

Міністерство освіти і науки України

Національний університет «Львівська Політехніка»

Кафедра ЕОМ



ЗВІТ

з лабораторної роботи № 2

з дисципліни: «Кросплатформні засоби програмування»

на тему: «Класи та пакети»

Виконав:

студент гр. КІ-306

Приймак Ю.О.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

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.

16. Аудіоплеєр

Виконання:

Main.java

```
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class Main {
    private static AudioPlayer[] audioPlayers;

    public static void main(String[] args) {
        AudioPlayer audioPlayer = new AudioPlayer(new Screen(7.8,
"720x1980"), new HardDisk(2000, "Harman"));

        audioPlayer.AddSong("Stepan Giga - Zoloto Karpat");
        audioPlayer.AddSong("Stepan Giga - Yvoruna");
        audioPlayer.AddSong("Victor Pavlic - Shikidim");
        audioPlayer.AddSong("Zhadan i Sobaku - Madona");
        audioPlayer.AddSong("Zhadan i Sobaku - Kobzon");

        audioPlayer.TurnOnPrevSong();
        audioPlayer.TurnOnPrevSong();
        audioPlayer.TurnOnPrevSong();

        audioPlayer.TurnOnPrevSong();
        audioPlayer.TurnOnPrevSong();
        audioPlayer.TurnOnPrevSong();
        audioPlayer.TurnOnPrevSong();
```

```
System.out.println(audioPlayer);
```

```
    AudioPlayer audioPlayer2 = new AudioPlayer(new Screen(7,  
"720x1980"), new HardDisk(1000, "Harman99"));
```

```
//    main1();
```

```
}
```

```
public static void main1() {
```

```
    AudioPlayer audioPlayer3 = new AudioPlayer(new Screen(8,  
"720x1980"), new HardDisk(1000, "Harman1"));
```

```
    AudioPlayer audioPlayer1 = new AudioPlayer(new Screen(9,  
"720x1980"), new HardDisk(1000, "Harman0"));
```

```
    AudioPlayer audioPlayer4 = new AudioPlayer(new Screen(7,  
"720x1980"), new HardDisk(2000, "Harman9"));
```

```
    AudioPlayer audioPlayer5 = new AudioPlayer(new Screen(7,  
"720x1980"), new HardDisk(1000, "Harman100"));
```

```
int totalMaxCapacityCount = 0;
```

```
for (AudioPlayer player : audioPlayers) {
```

```
    player.printCapacity();
```

```
    totalMaxCapacityCount += AudioPlayer.getMaxCapacityCount();
```

```
}
```

```
// Вивести у консоль загальну кількість аудіоплеєрів з найбільшим capacity
```

```
System.out.println("Кількість аудіоплеєрів з найбільшим capacity: " + totalMaxCapacityCount);
```

```
    }  
}
```

Logger.java

```
import java.io.*;
```

```
import java.text.SimpleDateFormat;
```

```
import java.util.*;
```

```
/**
```

```
 * Class Logger. Was created to log information, errors and warnings. Also there was implemented Singleton
```

```
 * @author
```

```
 * @version 1.0
```

```
 */
```

```
public class Logger
```

```
{
```

```
    private static Logger logger;
```

```
    private final String fileName;
```

```
    protected final String infoFlag = new String("[INFO] ");
```

```
    protected final String errorFlag = new String("[ERROR] ");
```

```
    protected final String warningFlag = new String("[WARNING] ");
```

```

/**
 * Constructor
 * @param fileName
 */
private Logger(String fileName)
{
    this.fileName = fileName;
    File loggerFile = null;
    FileWriter fout = null;
    try
    {
        loggerFile = new File(fileName);
        fout = new FileWriter(loggerFile, true);

        SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd
'at' HH:mm:ss z");

        Date date = new Date(System.currentTimeMillis());

        fout.write "[" + formatter.format(date) + "]" + "Logger start to
work\n");
    }
    catch (IOException e)
    {
        System.err.println("Something wrong with log file" + e.getMessage());
        System.exit(1);
    }
    finally
    {
        try
        {

