A Mini Project Report

On

Music Player

Submitted in partial fulfillment of requirements for the Course
CSE18R272 - JAVA PROGRAMMING

Bachelor's of Technology

In

Computer Science and Engineering

Submitted By

Patan Dilshad

9918004087

Manas Kumar Mishra

9918004063

Under the guidance of

Dr. R. RAMALAKSHMI

(Associate Professor)

Department of Computer Science and Engineering

Kalasalingam Academy of Research and Education

Anand Nagar, Krishnankoil-626126

APRIL 2020

ABSTRACT

In this Project, we created a PC Music Player using Java awt and swing package. Under this real time project, we use different packages of java, like scanner class, fileReader etc. We even created an executable .jar file and with the help of it we can directly run the project. We created different program files under this project, each having different purpose. Our project is a basic music player that at first reads the file to get the music files to play. We even add Pause and Stop button in it to pause and stop the music completely, respectively. Under this MP3 Player, we are only able to read ".wav" files. Others files, like mp3 format are not supported. Overall, its a real time Music Player Project that read files to play songs.

DECLARATION

I hereby declare that the work presented in this report entitled "**Music Player**", in partial fulfilment of the requirements for the course CSE18R272-Java Programming and submitted in **Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education** (**Deemed to be University**) is an authentic record of our own work carried out during the period from **Jan 2020** under the guidance of Mr. **Dr. R. Ramalakshmi** (Associate Professor).

The work reported in this has not been submitted by me for the award of any other degree of this or any other institute.

Patan Dilshad
9918004087
Manas Kumar Mishra
9918004063

ACKNOWLEDGEMENT

First and foremost, I wish to thank the **Almighty God** for his grace and benediction to complete this Project work successfully. I would like to convey my special thanks from the bottom of my heart to my dear **Parents** and affectionate **Family members** for their honest support for the completion of this project work

I express my deep gratitude to Mr. "kalvivallal" Thiru **T. Kalasalingam** B.Com, Founder and Chairman, "llayavallal" **Dr. K. Sridharan** Ph.d, Chancellor, **Dr. S. ShasiAnand**, Ph.d, Vice President (Academic), **Mr. S. Arjun Kalasalingam**, M.S., Vide President(Administration), **Dr.R.Nagaraj**, Vice Chancellor, **Dr.V.Vasudevan**, Ph.d, Registrar, **Dr.P.Deepalakshmi**, Ph.d, Dean(School Of Computing). And also a spcial thanks to **Dr.A.Francis Saviour Devaraj**, Head Department of CSE, Kalasalingam Academy of Research and Education for granting the permission and providing necessary facilities to carry out Project Work. .

I would like to express my special appreciation and profound thanks to my enthusiastic Project Supervisor **Dr.P.Ramalaxmi** Ph.d, Associate Professor at Kalasalingam Academy of Research and Education [KARE] fpr jer inspiring guidance, constant encouragement with my work during all stages. I am extremely glad taht I had a chance to do my Project under my guide, who truly practices and appreciates deep thinking. I will be forever indebted to my Guide for all the time he has spent with me in discussions. And during the most difficult times when writing this report, he gave me the moral support the freedom I needed to move on.

And last but not the least, I want to thank my teammate to be ny back, all the way through this project

Patan Dilshad

9918004087

Manas Kumar Mishra

9918004063

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

We created a GUI Music Player using Java awt and swing. Under this project, We used FileReader to read the ".wav" files to play it on our Music Player. It'a a basic Music Player that only contains some widgets. First widget is a play button. It first searches the ".wav" files from your computer and than it will play the music. The second widget is a pause button. As name signifies, it pause the music. But after pausing, we can play the music again by pressing directly play button. The third widget is a stop button and as usual it stops the music and if we want to play the music again, then we again have to choose the ".wav" files and play it from our computer. There is one more Slider widget under our Music Player that shows that runs with the music at real time. There are two times in this project too. One of the timer show the initial music time and the other tell the music duration.

