

# **JAVA MEDIA PLAYER**

## **A Project Work Report**

*Submitted in the partial fulfilment for the award of the degree of*

## **BACHELOR OF ENGINEERING**

**IN**

## **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**Submitted by:**

**HIMANSHU YADAV**

**19BCS6100**

**Under the Supervision of:**

**PROF. VIJAY SONI**



**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB**

**December, 2020**

Name and signature of student(s)  
Himanshu Yadav

Name and signature of Supervisor

---

# PROJECT COMPLETION CERTIFICATE

## JAVA MEDIA PLAYER

This is to certify that the Himanshu Yadav has successfully completed the project work titled “ Java Media Player ” *Submitted in the partial fulfilment for the award of the degree of* **BACHELOR OF ENGINEERING IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING.**

This project is the record of authentic work carried out during the academic year 2020-21.

---

Project Guide

**Date: December 04, 2020**

# DECLARATION

I the undersigned solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of Prof. Viay Soni. I assert the statements made and conclusions drawn are an outcome of my work. I further certify that the work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Himanshu Yadav  
( 19BCS6100 )

# **ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to (Name of your Organization Guide) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents and my department for their kind co-operation and encouragement which help me in completion of this project.

**THANKS AGAIN TO ALL WHO HELPED**

## **Content**

Chapter 1: Introduction to project

Chapter 2: Project Requirements

Chapter 3: Implementation Details

Chapter 4: Output Analysis

# Chapter 1: Introduction to project

‘Java Media Player’.

A media file is a file that contains something pictorial for a person to infer by watching without much requirement of aesthetics of reading generally.

For example, An audio or a video, this is what the media file is all about.

A media player is an Application or software that is used to show the content of a media file. Commonly known media players include VLC, WinAmp, Windows Mediaplayer, NeroEssentials MediaPlayer etc.

The basic idea of this project is to play an Audio and Video files’playlist using swing components/framework as base.



## **Chapter 2: Project Requirements**

### **Hardware:**

- Intel or AMD x86 Compatible CPU with SSE2
- 256MB RAM
- 300MB free hard drive space for installation (plus space for digital media)
- Sound device with a working WDM or ASIO driver and connecting cables
- Display Monitor and connecting cables

### **Software:**

The software required in order to build the system is as follows:

>> Java Development Kit

>> Java Virtual Machine

>> JAVA external Libraries : javax.swing, javafx (with all dependencies), java.io, java.util

>> Coding: Eclipse (Preffered) or NetBeans IDE

>> Operating System : Windows, Mac, Linux, JVM

## **Chapter 3: Implementation Details**

### **Algorithm:**

1. Start.
2. Check the given playlist directory for audio and video file
3. If audio/video not present :      EXIT  
    Else : Continue
4. Play the audio (.mp3) and (.mp4) files in the order one by one
5. If Skip pressed : Go to next file, if no next file then play first file
6. If Pause is pressed : Pause the mediaplayer and change Pause button to Play.
7. If Play is pressed : Play the mediaplayer and change Play button to Pause.
8. If File is completely traversed : Go to next file, if no next file then play first file
9. If Cross pressed :              EXIT