```

```

        fout.flush();
        fout.close();
    }
    catch (IOException e)
    {
        System.out.println(e.getMessage());
    }
}

/**
 * Method to do logging
 * @param massege
 */
public void log(String massege)
{
    File loggerFile = null;
    FileWriter fout = null;
    try
    {
        loggerFile = new File(this.fileName);
        fout = new FileWriter(loggerFile, true);

        SimpleDateFormat formatter= new SimpleDateFormat("yyyy-MM-dd
'at' HH:mm:ss z");

        Date date = new Date(System.currentTimeMillis());
        fout.write("[ " + formatter.format(date) + " ] " + massege + "\n");
    }
    catch (IOException e)

```

```

    {
        System.err.println("Something wrong with log file" + e.getMessage());
        System.exit(1);
    }
    finally
    {
        try
        {
            fout.flush();
            fout.close();
        }
        catch (IOException | NullPointerException e)
        {
            System.out.println(e.getMessage());
        }
    }
}

```

```
/**
```

```
 * Singleton implementation
```

```
 * @param fileName
```

```
 * @return
```

```
 */
```

```
public static Logger getLogger(String fileName)
```

```

{
    if (logger == null)
    {

```



```

        logger = new Logger(fileName);
    }
    return logger;
}

/**
 * Getter for logger
 * @return logger
 */
public static Logger getLogger()
{
    return logger;
}
}

```

AudioPlayer.java

```

import java.util.ArrayList;
import javax.sound.midi.Soundbank;
import org.w3c.dom.ls.LSOutput;
/**
 * Class AudioPlayer
 * @author
 * @version 1.0
 */
public class AudioPlayer

```

```

{
    private final Button nextSong = new Button("next song");
    private final Button prevSong = new Button("prev song");
    private final Button pause = new Button("pause");
    public int capacity;
    private Logger logger = Logger.getLogger("logs.txt");
    private Screen screen;
    private HardDisk hardDisk;
    private ArrayList<String> songs = new ArrayList<>();
    private int curSong = 0;
    static String name = "";
    private static double maxCapacity = 0;
    private static int maxCapacityCount = 0;

    public void printCapacity() {
        if (this.getHardDisk().getCapacity() == 1000) {
            System.out.println("AudioPlayer capacity with 1000: " +
this.getHardDisk().getCapacity());

            // Оновіть лічильник, якщо поточний об'єкт має максимальний
capacity

            if (this.getHardDisk().getCapacity() > getMaxCapacity()) {
                maxCapacity = this.getHardDisk().getCapacity();
                maxCapacityCount = 1;
            } else if (this.getHardDisk().getCapacity() == getMaxCapacity()) {
                maxCapacityCount++;
            }
        }
    }
}

```

```
public static double getMaxCapacity() {  
    return maxCapacity;  
}
```

```
public static int getMaxCapacityCount() {  
    return maxCapacityCount;  
}
```

```
public static void printMaxDiagonalLength(AudioPlayer... players) {  
    double maxDiagonal = 0;  
  
    for (AudioPlayer player : players) {  
        double diagonal =  
Math.sqrt(Math.pow(player.getScreen().getDiagonal(), 2) +  
            Math.pow(player.getScreen().getExpansion().length(), 2));  
  
        if (diagonal > maxDiagonal) {  
            maxDiagonal = diagonal;  
        }  
    }  
  
    System.out.println("Maximum diagonal length: " + maxDiagonal);  
}
```

```
/**
```

```
 * Method to be called in Main class to print max diagonal length
```

```

*/

public static void printMaxDiagonalLengthFromMain() {

    AudioPlayer audioPlayer = new AudioPlayer(new Screen(7.8,
"720x1980"), new HardDisk(1000, "Harman"));

    AudioPlayer audioPlayer1 = new AudioPlayer(new Screen(9,
"720x1980"), new HardDisk(1000, "Harman0"));

    AudioPlayer audioPlayer2 = new AudioPlayer(new Screen(7,
"720x1980"), new HardDisk(1000, "Harman9"));

    AudioPlayer audioPlayer3 = new AudioPlayer(new Screen(8,
"720x1980"), new HardDisk(1000, "Harman1"));

    printMaxDiagonalLength(audioPlayer, audioPlayer1, audioPlayer2,
audioPlayer3);

}

