Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Кафедра ЕОМ

Звіт

до лабораторної роботи № 3

З дисципліни: «Кросплатформенні засоби програмування»

На тему: **«КЛАСИ ТА ПАКЕТИ»**

Варіант 16

Виконав: ст.гр. КІ-301
Онисько М.М.
Прийняв: доцент каф.
Майдан М.В.

Львів 2023

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.Lab3;
    - 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 розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант завдання:
16. Аудіоплеєр

Виконання роботи:

Код програми:

```java
/**
* lab2 package
*/
package ki301OnyskoLab2;

import java.io.*;

/**
* Audioplayer Application class implements main method for AudioPlayer class possibilities demonstration
* @author MYKOLA ONYSKO
* @version 1.0
*/

public class AudioPlayerApp {
	/**
	* @param args
	* @throws FileNotFoundException
	*/
	public static void main(String[] args) throws FileNotFoundException{

		AudioPlayer audioplayer = new AudioPlayer("IPhone", "IPod 4", 99);
		Playlist playlistOfBroke = new Playlist("Плейлист броука");
		Song song1 = new Song("Скрябін - Говорили і курили", "3:31");
```

```
        Song song2 = new Song("Kanye West - Ok Ok", "3:24");
        Song song3 = new Song("The Prodigy - Breathe", "5:36");
        playlistOfBroke.addSong(song1);
        playlistOfBroke.addSong(song2);
        playlistOfBroke.addSong(song3);

        System.out.println("1.додати плейлист броук\n");
        audioplayer.addPlaylist(playlistOfBroke);

        Playlist playlistNirvana= new Playlist("Nirvana");
        Song song4 = new Song("Nirvana - Heart-Shaped Box", "4:41");
        Song song5 = new Song("Nirvana - Something In The Way", "3:52");
        Song song6 = new Song("Nirvana - Smells Like Teen Spirit", "5:01");
        playlistNirvana.addSong(song4);
        playlistNirvana.addSong(song5);
        playlistNirvana.addSong(song6);
        System.out.println("2.додати плейлист нірвана\n");
        audioplayer.addPlaylist(playlistNirvana);
        System.out.println("3.вибрати плейлист броук\n");
        audioplayer.setPlaylist(playlistOfBroke);
        System.out.println("4.вибір пісні2\n");
        audioplayer.setSong(song2);
        System.out.println("5.кнопка плей\n");
        audioplayer.playback();
        System.out.println("6.кнопка плей\n");
        audioplayer.playback();
        System.out.println("7.інформація плеєра\n");
        audioplayer.displayInfo();
        System.out.println("8.список пісень\n");
        audioplayer.displaySongs();
        System.out.println("9.вибрати плейлист нірвана\n");
        audioplayer.setPlaylist(playlistNirvana);
        System.out.println("10.список пісень\n");
        audioplayer.displaySongs();
        System.out.println("11.видалення пісні\n");
        audioplayer.removeSong(song4);
        System.out.println("12.список пісень\n");
        audioplayer.displaySongs();
        System.out.println("13.видалення плейлиста\n");
        audioplayer.removePlaylist(playlistNirvana);
        System.out.println("14.вибрати плейлист броук\n");
        audioplayer.setPlaylist(playlistOfBroke);
        System.out.println("15.вибір пісні1\n");
        audioplayer.setSong(song1);
        System.out.println("16.інформація плеєра\n");
        audioplayer.displayInfo();
        System.out.println("17.Наступна пісня\n");
        audioplayer.skipToNextSong();
        System.out.println("18.Наступна пісня\n");
        audioplayer.skipToNextSong();
        System.out.println("19.Наступна пісня\n");
        audioplayer.skipToNextSong();
        System.out.println("20.Наступна пісня\n");
        audioplayer.skipToNextSong();
        System.out.println("21.Попередня пісня\n");
        audioplayer.skipToPreviousSong();
        System.out.println("22.Попередня пісня\n");
        audioplayer.skipToPreviousSong();
        System.out.println("23.Попередня пісня\n");
