Міністерство освіти і науки України
Національний університет “Львівська політехніка”

Кафедра ЕОМ

ЗВІТ

до лабораторної роботи №2

З дисципліни: «Кросплатформні засоби програмування»

На тему: «КЛАСИ ТА ПАКЕТИ»

Варіант 25

Виконав:

ст. групи КІ-306

Тимків Н.В.

Прийняв:

доцент кафедри ЕОМ

Олексів М.В.

Львів – 2024

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

Завдання:
1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область 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;
- 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.

Тема згідно варіанту №25 – «Кондиціонер»
GitHub Repository:

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

Хід роботи

Код програми:

ConditionerDriver.java

```
package KI.Tymkiv.Lab2;

import java.io.IOException;


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

```
            conditioner.turnOn();
            conditioner.changeTemperature(23);
            conditioner.switchMode("Heat");
            conditioner.increaseFanSpeed();
            conditioner.decreaseFanSpeed();
            conditioner.cleanFilter();
            conditioner.checkFilterStatus();
            conditioner.performMaintenance();
            conditioner.getStatus();
            conditioner.turnOff();

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

Conditioner.java

```
package KI.Tymkiv.Lab2;

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 class Conditioner {
    private Compressor compressor;
    private Filter filter;
    private Thermostat thermostat;
    private Logger logger;
    private boolean isOn;
    private String mode;
    private 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()));
    }

    /**
     * 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.Lab2;

/**
 * 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

```
package KI.Tymkiv.Lab2;

/**
```

```
 * 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

```
package KI.Tymkiv.Lab2;

/**
 * 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.Lab2;

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());
            }
        }
    }
}
```

```
Кондиціонер увімкнено.
Температуру встановлено на 23.0 градусів.
Режим змінено на Heat.
Швидкість вентилятора збільшено до 2.
Швидкість вентилятора зменшено до 1.
Фільтр очищено.
Стан фільтра: чистий
Фільтр очищено.
Виконано технічне обслуговування кондиціонера.
Статус кондиціонера: увімкнено, Режим: Heat, Температура: 23.0, Швидкість вентилятора: 1
Кондиціонер вимкнено.

Process finished with exit code 0
```

Рис.1 Вивід логу у консоль

conditioner_log.txt | ConditionerDriver.java | Conditioner.java | Compressor.java | Filter.java

```
1  Кондиціонер KI.Tymkiv.Lab2.Conditioner@b4c966a створено.
2  Кондиціонер увімкнено.
3  Температуру встановлено на 23.0 градусів.
4  Режим змінено на Heat.
5  Швидкість вентилятора збільшено до 2.
6  Швидкість вентилятора зменшено до 1.
7  Фільтр очищено.
8  Стан фільтра: чистий
9  Фільтр очищено.
10 Виконано технічне обслуговування кондиціонера.
11 Статус кондиціонера: увімкнено, Режим: Heat, Температура: 23.0, Швидкість вентилятора: 1
12 Кондиціонер вимкнено.
13
```