Chapter 2

PROJECT DESCRIPTION

Under our project, we have three different code files in which we code different parts of our project. Under SwingAudioPlayer.java, we created the GUI. To create the GUI of Music Player, we used java awt and swing packages of java. Under awt, we used Dimesion, FlowLayout, Font, GridBagCOnstraints, GridBagLayout, Insets, ActionEvent, ActionListener, File, IOException and under Swing package, we used, ImageIcon, Jbutton, JFileChooser, JFrame, JLabel, JOptionPane, JPanel, JSlider, SwingUtilities, UIManager and FileFilter. We even used two sound package just for exception handling and those were, LineUnvailableException and UnspportedAudioFileException. The other file named, PlayingTimer only works with Time and Slider of the GUI Music Player. Slider that is being used under this project works on real time. Under this file, we used text, util, sound and swing package of java. Under text package, we used DateFormat and SimpleDateFormat. Under util package, we used Date and TimeZone package. Under sound package in this file, we just used Clip Package and under swing package we used JLabel and JSlider. Last but not the least, there is our last file under this project named, AudioPlayer, that contains java.io and javax.sound package. under io package, we used File and IOException and under sound package, we used, AudioFormat, AudioInputStream, AudioSystem, Clip, DataLine, LineEvent, LineListener, LineUnavailableException and finaly, UnsupportedAudioFileException. The objective of this project is just that we wanna create a simple GUI software using java that have less weight and works well in PC.

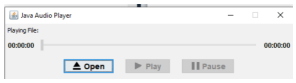

Figure 2.1: Figure Example

Chapter 3

CONCLUSION

This Project is based on GUI(Graphical User Interface) based Music Player. It displays the play, pause and stop button with a slider in it. The application is also accompanied with FileChooser which explains that how sophisticated this project is. This application could only read **.wav** files. There is also two timers specified in the application that tells the current time of the music and the duration of the music.

The future enhancement may include the addition of much UI and UX added in it and may be some animations that goes on with the music.

Appendices

SOURCE CODE

```
import java.io.File;
import java.io.IOException;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineEvent;
import javax.sound.sampled.LineListener;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.
    ↪ UnsupportedAudioFileException;


public class AudioPlayer implements LineListener {
        private static final int SECONDS_IN_HOUR = 60 *
            ↪  60;
        private static final int SECONDS_IN_MINUTE =
            ↪ 60;

        /**
         * this flag indicates whether the playback
             ↪ completes or not.
         */
        private boolean playCompleted;

        /**
         * this flag indicates whether the playback is
             ↪ stopped or not.
         */
        private boolean isStopped;

        private boolean isPaused;

        private Clip audioClip;
```

```
/**
 * Load audio file before playing back
 *
 * @param audioFilePath
 *            Path of the audio file.
 * @throws IOException
 * @throws UnsupportedAudioFileException
 * @throws LineUnavailableException
 */
public void load(String audioFilePath)
                throws
                    ↪ UnsupportedAudioFileException
                    ↪ , IOException,
                LineUnavailableException {
        File audioFile = new File(audioFilePath
            ↪ );

        AudioInputStream audioStream =
            ↪ AudioSystem
                        .getAudioInputStream(
                            ↪ audioFile);

        AudioFormat format = audioStream.
            ↪ getFormat();

        DataLine.Info info = new DataLine.Info(
            ↪ Clip.class, format);

        audioClip = (Clip) AudioSystem.getLine(
            ↪ info);

        audioClip.addLineListener(this);

        audioClip.open(audioStream);
}

public long getClipSecondLength() {
        return audioClip.getMicrosecondLength()
            ↪  / 1_000_000;
}
```

```
public String getClipLengthString() {
        String length = "";
        long hour = 0;
        long minute = 0;
        long seconds = audioClip.
           ↪ getMicrosecondLength() / 1
           ↪ _000_000;

        System.out.println(seconds);

        if (seconds >= SECONDS_IN_HOUR) {
                hour = seconds /
                   ↪ SECONDS_IN_HOUR;
                length = String.format("%02d:",
                   ↪  hour);
        } else {
                length += "00:";
        }

        minute = seconds - hour *
           ↪ SECONDS_IN_HOUR;
        if (minute >= SECONDS_IN_MINUTE) {
                minute = minute /
                   ↪ SECONDS_IN_MINUTE;
                length += String.format("%02d:"
                   ↪ , minute);

        } else {
                minute = 0;
                length += "00:";
        }

        long second = seconds - hour *
           ↪ SECONDS_IN_HOUR - minute *
           ↪ SECONDS_IN_MINUTE;

        length += String.format("%02d", second)
           ↪ ;
```

```
        return length;
    }

    /**
     * Play a given audio file.
     *
     * @throws IOException
     * @throws UnsupportedAudioFileException
     * @throws LineUnavailableException
     */
    void play() throws IOException {

        audioClip.start();

        playCompleted = false;
        isStopped = false;

        while (!playCompleted) {
            // wait for the playback
               ↪ completes
            try {
                Thread.sleep(1000);
            } catch (InterruptedException
               ↪ ex) {
                ex.printStackTrace();
                if (isStopped) {
                    audioClip.stop
                       ↪ ();
                    break;
                }
                if (isPaused) {
                    audioClip.stop
                       ↪ ();
                } else {
                    System.out.
                       ↪ println("
                       ↪ !!!!");
                    audioClip.start
                       ↪ ();
                }
```

```
                }
            }

            audioClip.close();

        }

        /**
         * Stop playing back.
         */
        public void stop() {
            isStopped = true;
        }

        public void pause() {
            isPaused = true;
        }

        public void resume() {
            isPaused = false;
        }

        /**
         * Listens to the audio line events to know
            ↪ when the playback completes.
         */
        @Override
        public void update(LineEvent event) {
            LineEvent.Type type = event.getType();
            if (type == LineEvent.Type.STOP) {
                    System.out.println("STOP_EVENT"
                        ↪ );
                    if (isStopped || !isPaused) {
                            playCompleted = true;
                    }
            }
        }

        public Clip getAudioClip() {
            return audioClip;
```

```
        }
}
```

