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 16

Виконав:

ст. групи КІ-306

Мілян М.О.

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 №16 – «Аудіоплеєр»

Хід роботи

Код програми:

AudioPlayerDriver.java

```
package KI.Milian.Lab2;

/**
 * Клас-драйвер для тестування аудіоплеєра.
 */
public class AudioPlayerDriver {
    public static void main(String[] args) {
        try {
            // Створюємо об'єкти для плеєра
            Speaker speaker = new Speaker(70);
            BluetoothModule bluetoothModule = new BluetoothModule();
            Battery battery = new Battery(4000);

            // Створюємо аудіоплеєр
            AudioPlayer player = new AudioPlayer(speaker, bluetoothModule,
battery);

            // Тестуємо функціонал з виведенням у консоль
            player.connectBluetooth("Bluetooth Speaker");
            player.playTrack("Song 1");
            player.increaseVolume();
            player.chargeBattery(500); // Заряджаємо батарею
```

```
            player.disconnectBluetooth();
            player.playTrack("Song 2");

            // Закриваємо ресурси
            player.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

AudioPlayer.java

```
package KI.Milian.Lab2;

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

/**
 * Клас, що описує аудіоплеєр.
 */
public class AudioPlayer {
    private Speaker speaker;
    private BluetoothModule bluetoothModule;
    private Battery battery;
    private FileWriter logWriter;

    public AudioPlayer(Speaker speaker, BluetoothModule bluetoothModule, Battery
battery) throws IOException {
        this.speaker = speaker;
        this.bluetoothModule = bluetoothModule;
        this.battery = battery;
        this.logWriter = new FileWriter("audio_player_log.txt", true);
    }

    public AudioPlayer() throws IOException {
        this(new Speaker(50), new BluetoothModule(), new Battery(3000));
    }

    /**
     * Відтворює трек і зменшує заряд батареї.
     */
    public void playTrack(String track) throws IOException {
        if (battery.getCapacity() <= 0) {
            log("Battery is empty. Cannot play track.");
            System.out.println("Battery is empty. Please charge the device.");
            return;
        }

        if (bluetoothModule.isConnected()) {
            log("Playing track: " + track + " on device: " +
bluetoothModule.getConnectedDevice());
            System.out.println("Playing track: " + track + " on Bluetooth
device: " + bluetoothModule.getConnectedDevice());
        } else {
            log("Playing track: " + track + " on built-in speaker.");
            System.out.println("Playing track: " + track + " on built-in
speaker.");
        }
        battery.drainBattery(50); // Витрачаємо 50 мАг на кожен трек
        log("Battery capacity after playing track: " + battery.getCapacity() +
"mAh");
        System.out.println("Battery capacity after playing track: " +
battery.getCapacity() + "mAh");
```

```
    }

    /**
     * Підключаємося до Bluetooth-пристрою і зменшуємо заряд батареї.
     */
    public void connectBluetooth(String device) throws IOException {
        if (battery.getCapacity() <= 0) {
            log("Battery is empty. Cannot connect to Bluetooth.");
            System.out.println("Battery is empty. Please charge the device.");
            return;
        }

        bluetoothModule.connectToDevice(device);
        log("Connected to Bluetooth device: " + device);
        System.out.println("Connected to Bluetooth device: " + device);
        battery.drainBattery(30); // Витрачаємо 30 мАг на підключення до
Bluetooth
        log("Battery capacity after connecting to Bluetooth: " +
battery.getCapacity() + "mAh");
        System.out.println("Battery capacity after connecting to Bluetooth: " +
battery.getCapacity() + "mAh");
    }

    public void disconnectBluetooth() throws IOException {
        bluetoothModule.disconnect();
        log("Disconnected from Bluetooth device.");
        System.out.println("Disconnected from Bluetooth device.");
    }

    public void increaseVolume() throws IOException {
        speaker.setVolume(speaker.getVolume() + 10);
        log("Increased volume to: " + speaker.getVolume());
        System.out.println("Increased volume to: " + speaker.getVolume());
    }

    public void decreaseVolume() throws IOException {
        speaker.setVolume(speaker.getVolume() - 10);
        log("Decreased volume to: " + speaker.getVolume());
        System.out.println("Decreased volume to: " + speaker.getVolume());
    }

    public void chargeBattery(int amount) throws IOException {
        battery.setCapacity(battery.getCapacity() + amount);
        log("Charged battery by " + amount + "mAh. New capacity: " +
battery.getCapacity() + "mAh");
        System.out.println("Charged battery by " + amount + "mAh. New capacity:
" + battery.getCapacity() + "mAh");
    }

    private void log(String message) throws IOException {
        logWriter.write(message + "\n");
    }

    public void close() throws IOException {
        if (logWriter != null) {
            logWriter.close();
        }
    }