## Code :

```
package pkg;

import java.io.*;
import java.util.*;
import javafx.application.Platform;
import javafx.beans.value.*;
import javafx.embed.swing.JFXPanel;
import javafx.event.*;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.media.*;
import javafx.util.Duration;
import javax.swing.*;

public class MPav {
    private static void initAndShowGUI() {
        // This method is invoked on Swing thread
        JFrame frame = new JFrame("JAVA Media PLayer");

        final JFXPanel fxPanel = new JFXPanel();
        frame.add(fxPanel);

        frame.setBounds(200, 100, 800, 550);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setVisible(true);

        Platform.runLater(new Runnable() {
            @Override public void run() {
                initFX(fxPanel);
            }
        });
    }

    private static void initFX(JFXPanel fxPanel) {
        // This method is invoked on JavaFX thread
        Scene scene = new SceneGenerator().createScene();
        fxPanel.setScene(scene);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override public void run() {
                initAndShowGUI();
            }
        });
    }
}
```

```

    });
}
}

class SceneGenerator {
    final Label currentlyPlaying = new Label();
    final ProgressBar progress = new ProgressBar();
    private ChangeListener<Duration> progressChangeListener;

    public Scene createScene() {
        final StackPane layout = new StackPane();

        // DEFAULT DIRECTORY
        File dir = new File("C:\\Users\\Owner\\Documents\\files AV");

        /*
        JFrame sel = new JFrame();
        sel.setBounds(30, 30, 200, 300);

        sel.setVisible(true);
        */
        JFileChooser jfc = new JFileChooser("CHOOSE THE PLAYLIST
        DIRECTORY/FOLDER");
        jfc.setFileSelectionMode( JFileChooser.DIRECTORIES_ONLY );
        int result = jfc.showOpenDialog( null );
        jfc.setVisible(true);
        if ( result != JFileChooser.CANCEL_OPTION )
        {
            dir = jfc.getSelectedFile();
            // add=null;
        }

        if (!dir.exists() || !dir.isDirectory()) {
            System.out.println("Cannot find video source directory: " + dir);
            Platform.exit();

            return null;
        }

        // create some media players.
        final List<MediaPlayer> players = new ArrayList<MediaPlayer>();
        for (String file : dir.list(new FilenameFilter() {
            @Override public boolean accept(File dir, String name) {
                return name.endsWith(".mp3") || name.endsWith(".mp4");
            }
        }))
            players.add(createPlayer("file:/// " + (dir + "\\ " + file).replace("\\ ",
            "/" ).replaceAll(" ", "%20")));
        if (players.isEmpty()) {
            System.out.println("No audio/video found in " + dir);
            Platform.exit();
            return null;
        }
    }
}

```

```

    }

    // create a view to show the mediaplayers.
    final MediaView mediaView = new MediaView(players.get(0));
    final Button skip = new Button("Skip");
    final Button play = new Button("Pause");

    // play each audio file in turn.
    for (int i = 0; i < players.size(); i++) {
        final MediaPlayer player = players.get(i);
        final MediaPlayer nextPlayer = players.get((i + 1) % players.size());
        player.setOnEndOfMedia(new Runnable() {
            @Override public void run() {

                player.currentTimeProperty().removeListener(progressChangeListener);
                mediaView.setMediaPlayer(nextPlayer);
                nextPlayer.play();
            }
        });
    }

    // allow the user to skip a track.
    skip.setOnAction(new EventHandler<ActionEvent>() {
        @Override public void handle(ActionEvent actionEvent) {
            final MediaPlayer curPlayer = mediaView.getMediaPlayer();
            MediaPlayer nextPlayer = players.get((players.indexOf(curPlayer) + 1)
% players.size());
            mediaView.setMediaPlayer(nextPlayer);

            curPlayer.currentTimeProperty().removeListener(progressChangeListener);
            curPlayer.stop();
            nextPlayer.play();
        }
    });

    // allow the user to play or pause a track.
    play.setOnAction(new EventHandler<ActionEvent>() {
        @Override public void handle(ActionEvent actionEvent) {
            if ("Pause".equals(play.getText())) {
                mediaView.getMediaPlayer().pause();
                play.setText("Play");
            } else {
                mediaView.getMediaPlayer().play();
                play.setText("Pause");
            }
        }
    });

    // display the name of the currently playing track.
    mediaView.mediaPlayerProperty().addListener(new
ChangeListener<MediaPlayer>() {