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

import javax.sound.sampled.Clip;
import javax.swing.JLabel;
import javax.swing.JSlider;


public class PlayingTimer extends Thread {
        private DateFormat dateFormater = new
           ↪ SimpleDateFormat("HH:mm:ss");
        private boolean isRunning = false;
        private boolean isPause = false;
        private boolean isReset = false;
        private long startTime;
        private long pauseTime;

        private JLabel labelRecordTime;
        private JSlider slider;
        private Clip audioClip;

        public void setAudioClip(Clip audioClip) {
                this.audioClip = audioClip;
        }

        PlayingTimer(JLabel labelRecordTime, JSlider
           ↪ slider) {
                this.labelRecordTime = labelRecordTime;
                this.slider = slider;
        }

        public void run() {
                isRunning = true;
```

```
                startTime = System.currentTimeMillis();

                while (isRunning) {
                        try {
                                Thread.sleep(100);
                                if (!isPause) {
                                        if (audioClip
                                            ↪ != null
                                            ↪ &&
                                            ↪ audioClip
                                            ↪ .
                                            ↪ isRunning
                                            ↪ ()) {
                                                labelRecordTime
                                                    ↪ .
                                                    ↪ setText
                                                    ↪ (
                                                    ↪ toTimeString
                                                    ↪ ()
                                                    ↪ )
                                                    ↪ ;
                                                int
                                                    ↪ currentSecond
                                                    ↪
                                                    ↪ =
                                                    ↪
                                                    ↪ (
                                                    ↪ int
                                                    ↪ )
                                                    ↪
                                                    ↪ audioClip
                                                    ↪ .
                                                    ↪ getMicrosecondPositio
                                                    ↪ ()
                                                    ↪
                                                    ↪ /
                                                    ↪
                                                    ↪ 1
                                                    ↪ _000_000
                                                    ↪ ;
```

```
                                        slider.
                                            ↪ setValue
                                            ↪ (
                                            ↪ currentSecond
                                            ↪ )
                                            ↪ ;
                                    }
                                } else {
                                    pauseTime +=
                                        ↪ 100;
                                }
                            } catch (InterruptedException
                                ↪ ex) {
                                ex.printStackTrace();
                                if (isReset) {
                                    slider.setValue
                                        ↪ (0);
                                    labelRecordTime
                                        ↪ .setText(
                                        ↪ "00:00:00
                                        ↪ ");
                                    isRunning =
                                        ↪ false;
                                        ↪
                                        ↪
                                    break;
                                }
                            }
                        }
                    }


                    /**
                     * Reset counting to "00:00:00"
                     */
                    void reset() {
                            isReset = true;
                            isRunning = false;
                    }
```

```
        void pauseTimer() {
                isPause = true;
        }

        void resumeTimer() {
                isPause = false;
        }

        /**
         * Generate a String for time counter in the
            ↪ format of "HH:mm:ss"
         * @return the time counter
         */
        private String toTimeString() {
                long now = System.currentTimeMillis();
                Date current = new Date(now - startTime
                   ↪  - pauseTime);
                dateFormater.setTimeZone(TimeZone.
                   ↪ getTimeZone("GMT"));
                String timeCounter = dateFormater.
                   ↪ format(current);
                return timeCounter;
        }
}
```