    @Override
    public String toString() {
        return "AudioPlayer{speaker=" + speaker + ", bluetoothModule=" +
bluetoothModule + ", battery=" + battery + "}";
    }
}
```

Battery.java

```java
package KI.Milian.Lab2;

/**
 * Клас, що описує акумулятор для аудіоплеєра.
 */
public class Battery {
    private int capacity;  // Ємність батареї в мАг (мілліампер-години)

    public Battery(int capacity) {
        this.capacity = capacity;
    }

    public int getCapacity() {
        return capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    /**
     * Зменшує заряд батареї на вказану кількість мАг.
     * Якщо заряд стає меншим за 0, встановлюється на 0.
     */
    public void drainBattery(int amount) {
        capacity -= amount;
        if (capacity < 0) {
            capacity = 0;
        }
    }

    @Override
    public String toString() {
        return "Battery{capacity=" + capacity + "mAh}";
    }
}
```

Speaker.java

```java
package KI.Milian.Lab2;

/**
 * Клас, що описує динамік аудіоплеєра.
 */
public class Speaker {
    private int volume; // Гучність динаміка

    public Speaker(int volume) {
        this.volume = volume;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }

    @Override
    public String toString() {
        return "Speaker{volume=" + volume + "}";
```

```
    }
}
```

BluetoothModule.java

```
package KI.Milian.Lab2;

/**
 * Клас, що описує Bluetooth модуль для аудіоплеєра.
 */
public class BluetoothModule {
    private boolean isConnected; // Стан підключення
    private String connectedDevice; // Підключений пристрій

    public BluetoothModule() {
        this.isConnected = false;
        this.connectedDevice = "None";
    }

    public void connectToDevice(String device) {
        this.isConnected = true;
        this.connectedDevice = device;
    }

    public void disconnect() {
        this.isConnected = false;
        this.connectedDevice = "None";
    }

    public boolean isConnected() {
        return isConnected;
    }

    public String getConnectedDevice() {
        return connectedDevice;
    }

    @Override
    public String toString() {
        return "BluetoothModule{isConnected=" + isConnected + ",
connectedDevice='" + connectedDevice + "'}";
    }
}
```

```
Connected to Bluetooth device: Bluetooth Speaker
Battery capacity after connecting to Bluetooth: 3970mAh
Playing track: Song 1 on Bluetooth device: Bluetooth Speaker
Battery capacity after playing track: 3920mAh
Increased volume to: 80
Charged battery by 500mAh. New capacity: 4420mAh
Disconnected from Bluetooth device.
Playing track: Song 2 on built-in speaker.
Battery capacity after playing track: 4370mAh