        audioplayer.skipToPreviousSong();
        System.out.println("24.Попередня пісня\n");
        audioplayer.skipToPreviousSong();
        System.out.println("25.Попередня пісня\n");
        audioplayer.skipToPreviousSong();
```

```
            audioplayer.isCharging();
            audioplayer.chargeBattery();
            audioplayer.isCharging();
            audioplayer.chargeBattery();
            audioplayer.isCharging();
            audioplayer.dischargeBattery(10);
            audioplayer.dischargeBattery(100);
            audioplayer.getCurrentLevel();
      }
}
```

```
/**
 * Lab2 package
 */
package ki301OnyskoLab2;
import java.io.*;
import java.util.ArrayList;
import java.io.FileNotFoundException;
/**
 * Class <code>Audioplayer</code> implements audioplayer
 * @author MYKOLA ONYSKO
 * @version 1.0
 */
public class AudioPlayer  {
      PrintWriter fout;
      String brand;
      String model;
      ArrayList <Playlist> playlists;
      Playlist playlist;
      Song song;
      Battery battery;
      /**
       * Constructor
       * @throws FileNotFoundException
       */
      public AudioPlayer() throws FileNotFoundException{

            String brand = "невідомий";
            String model = "невідома";
            playlists = new ArrayList <Playlist>();
            playlist = new Playlist();
            battery = new Battery();
            song = new Song();
            fout = new PrintWriter(new File("Log.txt"));
      }
      /**
       * Constructor
       * @param <code>brand</code> Brand of audioplayer
       * @param <code>model</code> Model of audioplayer
       * @param <code>charge</code> Battery charge
       * @throws FileNotFoundException
       */
      public AudioPlayer(String brand, String model, int charge) throws
FileNotFoundException{
            this.brand = brand;
            this.model = model;
            playlists = new ArrayList <Playlist>();
            playlist = new Playlist();
            battery = new Battery(charge);
            song = new Song();
            fout = new PrintWriter(new File("Log.txt"));
      }
```

```
	/**
	 * Method returns a playlist with a list of songs
	 * @param <code>playlist</code>
	 */
	public void addPlaylist(Playlist playlist) {
		this.playlists.add(playlist);
		System.out.println("Додано плейлист " + playlist.getPlaylistName() +
"\n");
		fout.println("Додано плейлист " + playlist.getPlaylistName() + "\n");
		fout.flush();
	}
	/**
	 * Method selects the current playlist
	 * @param <code>playlist</code>
	 */
	public void setPlaylist(Playlist playlist) {
		for(Playlist tmpPlaylist : playlists) {
			if(tmpPlaylist == playlist) {
				this.playlist = playlist;
			}
		}
		System.out.println("Вибрано плейлист " + playlist.getPlaylistName() +
"\n");
		fout.println("Вибрано плейлист " + playlist.getPlaylistName() + "\n");
		fout.flush();
	}
	//є
	/**
	 * Method selects the current music
	 * @param song
	 */
	public void setSong(Song song) {
		for(Song tmpSong : playlist.getSongs()) {
			if(tmpSong == song) {
				this.song = song;
			}
		}
		System.out.println("Вибрано пісню " + song.getSongName() + "\n");
		fout.println("Вибрано пісню " + song.getSongName() + "\n");
		fout.flush();
	}
	/**
	 * Method turns the music on and off
	 */
	public void playback() {
		if(song.playback()) {
			System.out.println("Грає пісня " + song.getSongName() + "\n");
			fout.println("Грає пісня " + song.getSongName() + "\n");
		} else {
			System.out.println("Призупинена пісня " + song.getSongName() +
"\n");
			fout.println("Призупинена пісня " + song.getSongName() + "\n");
		}
		fout.flush();
	}
	/**
	 * Method deletes the current playlist
	 * @param playlist
	 */
	public void removePlaylist(Playlist playlist) {
		int i = this.playlists.indexOf(playlist);
		this.playlists.remove(i);
		System.out.println("Видалено плейлист " + playlist.getPlaylistName() +
"\n");
```