```

```

/**

```

```

    * Constructor

```

```

    * @param screen

```

```

    * @param hardDisk

```

```

*/

```

```

public AudioPlayer(Screen screen, HardDisk hardDisk) {

    logger.log(logger.infoFlag + "AudioPlayer constructor called");

    this.screen = screen;

    this.hardDisk = hardDisk;

}

```

```

/**

```

```

    * Method to add new song to player

```

```

    * @param song

```

```

*/

public void AddSong(String song)
{
    songs.add(song);
    System.out.println(song + " was added to audio player");
    logger.log(logger.infoFlag + "AudioPlayer AddSong method was called");
}

/**
 * Method to turn on next song
 */
public void TurnOnNextSong()
{
    logger.log(logger.infoFlag + "TurnOnNextSong AudioPlayer method was
called");
    if(curSong == songs.size() - 1)
    {
        System.out.println("You push button " + nextSong.getAction());
        System.out.println("Now playing " + songs.get(curSong));
        curSong = 0;
    } else if (curSong < songs.size() - 1) {
        System.out.println("You push button " + nextSong.getAction());
        System.out.println("Now playing " + songs.get(curSong));
        curSong++;
    }
}
}

```

```

/**
 * Method to turn on prev song
 */
public void TurnOnPrevSong()
{
    logger.log(logger.infoFlag + "TurnPrevNextSong AudioPlayer method was
called");
    if(curSong == 0)
    {
        System.out.println("You push button " + prevSong.getAction());
        System.out.println("Now playing " + songs.get(curSong));
        curSong = songs.size() - 1;
    } else if (curSong > 0) {
        System.out.println("You push button " + prevSong.getAction());
        System.out.println("Now playing " + songs.get(curSong));
        curSong--;
    }
}

public Button getNextSong() {
    return nextSong;
}

public Button getPrevSong() {
    return prevSong;
}

public Button getPause() {

```

```
    return pause;
}
```

```
public Logger getLogger() {
    return logger;
}
```

```
public void setLogger(Logger logger) {
    this.logger = logger;
}
```

```
public Screen getScreen() {
    return screen;
}
```

```
public void setScreen(Screen screen) {
    this.screen = screen;
}
```

```
public HardDisk getHardDisk() {
    return hardDisk;
}
```

```
public void setHardDisk(HardDisk hardDisk) {
    this.hardDisk = hardDisk;
}
```

```
public ArrayList<String> getSongs() {  
    return songs;  
}
```

```
public void setSongs(ArrayList<String> songs) {  
    this.songs = songs;  
}
```

```
public int getCurSong() {  
    return curSong;  
}
```

```
public void setCurSong(int curSong) {  
    this.curSong = curSong;  
}
```

@Override

```
public String toString() {  
    StringBuilder stringBuilder = new StringBuilder();  
    stringBuilder.append("AudioPlayer{ ")  
        .append(" screen=").append(screen).append("\n")  
        .append(", hardDisk=").append(hardDisk).append("\n")  
        .append(", songs=").append(songs).append("\n")  
        .append(", curSong=").append(curSong).append("\n");  
  
    stringBuilder.append('}');  
    return stringBuilder.toString();  
}
```



```
}

    public void printCapacity1000() {
    }
}
```

