Department of
Science and
Technology

CST2550

Software Engineering Management & Development

Coursework 2

Karishma Sukhwani

M00697033

Autumn/Winter Term - 2019/20

Table of Contents

Abstract

This JavaFx Application is a kind of prototype for a Karaoke Media Player. This allows user to load a database library file, having songs and lets user create a playlist to play from it. This Karaoke application has functionalities like play, pause, stop, skip and start over for media file to be displayed.

Introduction

This application is built for a company requiring a karaoke player app for their clients. Their key requirements are to let user select songs from library which is customizable and add it to play-list which can be played further when user presses play. This paper is describing in detail about this application and development part.

This report will be firstly having Application specifications like what are the features and use cases of it. Moving next to Design & Implementation part where logic, pseudo-code, and GUI mock-up wireframes are defined. It will also be having performance analysis and testing approach for this project. Limitations and lesson learnings are cover at the end followed by the source files.

Design

Here is the outlined pseudocode for my karaoke application:

Pseudocode

Application Pseudocode

Step 1 – Load Library
Step 2 – Add songs to playlist
Step 3 (optional) – Show playlist
Step 4 – Play playlist

Step 5 (optional) – Pause song

Step 6 (optional) – Skip song

Step 7 (optional) – Start over/ Previous song

Step 8 (optional) – Stop player

Step 9 (optional) – Clear playlist

Step 10 (optional) – Delete song from playlist

Individual Steps Pseudocode

Load library

1. Define & initialize the library as a TreeMap object of song class

2. Define File object and initialize it from the read arguments (file name/ file path)

3. Create a Reader to read through the file

4. Defile Song object to store songs temporarily

5. Loop (till end of the file):

a. Split read line into 4 parts and store it in a string array

b. Initialize the song class and set attributes (e.g, title, artist, etc) from String-Array

c. Add song object in library

Add songs to playlist

1. Show library scene

2. Initialize playlist with LinkedList interface (Queue class)

3. Repeat until user is done with creating playlist:

a. Select song from listView

b. Press Add to playlist button

get selected song from listview and add it at the end of playlist

c. Add song object in library

Show playlist

1. Show playlist scene

2. Show playlist object as listView

3. Display buttons as delete song and clear playlist

User can delete song or playlist from here

Play playlist

If mediaPlayer is not initialized:

If playlist is not empty:

a. Create a new MediaPlayer with Media instance

b. Add mediaPlayer to MediaView

c. Add event handlers/ listeners for new mediaPlayer

//event handlers & listeners will handle further

If playlist is empty:

Show alert for empty playlist

Else:

If mediaPlayer.Status is STOPPED && it is not due to stop button:

If playlist is not empty:

a. Create a new MediaPlayer with Media instance

b. Add mediaPlayer to MediaView

c. Add event handlers/ listeners for new mediaPlayer

//event handlers & listeners will handle further

Else:

Show alert for empty playlist

If STOPPED status is because of stop Button:

Play current mediaPlayer of mediaView

Pause song

Get current mediaPlayer of mediaView and Pause it

Skip song

1. Get current mediaPlayer of mediaView and Stop it
2. If next song in playlst is not null:
    - **a.** Create a new MediaPlayer with Media instance
    - **b.** Add mediaPlayer to MediaView
    - **c.** Add event handlers/ listeners for new mediaPlayer

        //event handlers & listeners will handle further

Else:

Show alert for empty playlist

Start over/ Previous song

If current song for mediaPlayer is not null:

- **a.** Seek zero Duration for current mediaPlayer
- **b.** Set stopTime of mediaPlayer as end of duration of currentSong
- **c.** play mediaPlayer

Else:

Show alert for empty playlist

Stop Player

1. Get current mediaPlayer of mediaView and Stop it
2. Set stopButton flag as up (true)

Clear playlist

1. Show confirmation alert
2. If yes:
   Playlist.clear()
   Else:
   Close alert

Delete song

1. Select song from listView
2. Press Delete button
   get selected song from listview and remove from playlist (playlist.remove())
3. Update listView with new playlist

Efficiency Analysis

This Application uses **TreeMap Class** for storing library which implements Red-Black Tree Data Structure in Java (Oracle Java TreeMap API Documentation). So, time complexity for different operations and space complexity would be according to **Red-Black Tree Data structure**.

For storing playlist in Application, **Queue Class** has been used which implements LinkedList interface and Queue Data Structure in java. S, time complexity and space complexity of different operations on playlist will be according to **Queue Data Structure**.

