Міністерство освіти і науки України
Національний університет “Львівська політехніка”

Кафедра ЕОМ

ЗВІТ

до лабораторної роботи №3

З дисципліни: «Кросплатформні засоби програмування»

На тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

Варіант 25

Виконала:

ст. групи КІ-306

Тимків Н.В.

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

Львів – 2024

Мета: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Завдання:
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.

Тема згідно варіанту №25 – «Пристрій кліматконтролю»
GitHub Repository:

https://github.com/NazarTymkiiv/CPPT_Tymkiv_NV_KI-36_2.git

Хід роботи

Код програми:

ClimateControlDeviceDriver.java

```
package KI.Tymkiv.Lab3;

import java.io.IOException;

/**
 * Клас {@code ConditionerDriver} демонструє використання класу {@code
Conditioner}.
 */
public class ClimateControlDeviceDriver {
    /**
     * Головний метод.
     *
     * @param args аргументи командного рядка (не використовуються).
     */
    public static void main(String[] args) {
        try {
            ClimateControlDevice device = new ClimateControlDevice();

            device.turnOn();
            device.changeTemperature(22);
            device.switchMode("Cool");
            device.increaseFanSpeed();
            device.enableEnergySavingMode();
            device.autoAdjustTemperature(30);
            device.performSpecificMaintenance();
            device.getStatus();
            device.disableEnergySavingMode();
            device.turnOff();

            device.closeLogger();
        } catch (IOException e) {
```

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

ClimateControlDevice.java

```
package KI.Tymkiv.Lab3;

import java.io.IOException;

/**
 * Клас {@code ClimateControlDevice} представляє пристрій для кліматконтролю.
 * Він розширює функціональність базового кондиціонера та реалізує інтерфейс
{@link EnergySaving}.
 * Підтримує режими енергозбереження та автоматичне регулювання температури.
 */
public class ClimateControlDevice extends Conditioner implements EnergySaving {
    private boolean energySavingMode;

    /**
     * Конструктор для створення пристрою кліматконтролю.
     *
     * @throws IOException якщо виникає помилка при створенні або записі у файл
логів
     */
    public ClimateControlDevice() throws IOException {
        super();
        this.energySavingMode = false;
    }

    /**
     * Вмикає режим енергозбереження.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    @Override
    public void enableEnergySavingMode() throws IOException {
        this.energySavingMode = true;
        logger.log("Режим енергозбереження увімкнено.");
        System.out.println("Режим енергозбереження увімкнено.");
    }

    /**
     * Вимикає режим енергозбереження.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    @Override
    public void disableEnergySavingMode() throws IOException {
        this.energySavingMode = false;
        logger.log("Режим енергозбереження вимкнено.");
        System.out.println("Режим енергозбереження вимкнено.");
    }

    /**
     * Перевіряє, чи увімкнено режим енергозбереження.
     *
     * @return {@code true}, якщо режим енергозбереження увімкнено, інакше
{@code false}
     */
    @Override
    public boolean isEnergySavingModeEnabled() {
```

```
        return this.energySavingMode;
    }

    /**
     * Виконує специфічне обслуговування пристрою кліматконтролю.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    @Override
    public void performSpecificMaintenance() throws IOException {
        logger.log("Виконується специфічне обслуговування пристрою
кліматконтролю.");
        System.out.println("Виконується специфічне обслуговування пристрою
кліматконтролю.");
    }

    /**
     * Автоматично регулює температуру на основі зовнішніх умов.
     *
     * @param outdoorTemperature зовнішня температура
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void autoAdjustTemperature(double outdoorTemperature) throws
IOException {
        if (isOn) {
            double targetTemperature;
            if (energySavingMode) {
                targetTemperature = outdoorTemperature + 2; // Менша різниця для
економії енергії
            } else {
                targetTemperature = outdoorTemperature - 2; // Більша різниця
для комфорту
            }
            changeTemperature(targetTemperature);
            logger.log(String.format("Автоматично встановлено температуру на %s
градусів.", targetTemperature));
            System.out.printf("Автоматично встановлено температуру на %s
градусів.\n", targetTemperature);
        } else {
            logger.log("Неможливо автоматично регулювати температуру. Пристрій
вимкнено.");
            System.out.println("Неможливо автоматично регулювати температуру.
Пристрій вимкнено.");
        }
    }
}
```