```

```

        @Override public void changed(ObservableValue<? extends MediaPlayer>
observableValue, MediaPlayer oldPlayer, MediaPlayer newPlayer) {
            setCurrentlyPlaying(newPlayer);
        }
    });

    // start playing the first track.
    mediaView.setMediaPlayer(players.get(0));
    mediaView.getMediaPlayer().play();
    setCurrentlyPlaying(mediaView.getMediaPlayer());

    // silly invisible button used as a template to get the actual preferred
    size of the Pause button.
    Button invisiblePause = new Button("Pause");
    invisiblePause.setVisible(false);
    play.prefHeightProperty().bind(invisiblePause.heightProperty());
    play.prefWidthProperty().bind(invisiblePause.widthProperty());

    // layout the scene.
    layout.setStyle("-fx-background-color: cornsilk; -fx-font-size: 20; -fx-
padding: 20; -fx-alignment: center;");
    layout.getChildren().addAll(
        invisiblePause,
        VBoxBuilder.create().spacing(10).alignment(Pos.CENTER).children(
            currentlyPlaying,
            mediaView,
            HBoxBuilder.create().spacing(10).alignment(Pos.CENTER).children(skip,
play, progress).build()
        ).build()
    );
    progress.setMaxWidth(Double.MAX_VALUE);
    HBox.setHgrow(progress, Priority.ALWAYS);
    return new Scene(layout, 800, 600);
}

/** sets the currently playing label to the label of the new media player
and updates the progress monitor. */
private void setCurrentlyPlaying(final MediaPlayer newPlayer) {
    progress.setProgress(0);
    progressChangeListener = new ChangeListener<Duration>() {
        @Override public void changed(ObservableValue<? extends Duration>
observableValue, Duration oldValue, Duration newValue) {
            progress.setProgress(1.0 * newPlayer.getCurrentTime().toMillis() /
newPlayer.getTotalDuration().toMillis());
        }
    };
    newPlayer.currentTimeProperty().addListener(progressChangeListener);

    String source = newPlayer.getMedia().getSource();
    source = source.substring(0, source.length() - ".mp4".length());

```

```




        source = source.substring(source.lastIndexOf("/") + 1).replaceAll("%20",
" ");
        currentlyPlaying.setText("Now Playing: " + source);
    }

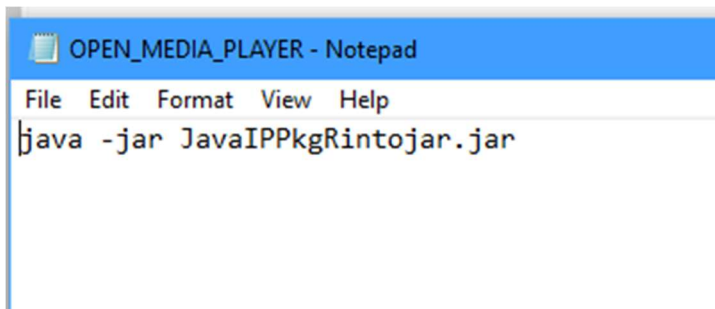
    /** @return a MediaPlayer for the given source which will report any errors
it encounters */
    private MediaPlayer createPlayer(String aMediaSrc) {
        System.out.println("Creating player for: " + aMediaSrc);
        final MediaPlayer player = new MediaPlayer(new Media(aMediaSrc));
        player.setOnError(new Runnable() {
            @Override public void run() {
                System.out.println("Media error occurred: " + player.getError());
            }
        });
        return player;
    }
}

```

---

R (H:) > HIMANSHU\_19BCS6100

Name	Date modified	Type	Size
 JavaIPpkgRintojar	12/12/2020 12:20 AM	Executable Jar File	12 KB
 OPEN_MEDIA_PLAYER	12/12/2020 12:21 AM	Windows Batch File	1 KB
 ReadMe	12/12/2020 12:38 AM	Text Document	5 KB



```

OPEN_MEDIA_PLAYER - Notepad
File Edit Format View Help
java -jar JavaIPpkgRintojar.jar

```





## Chapter 4: Output Analysis






MY COMPUTER > Documents > files AV		
	Sample_BeeMoved_48kHz16bit.m4a	Date modified: 12/4/2020 5:07 PM
	aaa dbz14 Length: 00:19:11	Frame height: 272 Frame width: 512 Date modified: 11/11/2017 1:09 AM Size: 44.3 MB
	Alan Walker - Alone www.my-free-mp3.net	Length: 00:02:39 Size: 6.07 MB
	Axwell _ Ingrosso - Dreamer www.my-free-mp3.net	Length: 00:04:11 Size: 9.58 MB
	K-391, Alan Walker - Ignite 320kbps(Mp3go.in) Mp3go.in	Album: Ignite - K-391 & Alan Walker - Mp3go.in Genre: Club Length: 00:03:49 Size: 8.81 MB

