoteControl remoteControl)
throws IOException {
        super(screen, speaker, remoteControl);
        this.favoriteChannels = new ArrayList<>();
    }

    /**
     * Вмикає телевізор з тюнером.
     *
```

```java
     * @throws IOException якщо виникає помилка під час увімкнення
     */
    @Override
    public void turnOn() throws IOException {
        isOn = true;
        logger.log("Телевізор з тюнером увімкнено");
        System.out.println("Телевізор з тюнером увімкнено");
    }

    /**
     * Вимикає телевізор з тюнером.
     *
     * @throws IOException якщо виникає помилка під час вимкнення
     */
    @Override
    public void turnOff() throws IOException {
        isOn = false;
        logger.log("Телевізор з тюнером вимкнено");
        System.out.println("Телевізор з тюнером вимкнено");
    }

    /**
     * Сканує доступні телевізійні канали.
     *
     * @throws IOException якщо виникає помилка під час сканування
     * @throws InterruptedException якщо процес сканування перервано
     */
    @Override
    public void scanChannels() throws IOException, InterruptedException {
        if (isOn) {
            logger.log("Сканування каналів...");
            System.out.println("Сканування каналів...");
            // Імітація сканування каналів
            Thread.sleep(2000);
            logger.log("Сканування завершено. Знайдено 50 каналів.");
            System.out.println("Сканування завершено. Знайдено 50 каналів.");
        }
    }

    /**
     * Додає канал до списку улюблених.
     *
     * @param channel Номер каналу, який необхідно додати
     * @throws IOException якщо виникає помилка під час додавання каналу
     */
    @Override
    public void addFavoriteChannel(int channel) throws IOException {
        if (!favoriteChannels.contains(channel)) {
            favoriteChannels.add(channel);
            logger.log("Канал " + channel + " додано до улюблених");
            System.out.println("Канал " + channel + " додано до улюблених");
        }
    }

    /**
     * Видаляє канал зі списку улюблених.
     *
     * @param channel Номер каналу, який необхідно видалити
     * @throws IOException якщо виникає помилка під час видалення каналу
     */
    @Override
    public void removeFavoriteChannel(int channel) throws IOException {
        if (favoriteChannels.remove(Integer.valueOf(channel))) {
```

```
            logger.log("Канал " + channel + " видалено з улюблених");
            System.out.println("Канал " + channel + " видалено з улюблених");
        }
    }
}
```

TunerCapable.java

```
package KI303LEVCHENKOLAB3;

import java.io.IOException;

/**
 * Інтерфейс для телевізорів з тюнером, що дозволяє сканувати канали
 * та керувати улюбленими каналами.
 */
public interface TunerCapable {

    /**
     * Сканує доступні телевізійні канали.
     *
     * @throws IOException якщо виникає помилка під час сканування.
     * @throws InterruptedException якщо процес сканування перервано.
     */
    void scanChannels() throws IOException, InterruptedException;

    /**
     * Додає канал до списку улюблених.
     *
     * @param channel Номер каналу, який необхідно додати.
     * @throws IOException якщо виникає помилка під час додавання каналу.
     */
    void addFavoriteChannel(int channel) throws IOException;

    /**
     * Видаляє канал зі списку улюблених.
     *
     * @param channel Номер каналу, який необхідно видалити.
     * @throws IOException якщо виникає помилка під час видалення каналу.
     */
    void removeFavoriteChannel(int channel) throws IOException;
}
```

Television.java

```
package KI303LEVCHENKOLAB3;

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 abstract class Television {
    protected Screen screen;
    protected Speaker speaker;
    protected RemoteControl remoteControl;
    protected Logger logger;
    protected boolean isOn;
    protected int currentChannel;
    protected 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()));
    }

    /**
     * Вмикає телевізор.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
     */
    public abstract void turnOn() throws IOException;

    /**
     * Вимикає телевізор.
     *
     * @throws IOException якщо виникає помилка при роботі з логом.
```