Table below analysis the efficiency for various operations:

Average Time Complexity:

Data Structure	Access	Search	Insertion	Deletion
Red-Black Tree (Tree Map)	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$
Queue	$\boldsymbol{\theta(n)}$	$\boldsymbol{\theta(n)}$	$\boldsymbol{\theta(1)}$	$\boldsymbol{\theta(1)}$

Worst Time Complexity:

Data Structure	Access	Search	Insertion	Deletion
Red-Black Tree (Tree Map)	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$	$\boldsymbol{\theta(\log n)}$
Queue	$\boldsymbol{\theta(n)}$	$\boldsymbol{\theta(n)}$	$\boldsymbol{\theta(1)}$	$\boldsymbol{\theta(1)}$

Space Complexity:

Data Structure	Access
Red-Black Tree (Tree Map)	$\boldsymbol{\theta(n)}$
Queue	$\boldsymbol{\theta(n)}$

This application is not performing access and search operations on Queue as we don't require it.

GUI Mock-up Wireframes

Main page:

Fig. WireFrame 1

After pressing Library Button

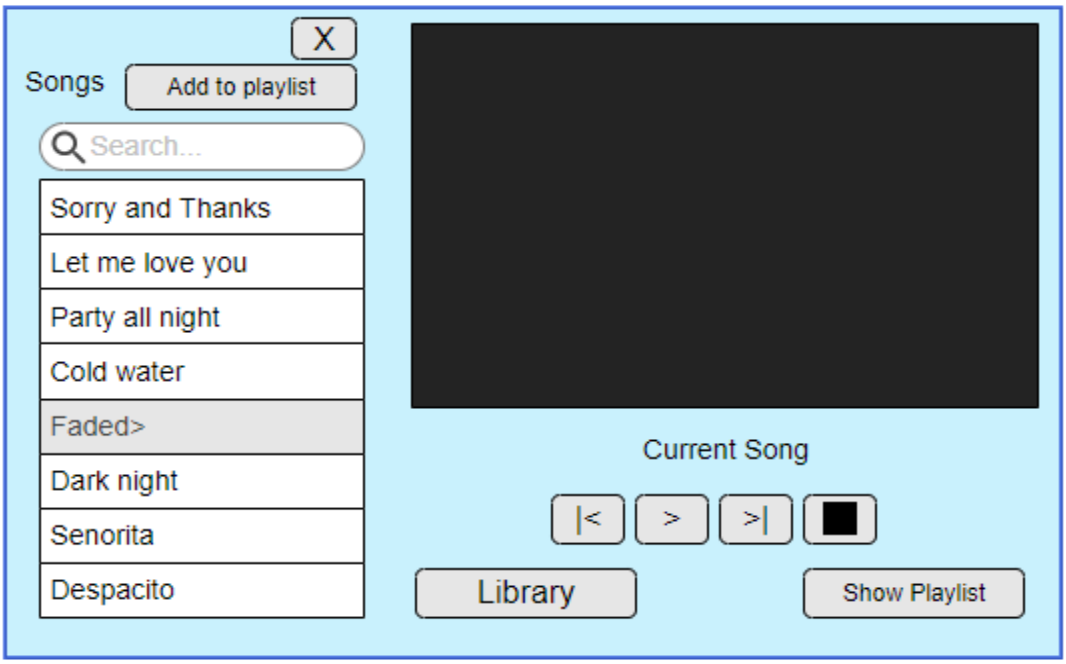

Fig. Wireframe 2

After pressing Show Playlist Button

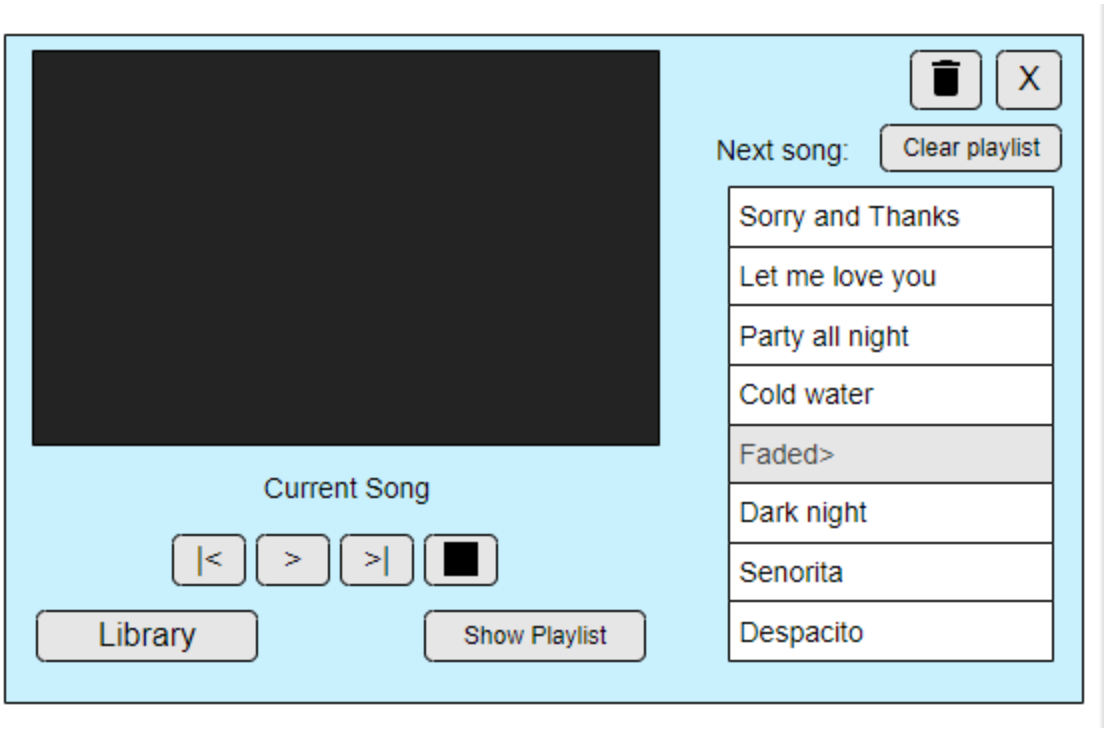

Fig. Wireframe 3

Testing

For karaoke application, I have performed manual testing with each use case under various scenarios. I have thoroughly verified the modular functionalities of all the components, and also, when integrated into the application with different modules.

This Application has also been tested on different system having NetBeans platform. For running this on Linux platform, we need to change the parameter/ argument format for main class because NetBeans platform takes parameters in the form of Map Entry.

Testcase	Description	Scenario/ Input	Anticipated O/P	Result
1	loadLibrary() function	Correct file path argument	Song Library loaded	Pass
2	loadLibrary() function	Incorrect file path argument	FileNotFound message printed	Pass
3	Display mediaView	Without setting mediaPlayer	Black screen displayed	Pass
4	Display mediaView	After setting mediaPlayer	Media File Displayed	Pass
5	Adding songs to playlist	From listView, library of TreeMap class	Playlist updated	Pass