```
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;

import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.
   ↪ UnsupportedAudioFileException;
import javax.swing.ImageIcon;
```

```
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.filechooser.FileFilter;


public class SwingAudioPlayer extends JFrame implements
    ↪ ActionListener {
        private AudioPlayer player = new AudioPlayer();
        private Thread playbackThread;
        private PlayingTimer timer;

        private boolean isPlaying = false;
        private boolean isPause = false;

        private String audioFilePath;
        private String lastOpenPath;

        private JLabel labelFileName = new JLabel("
            ↪ Playing␣File:");
        private JLabel labelTimeCounter = new JLabel("
            ↪ 00:00:00");
        private JLabel labelDuration = new JLabel("
            ↪ 00:00:00");

        private JButton buttonOpen = new JButton("Open"
            ↪ );
        private JButton buttonPlay = new JButton("Play"
            ↪ );
        private JButton buttonPause = new JButton("
            ↪ Pause");

        private JSlider sliderTime = new JSlider();
```

```
// Icons used for buttons
private ImageIcon iconOpen = new ImageIcon(
    ↪ getClass().getResource(
                "/net/codejava/audio/images/
                    ↪ Open.png"));
private ImageIcon iconPlay = new ImageIcon(
    ↪ getClass().getResource(
                "/net/codejava/audio/images/
                    ↪ Play.gif"));
private ImageIcon iconStop = new ImageIcon(
    ↪ getClass().getResource(
                "/net/codejava/audio/images/
                    ↪ Stop.gif"));
private ImageIcon iconPause = new ImageIcon(
    ↪ getClass().getResource(
                "/net/codejava/audio/images/
                    ↪ Pause.png"));


public SwingAudioPlayer() {
        super("Java␣Audio␣Player");
        setLayout(new GridBagLayout());
        GridBagConstraints constraints = new
            ↪ GridBagConstraints();
        constraints.insets = new Insets(5, 5,
            ↪ 5, 5);
        constraints.anchor = GridBagConstraints
            ↪ .WEST;

        buttonOpen.setFont(new Font("Sans",
            ↪ Font.BOLD, 14));
        buttonOpen.setIcon(iconOpen);

        buttonPlay.setFont(new Font("Sans",
            ↪ Font.BOLD, 14));
        buttonPlay.setIcon(iconPlay);
        buttonPlay.setEnabled(false);

        buttonPause.setFont(new Font("Sans",
            ↪ Font.BOLD, 14));
```

```
buttonPause.setIcon(iconPause);
buttonPause.setEnabled(false);

labelTimeCounter.setFont(new Font("Sans
    ↪ ", Font.BOLD, 12));
labelDuration.setFont(new Font("Sans",
    ↪ Font.BOLD, 12));

sliderTime.setPreferredSize(new
    ↪ Dimension(400, 20));
sliderTime.setEnabled(false);
sliderTime.setValue(0);

constraints.gridx = 0;
constraints.gridy = 0;
constraints.gridwidth = 3;
add(labelFileName, constraints);

constraints.anchor = GridBagConstraints
    ↪ .CENTER;
constraints.gridy = 1;
constraints.gridwidth = 1;
add(labelTimeCounter, constraints);

constraints.gridx = 1;
add(sliderTime, constraints);

constraints.gridx = 2;
add(labelDuration, constraints);

JPanel panelButtons = new JPanel(new
    ↪ FlowLayout(FlowLayout.CENTER, 20,
    ↪  5));
panelButtons.add(buttonOpen);
panelButtons.add(buttonPlay);
panelButtons.add(buttonPause);

constraints.gridwidth = 3;
constraints.gridx = 0;
constraints.gridy = 2;
```

```
            add(panelButtons, constraints);

            buttonOpen.addActionListener(this);
            buttonPlay.addActionListener(this);
            buttonPause.addActionListener(this);

            pack();
            setResizable(false);
            setDefaultCloseOperation(JFrame.
                ↪ EXIT_ON_CLOSE);
            setLocationRelativeTo(null);
    }