EnergySaving.java

```
package KI.Tymkiv.Lab3;

import java.io.IOException;

/**
 * Інтерфейс {@code EnergySaving} визначає функціональність для пристроїв, що
підтримують режим енергозбереження.
 * Він описує методи для ввімкнення, вимкнення та перевірки стану режиму
енергозбереження.
 */
public interface EnergySaving {

    /**
     * Вмикає режим енергозбереження.
     *
     * @throws IOException якщо виникають помилки при логуванні
```

```
     */
    void enableEnergySavingMode() throws IOException;

    /**
     * Вимикає режим енергозбереження.
     *
     * @throws IOException якщо виникають помилки при логуванні
     */
    void disableEnergySavingMode() throws IOException;

    /**
     * Перевіряє, чи увімкнено режим енергозбереження.
     *
     * @return {@code true}, якщо режим енергозбереження увімкнено, інакше
{@code false}
     * @throws IOException якщо виникають помилки при логуванні
     */
    boolean isEnergySavingModeEnabled() throws IOException;
}
```

Conditioner.java

```
package KI.Tymkiv.Lab3;

import java.io.IOException;

/**
 * Клас для моделювання роботи кондиціонера.
 * Він включає в себе такі компоненти, як компресор, фільтр, термостат, і
дозволяє керувати станом кондиціонера.
 * Також ведеться логування дій кондиціонера.
 */
public abstract class Conditioner {
    protected Compressor compressor;
    protected Filter filter;
    protected Thermostat thermostat;
    protected Logger logger;
    protected boolean isOn;
    protected String mode;
    protected int fanSpeed;

    /**
     * Конструктор за замовчуванням. Створює новий об'єкт кондиціонера з
початковими налаштуваннями.
     * Логування початкового стану записується у файл.
     *
     * @throws IOException якщо виникає помилка при створенні або запису у файл
логів
     */
    public Conditioner() throws IOException {
        this.compressor = new Compressor();
        this.filter = new Filter();
        this.thermostat = new Thermostat();
        this.isOn = false;
        this.mode = "Cool";
        this.fanSpeed = 1;

        this.logger = new Logger("conditioner_log.txt");
        logger.log(String.format("Кондиціонер %s створено.", this.toString()));
    }

    /**
     * Конструктор з параметрами. Дозволяє створити кондиціонер з заданими
компонентами.
     * Логування початкового стану записується у файл.
```

```
     *
     * @param compressor об'єкт компресора
     * @param filter об'єкт фільтра
     * @param thermostat об'єкт термостата
     * @throws IOException якщо виникає помилка при створенні або запису у файл
логів
     */
    public Conditioner(Compressor compressor, Filter filter, Thermostat
thermostat) throws IOException {
        this.compressor = compressor;
        this.filter = filter;
        this.thermostat = thermostat;
        this.isOn = false;
        this.mode = "Cool";
        this.fanSpeed = 1;

        this.logger = new Logger("conditioner_log.txt");
        logger.log(String.format("Кондиціонер %s створено.", this.toString()));
    }


    /**
     * Абстрактний метод для виконання специфічного обслуговування.
     * Повинен бути реалізований у підкласах.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public abstract void performSpecificMaintenance() throws IOException;

    /**
     * Увімкнення кондиціонера. Запускає компресор, змінює стан кондиціонера на
"увімкнено" та записує це у лог.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void turnOn() throws IOException {
        if (!isOn) {
            compressor.start();
            isOn = true;
            logger.log("Кондиціонер увімкнено.");
            System.out.println("Кондиціонер увімкнено.");
        } else {
            logger.log("Кондиціонер вже увімкнено.");
            System.out.println("Кондиціонер вже увімкнено.");
        }
    }

    /**
     * Вимкнення кондиціонера. Зупиняє компресор, змінює стан кондиціонера на
"вимкнено" та записує це у лог.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void turnOff() throws IOException {
        if (isOn) {
            compressor.stop();
            isOn = false;
            logger.log("Кондиціонер вимкнено.");
            System.out.println("Кондиціонер вимкнено.");
        } else {
            logger.log("Кондиціонер вже вимкнено.");
            System.out.println("Кондиціонер вже вимкнено.");
        }
    }

    /**
```

```
     * Зміна температури кондиціонера. Змінює налаштування термостата, якщо
кондиціонер увімкнено.
     *
     * @param temperature нова температура
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void changeTemperature(double temperature) throws IOException {
        if (isOn) {
