Developing an MP3 Music Player Application with JavaFX

ABSTRACT

This project involves the development of an MP3 music player application using JavaFX technology.

The application provides users with a user-friendly interface to browse and play their MP3 music collection. The player features basic playback controls such as play, pause, stop, and next/previous track buttons, as well as volume and track progress sliders.

The project also utilizes Object-Oriented Programming concepts to ensure code modularity and maintainability. The application is built on the Model-View-Controller (MVC) design pattern, which separates the application into three components for better organization and flexibility.

The completed project showcases the power and versatility of JavaFX technology in developing modern and user-friendly applications.

KEYWORDS

JavaFX, MP3 Music Player, User Interface, Object-Oriented Programming, Mode-View-Controller, Code modularity, Maintainability.

INTRODUCTION

The widespread use of digital audio formats and the increasing availability of music on the internet have made music players an essential tool for listening to music. The purpose of this project is to develop an MP3 music player application using JavaFX technology. JavaFX is a powerful and versatile platform for building rich, interactive user interfaces for desktop applications. The project involves designing and implementing a user-friendly interface for browsing and playing MP3 music files. The application will include basic playback controls such as play, pause, stop, and next/previous track buttons, as well as volume and track progress sliders. The user interface will display song information to enhance the user's listening experience.

The project will also utilize Object-Oriented Programming concepts to ensure code modularity and maintainability. The completed project will showcase the power and versatility of JavaFX technology in developing modern and user-friendly applications.

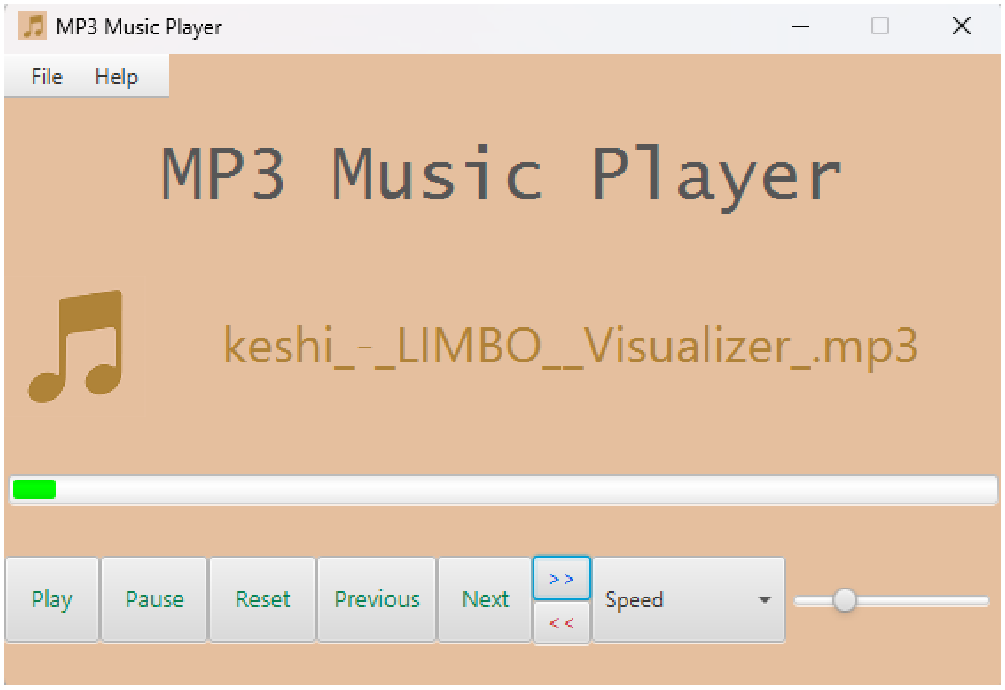
Requirements

The minimum requirements for running the software required to run the MP3 Music Player listed below:

- Microsoft Windows 7/8/10/11
- IntelliJ IDEA / Eclipse
- Scene builder
- JDK 20
- JavaFX 20

USER INTERFACE

The opening user interface (UI) would look like the figure given below:



USER GUIDELINES

1.Music Selection

We can either play the default playlist or select any audio files stored in our system. There are two methods to accomplish this task.

Firstly, if we want to play the default playlist, after opening the software/application we only need to press the ***Play*** button to play an audio file which is already included and appears first in the playlist.

Secondly, if we want to select a specific audio file and play the audio, we need to go to the menu bar and select ***File*** >> ***Open*** >> select "specific audio file" >> ***Open***.

2.Add to Playlist

To add an audio file to our playlist, we need go to the menu bar and select ***File*** >> ***Open & add to Playlist*** >> select "specific audio file" >> ***Open***.

3.Basic playback controls

- ***Play***: This button initiates the media player & the audio file starts to play.
- ***Pause***: We can use this button to pause the playing audio file.
- ***Reset***: This button will take the media player to the starting point of the playing audio file.

- ***Previous***: To select the previous audio file from the playlist.
- ***Next***: To select the next audio file from the playlist.
- **>>**: This button will skip forward 10 seconds of the playing audio file.
- **<<**: This button will skip backward 10 seconds of the playing audio file.
- ***Speed***: This combo box provides options for selecting the playing speed of the running audio file.

4.Volume Control

There is a horizontal slider to control the volume of the music player at users' comfort. The slider ranges from 0 to 200 level and the default are 100.

5.Further Information

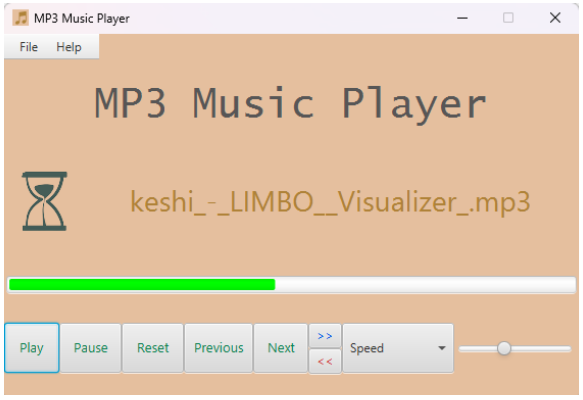
To know about the developer information about this MP3 Music Player, select ***Help*** >> ***About***.

6.Closing the player

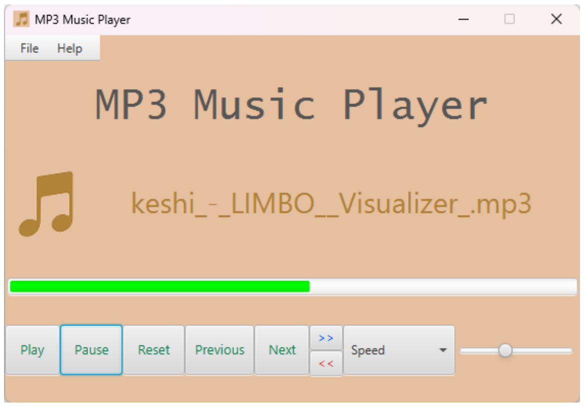
To close or exit from the MP3 Music Player, either select the top right cross(x) icon or select ***File*** >> ***Exit***.