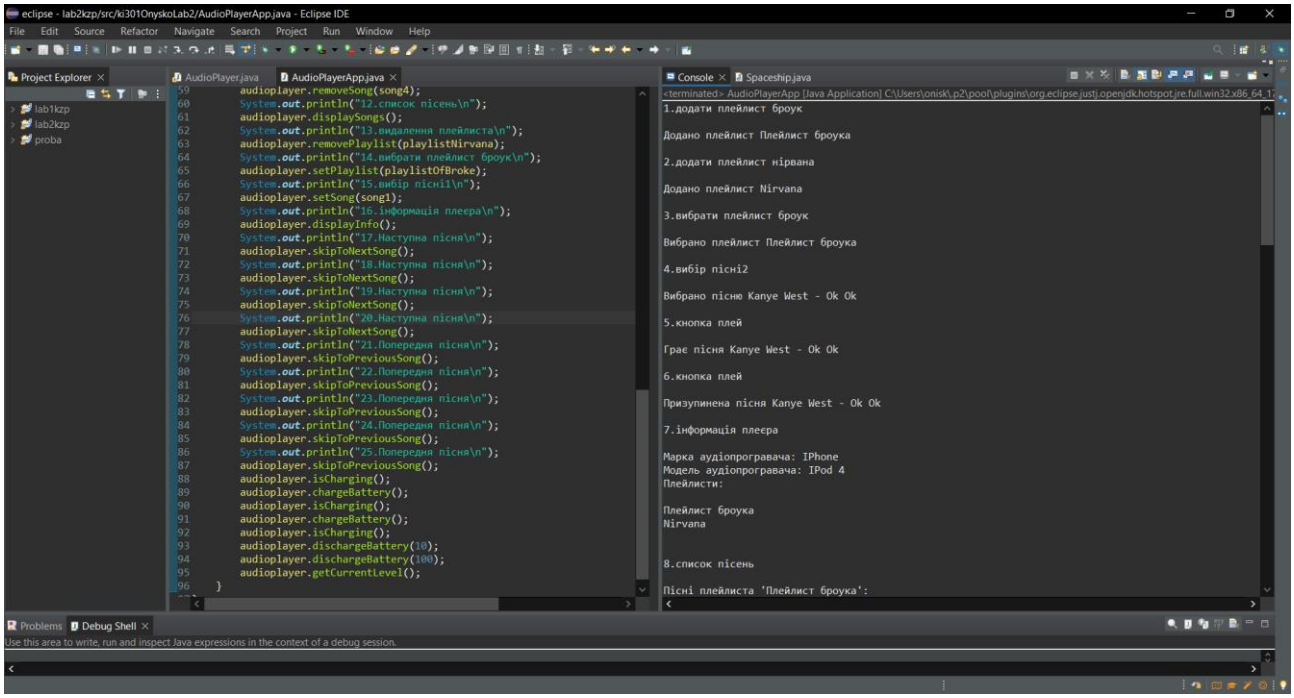
```
            fout.println("Видалено плейлист " + playlist.getPlaylistName() + "\n");
            fout.flush();
      }
      /**
       * Method switches to the next song in the playlist
       */
      public void skipToNextSong() {

            int i = playlist.getSongs().indexOf(song);
            if(i >= 0 && i < playlist.getSongs().size() - 1) {
                  song = playlist.getSongs().get(++i);
            }else {
                  song = playlist.getSongs().get(0);
            }
            System.out.println("Тепер грає пісня: " + song.getSongName());
            fout.println("Тепер грає пісня: " + song.getSongName());
            fout.flush();
      }
      /**
       * Method switches to the previous song in the playlist
       */
      public void skipToPreviousSong() {
            int i = playlist.getSongs().indexOf(song);
            if(i > 0) {
                  song = playlist.getSongs().get(--i);
            }else if(i==0){
                  int lastIndex = playlist.getSongs().size()-1;
                  song = playlist.getSongs().get(lastIndex);
            }
            System.out.println("Тепер грає пісня: " + song.getSongName());
            fout.println("Тепер грає пісня: " + song.getSongName());
            fout.flush();
      }
      /**
       * method displays information about the audio player
       */
      public void displayInfo( ) {

            System.out.println("Марка аудіопрогравача: " + brand +
                                    "\nМодель аудіопрогравача: "  + model +
                                          "\nПлейлисти:\n ");
            fout.println("Марка аудіопрогравача: " + brand +
                        "\nМодель аудіопрогравача: "  + model +
                              "\nПлейлисти:\n ");
            for(Playlist playlist : playlists) {
                  System.out.println(playlist.getPlaylistName());
                  fout.println(playlist.getPlaylistName());
            }
            System.out.println("\n");
            fout.println("\n");
            fout.flush();
      }
      /**
       * Method displays the songs in the selected playlist
       */
      public void displaySongs() {
            System.out.println("Пісні плейлиста '"+
this.playlist.getPlaylistName()+"':\n");
            fout.println("Пісні плейлиста '"+
this.playlist.getPlaylistName()+"':\n");
            for (Song song : this.playlist.getSongs()){
                  System.out.println(song.getSongName());
                  fout.println(song.getSongName());
            }
```