            thermostat.setTemperature(temperature);
            logger.log(String.format("Температуру встановлено на %s градусів.",
temperature));
            System.out.printf("Температуру встановлено на %s градусів.\n",
temperature);
        } else {
            logger.log("Неможливо змінити температуру. Кондиціонер вимкнено.");
            System.out.println("Неможливо змінити температуру. Кондиціонер
вимкнено.");
        }
    }

    /**
     * Перемикає режим роботи кондиціонера.
     *
     * @param newMode новий режим роботи (наприклад, "Cool", "Heat")
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void switchMode(String newMode) throws IOException {
        if (isOn) {
            this.mode = newMode;
            logger.log(String.format("Режим змінено на %s.", newMode));
            System.out.printf("Режим змінено на %s.\n", newMode);
        } else {
            logger.log("Неможливо змінити режим. Кондиціонер вимкнено.");
            System.out.println("Неможливо змінити режим. Кондиціонер
вимкнено.");
        }
    }

    /**
     * Збільшує швидкість вентилятора на 1, якщо вона ще не досягла максимуму.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void increaseFanSpeed() throws IOException {
        if (isOn) {
            if (fanSpeed < 3) {
                fanSpeed++;
                logger.log(String.format("Швидкість вентилятора збільшено до
%d.", fanSpeed));
                System.out.printf("Швидкість вентилятора збільшено до %d.\n",
fanSpeed);
            } else {
                logger.log("Досягнуто максимальну швидкість вентилятора.");
                System.out.println("Досягнуто максимальну швидкість
вентилятора.");
            }
        } else {
            logger.log("Неможливо змінити швидкість вентилятора. Кондиціонер
вимкнено.");
            System.out.println("Неможливо змінити швидкість вентилятора.
Кондиціонер вимкнено.");
        }
    }

    /**
     * Зменшує швидкість вентилятора на 1, якщо вона ще не досягла мінімуму.
```

```
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void decreaseFanSpeed() throws IOException {
        if (isOn) {
            if (fanSpeed > 1) {
                fanSpeed--;
                logger.log(String.format("Швидкість вентилятора зменшено до
%d.", fanSpeed));
                System.out.printf("Швидкість вентилятора зменшено до %d.\n",
fanSpeed);
            } else {
                logger.log("Досягнуто мінімальну швидкість вентилятора.");
                System.out.println("Досягнуто мінімальну швидкість
вентилятора.");
            }
        } else {
            logger.log("Неможливо змінити швидкість вентилятора. Кондиціонер
вимкнено.");
            System.out.println("Неможливо змінити швидкість вентилятора.
Кондиціонер вимкнено.");
        }
    }

    /**
     * Очищує фільтр кондиціонера та веде запис у лог.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void cleanFilter() throws IOException {
        filter.clean();
        logger.log("Фільтр очищено.");
        System.out.println("Фільтр очищено.");
    }

    /**
     * Перевіряє стан фільтра (чистий або забруднений).
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void checkFilterStatus() throws IOException {
        boolean isClean = filter.isClean();
        logger.log(String.format("Стан фільтра: %s", isClean ? "чистий" :
"забруднений"));
        System.out.printf("Стан фільтра: %s\n", isClean ? "чистий" :
"забруднений");
    }

    /**
     * Виконує технічне обслуговування кондиціонера, включаючи очищення фільтра
та перезапуск компресора.
     *
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void performMaintenance() throws IOException {
        cleanFilter();
        compressor.stop();
        compressor.start();
        logger.log("Виконано технічне обслуговування кондиціонера.");
        System.out.println("Виконано технічне обслуговування кондиціонера.");
    }

    /**
     * Виводить поточний статус кондиціонера, включаючи стан, режим, температуру
та швидкість вентилятора.
     *
```

```
     * @throws IOException якщо виникає помилка при записі у файл логів
     */
    public void getStatus() throws IOException {
        logger.log(String.format("Статус кондиціонера: %s, Режим: %s,
Температура: %s, Швидкість вентилятора: %d",
                isOn ? "увімкнено" : "вимкнено", mode,
thermostat.getTemperature(), fanSpeed));
        System.out.printf("Статус кондиціонера: %s, Режим: %s, Температура: %s,
Швидкість вентилятора: %d\n",
                isOn ? "увімкнено" : "вимкнено", mode,
thermostat.getTemperature(), fanSpeed);
    }

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

Compressor.java