UI MODULES

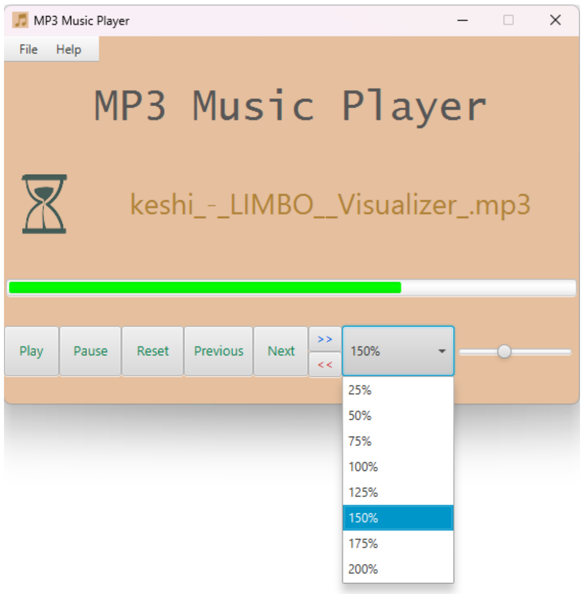
- At the time of playing music:


MP3 Music Player
File Help
MP3 Music Player
keshi_-_LIMBO__Visualizer_.mp3
Play Pause Reset Previous Next >> << Speed

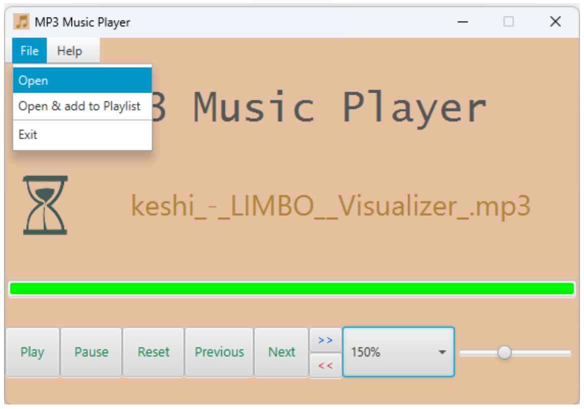
- At the time of pause state:

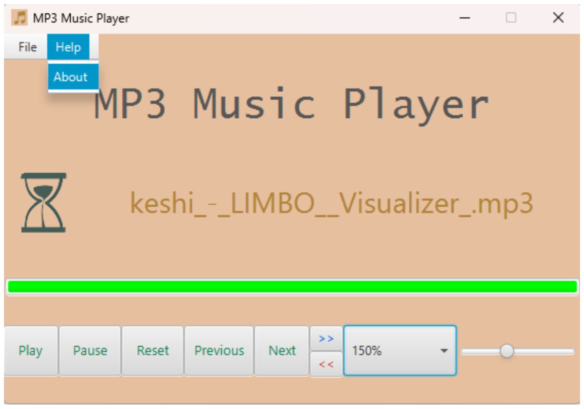

MP3 Music Player
File Help
MP3 Music Player
keshi_-_LIMBO__Visualizer_.mp3
Play Pause Reset Previous Next >> << Speed

- Speed manipulation:

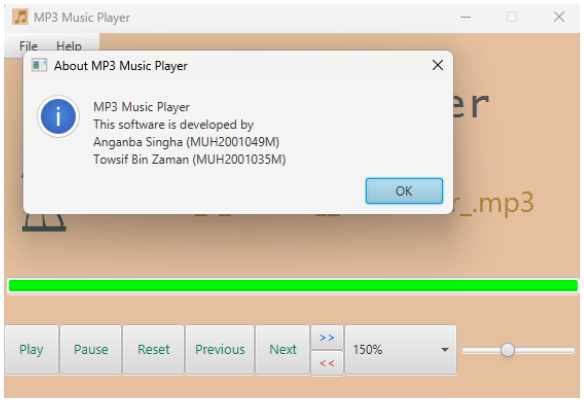

MP3 Music Player
File Help
MP3 Music Player
keshi_-_LIMBO__Visualizer_.mp3
Play Pause Reset Previous Next >> << 150%
25%
50%
75%
100%
125%
150%
175%
200%

- Menu bar:


MP3 Music Player
File Help
Open
Open & add to Playlist
Exit
3 Music Player
keshi_-_LIMBO__Visualizer_.mp3
Play Pause Reset Previous Next >> << 150%


MP3 Music Player
File Help
About
MP3 Music Player
keshi_-_LIMBO__Visualizer_.mp3
Play Pause Reset Previous Next >> << 150%

- About information:

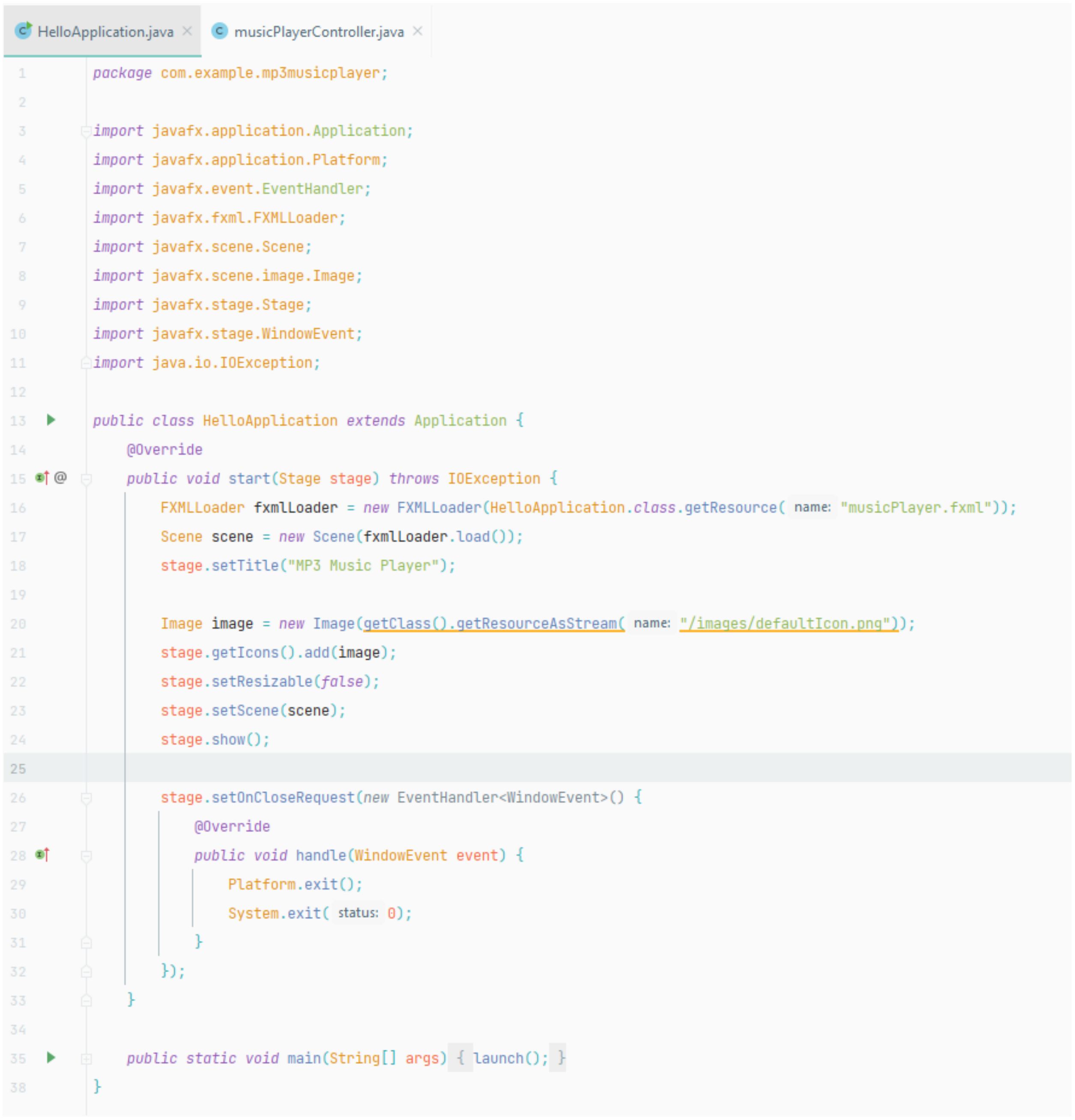


MP3 Music Player
This software is developed by
Anganba Singha (MUH2001049M)
Towsif Bin Zaman (MUH2001035M)

SOURCE CODE

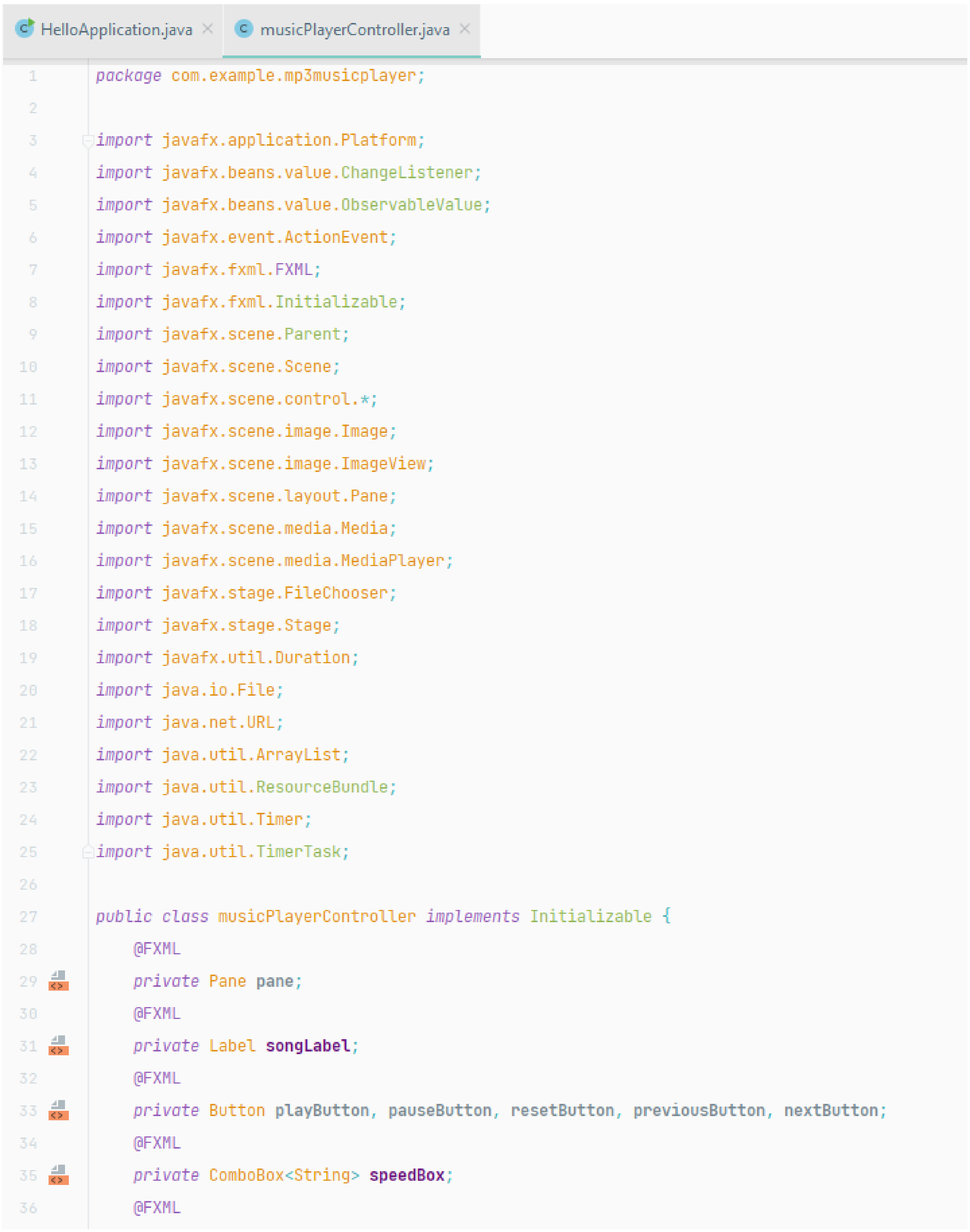
Administrative module:

1. HelloApplication.java

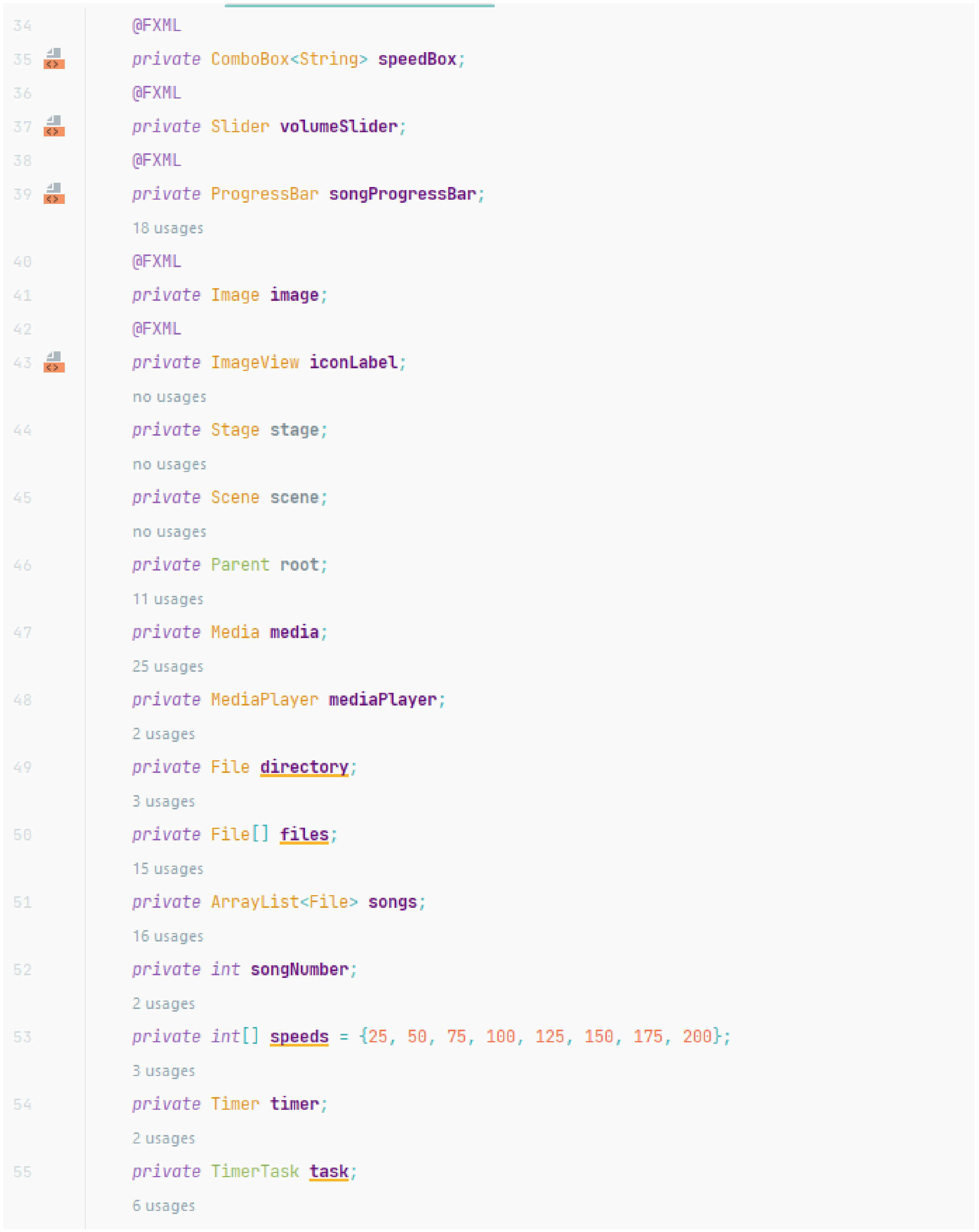


```java
package com.example.mp3musicplayer;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import java.io.IOException;

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "musicPlayer.fxml"));
        Scene scene = new Scene(fxmlLoader.load());
        stage.setTitle("MP3 Music Player");

        Image image = new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
        stage.getIcons().add(image);
        stage.setResizable(false);
        stage.setScene(scene);
        stage.show();

        stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
            @Override
            public void handle(WindowEvent event) {
                Platform.exit();
                System.exit( status: 0);
            }
        });
    }

    public static void main(String[] args) { launch(); }
}
```