    /**
     * Handle click events on the buttons.
     */
    @Override
    public void actionPerformed(ActionEvent event)
        ↪ {
            Object source = event.getSource();
            if (source instanceof JButton) {
                    JButton button = (JButton)
                        ↪ source;
                    if (button == buttonOpen) {
                            openFile();
                    } else if (button == buttonPlay
                        ↪ ) {
                            if (!isPlaying) {
                                    playBack();
                            } else {
                                    stopPlaying();
                            }
                    } else if (button ==
                        ↪ buttonPause) {
                            if (!isPause) {
                                    pausePlaying();
                            } else {
                                    resumePlaying()
                                        ↪ ;
                            }
```

```
                }
            }
        }

        private void openFile() {
            JFileChooser fileChooser = null;

            if (lastOpenPath != null &&
               ↪ lastOpenPath.equals("")) {
                fileChooser = new JFileChooser(
                   ↪ lastOpenPath);
            } else {
                fileChooser = new JFileChooser
                   ↪ ();
            }

            FileFilter wavFilter = new FileFilter()
               ↪ {
                @Override
                public String getDescription()
                   ↪ {
                    return "Sound file (*.
                       ↪ WAV)";
                }

                @Override
                public boolean accept(File file
                   ↪ ) {
                    if (file.isDirectory())
                       ↪ {
                        return true;
                    } else {
                        return file.
                           ↪ getName()
                           ↪ .
                           ↪ toLowerCase
                           ↪ ().
                           ↪ endsWith(
                           ↪ ".wav");
                    }
```

```
            }
        };


        fileChooser.setFileFilter(wavFilter);
        fileChooser.setDialogTitle("Open Audio
            ↪ File");
        fileChooser.setAcceptAllFileFilterUsed(
            ↪ false);

        int userChoice = fileChooser.
            ↪ showOpenDialog(this);
        if (userChoice == JFileChooser.
            ↪ APPROVE_OPTION) {
            audioFilePath = fileChooser.
                ↪ getSelectedFile().
                ↪ getAbsolutePath();
            lastOpenPath = fileChooser.
                ↪ getSelectedFile().
                ↪ getParent();
            if (isPlaying || isPause) {
                stopPlaying();
                while (player.
                    ↪ getAudioClip().
                    ↪ isRunning()) {
                    try {
                        Thread.
                            ↪ sleep
                            ↪ (100)
                            ↪ ;
                    } catch (
                        ↪ InterruptedException
                        ↪  ex) {
                        ex.
                            ↪ printStackTrace
                            ↪ ()
                            ↪ ;
                    }
                }
            }
```

```
                playBack();
            }
        }

        /**
         * Start playing back the sound.
         */
        private void playBack() {
            timer = new PlayingTimer(
                ↪ labelTimeCounter, sliderTime);
            timer.start();
            isPlaying = true;
            playbackThread = new Thread(new
                ↪ Runnable() {

                    @Override
                    public void run() {
                        try {

                            buttonPlay.
                                ↪ setText("
                                ↪ Stop");
                            buttonPlay.
                                ↪ setIcon(
                                ↪ iconStop)
                                ↪ ;
                            buttonPlay.
                                ↪ setEnabled
                                ↪ (true);

                            buttonPause.
                                ↪ setText("
                                ↪ Pause");
                            buttonPause.
                                ↪ setEnabled
                                ↪ (true);

                            player.load(
                                ↪ audioFilePath
                                ↪ );
```

```
                timer.
                    ↪ setAudioClip
                    ↪ (player.
                    ↪ getAudioClip
                    ↪ ());
                labelFileName.
                    ↪ setText("
                    ↪ Playing␣
                    ↪ File:␣" +
                    ↪
                    ↪ audioFilePath
                    ↪ );
                sliderTime.
                    ↪ setMaximum
                    ↪ ((int)
                    ↪ player.
                    ↪ getClipSecondLength
                    ↪ ());

                labelDuration.
                    ↪ setText(
                    ↪ player.
                    ↪ getClipLengthString
                    ↪ ());
                player.play();

                resetControls()
                    ↪ ;

            } catch (
                ↪ UnsupportedAudioFileException
                ↪  ex) {
                    JOptionPane.
                        ↪ showMessageDialog
                        ↪ (
                        ↪ SwingAudioPlayer
                        ↪ .this,
                                    "
                                        ↪ The
                                        ↪ ␣
```