```
package KI.Tymkiv.Lab3;

/**
 * Клас, що моделює роботу компресора кондиціонера.
 * Компресор може бути увімкнений або вимкнений.
 */
public class Compressor {
    private boolean running;

    /**
     * Конструктор за замовчуванням. Створює компресор у вимкненому стані.
     */
    public Compressor() {
        running = false;
    }

    /**
     * Увімкнення компресора. Змінює стан компресора на "увімкнено".
     */
    public void start() {
        running = true;
    }

    /**
     * Вимкнення компресора. Змінює стан компресора на "вимкнено".
     */
    public void stop() {
        running = false;
    }

    /**
     * Перевіряє, чи працює компресор.
     *
     * @return {@code true}, якщо компресор увімкнено, і {@code false}, якщо
вимкнено.
     */
    public boolean isRunning() {
        return running;
    }
}
```

Filter.java

```java
package KI.Tymkiv.Lab3;

/**
 * Клас Filter моделює фільтр кондиціонера.
 * Фільтр може бути чистим або забрудненим.
 */
public class Filter {
    private boolean clean;

    /**
     * Конструктор з параметром, який дозволяє встановити стан фільтра (чистий або забруднений).
     *
     * @param clean стан фільтра: {@code true} для чистого фільтра, {@code false} для забрудненого
     */
    public Filter(boolean clean) {
        this.clean = clean;
    }

    /**
     * Конструктор за замовчуванням, який створює чистий фільтр.
     */
    public Filter() {
        this.clean = true;
    }

    /**
     * Перевіряє, чи є фільтр чистим.
     *
     * @return {@code true}, якщо фільтр чистий, і {@code false}, якщо він забруднений
     */
    public boolean isClean() {
        return clean;
    }

    /**
     * Очищає фільтр, встановлюючи його стан як чистий.
     */
    public void clean() {
        clean = true;
    }

    /**
     * Забруднює фільтр, встановлюючи його стан як забруднений.
     */
    public void soil() {
        clean = false;
    }
}
```

Thermostat.java

```java
package KI.Tymkiv.Lab3;

/**
 * Клас {@code Thermostat} представляє термостат, який зберігає та управляє
```

```
температурою.
 */
public class Thermostat {
    private double temperature;

    /**
     * Створює новий термостат з заданою температурою.
     *
     * @param temperature температура термостата
     */
    public Thermostat(double temperature) {
        this.temperature = temperature;
    }

    /**
     * Створює новий термостат з початковою температурою 0 градусів.
     */
    public Thermostat() {
        this.temperature = 0;
    }

    /**
     * Повертає поточну температуру термостата.
     *
     * @return температура термостата
     */
    public double getTemperature() {
        return temperature;
    }

    /**
     * Встановлює нову температуру термостата.
     *
     * @param temperature нова температура термостата
     */
    public void setTemperature(double temperature) {
        this.temperature = temperature;
    }
}
```

Logger.java

```
package KI.Tymkiv.Lab3;

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);
    }
```

```java
    /**
     * 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());
            }
        }
    }
}
```

```
Кондиціонер увімкнено.
Температуру встановлено на 22.0 градусів.
Режим змінено на Cool.
Швидкість вентилятора збільшено до 2.
Режим енергозбереження увімкнено.
Температуру встановлено на 32.0 градусів.
Автоматично встановлено температуру на 32.0 градусів.
Виконується специфічне обслуговування пристрою кліматконтролю.
Статус кондиціонера: увімкнено, Режим: Cool, Температура: 32.0, Швидкість вентилятора: 2
Режим енергозбереження вимкнено.
Кондиціонер вимкнено.