6	Search song (startswith method)	Regardless of upper/lower case	Selectable searched song displayed in the listView	Pass
7	Play Button	Without creating playlist even once	Button disabled	Pass
8	Play Button	Without creating playlist	Error Alert for empty playlist	Pass
9	Play Button	After creating playlist	Video should be played with current Song's name	Pass
10	Play Button	After pressing Stop Button	Current song should be played from beginning	Pass
11	Start over/ previous Button	No songs in playlist, and no playlist was played before	Button disabled	Pass
12	Start over/ previous Button	No songs in playlist, and playlist was played before	Play last song of previous playlist	Pass
13	Start over/ previous Button	Playlist created, no song from current playlist has been played, no previous playlist	Error Alert for empty playlist	Pass
14	Start over/ previous Button	Playlist created, already played first song from playlist	Play previous song	Pass
15	Next Button	Empty playlist	Error Alert for empty playlist	Pass

16	Next Button	songs in playlist	Next song played	Pass
17	Stop Button	no song is being played	Do nothing	Pass
18	Stop Button	Any song is played	Bring to stop state	Pass
19	Show Playlist Button	No songs in Playlist	Show empty playlist error	Pass
20	Show Playlist Button	Playlist exist	Display playlist scene	Pass
21	Delete song Button	Regardless of song selected or not	Deletes selected song from playlist/ not deleted if not selected	Pass
22	Clear Playlist	N/A	Clear playlist with confirmation and brings back main display	Pass
23	Song to be played	When can't find the video-file attached with song	File Not found error	Pass
24	Play Button	After stop button, no more songs in playlist	Error Alert for empty playlist	Pass

Conclusion

By the end of this project, I have developed a Karaoke player which allows user to select a song from library and play it with background music and lyrics written in video file. This Application has play, pause, skip song, previous song and stop buttons to control the media File. User can also delete a song from playlist or playlist as of whole according to their choice.