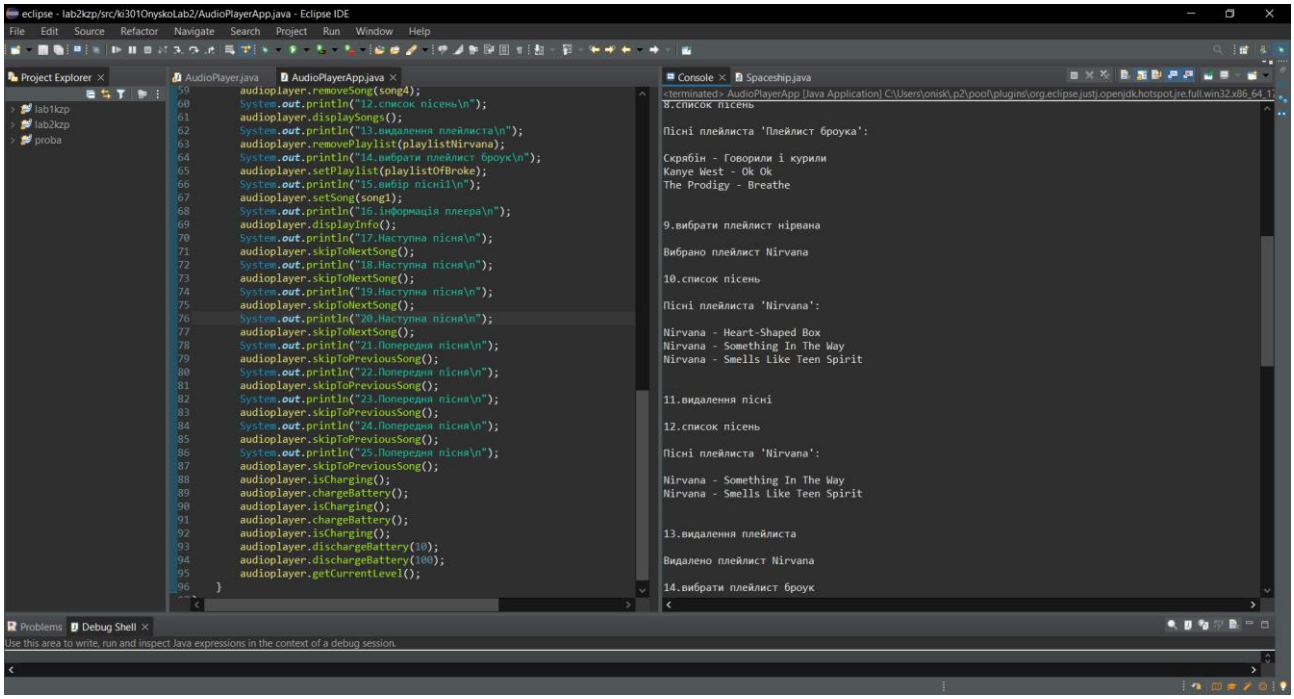
```
            System.out.println("\n");
            fout.println("\n");
            fout.flush();
      }
      /**
       * Method deletes the current music
       * @param song
       */
      public void removeSong(Song song) {
            playlist.removeSong(song);
      }
      /**
       * Method displays the current battery charge
       */
      public void getCurrentLevel() {
            battery.getCurrentLevel();
            System.out.println("Поточна батарея:" + battery.getCurrentLevel() +
"\n");
            fout.println("Поточна батарея:" + battery.getCurrentLevel() + "\n");
            fout.flush();
      }
      /**
       * Method displays whether the battery is currently charging
       */
      public void isCharging() {
            battery.isCharging();
            if(battery.isCharging()==true){
                  System.out.println("Батарея заряджається\n");
                  fout.println("Батарея заряджається\n");
                  fout.flush();
            }else {
                  System.out.println("Батарея не заряджається\n");
                  fout.println("Батарея не заряджається\n");
                  fout.flush();
            }
      }
      /**
       * Method charges the battery
       */
      public void chargeBattery() {
            battery.chargeBattery();
            if(battery.isCharging()) {
                  if(battery.getCurrentLevel()<100) {
                        System.out.println("Батарея почала заряджатись\n");
                    fout.println("Батарея почала заряджатись\n");
                  }
                  else {
                        System.out.println("Батарея не потребує зарядки\n");
                    fout.println("Батарея не потребує зарядки\n");
                  }
            }else {
                  System.out.println("Батарея перестала заряджатись\n");
              fout.println("Батарея перестала заряджатись\n");
            }
            fout.flush();
      }
      /**
       * Method dicharges battery
       * @param energyConsumed
       */
      public void dischargeBattery(int energyConsumed) {
             System.out.println("Заряд батареї зменшився");
             fout.println("Заряд батареї зменшився");
             int t = battery.dischargeBattery(energyConsumed);
```

```
                if(t <= 0)
                {
                        System.out.println("Батарея повністю розряджена\n");
                        fout.println("Батарея повністю розряджена\n");
                }else {
                        System.out.println("Стан батареї:" + t + "\n");
                       fout.println("Батарея повністю розряджена\n");
                }
        fout.flush();
        }

        public void dispose()
        {
        fout.close();
        }

}
class Song{

        private String songName;
        private String songLength;
        private boolean isPlayed;
        /*
         * Constructor
         */
        public Song(){
                songName = "невідома";
                songLength = "00:00";
                isPlayed = false;
        }
        /**
         * Constructor
         * @param <code>songName</code> Name of song
         * @param <code>songLength</code> Song length
         */
        public Song(String songName, String songLength){
                this.songName = songName;
                this.songLength = songLength;
                isPlayed = false;
        }
        /**
         * Method returns the name of the song
         * @return songName
         */
        public String getSongName() {
                return songName;
        }
        /*