Fig 1: The Contents of Playlist Folder (files AV)

```
Try [Java Application] C:\Program Files (x86)\Java\jre1.8.0_251\bin\javaw.exe (Dec 4, 2020, 9:22:05 PM)
Creating player for: file:///C:/Users/Owner/Documents/files%20AV/aaa%20dbz14.mp4
Creating player for: file:///C:/Users/Owner/Documents/files%20AV/Alan%20Walker%20-%20Alone%20www.my-free-mp3.net%20.mp3
Creating player for: file:///C:/Users/Owner/Documents/files%20AV/Axwell%20_%20Ingrosso%20-%20Dreamer%20www.my-free-mp3.net%20.mp3
Creating player for: file:///C:/Users/Owner/Documents/files%20AV/K-391,%20Alan%20Walker%20-%20Ignite%20320kbps(Mp3go.in).mp3
```

Fig 2: The Console output informing of the order acquired

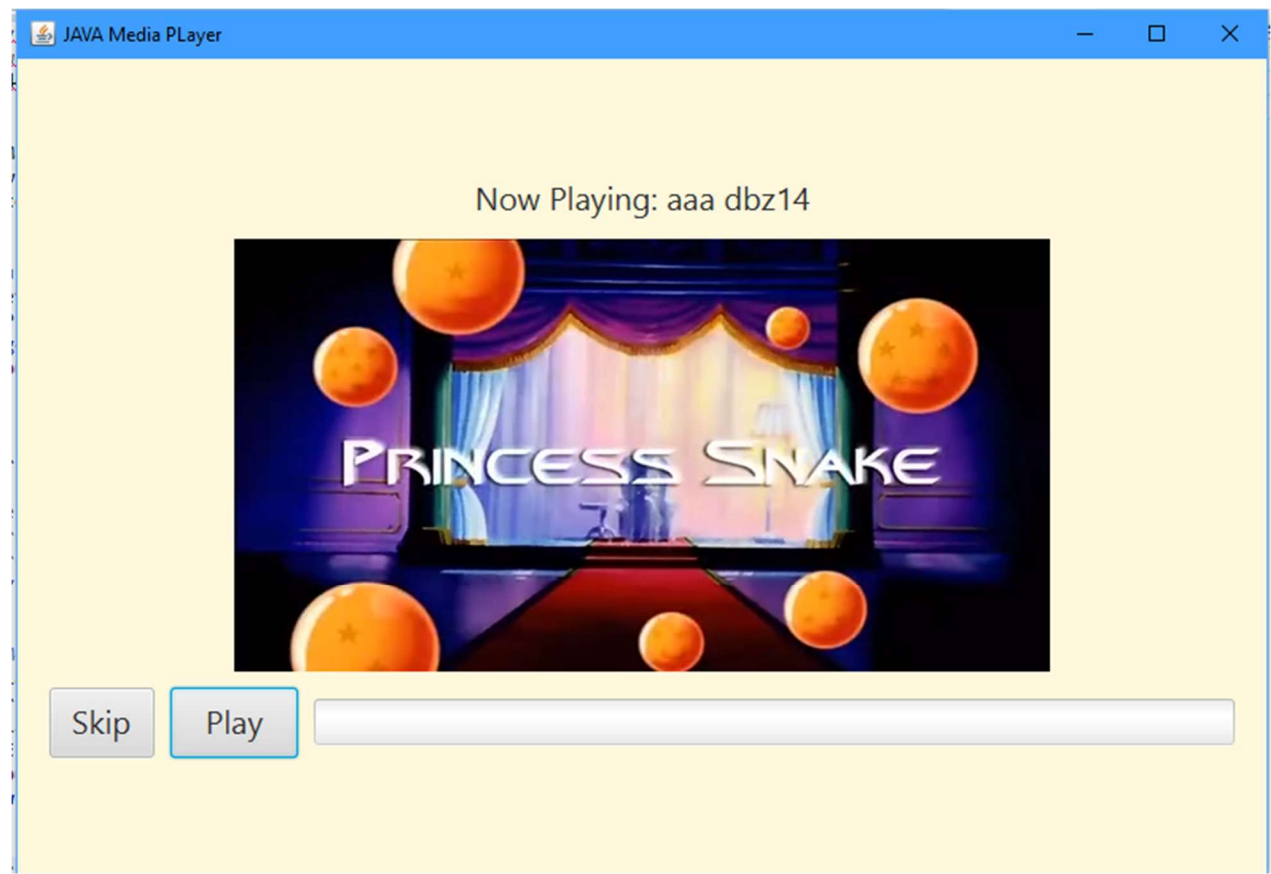


Fig 3: (Paused) First Media File of the playlist

[VIDEO]



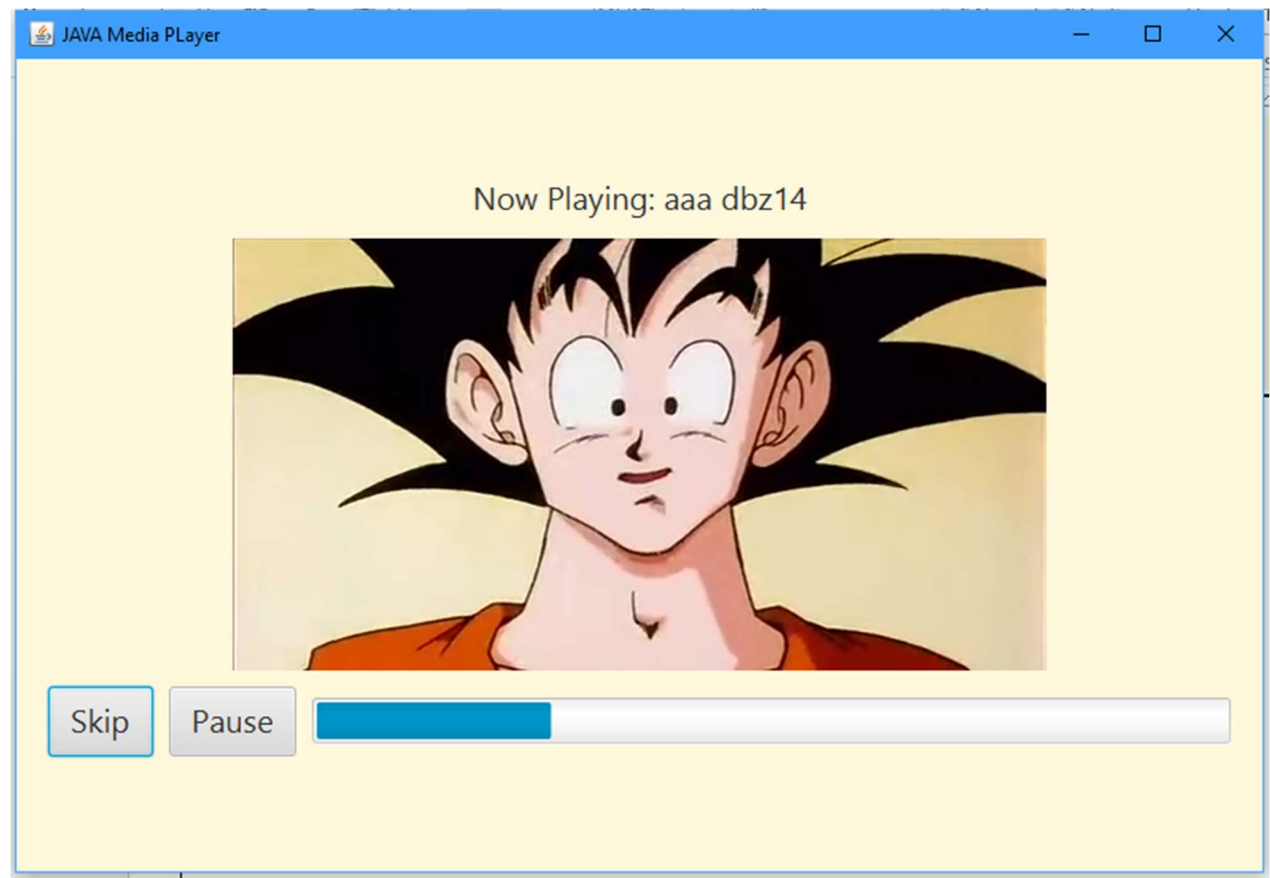


Fig 4: (Play) First Media File of the playlist [VIDEO]