Process finished with exit code 0
```

Рис.1 Вивід логу у консоль

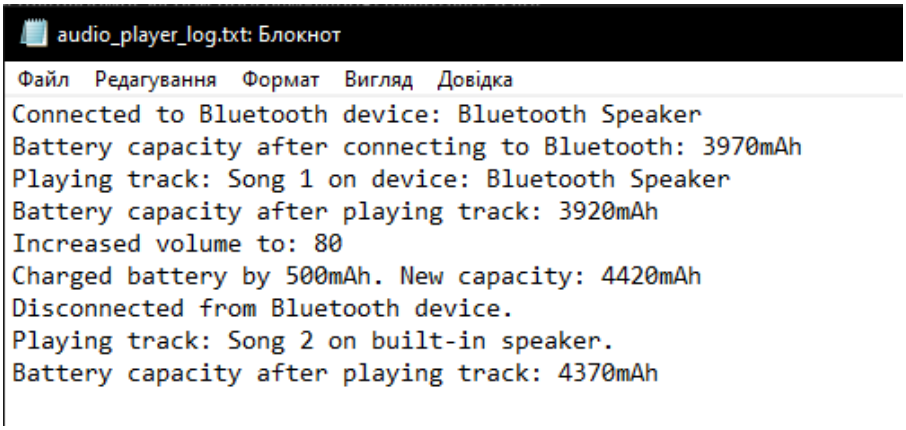


Рис.2 Вивід логу у текстовий файл

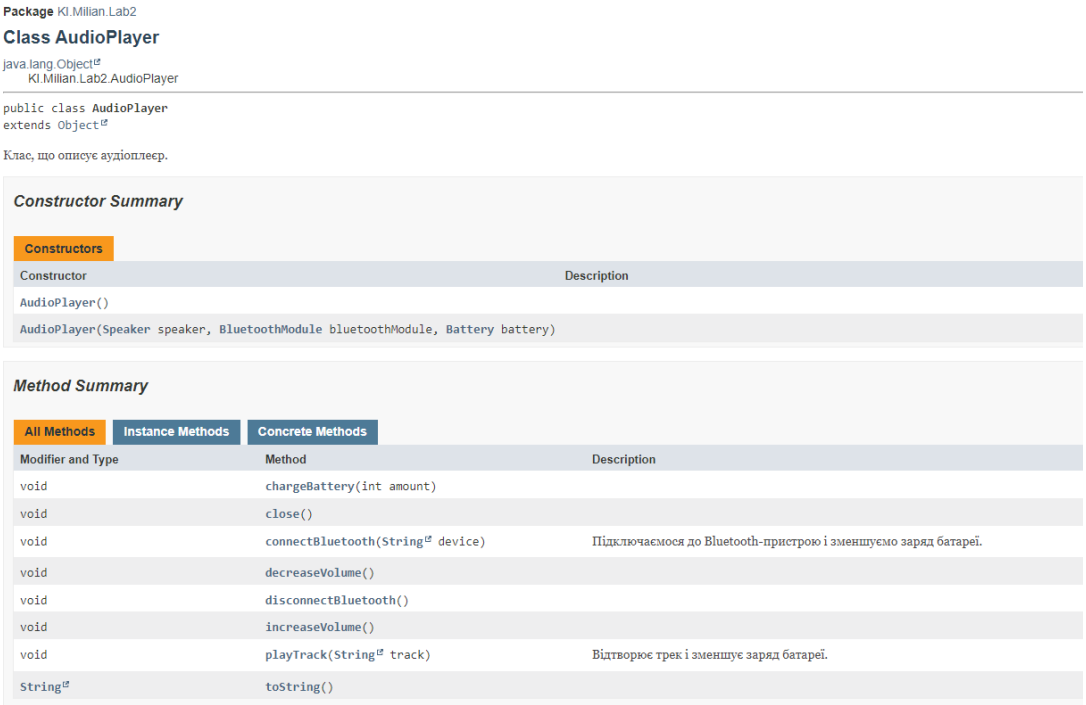
Package KI.Milian.Lab2

package KI.Milian.Lab2

Classes

Class	Description
AudioPlayer	Клас, що описує аудіоплеєр.
AudioPlayerDriver	Клас-драйвер для тестування аудіоплеєра.
Battery	Клас, що описує акумулятор для аудіоплеєра.
BluetoothModule	Клас, що описує Bluetooth модуль для аудіоплеєра.
Speaker	Клас, що описує динамік аудіоплеєра.

Рис.3.1 Фрагмент згенерованої документації



Package KI.Milian.Lab2

Class AudioPlayer

java.lang.Object
KI.Milian.Lab2.AudioPlayer

public class **AudioPlayer**
extends Object

Клас, що описує аудіоплеєр.

Constructor Summary

Constructors

Constructor	Description
AudioPlayer()	
AudioPlayer(Speaker speaker, BluetoothModule bluetoothModule, Battery battery)	

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
void	chargeBattery(int amount)	
void	close()	
void	connectBluetooth(String device)	Підключаємося до Bluetooth-пристрою і зменшуємо заряд батареї.
void	decreaseVolume()	
void	disconnectBluetooth()	
void	increaseVolume()	
void	playTrack(String track)	Відтворює трек і зменшує заряд батареї.
String	toString()	

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.