         * Method returns the length of the song
         * @return songLength
         */
        public String getSongLength() {
                return songLength;
        }
        /**
         * Method to play/pause music
         * @return true, if the song is not playing, false in other cases
         */
        public boolean playback() {

                if(isPlayed == false) {
                        isPlayed = true;
                        return isPlayed;
                }else{
```

```
                    isPlayed = false;
                    return isPlayed;
                }
        }
}
class Playlist{

        private ArrayList <Song> songs;
        private String playlistName;
        /**
         * Constructor
         */
        public Playlist(){
                playlistName = "невідома";
                songs = new ArrayList<Song>();
        }
        /**
         * Constructor
         * @param <code>playlistName</code> Name of playlist
         */
        public Playlist(String playlistName){
                this.playlistName = playlistName;
                songs = new ArrayList<Song>();
        }
        /**
         * Method adds a song
         * @param song
         */
        public void addSong(Song song) {
                this.songs.add(song);
        }
        /**
         * Method returns the name of the playlist
         * @return playlistName
         */
        public String getPlaylistName() {
                return playlistName;
        }
        /**
         * Method sets the current playlist
         * @param playlistName
         * @return playlistName
         */
        public void setPlaylistName(String playlistName) {
                this.playlistName = playlistName;
        }
        /**
         * Method deletes the song
         * @param song
         */
        public void removeSong(Song song) {
                int i = this.songs.indexOf(song);
                this.songs.remove(i);
        }
        /**
         * Method returns songs
         * @return songs
         */
        public ArrayList <Song> getSongs(){
                return songs;
        }
}
class Battery{
```

```java
    private int currentLevel;
    private boolean charging;
    private int energyConsumed;
    /**
     * Constructor
     */
    public Battery() {
          int currentLevel = 100;
          boolean charging = false;
          int energyConsumed = 0;
    }
    /**
     * Constructor
     * @param <code>currentLevel</code>
     */
    public Battery(int currentLevel) {
          this.currentLevel = currentLevel;
          boolean charging = false;
          int energyConsumed = 0;
    }
    /**
   * Method to get the current battery level.
   * @return current battery level in percent
   */
     public int getCurrentLevel() {
       return currentLevel;
   }
   /**
    * Method to check if the battery is charging.
    * @return true, if the battery is charging, false in other cases
    */
   public boolean isCharging() {
       return charging;
   }
   /**
    * Method for charging the battery.
    * Checks the battery for charging and for reaching the maximum charge level.
    */
   public boolean chargeBattery() {
     if(!charging)
     {
          charging = true;
          return charging;

     }
     else {
           charging = false;
          return charging;
     }
   }
   /**
    * A method for discharging the battery of an audioplayer
    * @param energyConsumed
    */
   public int dischargeBattery(int energyConsumed) {
       currentLevel -= energyConsumed;

     if(currentLevel <= 0)
     {
           currentLevel = 0;
       }
     return currentLevel;
   }
}
```