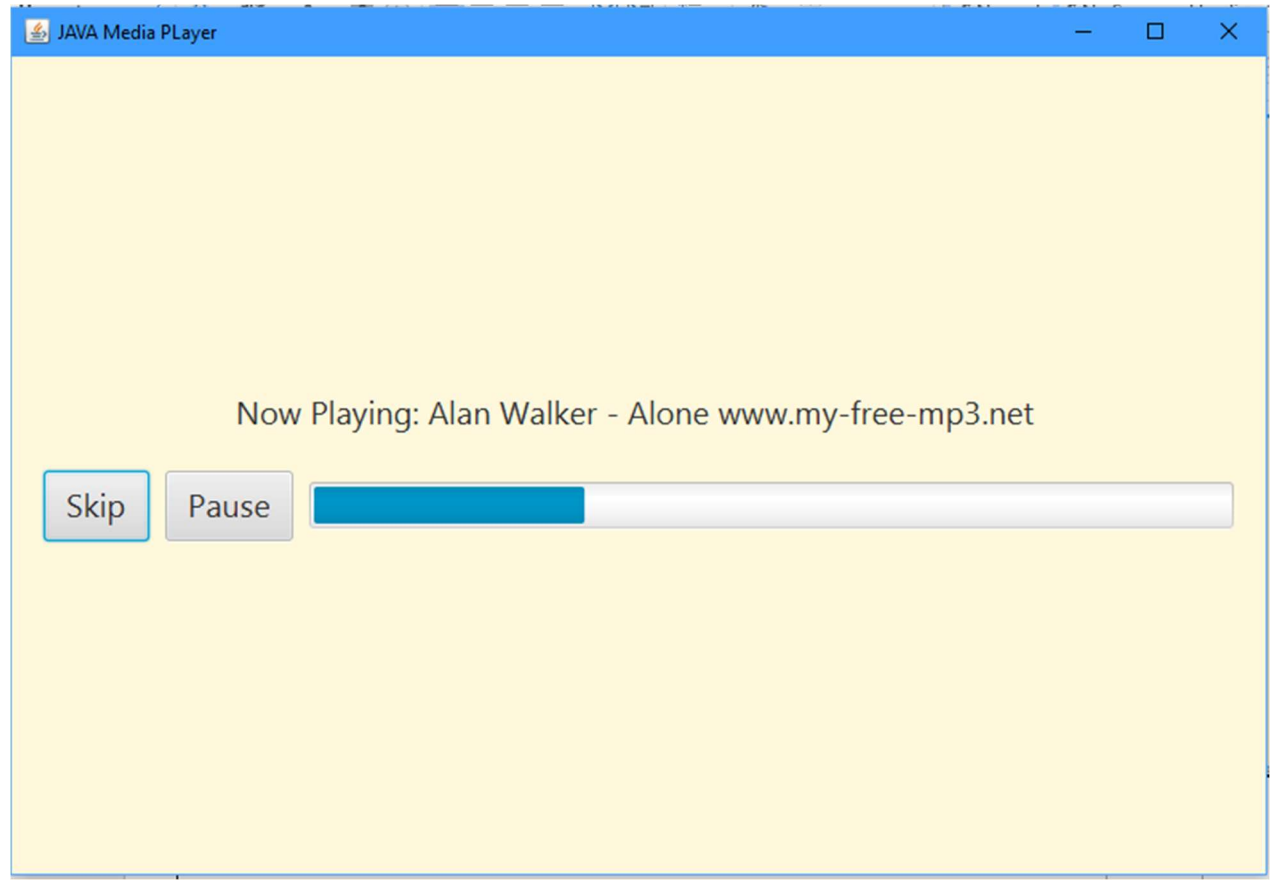


Fig 5: Second Media File of the playlist

[AUDIO]