Limitation of this Application includes – a video file should be having subtitles for lyrics. Without that, it can be media player but not a Karaoke player. Also, it does not have Slider to track and control current time of player, so user cannot skip/ rewind the song for next few seconds. User can just access the last song played, not the one before it. This Application uses system volume for now instead of Application volume for media file.

Talking about having such project in future, I would try to eliminate limitations of this project and would add extra controls for user. Instead of deleting songs from playlist, we can add it to another

Queue to save the playlist for next time usage. This can be extended to creating a new file for saving user data like favorite songs and playlist by adding a login interface. Furthermore, for making it feel like a Karaoke, we can add a background file having BeatWaves which fluctuates considering the audio property of the Karaoke file. There might be option to highlight the lyrics to be sung and de-shade already sung lyrics for the Application.

References

Oracle Java MediaPlayer.Status API Documentation -

Available online on -

https://docs.oracle.com/javafx/2/api/javafx/scene/media/MediaPlayer.Status.html

Oracle Java TreeMap API Documentation -

Available online on - https://docs.oracle.com/javase/1.5.0/docs/api/java/util/TreeMap.html

Oracle Java Adding a Media Tutorial -

Available online on - https://docs.oracle.com/javase/8/javafx/media-tutorial/playercontrol.htm

Basic explanation and efficiency of Data structure types -

Available online on - https://medium.com/omarelgabrys-blog/data-structures-a-quick-comparison-6689d725b3b0 - time/

Appendices

Filename: **Song.java**

```
public class Song {
    private String title;
    private String artist;
    private int duration; //in seconds
    private String videoFile;
```

```
public Song() {
    title="";
    artist="";
    duration=0;
    videoFile="";
}

public Song(String title, String artist, int duration, String videoFile) {
    this.title = title;
    this.artist = artist;
    this.duration = duration;
    this.videoFile = videoFile;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getArtist() {
    return artist;
}

public void setArtist(String artist) {
    this.artist = artist;
}
```

```
    public int getDuration() {
        return duration;
    }

    public void setDuration(int duration) {
        this.duration = duration;
    }

    public String getVideoFile() {
        return videoFile;
    }

    public void setVideoFile(String videoFile) {
        this.videoFile = videoFile;
    }

    @Override
    public String toString() {

        return title + "\tby\t" + artist ;
    }

}
```

Filename: **Karaoke_cw.java**

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Optional;
import java.util.Queue;
import java.util.TreeMap;
import java.util.function.Predicate;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.collections.transformation.FilteredList;
import javafx.event.ActionEvent;
import javafx.geometry.Insets;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonBar.ButtonData;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.SelectionMode;
```

```
import javafx.scene.control.Slider;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaPlayer.Status;
import javafx.scene.media.MediaView;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.util.Duration;

public class Karaoke_cw extends Application {



    MediaView mediaView;
    Slider timeSlider;
    Button playB;
    Button prevB;
    Button nextB;
    Button stopB;
    Button libButton = new Button("Show Library");
```

```
Button plButton = new Button("Show Playlist");
Button plCloseB = new Button(" X ");
Button libCloseB = new Button(" X ");
Button plDeleteB = new Button();
TextField searchField = new TextField();
Button addButton = new Button( "Add to playlist ");
Button clrPLB = new Button( "Clear playlist ");
boolean isPlaying=false;
boolean isPlVisible=false;
boolean isLibVisible=false;
boolean isStopPressed=false;
Map<String, Song> library;
Queue<Song> playList;
ObservableList<Song> lib;
ObservableList<Song> pl;
ListView<Song> libLV;
ListView<Song> plLV;
Song currS = new Song();
//Runnable r1;
Runnable r2;
Runnable r3;



@Override
public void start(Stage primaryStage) throws Exception {

    //System.out.println(getParameters().getNamed().get("sampleFile"));

    playList = new LinkedList<>();
```

```
plCloseB.setStyle("-fx-background-color: red;");
libCloseB.setStyle("-fx-background-color: red;");

//Layout
GridPane gridPane = new GridPane();
gridPane.setPadding(new Insets(20));
gridPane.setHgap(10);
gridPane.setVgap(10);