```
                                                    ↪ audio
                                                    ↪ ␣
                                                    ↪ format
                                                    ↪ ␣
                                                    ↪ is
                                                    ↪ ␣
                                                    ↪ unsupported
                                                    ↪ !
                                                    ↪ "
                                                    ↪ ,
                                                    ↪
                                                    ↪ "
                                                    ↪ Error
                                                    ↪ "
                                                    ↪ ,
                                                    ↪
                                                    ↪ JOptionPane
                                                    ↪ .
                                                    ↪ ERROR_MESSAGE
                                                    ↪ )
                                                    ↪ ;
                                                    ↪
                                resetControls()
                                    ↪ ;
                                ex.
                                    ↪ printStackTrace
                                    ↪ ();
                            } catch (
                                ↪ LineUnavailableException
                                ↪  ex) {
                                JOptionPane.
                                    ↪ showMessageDialog
                                    ↪ (
                                    ↪ SwingAudioPlayer
                                    ↪ .this,
                                                    "
                                                    ↪ Could
                                                    ↪ ␣
                                                    ↪ not
                                                    ↪ ␣
```

```
                                                    ↪ play
                                                    ↪ ␣
                                                    ↪ the
                                                    ↪ ␣
                                                    ↪ audio
                                                    ↪ ␣
                                                    ↪ file
                                                    ↪ ␣
                                                    ↪ because
                                                    ↪ ␣
                                                    ↪ line
                                                    ↪ ␣
                                                    ↪ is
                                                    ↪ ␣
                                                    ↪ unavailable
                                                    ↪ !
                                                    ↪ "
                                                    ↪ ,
                                                    ↪
                                                    ↪ "
                                                    ↪ Error
                                                    ↪ "
                                                    ↪ ,
                                                    ↪
                                                    ↪ JOptionPane
                                                    ↪ .
                                                    ↪ ERROR_MESSAGE
                                                    ↪ )
                                                    ↪ ;
                                                    ↪
                        resetControls()
                            ↪ ;
                        ex.
                            ↪ printStackTrace
                            ↪ ();
                } catch (IOException ex
                    ↪ ) {
                        JOptionPane.
                            ↪ showMessageDialog
                            ↪ (
```

```
                    ↪ SwingAudioPlayer
                    ↪ .this,
                                "
                                    ↪ I
                                    ↪ /
                                    ↪ O
                                    ↪ ␣
                                    ↪ error
                                    ↪ ␣
                                    ↪ while
                                    ↪ ␣
                                    ↪ playing
                                    ↪ ␣
                                    ↪ the
                                    ↪ ␣
                                    ↪ audio
                                    ↪ ␣
                                    ↪ file
                                    ↪ !
                                    ↪ "
                                    ↪ ,
                                    ↪
                                    ↪ "
                                    ↪ Error
                                    ↪ "
                                    ↪ ,
                                    ↪
                                    ↪ JOptionPane
                                    ↪ .
                                    ↪ ERROR_MESSAGE
                                    ↪ )
                                    ↪ ;
                                    ↪
                resetControls()
                    ↪ ;
                ex.
                    ↪ printStackTrace
                    ↪ ();
            }
```

```
                }
            });

            playbackThread.start();
        }

        private void stopPlaying() {
            isPause = false;
            buttonPause.setText("Pause");
            buttonPause.setEnabled(false);
            timer.reset();
            timer.interrupt();
            player.stop();
            playbackThread.interrupt();
        }

        private void pausePlaying() {
            buttonPause.setText("Resume");
            isPause = true;
            player.pause();
            timer.pauseTimer();
            playbackThread.interrupt();
        }

        private void resumePlaying() {
            buttonPause.setText("Pause");
            isPause = false;
            player.resume();
            timer.resumeTimer();
            playbackThread.interrupt();
                ↪
        }

        private void resetControls() {
            timer.reset();
            timer.interrupt();

            buttonPlay.setText("Play");
            buttonPlay.setIcon(iconPlay);
```

```
                buttonPause.setEnabled(false);

                isPlaying = false;
        }

        /**
         * Launch the program
         */
        public static void main(String[] args) {

                try {
                        UIManager.setLookAndFeel(
                           ↪ UIManager.
                           ↪ getSystemLookAndFeelClassName
                           ↪ ());
                } catch (Exception ex) {
                        ex.printStackTrace();
                }

                SwingUtilities.invokeLater(new Runnable
                   ↪ () {

                        @Override
                        public void run() {
                                new SwingAudioPlayer().
                                   ↪ setVisible(true);
                        }
                });
        }


}
```