学号 X41714008 专业 计算机科学与技术 姓名 李逢时

实验日期 **19/4/7**  教师签字 成绩

实验报告

【实验名称】 实验六 图形用户界面设计——播放器

【实验目的】

1. 学习java图形用户界面设计
2. 学习java事件处理

【实验原理】

1. Java程序设计基本原理
2. 面向对象程序设计基本原理

【实验内容】

**实验内容:** 根据已给出的播放器的例子程序,设计一个完整的播放器软件。

**实验要求：**

1 设计一个完整的播放器界面，并添加按钮响应；

2 实现歌词根据歌曲内容逐句提醒；

3 实现进度条根据歌曲播放时间移动；

4 导入一个gif动画图片；

5 歌曲列表右侧添加歌曲时间的列；

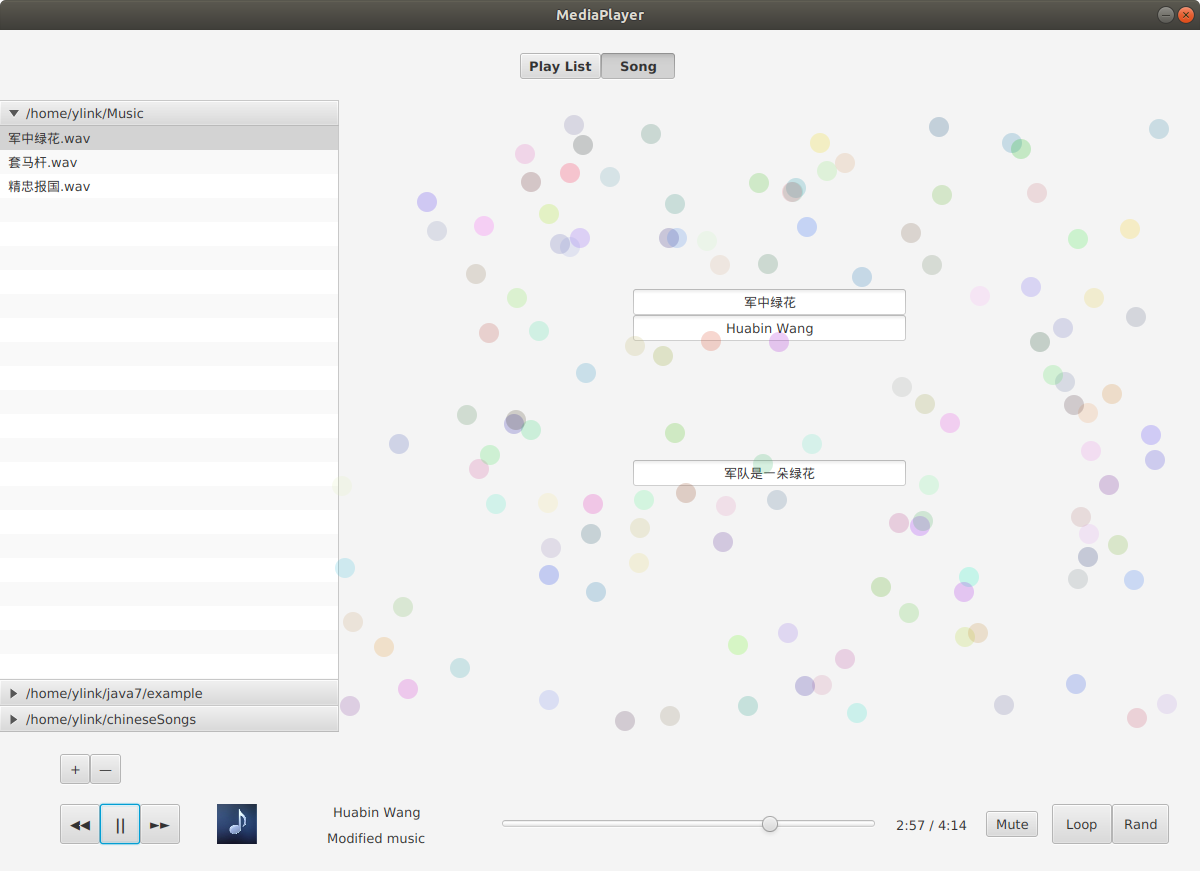