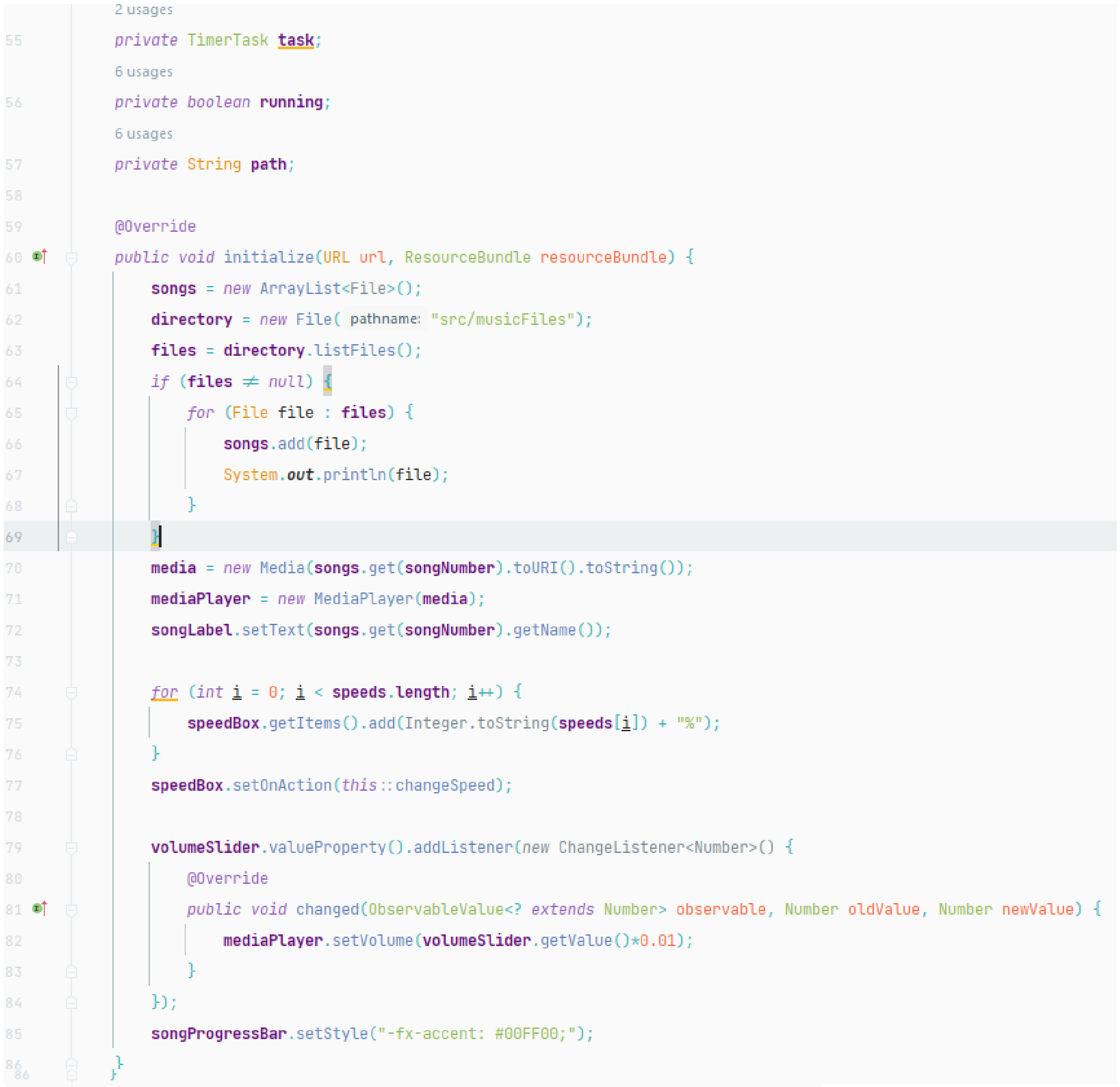
2.musicPlayerController.java



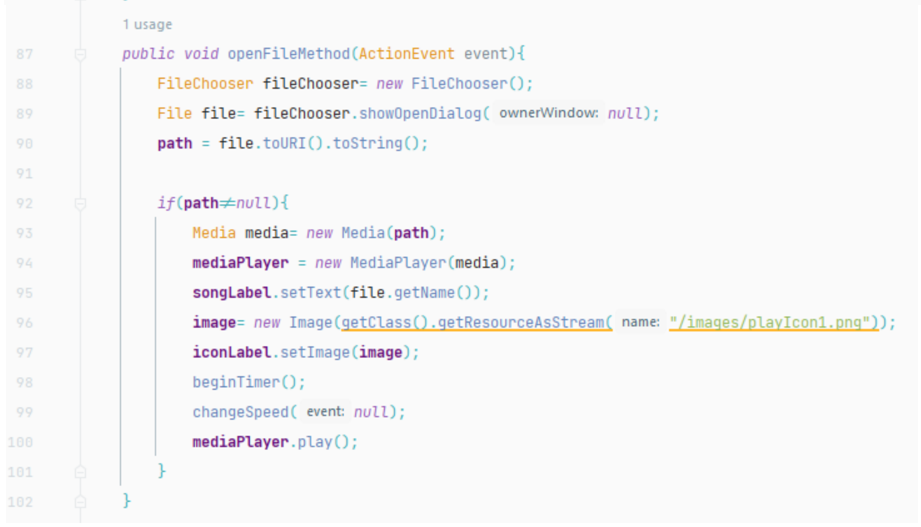
```java
package com.example.mp3musicplayer;

import javafx.application.Platform;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.util.Duration;
import java.io.File;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import java.util.Timer;
import java.util.TimerTask;

public class musicPlayerController implements Initializable {
    @FXML
    private Pane pane;
    @FXML
    private Label songLabel;
    @FXML
    private Button playButton, pauseButton, resetButton, previousButton, nextButton;
    @FXML
    private ComboBox<String> speedBox;
    @FXML
```



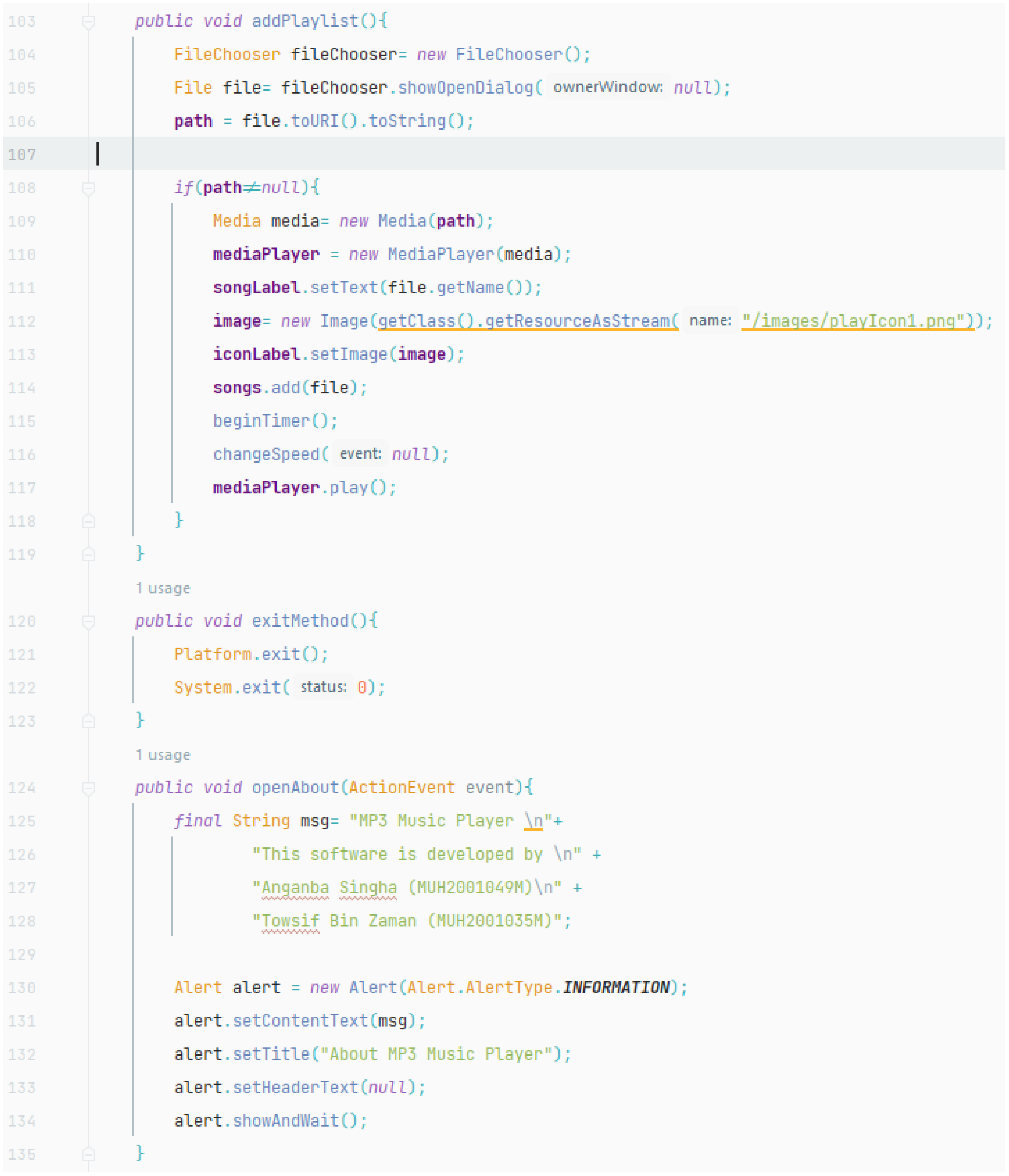
```java
    @FXML
    private ComboBox<String> speedBox;
    @FXML
    private Slider volumeSlider;
    @FXML
    private ProgressBar songProgressBar;
    18 usages
    @FXML
    private Image image;
    @FXML
    private ImageView iconLabel;
    no usages
    private Stage stage;
    no usages
    private Scene scene;
    no usages
    private Parent root;
    11 usages
    private Media media;
    25 usages
    private MediaPlayer mediaPlayer;
    2 usages
    private File directory;
    3 usages
    private File[] files;
    15 usages
    private ArrayList<File> songs;
    16 usages
    private int songNumber;
    2 usages
    private int[] speeds = {25, 50, 75, 100, 125, 150, 175, 200};
    3 usages
    private Timer timer;
    2 usages
    private TimerTask task;
    6 usages
```