```
     */
    public abstract void turnOff() throws IOException;

    /**
     * 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.
     *
     * @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

```java
package KI303LEVCHENKOLAB3;

/**
 * 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 KI303LEVCHENKOLAB3;

/**
 * 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

```
package KI303LEVCHENKOLAB3;

/**
 * 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;
    }
}
```

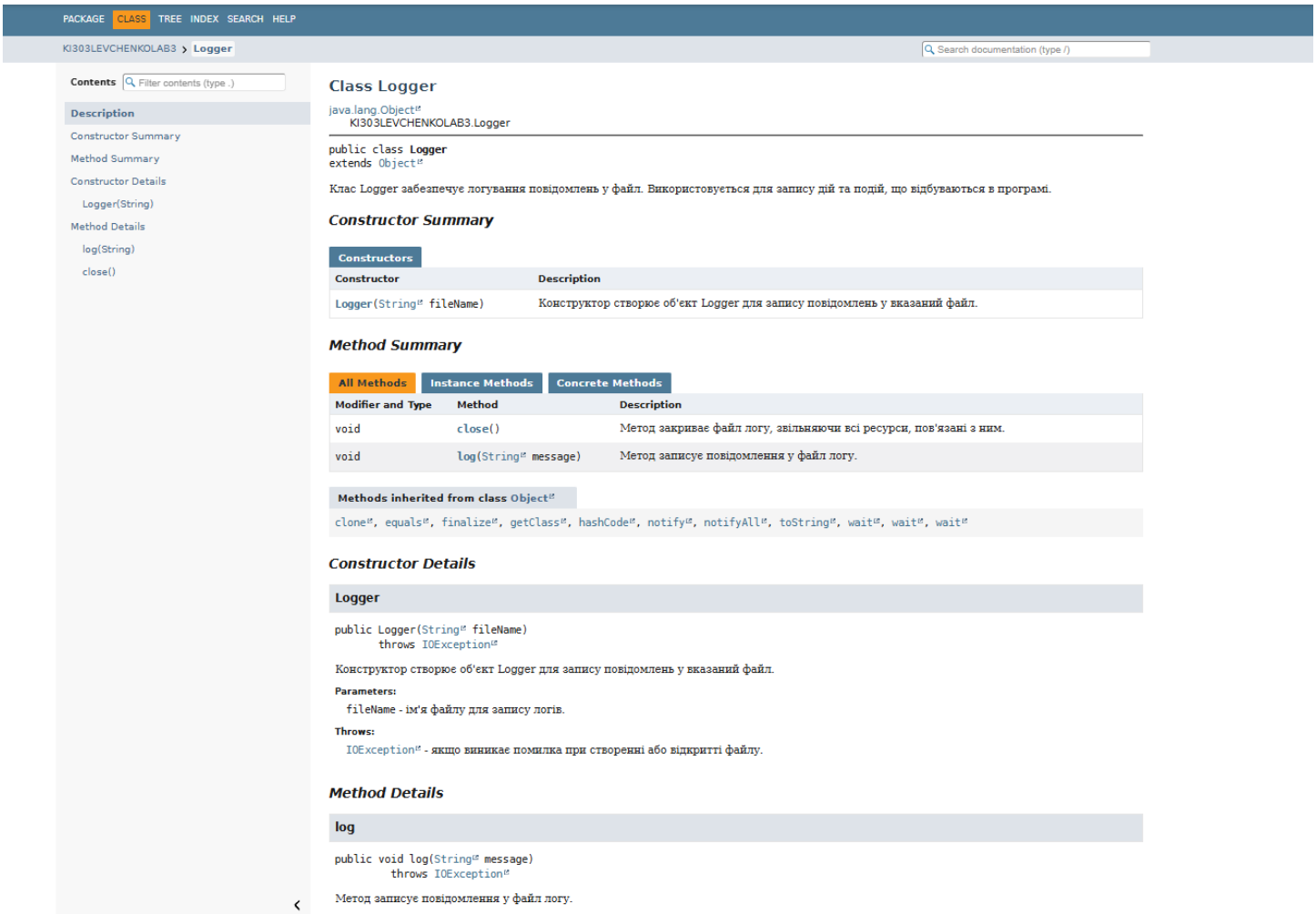
Logger.java