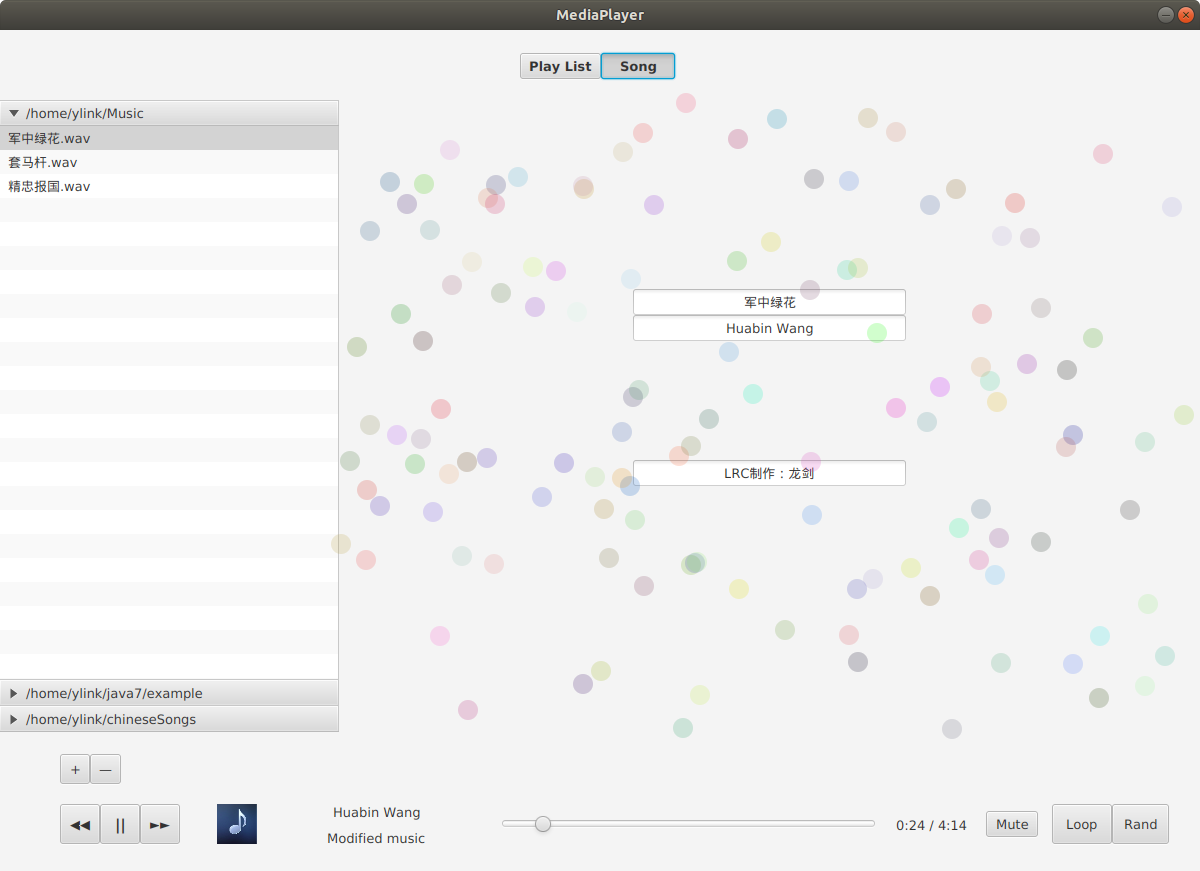
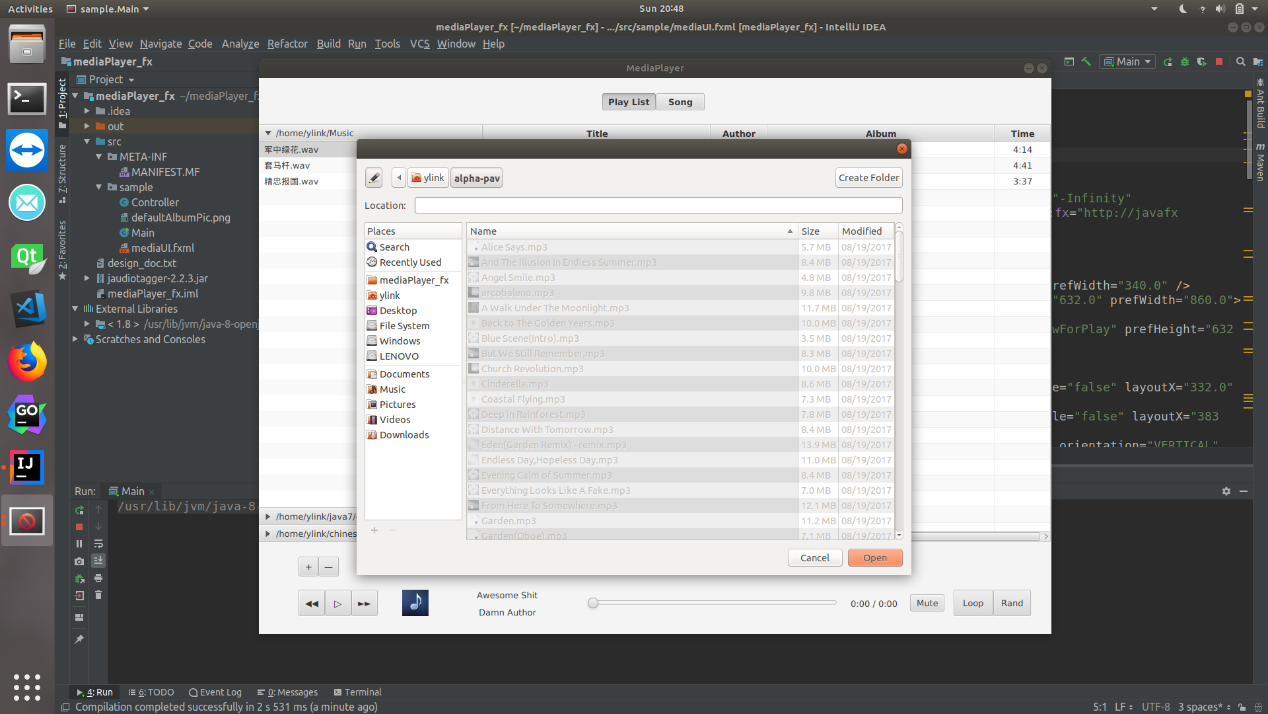
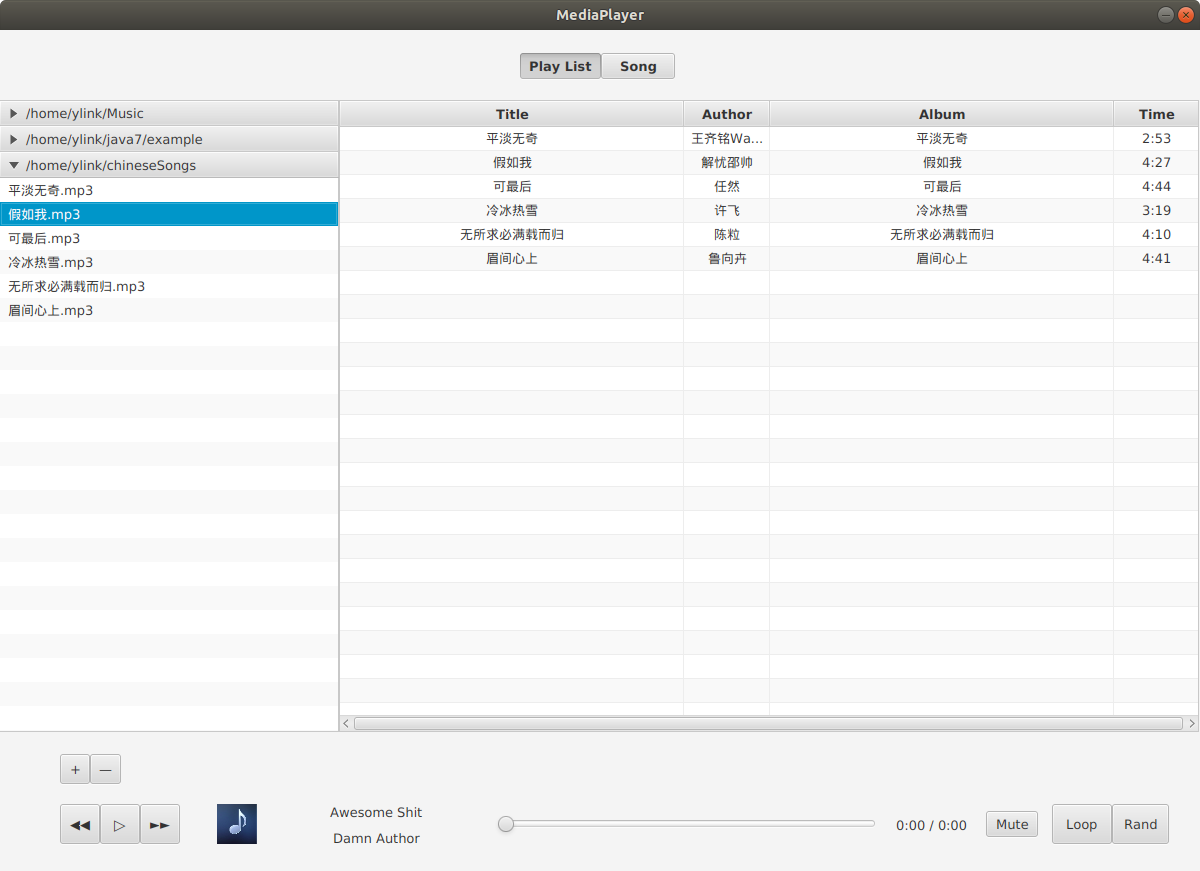
6 进度条附件添加一个当前播放时间的显示；