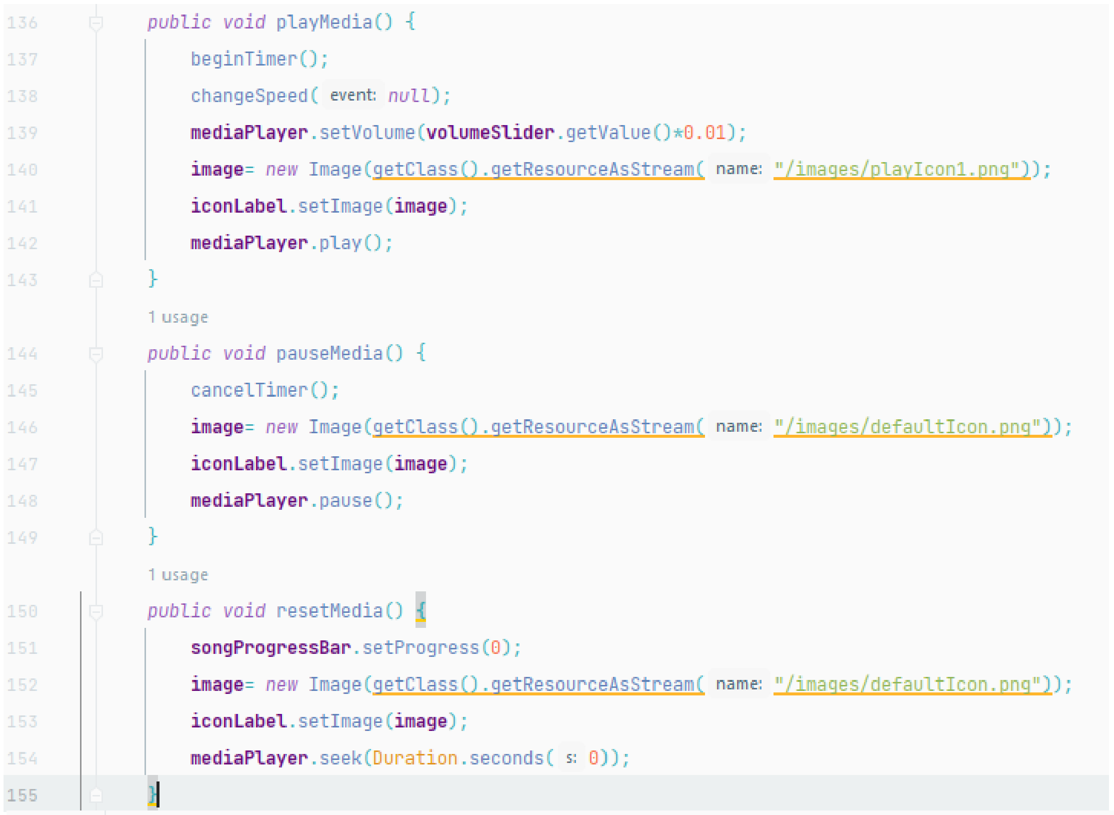
```java
    2 usages
    private TimerTask task;
    6 usages
    private boolean running;
    6 usages
    private String path;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        songs = new ArrayList<File>();
        directory = new File( pathname: "src/musicFiles");
        files = directory.listFiles();
        if (files != null) {
            for (File file : files) {
                songs.add(file);
                System.out.println(file);
            }
        }
        media = new Media(songs.get(songNumber).toURI().toString());
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(songs.get(songNumber).getName());

        for (int i = 0; i < speeds.length; i++) {
            speedBox.getItems().add(Integer.toString(speeds[i]) + "%");
        }
        speedBox.setOnAction(this::changeSpeed);

        volumeSlider.valueProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> observable, Number oldValue, Number newValue) {
                mediaPlayer.setVolume(volumeSlider.getValue()*0.01);
            }
        });
        songProgressBar.setStyle("-fx-accent: #00FF00;");
    }
```



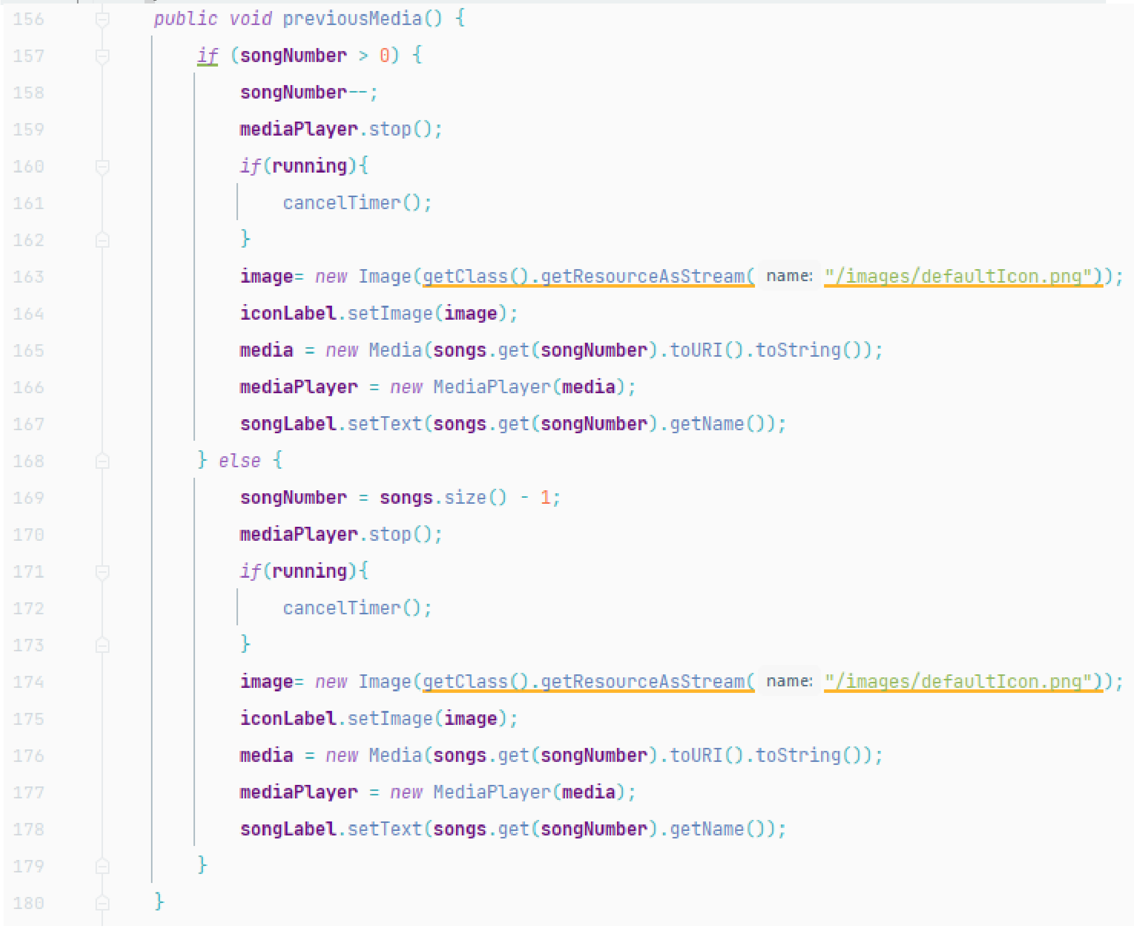
```java
    1 usage
    public void openFileMethod(ActionEvent event){
        FileChooser fileChooser= new FileChooser();
        File file= fileChooser.showOpenDialog( ownerWindow: null);
        path = file.toURI().toString();

        if(path!=null){
            Media media= new Media(path);
            mediaPlayer = new MediaPlayer(media);
            songLabel.setText(file.getName());
            image= new Image(getClass().getResourceAsStream( name: "/images/playIcon1.png"));
            iconLabel.setImage(image);
            beginTimer();
            changeSpeed( event: null);
            mediaPlayer.play();
        }
    }
```