Process finished with exit code 0
```

Рис.1 Вивід логу у консоль

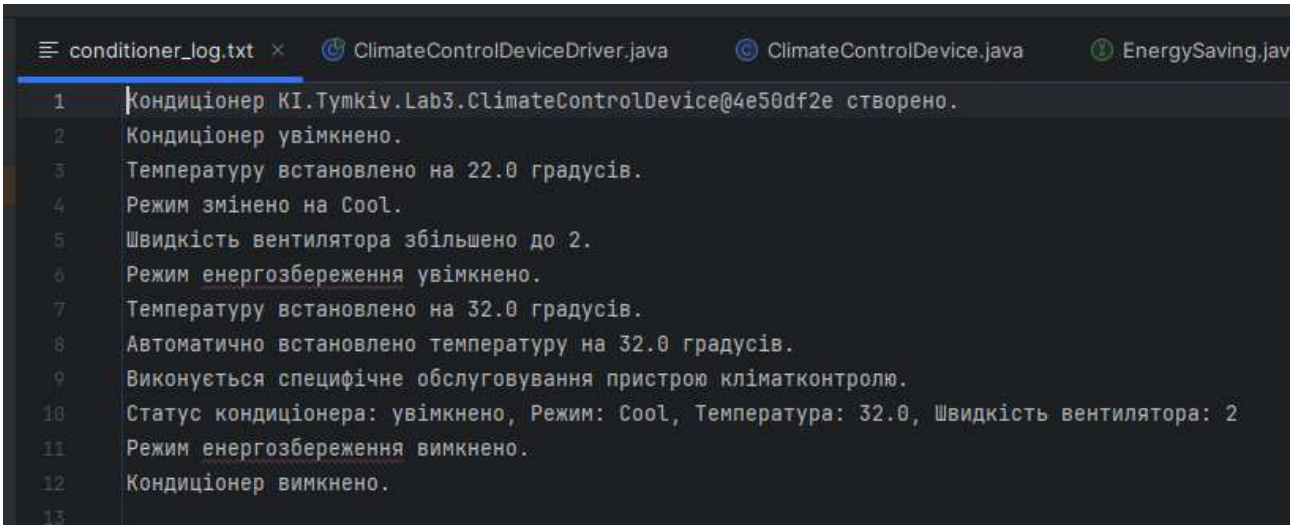

Рис.2 Вивід логу у текстовий файл

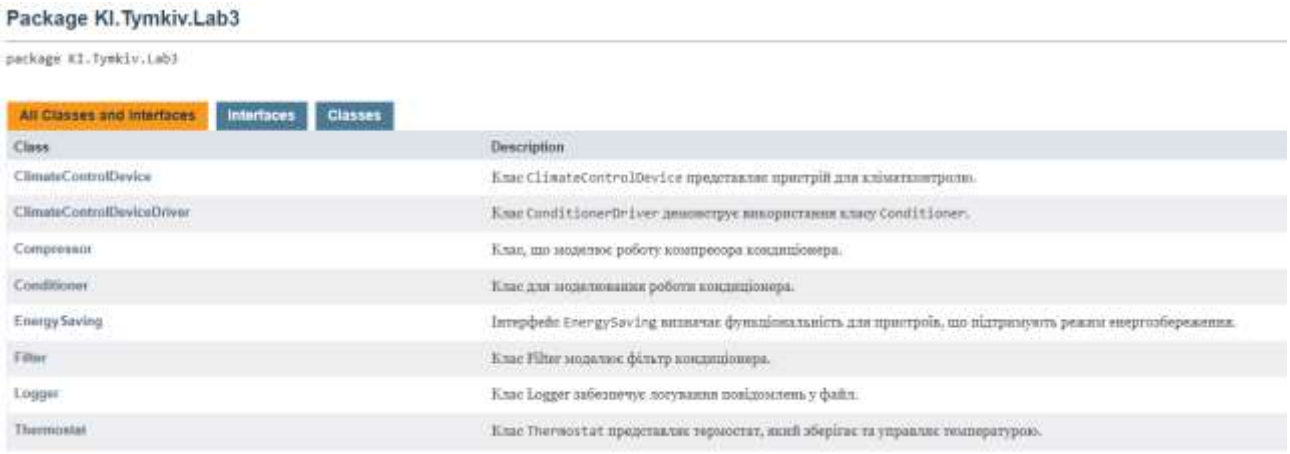