//MediaView for holding mediaPlayer
VBox mvContainer = new VBox();
mediaView = new MediaView();
mediaView.setFitHeight(210);
mediaView.setFitWidth(360);
mvContainer.setStyle("-fx-background-color: black");
mvContainer.getChildren().add(mediaView);
gridPane.add(mvContainer, 0, 0);
GridPane.setColumnSpan(mvContainer, 2);

//Label of current song being played
VBox mp3L = new VBox();
Label curMp3L = new Label("");
mp3L.getChildren().add(curMp3L);
gridPane.add(mp3L, 0, 1);
//Text mpStatus = new Text();
//mp3L.getChildren().add(mpStatus);
GridPane.setColumnSpan(mp3L, 2);
mp3L.setAlignment(Pos.BASELINE_CENTER);
```

```
        //alert for empty playlist
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("Warning");
        alert.setHeaderText("Empty Playlist!!!");
        alert.setContentText("First add songs in Playlist");

        /*
        //Listener for mediaPlayer.Status
        r1 = () -> {
            try{
                mediaView.getMediaPlayer().statusProperty().addListener((observable,
oldValue, newValue) -> {
                        mpStatus.setText(newValue.toString());


                });
            }
            catch(NullPointerException ex){
                //do nothing, exception has occured because mediaPlayer
                //..for mediaView has not been initialized yet
            }
            catch(Exception ex){
                ex.printStackTrace();
            }
        };
        new Thread(r1).start();
        */

        //when stopTime has come for mediaPlayer
        r2 = () -> {
            try{
```

```
            mediaView.getMediaPlayer().setOnEndOfMedia(() -> {


                if(playList.peek() != null){
                    try{
                        Media video = new Media(new
File(playList.peek().getVideoFile()).toURI().toString());
                        MediaPlayer mediaPlayer = new MediaPlayer(video);
                        mediaView.setMediaPlayer(mediaPlayer);
                        //new Thread(r1).start();
                        new Thread(r2).start();
                        new Thread(r3).start();
                    }catch(Exception ex){
                        System.out.println("File not found");
                    }
                }
                else{
                    curMp3L.setText("");
                    isPlaying = false;
                    playB.setText(" > ");
                }


                //set status as stop - if status is READY then it won't work
                //.. but this is required as we need to play songs from next
playlist
                mediaView.getMediaPlayer().stop();
            });
        }
        catch(NullPointerException ex){
            //do nothing, exception has occured because mediaPlayer
            //..for mediaView has not been initialized yet
```

```
            }
            catch(Exception ex){
                ex.printStackTrace();
            }
        };
        new Thread(r2).start();

        //for autoplying next song from queue if status=READY
        r3 = () -> {
            try{
                mediaView.getMediaPlayer().setOnReady(() -> {

                    if((currS = playList.poll()) != null){


mediaView.getMediaPlayer().setStopTime(Duration.seconds(currS.getDuration())
);
                        curMp3L.setText(currS.toString());
                        isPlaying = true;
                        playB.setText(" | | ");
                        mediaView.getMediaPlayer().play();

                        if(isPlVisible){
                            pl = FXCollections.<Song>observableList((List)playList);
                            plLV.setItems(pl);
                        }
                    }

                });
            }
```

```
            catch(NullPointerException ex){
                //do nothing, exception has occured because mediaPlayer
                //..for mediaView has not been initialized yet
            }
            catch(Exception ex){
                ex.printStackTrace();
            }
        };
        new Thread(r3).start();

        //load songs in library
        try{
            loadLibrary();
        } catch (IOException ex) {
            System.out.println("something wrong with loadLibrary()");
            //Logger.getLogger(Karaoke_cw.class.getName()).log(Level.SEVERE,
null, ex);
        }

        lib = FXCollections.observableArrayList();
        lib.setAll(library.values());
        libLV = new ListView<>(lib);

        //container for control buttons of player
        HBox mControllerBox = new HBox();

        playB = new Button(" > ");
        playB.setTooltip(new Tooltip("Play/ Pause Button"));
        playB.setOnAction((ActionEvent e) -> {
            //to handle play/ pause event
```

```
        if(isPlaying){   //user is pressing on pause button
            mediaView.getMediaPlayer().pause();
            isPlaying = false;
            playB.setText(" > ");
        }
        else{   //user is pressing on play button