eclipse - lab2kzp/src/ki301OnyskoLab2/AudioPlayerApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
lab1kzp
lab2kzp
proba
AudioPlayer.java
AudioPlayerApp.java
audioplayer.removeSong(song4);
System.out.println("12.список пісень\n");
audioplayer.displaySongs();
System.out.println("13.видалення плейлиста\n");
audioplayer.removePlaylist(playlistNirvana);
System.out.println("14.вибрати плейлист броук\n");
audioplayer.setPlaylist(playlistOfBroke);
System.out.println("15.вибір пісні1\n");
audioplayer.setSong(song1);
System.out.println("16.інформація плеєра\n");
audioplayer.displayInfo();
System.out.println("17.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("18.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("19.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("20.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("21.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("22.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("23.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("24.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("25.Попередня пісня\n");
audioplayer.skipToPreviousSong();
audioplayer.isCharging();
audioplayer.chargeBattery();
audioplayer.isCharging();
audioplayer.chargeBattery();
audioplayer.isCharging();
audioplayer.dischargeBattery(10);
audioplayer.dischargeBattery(100);
audioplayer.getCurrentLevel();
}
Console
Spaceship.java
<terminated> AudioPlayerApp [Java Application] C:\Users\onisk\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17
1.додати плейлист броук
Додано плейлист Плейлист броука
2.додати плейлист нірвана
Додано плейлист Nirvana
3.вибрати плейлист броук
Вибрано плейлист Плейлист броука
4.вибір пісні2
Вибрано пісню Kanye West - Ok Ok
5.кнопка плей
Грає пісня Kanye West - Ok Ok
6.кнопка плей
Призупинена пісня Kanye West - Ok Ok
7.інформація плеєра
Марка аудіопрогравача: IPhone
Модель аудіопрогравача: IPod 4
Плейлисти:
Плейлист броука
Nirvana
8.список пісень
Пісні плейлиста 'Плейлист броука':
Problems
Debug Shell
Use this area to write, run and inspect Java expressions in the context of a debug session.