Рис.3.1 Фрагмент згенерованої документації

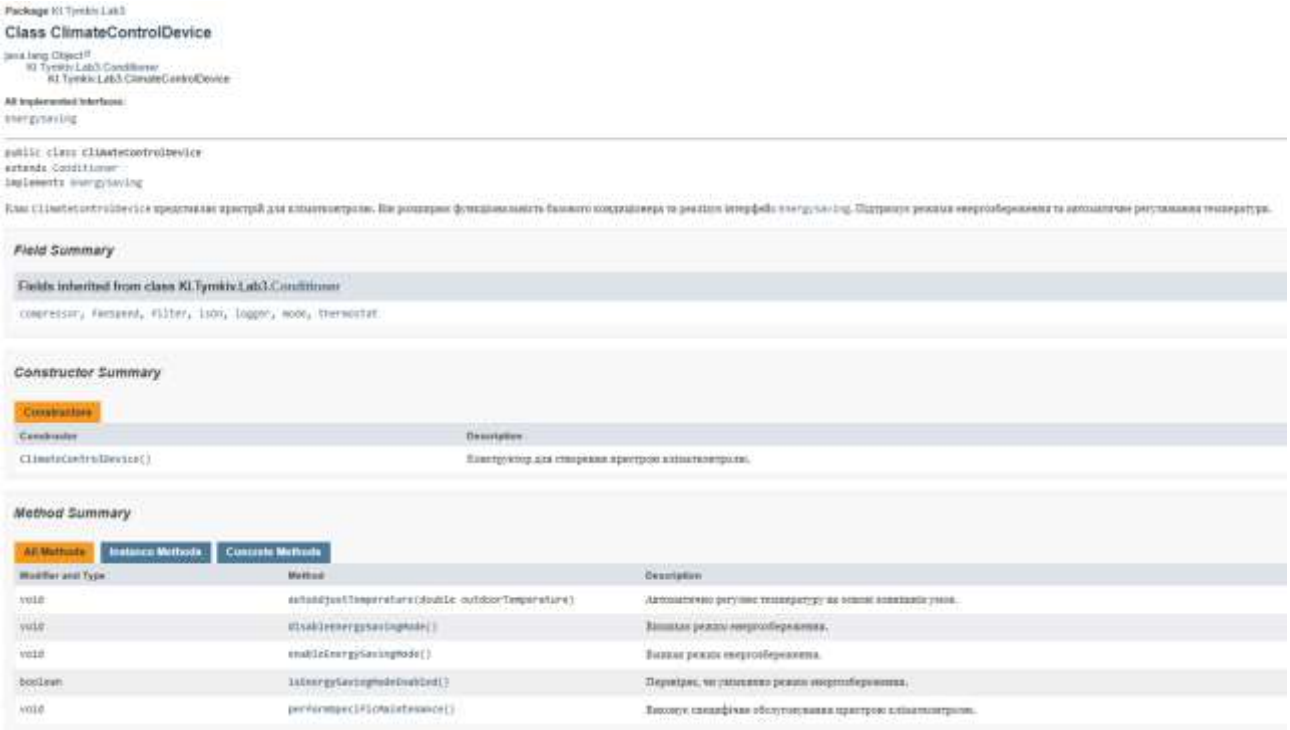

Рис.3.2 Фрагмент згенерованої документації

Висновок: На лабораторній роботі я ознайомився зі спадкуванням та інтерфейсами у мові Java.