            //if mediaPlayer has not been initialized yet
            if(mediaView.getMediaPlayer() == null ){

                if(playList.peek() != null){
                    try{
                        Media video = new Media(new
File(playList.peek().getVideoFile()).toURI().toString());
                        MediaPlayer mediaPlayer = new MediaPlayer(video);
                        mediaView.setMediaPlayer(mediaPlayer);
                        //new Thread(r1).start();
                        new Thread(r2).start();
                        new Thread(r3).start();
                    }catch(Exception ex){
                        System.out.println("File not found");
                    }
                    //Gets READY state and event handler handles further
                }
                else{
                    alert.showAndWait();
                }

            }
            else{
```

```
                //for new playlist after one has been played
                if(mediaView.getMediaPlayer().getStatus() == Status.STOPPED
                        && !isStopPressed){ //&& stop is not pressed


                    //get next element of playlist and play it
                    if(playList.peek() != null){


                        try{
                            Media video = new Media(new
File(playList.peek().getVideoFile()).toURI().toString());
                            MediaPlayer mediaPlayer = new MediaPlayer(video);
                            mediaView.setMediaPlayer(mediaPlayer);
                            //new Thread(r1).start();
                            new Thread(r2).start();
                            new Thread(r3).start();
                        }catch(Exception ex){
                            System.out.println("File not found");
                        }
                        //Gets READY state and event handler handles further


                    }
                    else{
                        alert.showAndWait();
                    }
                }
                else if(isStopPressed){
                    isStopPressed = false;
                    if(playList.peek() != null){//stop is pressed and user is playing
play
                        mediaView.getMediaPlayer().play();
```

```
                    isPlaying = true;
                    playB.setText(" | | ");
                }
                else {alert.showAndWait();  }
            }
            else{
                mediaView.getMediaPlayer().play();
                isPlaying = true;
                playB.setText(" | | ");
            }
        }
    }
});

prevB = new Button(" |< ");
prevB.setTooltip(new Tooltip("Previous song/ start over"));
prevB.setOnAction((ActionEvent e) -> {
    try{
        if(mediaView.getMediaPlayer().getStatus() == Status.STOPPED ){

            if(currS != null ){

                curMp3L.setText(currS.toString());
                mediaView.getMediaPlayer().seek(Duration.ZERO);

mediaView.getMediaPlayer().setStopTime(Duration.seconds(currS.getDuration())
);
                mediaView.getMediaPlayer().play();
                isPlaying = true;
                playB.setText(" | | ");
```

```
                if(isPlVisible){
                    pl = FXCollections.<Song>observableList((List)playList);
                    plLV.setItems(pl);
                }
            }
            else{
                alert.showAndWait();
            }
        }
        else{
            if(currS != null){
                curMp3L.setText(currS.toString());
                mediaView.getMediaPlayer().seek(Duration.ZERO);
                mediaView.getMediaPlayer().play();
                isPlaying = true;
                playB.setText(" | | ");

            }
        }
    }
    catch(NullPointerException ex){
        //playlist is empty
        alert.showAndWait();
    }
});
nextB = new Button(" >| ");
nextB.setTooltip(new Tooltip("Skip/Next song"));
nextB.setOnAction((ActionEvent e) -> {
```

```
        if(playList.peek() != null){
            try{
                mediaView.getMediaPlayer().stop();
            }catch(NullPointerException exx){ //do nothing if player is not
defined
            }
            try{
                Media video = new Media(new
File(playList.peek().getVideoFile()).toURI().toString());
                MediaPlayer mediaPlayer = new MediaPlayer(video);
                mediaView.setMediaPlayer(mediaPlayer);
                //new Thread(r1).start();
                new Thread(r2).start();
                new Thread(r3).start();
            }catch(Exception ex){
                System.out.println("File not found");
            }

        }
        else{
            curMp3L.setText("");
            isPlaying = false;

            try{
                mediaView.getMediaPlayer().stop();
            }catch(NullPointerException exx){
                //do nothing if player is not defined
            }
```

```
            playB.setText(" > ");
            alert.showAndWait();
        }
    });

    //stop button
    stopB = new Button(" II ");
    stopB.setOnAction((ActionEvent e) -> {
        try{
            mediaView.getMediaPlayer().stop();
            isPlaying = false;
            playB.setText(" > ");
            isStopPressed = true;
        }
        catch(NullPointerException ex){}
    });
    stopB.setTooltip(new Tooltip("Stop Button"));

    mControllerBox.getChildren().addAll(prevB, playB, nextB, stopB);
    mControllerBox.setAlignment(Pos.BASELINE_CENTER);
    gridPane.add(mControllerBox, 0, 2);
    GridPane.setColumnSpan(mControllerBox, 2);