```java
package KI303LEVCHENKOLAB3;

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 з тюнером увімкнено
Сканування каналів...
Сканування завершено. Знайдено 50 каналів.
Перемкнуто на канал 1
Канал 5 додано до улюблених
Канал 10 додано до улюблених
Гучність збільшено до 1
Яскравість екрану встановлено на 75
Канал 5 видалено з улюблених
Перевірено стан: Телевізор увімкнено. Канал: 1. Гучність: 1
Телевізор з тюнером вимкнено
```

Рис 1.1 (Знімок результату з television_log.txt)

Фрагменти документації до коду



PACKAGE CLASS TREE INDEX SEARCH HELP

KI303LEVCHENKOLAB3 › Logger

Search documentation (type /)

Contents — Filter contents (type .)

- Description
- Constructor Summary
- Method Summary
- Constructor Details
  - Logger(String)
- Method Details
  - log(String)
  - close()

Class Logger

java.lang.Object
  KI303LEVCHENKOLAB3.Logger

public class **Logger**
extends Object

Клас Logger забезпечує логування повідомлень у файл. Використовується для запису дій та подій, що відбуваються в програмі.

Constructor Summary

Constructors

Constructor	Description
Logger(String fileName)	Конструктор створює об'єкт Logger для запису повідомлень у вказаний файл.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	close()	Метод закриває файл логу, звільняючи всі ресурси, пов'язані з ним.
void	log(String message)	Метод записує повідомлення у файл логу.

Methods inherited from class Object

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

Constructor Details

Logger

public Logger(String fileName)
       throws IOException

Конструктор створює об'єкт Logger для запису повідомлень у вказаний файл.

Parameters:

fileName - ім'я файлу для запису логів.

Throws:

IOException - якщо виникає помилка при створенні або відкритті файлу.

Method Details

log

public void log(String message)
         throws IOException

Метод записує повідомлення у файл логу.

Відповіді на контрольні питання

1. Синтаксис реалізації спадкування. *class Subclass extends Superclass { }*
2. Що таке суперклас та підклас? *Суперклас — батьківський клас; підклас — клас, який успадковує від суперкласу.*
3. Як звернутися до членів суперкласу з підкласу? *Використовують ключове слово super*
4. Коли використовується статичне зв'язування при виклику методу? *Використовується для статичних методів або методів, які не переозначені в підкласі (компіляція на етапі компіляції).*
5. Як відбувається динамічне зв'язування при виклику методу? *Відбувається під час виконання програми (runtime), коли викликаються методи через поліморфізм.*
6. Що таке абстрактний клас та як його реалізувати? *Клас, який не можна створювати як об'єкт і містить абстрактні методи без реалізації: abstract class ClassName { abstract void methodName(); }*
7. Для чого використовується ключове слово instanceof? *Використовується для перевірки, чи є об'єкт екземпляром певного класу або його підкласу: object instanceof ClassName.*
8. Як перевірити чи клас є підкласом іншого класу? *Використовується метод isAssignableFrom Superclass.class.isAssignableFrom(Subclass.class)*
9. Що таке інтерфейс? *Абстрактний тип, що містить тільки оголошення методів, без їх реалізації.*
10. Як оголосити та застосувати інтерфейс?
    Оголошення:
    interface InterfaceName { void methodName(); }
    Застосування:
    class ClassName implements InterfaceName { void methodName() { // реалізація } }

Висновок: Я ознайомився з спадкуванням та інтерфейсами у мові Java.