Sreen.java

```
/**
 * Class Screen
 * @author
 * @version 1.0
 */
public class Screen
{
    private double diagonal;
    private String expansion;

    /**
     * Constructor
     * @param diagonal
     * @param expansion
     */
    public Screen(double diagonal, String expansion) {
        this.diagonal = diagonal;
        this.expansion = expansion;
    }
}
```

```
/**
 * Getter for Diagonal
 * @return diagonal
 */
public double getDiagonal() {
    return diagonal;
}

/**
 * Setter for diagonal
 * @param diagonal
 */
public void setDiagonal(double diagonal) {
    this.diagonal = diagonal;
}

/**
 * Getter for expansion
 * @return
 */
public String getExpansion() {
    return expansion;
}

/**
 * Setter for expansion
```

```

    * @param expansion
    */
    public void setExpansion(String expansion) {
        this.expansion = expansion;
    }

    @Override
    public String toString() {
        return "Screen{ " +
            "diagonal = " + diagonal +
            ", expansion = '" + expansion + "'" +
            '}';
    }
}

```

HardDisk.java

```

/**
 * Class Hard Disk
 * @author
 * @version
 */
public class HardDisk
{
    private double capacity;
    private String producer;

```

```
/**
 * Constructor
 * @param capacity
 * @param producer
 */
public HardDisk(double capacity, String producer)
{
    this.capacity = capacity;
    this.producer = producer;
}
```

```
/**
 * Getter for capacity
 * @return capacity
 */
public double getCapacity() {
    return capacity;
}
```

```
/**
 * Setter for capacity
 * @param capacity
 */
public void setCapacity(double capacity) {
    this.capacity = capacity;
}
```

```

/**
 * Getter for producer
 * @return producer
 */
public String getProducer() {
    return producer;
}

/**
 * Setter for producer
 * @param producer
 */
public void setProducer(String producer) {
    this.producer = producer;
}

@Override
public String toString() {
    return "HardDisk{ " +
        "capacity = " + capacity + " mb." +
        ", producer = " + producer + "\" +
        '\"';
}
}

```

Button.java

```
/**
 * Class Button
 * @author
 * @version 1.0
 */
public class Button
{
    private String action;

    /**
     * Constructor
     * @param action
     */
    public Button(String action) {
        this.action = action;
    }

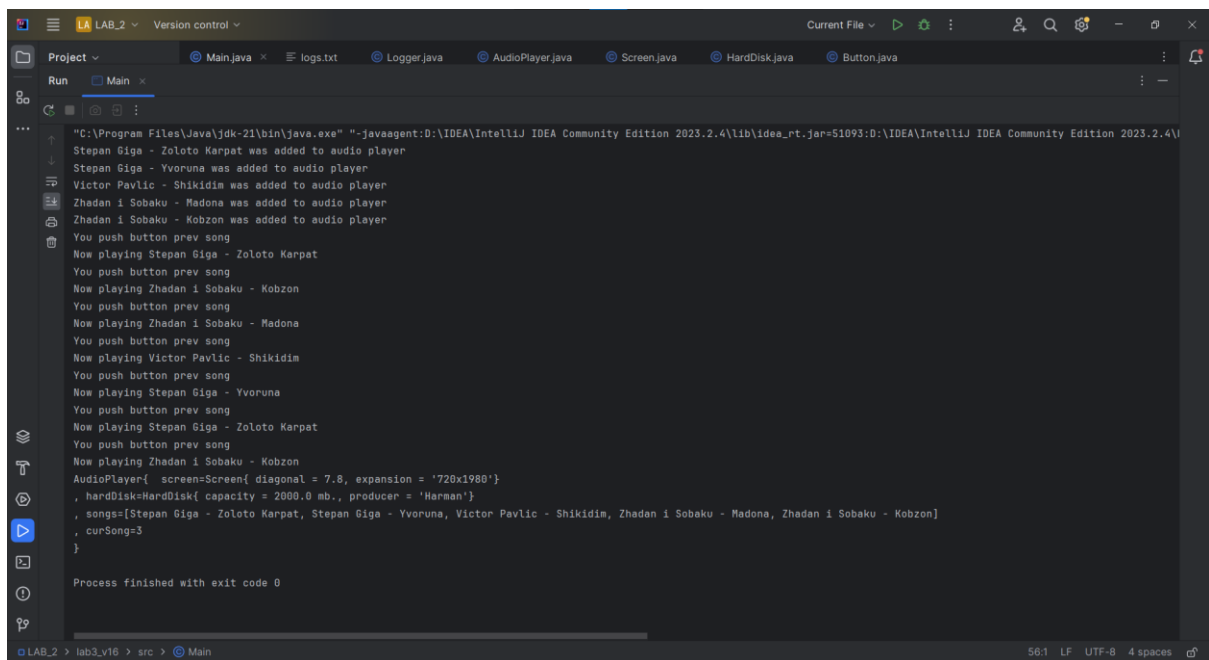
    /**
     * Getter for action
     * @return action
     */
    public String getAction() {
        return action;
    }

    /**
```

```
* Setter for action
* @param action
*/
public void setAction(String action) {
    this.action = action;
}

@Override
public String toString() {
    return "Button{ " +
        "action = " + action + "\" +
        '}'";
}
}
```