    //keep buttons disabled if no mediaPlayer
    if(mediaView.getMediaPlayer() == null){
        playB.setDisable(true);
        prevB.setDisable(true);
        nextB.setDisable(true);
    }
```

```
HBox libB = new HBox();
libB.getChildren().add(libButton);
libB.setAlignment(Pos.BOTTOM_LEFT);
gridPane.add(libB, 0, 3);



libButton.setOnAction((ActionEvent e) -> {

    //to avoid malfunctioning of application
    libButton.setDisable(true);
    plButton.setDisable(true);

    if(!isLibVisible){

        GridPane gp = new GridPane();
        gp.setVgap(5);
        gp.setHgap(5);
        gp.setPadding(new Insets(10));

        HBox hb1 = new HBox();
        hb1.getChildren().add(libCloseB);
        hb1.setAlignment(Pos.BASELINE_RIGHT);
        gp.add(hb1, 1, 0);

        HBox hb2 = new HBox();
        Text sText = new Text("Search Song: ");
        hb2.getChildren().add(sText);
        hb2.setAlignment(Pos.BASELINE_LEFT);
```

```
sText.setFont(Font.font("Verdana", FontWeight.SEMI_BOLD, 14));
gp.add(hb2, 0, 1);

HBox hb3 = new HBox();
hb3.getChildren().add(addButton);
hb3.setAlignment(Pos.BASELINE_RIGHT);
gp.add(hb3, 1, 1);

//searchField
gp.add(searchField, 0, 2);
searchField.setPromptText("Search");
GridPane.setColumnSpan(searchField, 2);

//listView of library
libLV.setOrientation(Orientation.VERTICAL);
libLV.setPrefSize(210, 300);
gp.add(libLV, 0, 3);
GridPane.setColumnSpan(libLV, 2);



//Adding borderPane to carry gridPane and playlist
BorderPane borderPane = new BorderPane();
borderPane.setPadding(new Insets(10));
borderPane.setCenter(gridPane);
borderPane.setLeft(gp);
borderPane.setStyle("-fx-background-color: lightblue");
Scene s1 = new Scene(borderPane, 630, 370);
primaryStage.setScene(s1);
```

```
            isLibVisible = true;
        }
    });

    libCloseB.setOnAction((ActionEvent e) -> {
        //enabling disabled buttons again
        plButton.setDisable(false);
        libButton.setDisable(false);

        //Adding borderpane to carry gridPane as same Pane cannot be root to more than 1 scene
        BorderPane borderPane = new BorderPane();
        borderPane.setCenter(gridPane);
        Scene s1 = new Scene(borderPane, 400, 370);
        primaryStage.setScene(s1);

        isLibVisible=false;
    });

    FilteredList<Song> filteredLib = new FilteredList<>(lib, p -> true); //p is predicator
    searchField.setOnKeyReleased(event -> {
        searchField.textProperty().addListener((observableValue, oldValue, newValue) -> {
            filteredLib.setPredicate((Predicate<? super Song>) song -> {
                if(newValue == null || newValue.isEmpty())
                    return true; //return whole list as it was
                else return song.getTitle().toLowerCase().startsWith(newValue.toLowerCase());
                    //only return filtered Songs
            });
```

```
            });

            libLV.setItems(filteredLib);
        });

        addButton.setOnAction((ActionEvent e) -> {
            Song selectedItem = libLV.getSelectionModel().getSelectedItem();
            playList.offer(selectedItem); //adds to the end of queue

            playB.setDisable(false);
            prevB.setDisable(false);
            nextB.setDisable(false);
            pl = FXCollections.<Song>observableList((List)playList);
            plLV = new ListView<>(pl);
        });

        try{
            plDeleteB.setGraphic(new ImageView( new Image(  new
File("download.jpg").toURI().toString(), 20, 20, true, true)));
        }catch(Exception e){
            plDeleteB.setText("Delete");
        }
        plDeleteB.setTooltip(new Tooltip("Delete selected song"));
        plDeleteB.setOnAction((ActionEvent e) -> {

            ObservableList selectedIndices =
plLV.getSelectionModel().getSelectedItems();
            for(Object o : selectedIndices){
                //System.out.println("o = " + o + " (" + o.getClass() + ")");
                playList.remove(o);
```

```
    }
    //System.out.println(playList);
    pl = FXCollections.<Song>observableList((List)playList);
    plLV.setItems(pl);
});