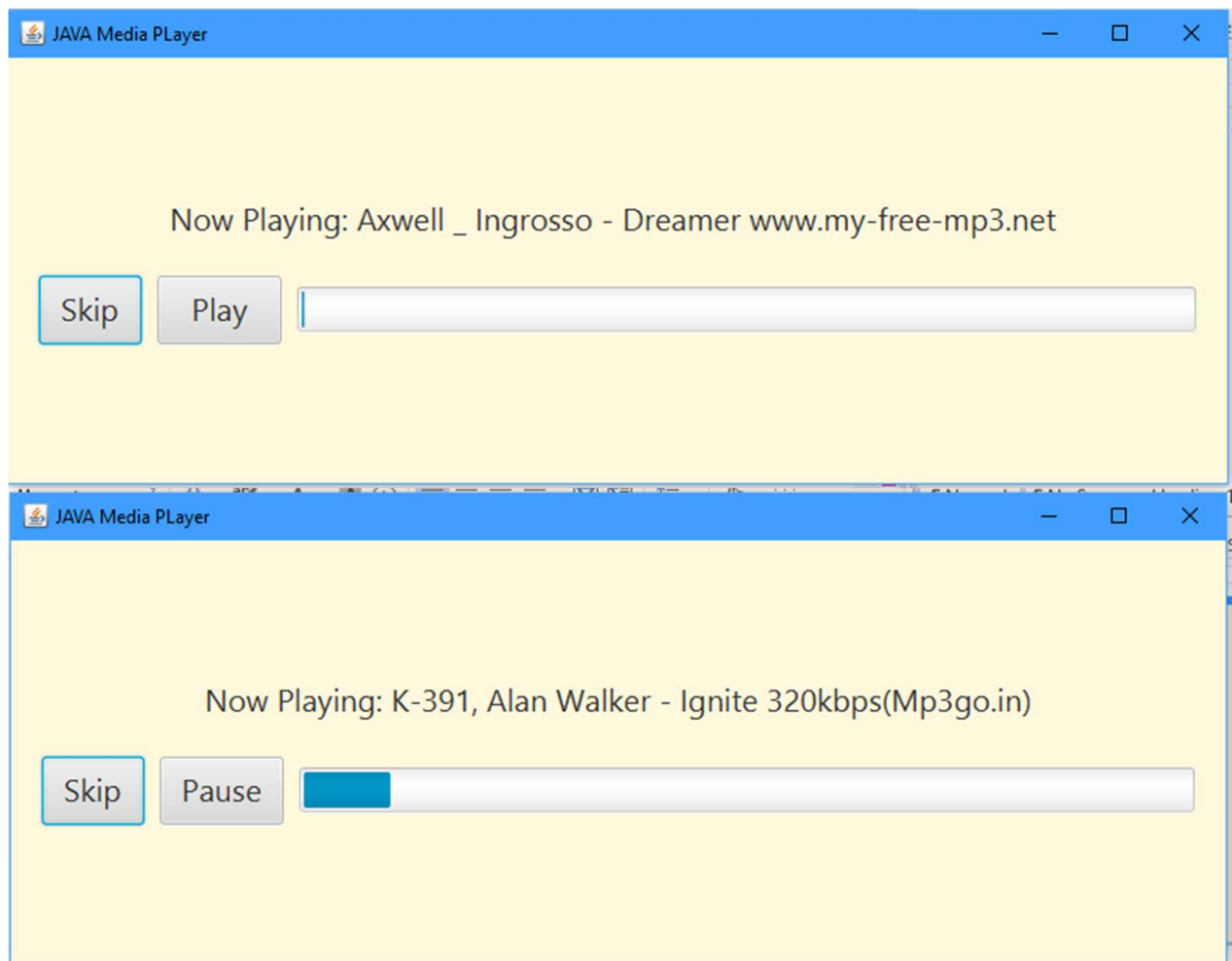


Fig 6: Third and Fourth Media File of the playlist [AUDIOS]