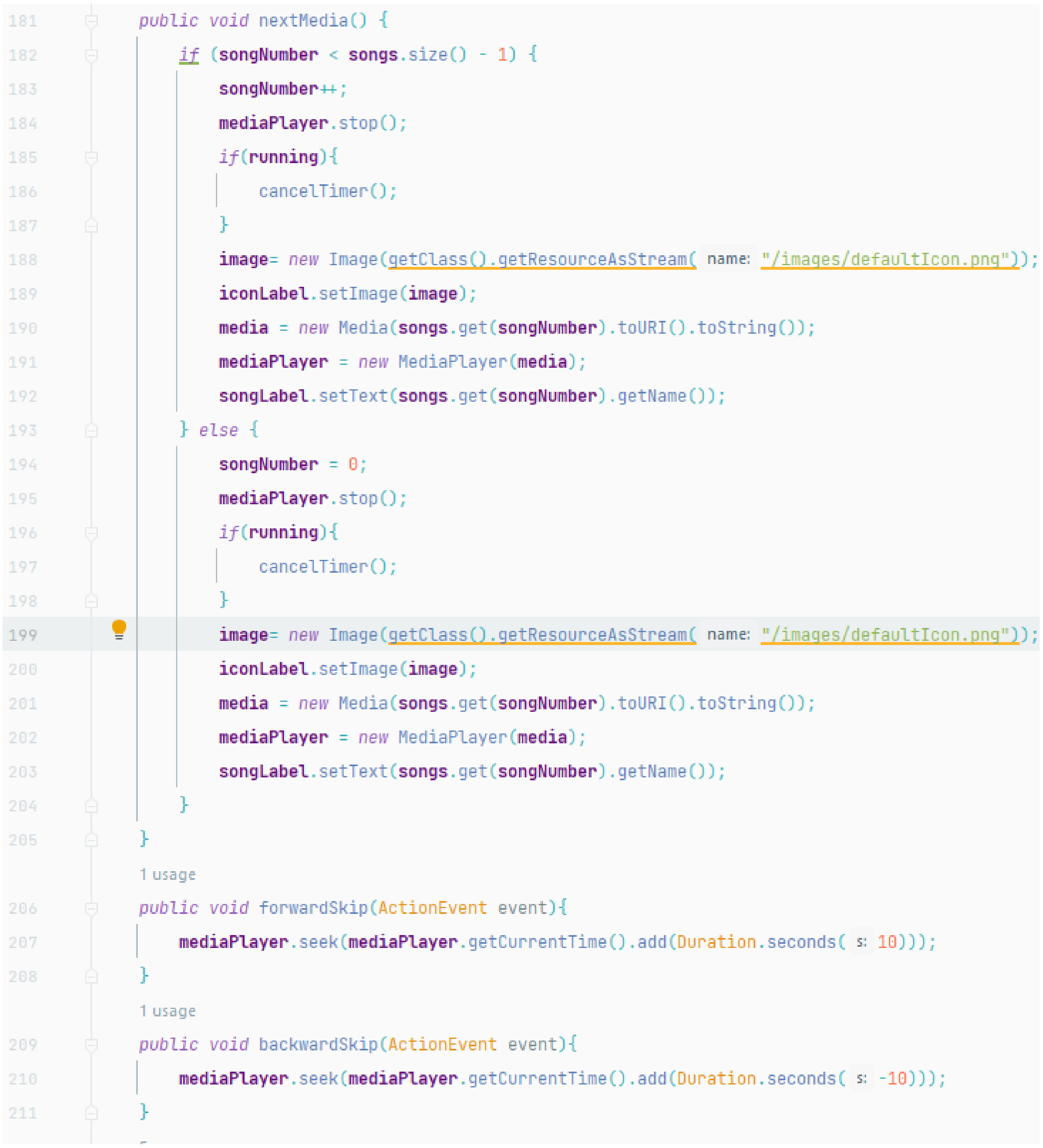
```
public void addPlaylist(){
    FileChooser fileChooser= new FileChooser();
    File file= fileChooser.showOpenDialog( ownerWindow: null);
    path = file.toURI().toString();

    if(path≠null){
        Media media= new Media(path);
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(file.getName());
        image= new Image(getClass().getResourceAsStream( name: "/images/playIcon1.png"));
        iconLabel.setImage(image);
        songs.add(file);
        beginTimer();
        changeSpeed( event: null);
        mediaPlayer.play();
    }
}
1 usage
public void exitMethod(){
    Platform.exit();
    System.exit( status: 0);
}
1 usage
public void openAbout(ActionEvent event){
    final String msg= "MP3 Music Player \n"+
            "This software is developed by \n" +
            "Anganba Singha (MUH2001049M)\n" +
            "Towsif Bin Zaman (MUH2001035M)";

    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setContentText(msg);
    alert.setTitle("About MP3 Music Player");
    alert.setHeaderText(null);
    alert.showAndWait();
}
```