Рис.2 Вивід логу у текстовий файл

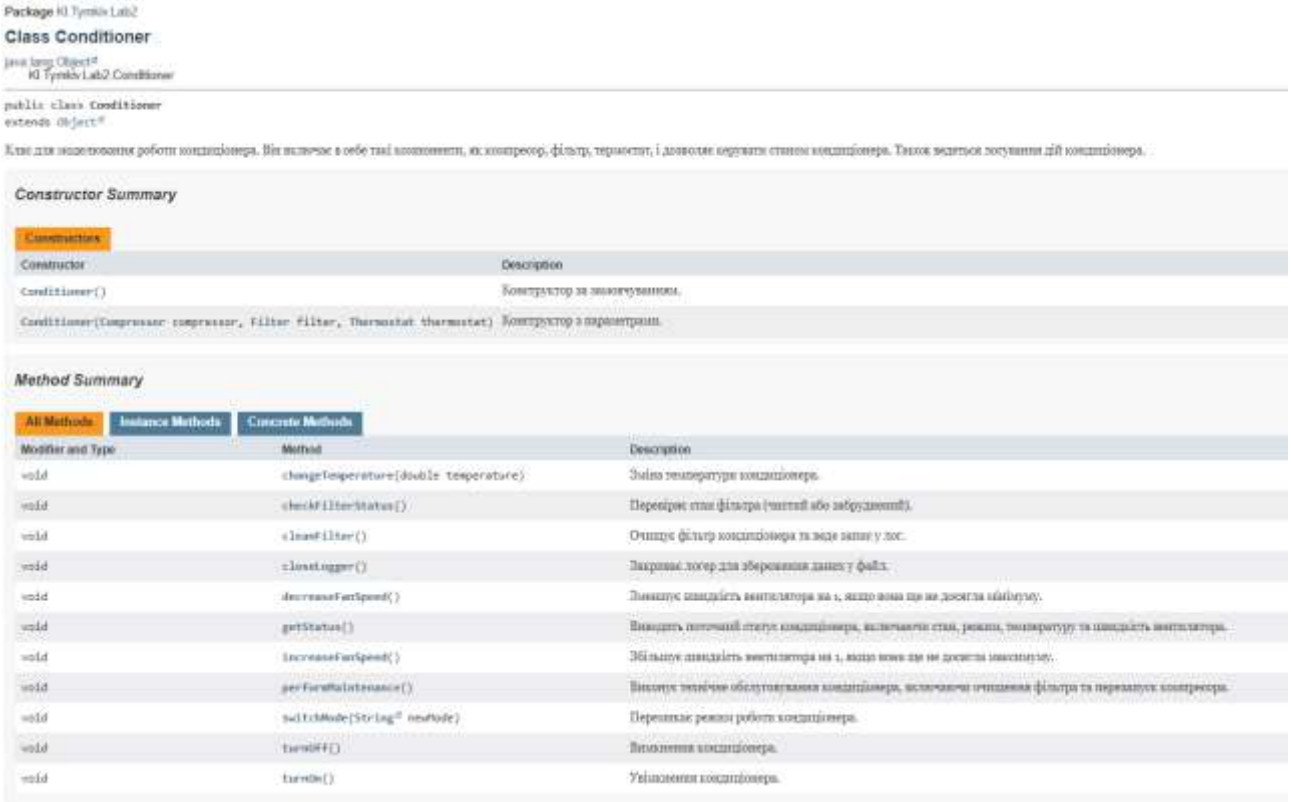
Package KI.Tymkiv.Lab2

package KI.Tymkiv.Lab2

Classes

Class	Description
Compressor	Клас, що моделює роботу компресора кондиціонера.
Conditioner	Клас для моделювання роботи кондиціонера.
ConditionerDriver	Клас `ConditionerDriver` демонструє використання класу `Conditioner`.
Filter	Клас Filter моделює фільтр кондиціонера.
Logger	Клас Logger забезпечує логування повідомлень у файл.
Thermostat	Клас `Thermostat` представляє термостат, який зберігає та управляє температурою.

Рис.3.1 Фрагмент згенерованої документації



Package KI.Tymkiv.Lab2

Class Conditioner

java.lang.Object
KI.Tymkiv.Lab2.Conditioner

public class Conditioner
extends Object

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.

Constructor Summary

Constructors

Constructor	Description
`Conditioner()`	Конструктор за замовчуванням.
`Conditioner(Compressor compressor, Filter filter, Thermostat thermostat)`	Конструктор з параметрами.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	`changeTemperature(double temperature)`	Зміна температури кондиціонера.
void	`checkFilterStatus()`	Перевіряє стан фільтра (чистий або забруднений).
void	`cleanFilter()`	Очищує фільтр кондиціонера та веде запис у лог.
void	`closeLogger()`	Закриває логер для збереження даних у файл.
void	`decreaseFanSpeed()`	Зменшує швидкість вентилятора на 1, якщо вона ще не досягла мінімуму.
void	`getStatus()`	Виводить поточний статус кондиціонера, включаючи стан, режим, температуру та швидкість вентилятора.
void	`increaseFanSpeed()`	Збільшує швидкість вентилятора на 1, якщо вона ще не досягла максимуму.
void	`performMaintenance()`	Виконує технічне обслуговування кондиціонера, включаючи очищення фільтра та перезапуск компресора.
void	`switchMode(String newMode)`	Перемикає режим роботи кондиціонера.
void	`turnOff()`	Вимкнення кондиціонера.
void	`turnOn()`	Увімкнення кондиціонера.

Рис.3.2 Фрагмент згенерованої документації

Висновок: На лабораторній роботі я ознайомився з базовими конструкціями мови Java та оволодів навиками написання й автоматичного документування простих консольних програм мовою Java.