HBox plB = new HBox();
plB.getChildren().add(plButton);
plB.setAlignment(Pos.BOTTOM_RIGHT);
gridPane.add(plB, 1, 3);

plButton.setOnAction((ActionEvent e) -> {
    if(playList.isEmpty()){
        //show alert for empty PL
        alert.showAndWait();
    }
    else{
        //to avoid malfunctioning of application
        libButton.setDisable(true);
        plButton.setDisable(true);

        if(!isPlVisible){
            GridPane gp = new GridPane();
            gp.setHgap(5);
            gp.setVgap(5);
            gp.setPadding(new Insets(10));

            HBox hb1 = new HBox();
            hb1.getChildren().addAll(plDeleteB, plCloseB);
```

```
        hb1.setPadding(new Insets(0,20,0,20));
        hb1.setAlignment(Pos.BASELINE_RIGHT);
        gp.add(hb1, 1, 0);

        HBox hb2 = new HBox();
        hb2.setAlignment(Pos.BASELINE_LEFT);
        Text nsText = new Text("Next Song: ");
        hb2.getChildren().add(nsText);
        nsText.setFont(Font.font("Verdana", FontWeight.SEMI_BOLD,
14));
        gp.add(hb2, 0, 1);

        HBox hb3 = new HBox();
        hb3.getChildren().add(clrPLB);
        hb3.setAlignment(Pos.BASELINE_RIGHT);
        gp.add(hb3, 1, 1);


        plLV.setOrientation(Orientation.VERTICAL);
        plLV.setPrefSize(210, 280);
        gp.add(plLV,0, 2);
        GridPane.setColumnSpan(plLV, 2);

        //Adding borderPane to carry gridPane and playlist
        BorderPane borderPane = new BorderPane();
        borderPane.setPadding(new Insets(10));
        borderPane.setCenter(gridPane);
        borderPane.setRight(gp);
        borderPane.setStyle("-fx-background-color: lightblue");
        Scene s1 = new Scene(borderPane, 640, 370);
```

```
                primaryStage.setScene(s1);

                isPlVisible = true;
            }
        }
    });

    plCloseB.setOnAction((ActionEvent e) -> {
        //enabling disabled buttons again
        plButton.setDisable(false);
        libButton.setDisable(false);

        //Adding borderPane to carry gridPane as it cannot be root again
        BorderPane borderPane = new BorderPane();
        borderPane.setCenter(gridPane);
        Scene s1 = new Scene(borderPane, 400, 370);
        primaryStage.setScene(s1);

        isPlVisible=false;
    });

    clrPLB.setOnAction((ActionEvent e) -> {
        //alert for empty playlist
        Alert a = new Alert(AlertType.CONFIRMATION);
        a.setTitle("Comfirm");
        a.setHeaderText("Clearing your playlist!!!");
        a.setContentText("Are you sure you want to remove all the songs from
playlist?");

        ButtonType btYes = new ButtonType(" Yes ", ButtonData.YES);
```

```
        ButtonType btCancel = new ButtonType("Cancel",
ButtonData.CANCEL_CLOSE);


        a.getButtonTypes().setAll(btYes, btCancel);


        Optional<ButtonType> result = a.showAndWait();
        if(!result.isPresent()){}
        else if(result.get() == btYes){
            playList.clear();
            pl.clear();
            plLV.setItems(null);
            plCloseB.fire();
        }
        else{}
    });


    gridPane.setStyle("-fx-background-color: lightblue");
    Scene scene = new Scene(gridPane, 400, 370); //(width*height)
    primaryStage.setTitle("Karaoke Player");
    primaryStage.setScene(scene);
    primaryStage.show();
  }


  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
    launch(args);
  }
```

```
    public void loadLibrary() throws IOException{

        library = new TreeMap<>(); //Creating TreeMap data structure for Red
Black Tree implemetation
        try{
            File file = new File(getParameters().getNamed().get("sampleFile"));
            BufferedReader br = new BufferedReader(new FileReader(file));

            Song s; //for storing songs taken out from file temporarily
            String st;

            while ((st = br.readLine()) != null) {
                String[] words = st.split("   ", 4);
                s = new Song();
                s.setTitle(words[0]);
                s.setArtist(words[1]);
                s.setDuration(Integer.valueOf(words[2]));
                s.setVideoFile(words[3]);

                library.put(s.getTitle(), s);
            }
        }catch(Exception ex){
            System.out.println("File not found");
        }

    }
}
```