eclipse - lab2kzp/src/ki301OnyskoLab2/AudioPlayerApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
lab1kzp
lab2kzp
proba
AudioPlayer.java
AudioPlayerApp.java
audioplayer.removeSong(song4);
System.out.println("12.список пісень\n");
audioplayer.displaySongs();
System.out.println("13.видалення плейлиста\n");
audioplayer.removePlaylist(playlistNirvana);
System.out.println("14.вибрати плейлист броук\n");
audioplayer.setPlaylist(playlistOfBroke);
System.out.println("15.вибір пісні1\n");
audioplayer.setSong(song1);
System.out.println("16.інформація плеєра\n");
audioplayer.displayInfo();
System.out.println("17.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("18.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("19.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("20.Наступна пісня\n");
audioplayer.skipToNextSong();
System.out.println("21.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("22.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("23.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("24.Попередня пісня\n");
audioplayer.skipToPreviousSong();
System.out.println("25.Попередня пісня\n");
audioplayer.skipToPreviousSong();
audioplayer.isCharging();
audioplayer.chargeBattery();
audioplayer.isCharging();
audioplayer.chargeBattery();
audioplayer.isCharging();
audioplayer.dischargeBattery(10);
audioplayer.dischargeBattery(100);
audioplayer.getCurrentLevel();
}
Console
Spaceship.java
<terminated> AudioPlayerApp [Java Application] C:\Users\onisk\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17
8.список пісень
Пісні плейлиста 'Плейлист броука':
Скрябін - Говорили і курили
Kanye West - Ok Ok
The Prodigy - Breathe
9.вибрати плейлист нірвана
Вибрано плейлист Nirvana
10.список пісень
Пісні плейлиста 'Nirvana':
Nirvana - Heart-Shaped Box
Nirvana - Something In The Way
Nirvana - Smells Like Teen Spirit
11.видалення пісні
12.список пісень
Пісні плейлиста 'Nirvana':
Nirvana - Something In The Way
Nirvana - Smells Like Teen Spirit
13.видалення плейлиста
Видалено плейлист Nirvana
14.вибрати плейлист броук
Problems
Debug Shell
Use this area to write, run and inspect Java expressions in the context of a debug session.