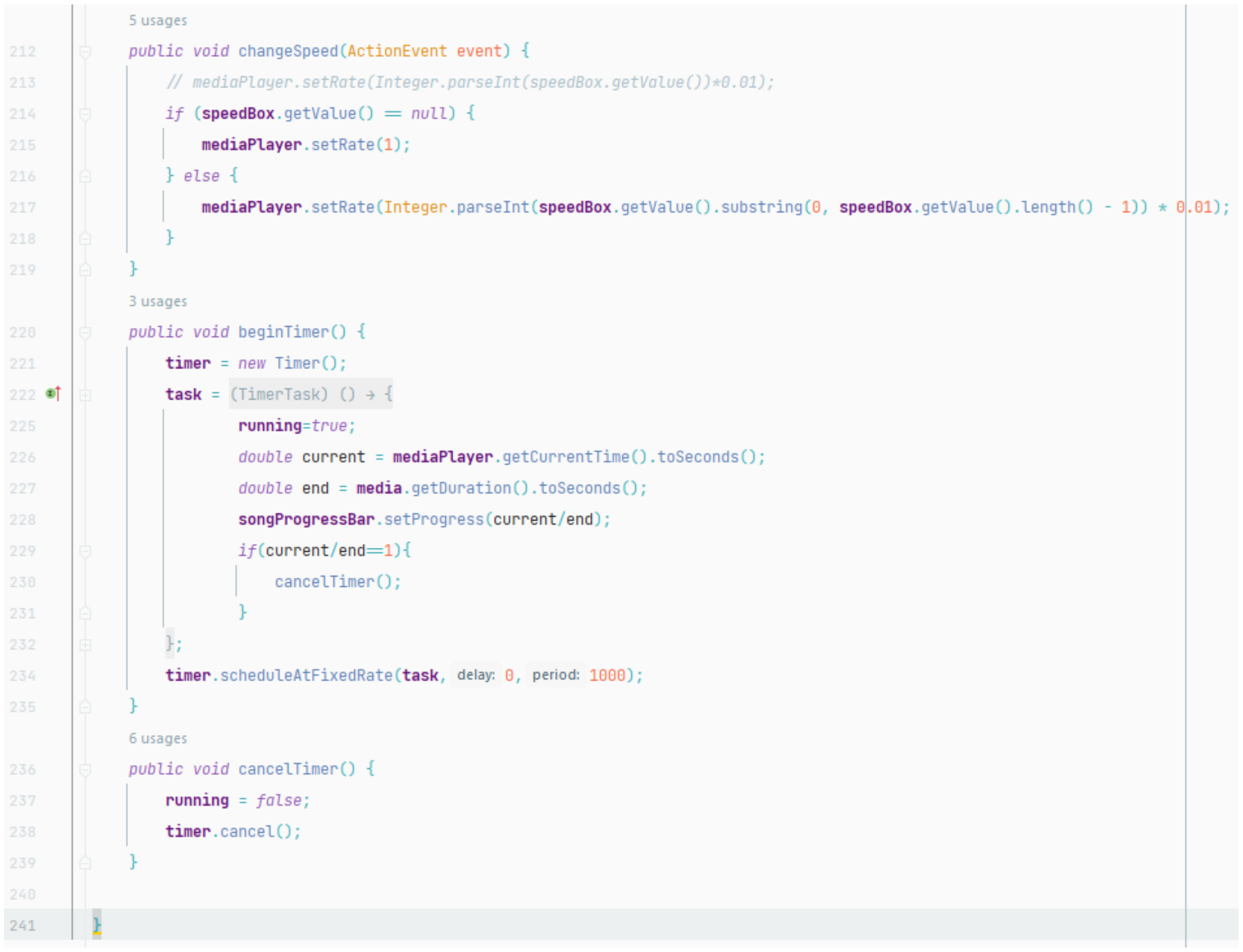
```
public void playMedia() {
    beginTimer();
    changeSpeed( event: null);
    mediaPlayer.setVolume(volumeSlider.getValue()*0.01);
    image= new Image(getClass().getResourceAsStream( name: "/images/playIcon1.png"));
    iconLabel.setImage(image);
    mediaPlayer.play();
}
1 usage
public void pauseMedia() {
    cancelTimer();
    image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
    iconLabel.setImage(image);
    mediaPlayer.pause();
}
1 usage
public void resetMedia() {
    songProgressBar.setProgress(0);
    image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
    iconLabel.setImage(image);
    mediaPlayer.seek(Duration.seconds( s: 0));
}
```



```
public void previousMedia() {
    if (songNumber > 0) {
        songNumber--;
        mediaPlayer.stop();
        if(running){
            cancelTimer();
        }
        image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
        iconLabel.setImage(image);
        media = new Media(songs.get(songNumber).toURI().toString());
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(songs.get(songNumber).getName());
    } else {
        songNumber = songs.size() - 1;
        mediaPlayer.stop();
        if(running){
            cancelTimer();
        }
        image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
        iconLabel.setImage(image);
        media = new Media(songs.get(songNumber).toURI().toString());
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(songs.get(songNumber).getName());
    }
}
```



```java
public void nextMedia() {
    if (songNumber < songs.size() - 1) {
        songNumber++;
        mediaPlayer.stop();
        if(running){
            cancelTimer();
        }
        image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
        iconLabel.setImage(image);
        media = new Media(songs.get(songNumber).toURI().toString());
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(songs.get(songNumber).getName());
    } else {
        songNumber = 0;
        mediaPlayer.stop();
        if(running){
            cancelTimer();
        }
        image= new Image(getClass().getResourceAsStream( name: "/images/defaultIcon.png"));
        iconLabel.setImage(image);
        media = new Media(songs.get(songNumber).toURI().toString());
        mediaPlayer = new MediaPlayer(media);
        songLabel.setText(songs.get(songNumber).getName());
    }
}
1 usage
public void forwardSkip(ActionEvent event){
    mediaPlayer.seek(mediaPlayer.getCurrentTime().add(Duration.seconds( s: 10)));
}
1 usage
public void backwardSkip(ActionEvent event){
    mediaPlayer.seek(mediaPlayer.getCurrentTime().add(Duration.seconds( s: -10)));
}
```



```
5 usages
public void changeSpeed(ActionEvent event) {
    // mediaPlayer.setRate(Integer.parseInt(speedBox.getValue())*0.01);
    if (speedBox.getValue() == null) {
        mediaPlayer.setRate(1);
    } else {
        mediaPlayer.setRate(Integer.parseInt(speedBox.getValue().substring(0, speedBox.getValue().length() - 1)) * 0.01);
    }
}
3 usages
public void beginTimer() {
    timer = new Timer();
    task = (TimerTask) () → {
            running=true;
            double current = mediaPlayer.getCurrentTime().toSeconds();
            double end = media.getDuration().toSeconds();
            songProgressBar.setProgress(current/end);
            if(current/end==1){
                cancelTimer();
            }
    };
    timer.scheduleAtFixedRate(task, delay: 0, period: 1000);
}
6 usages
public void cancelTimer() {
    running = false;
    timer.cancel();
}

}
```

CONCLUSION

Finally, creating an MP3 music player application with JavaFX may be both a rewarding and difficult undertaking for software developers. It's crucial to take into account technical specifications like audio playback, file I/O, and user interface design, as well as functionality like playlist management, audio controls, and music service integration, while developing a successful application. The program should also be fully documented, as well as optimized for performance and security. A JavaFX-based MP3 music player application can offer users a flawless and pleasurable music-listening experience with careful planning and execution.