Результати:



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\IDEA\IntelliJ IDEA Community Edition 2023.2.4\lib\idea_rt.jar=51093:D:\IDEA\IntelliJ IDEA Community Edition 2023.2.4\bin" -Dfile.encoding=UTF-8
Stepan Giga - Zoloto Karpat was added to audio player
Stepan Giga - Yvورونا was added to audio player
Victor Pavlic - Shikidim was added to audio player
Zhadan i Sobaku - Madona was added to audio player
Zhadan i Sobaku - Kobzon was added to audio player
You push button prev song
Now playing Stepan Giga - Zoloto Karpat
You push button prev song
Now playing Zhadan i Sobaku - Kobzon
You push button prev song
Now playing Zhadan i Sobaku - Madona
You push button prev song
Now playing Victor Pavlic - Shikidim
You push button prev song
Now playing Stepan Giga - Yvورونا
You push button prev song
Now playing Stepan Giga - Zoloto Karpat
You push button prev song
Now playing Zhadan i Sobaku - Kobzon
AudioPlayer{ screen=Screen{ diagonal = 7.8, expansion = '720x1980'}
, hardDisk=HardDisk{ capacity = 2000.0 mb., producer = 'Harman'}
, songs=[Stepan Giga - Zoloto Karpat, Stepan Giga - Yvورونا, Victor Pavlic - Shikidim, Zhadan i Sobaku - Madona, Zhadan i Sobaku - Kobzon]
, curSong=3
}

Process finished with exit code 0
```

1	[2022-10-27 at 23:24:44 EEST]	Logger start to work
2	[2022-10-27 at 23:24:44 EEST]	[INFO] AudioPlayer constructor called
3	[2022-10-27 at 23:25:18 EEST]	Logger start to work
4	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer constructor called
5	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer AddSong method was called
6	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer AddSong method was called
7	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer AddSong method was called
8	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer AddSong method was called
9	[2022-10-27 at 23:25:18 EEST]	[INFO] AudioPlayer AddSong method was called
10	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
11	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
12	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
13	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
14	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
15	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
16	[2022-10-27 at 23:25:18 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called
17	[2022-10-27 at 23:26:11 EEST]	Logger start to work
18	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer constructor called
19	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer AddSong method was called
20	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer AddSong method was called
21	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer AddSong method was called
22	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer AddSong method was called
23	[2022-10-27 at 23:26:11 EEST]	[INFO] AudioPlayer AddSong method was called
24	[2022-10-27 at 23:26:11 EEST]	[INFO] TurnPrevNextSong AudioPlayer method was called

Висновок: у ході данної лабораторної роботи я ознайомився з процесом розробки класів та пакетів мовою Java.