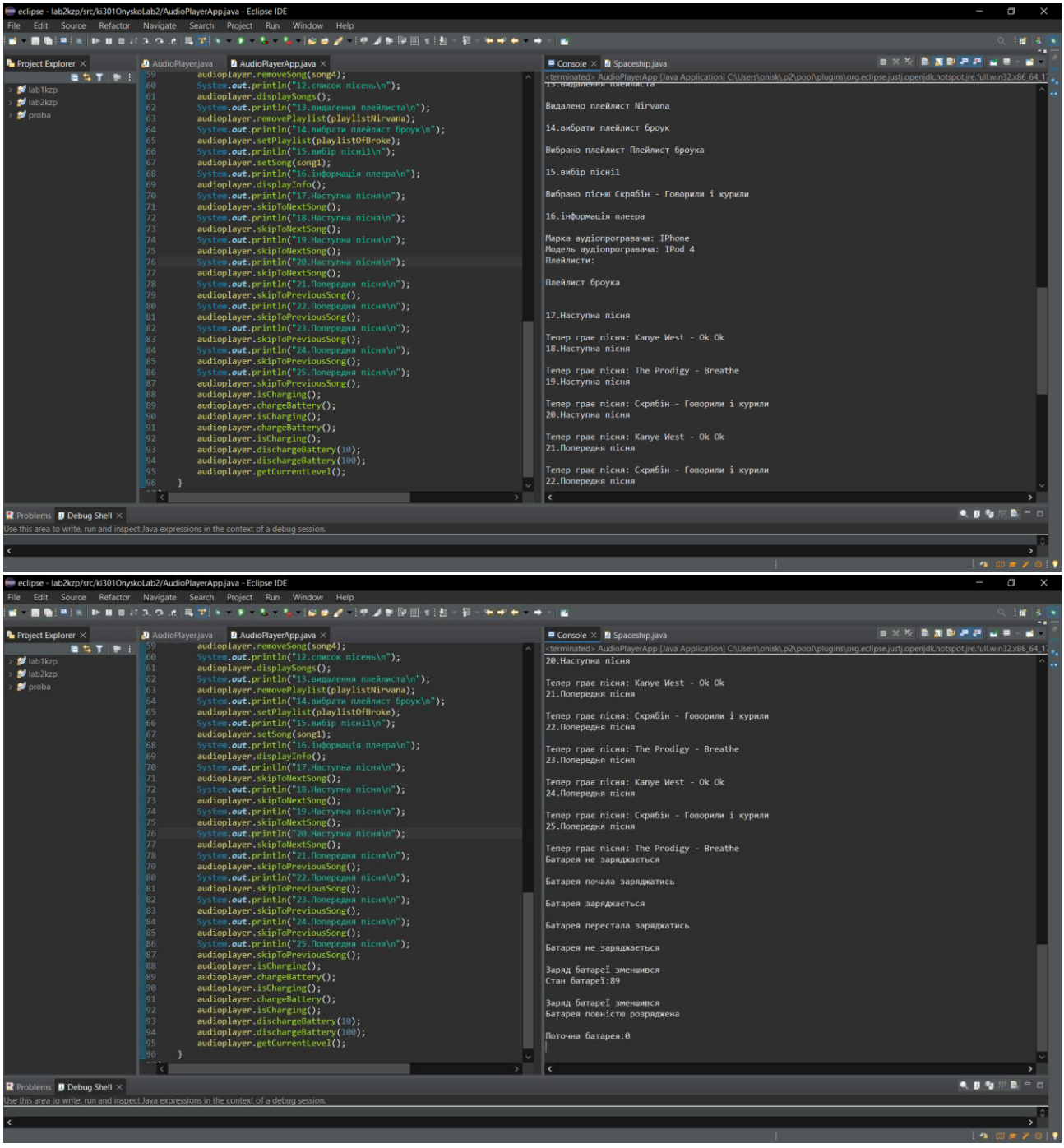


Рис.1 Код та вивід програми

PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package ki301OnyskoLab2

Class AudioPlayer

java.lang.Object
ki301OnyskoLab2.AudioPlayer

```
public class AudioPlayer
extends Object
```

Class Audioplayer implements audioplayer

Version:

1.0

Author:

MYKOLA ONYSKO

Constructor Summary

Constructors

Constructor	Description
`AudioPlayer()`	Constructor
`AudioPlayer(String brand, String model, int charge)`	Constructor

Method Summary

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

Modifier and Type	Method	Description
void	`addPlaylist(ki301OnyskoLab2.Playlist playlist)`	Method returns a playlist with a list of songs
void	`chargeBattery()`	Method charges the battery
void	`dischargeBattery(int energyConsumed)`	Method dicharges battery
void	`displayInfo()`	method displays information about the audio player
void	`displaySongs()`	Method displays the songs in the selected playlist

PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH: Search

Package ki301OnyskoLab2

Class AudioPlayerApp

java.lang.Object
ki301OnyskoLab2.AudioPlayerApp

```
public class AudioPlayerApp
extends Object
```

Audioplayer Application class implements main method for AudioPlayer class possibilities demonstration

Version:

1.0

Author:

MYKOLA ONYSKO

Constructor Summary

Constructors

Constructor	Description
`AudioPlayerApp()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
static void	`main(String[] args)`	

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructor Details

Рис.2 Автоматично згенерована документація до розробленої програми.

Висновок:
Під час виконання завдання, я розробив програму, яка моделює аудіоплеєр, реалізувавши класи та методи. Навчився створювати пакети, конструктори, методи та генерувати документацію Java.

Контрольні питання:
1. Синтаксис визначення класу:
```
access_modifier class ClassName {
    // тіло класу
}
```

2. Синтаксис визначення методу:
```
access_modifier return_type methodName(parameters) {
    // тіло методу
}
```

3. Синтаксис оголошення поля:
```
access_modifier data_type fieldName;
```

4. Як оголосити та ініціалізувати константне поле:
```
static final data_type CONSTANT_NAME = value;
```

5. Способи ініціалізації полів:
- Пряма ініціалізація в оголошенні поля.
- Ініціалізація в конструкторі класу.
- Ініціалізація в блоку ініціалізації.

6. Синтаксис визначення конструктора:
```
ClassName(parameters) {
    // тіло конструктора
}
```

7. Синтаксис оголошення пакету:
```
package package_name;
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах:
Використовуйте імпорт:
```
import package_name.ClassName;
```

9. Суть статичного імпорту пакетів:

Статичний імпорт дозволяє вам використовувати статичні (static) члени класу без зазначення імені класу. Наприклад: `import static package_name.ClassName.staticMember;`.

10. Вимоги до файлів і каталогів при використанні пакетів:
- Файли класів повинні розташовуватися в каталозі, який відповідає їхньому пакету.
- Назви каталогів і пакетів повинні відповідати ієрархії пакетів.
- Компілятор Java автоматично знаходить класи у відповідних каталогах згідно із заявленими пакетами.