7 双击列表播放该歌曲；

8 添加你认为需要的功能。。。

9 最后，把播放器程序打包成jar文件。

截图：



**源代码：**

**Controller.java**

**package sample;**

**import java.io.\*;**

**import java.net.URL;**

**import java.util.ArrayList;**

**import java.util.Random;**

**import java.util.ResourceBundle;**

**import javafx.beans.value.ChangeListener;**

**import javafx.collections.FXCollections;**

**import javafx.collections.ObservableList;**

**import javafx.event.EventHandler;**

**import javafx.fxml.FXML;**

**import javafx.fxml.Initializable;**

**import javafx.scene.control.\*;**

**import javafx.scene.control.cell.PropertyValueFactory;**

**import javafx.scene.input.MouseEvent;**

**import javafx.scene.layout.Pane;**

**import javafx.scene.media.Media;**

**import javafx.scene.media.MediaPlayer;**

**import javafx.scene.paint.Color;**

**import javafx.scene.shape.Circle;**

**import javafx.stage.DirectoryChooser;**

**import org.jaudiotagger.audio.AudioFile;**

**import org.jaudiotagger.audio.AudioFileIO;**

**import org.jaudiotagger.tag.FieldKey;**

**import org.jaudiotagger.tag.Tag;**

**import javafx.util.\*;**

**public class Controller implements Initializable {**

**//Buttons**

**@FXML**

**private Button backBtn;**

**@FXML**

**private Button playBtn;**

**@FXML**

**private Button forwardBtn;**

**@FXML**

**private ToggleButton muteBtn;**

**@FXML**

**private ToggleButton loopBtn;**

**@FXML**

**private ToggleButton shuffleBtn;**

**@FXML**

**private ToggleButton playListBtn;**

**@FXML**

**private ToggleButton songBtn;**

**//Labels**

**@FXML**

**private Label singerLabel;**

**@FXML**

**private Label albumLabel;**

**@FXML**

**private Label timeLabel;**

**//Slide**

**@FXML**

**private Slider timeSlide;**

**//Content Pane**

**@FXML**

**private Pane contentPane;**

**//Directory list**

**@FXML**

**private Accordion fileAccordion;**

**//Current Play List**

**@FXML**

**private TableView<MusicMetaInfo> playListTable;**

**@FXML**

**private TextField titleField;**

**@FXML**

**private TextField authorField;**

**@FXML**

**private TextField lyricField;**

**private TableColumn<MusicMetaInfo, String> titleCol = new TableColumn<>("Title");**

**private TableColumn<MusicMetaInfo, String> authorCol = new TableColumn<>("Author");**

**private TableColumn<MusicMetaInfo, String> albumCol = new TableColumn<>("Album");**

**private TableColumn<MusicMetaInfo, String> timeCol = new TableColumn<>("Time");**

**private String tempFilePath = System.getProperties().getProperty("user.home") + "/md.tmp";**

**private ArrayList<MusicIdentifier> musicFullList = new ArrayList<>();**

**private ArrayList<MusicIdentifier> backgroundPlayList = new ArrayList<>();**

**private Media mainMedia;**

**private MediaPlayer mainPlayer;**

**private int currentPlayingMusicIndexInPlayList = -1;**

**private ChangeListener<Duration> progressListener;**

**private ToggleGroup muteGroup;**

**private javafx.scene.Group vizContainer;**

**private ArrayList<String> lrcContainer;**

**//One of the switchable contents**

**//Current play list, and its assistance class**

**public class MusicMetaInfo {**

**private String title;**

**private String author;**

**private String album;**

**private String timeLen;**

**MusicMetaInfo() {}**

**MusicMetaInfo(String tl, String au, String al, String tm) {**

**title = tl;**

**author = au;**

**album = al;**

**timeLen = tm;**

**}**

**public String getTitle() {**

**return title;**

**}**

**public void setTitle(String title) {**

**this.title = title;**

**}**

**public String getAuthor() {**

**return author;**

**}**

**public void setAuthor(String author) {**

**this.author = author;**

**}**

**public String getAlbum() {**

**return album;**

**}**

**public void setAlbum(String album) {**

**this.album = album;**

**}**

**public String getTimeLen() {**

**return timeLen;**

**}**

**public void setTimeLen(String timeLen) {**

**this.timeLen = timeLen;**

**}**

**}**

**class MusicIdentifier {**

**private MusicMetaInfo info;**

**private String absolutePath;**

**MusicIdentifier(MusicMetaInfo minfo, String path) {**

**info = minfo;**

**absolutePath = path;**

**}**

**public MusicMetaInfo getInfo() {**

**return info;**

**}**

**public void setInfo(MusicMetaInfo info) {**

**this.info = info;**

**}**

**public String getAbsolutePath() {**

**return absolutePath;**

**}**

**public void setAbsolutePath(String absolutePath) {**

**this.absolutePath = absolutePath;**

**}**

**}**

**//Read dirs from temp file, then return an ArrayList<String> containing directories**

**ArrayList<String> readTempFile()**

**{**

**File file = new File(tempFilePath);**

**ArrayList<String> lst = new ArrayList<>();**

**if(!file.exists()) return lst;**

**try {**

**InputStreamReader read = new InputStreamReader(new FileInputStream(file), "utf8");**

**BufferedReader bufferedReader = new BufferedReader(read);**

**String lineTxt;**

**while ((lineTxt = bufferedReader.readLine()) != null)**

**{**

**lst.add(lineTxt);**

**}**

**bufferedReader.close();**

**read.close();**

**} catch(Exception e) { }**

**return lst;**

**}**

**//Currently supports mp3 files because of encoding issues**

**Boolean validateDirectory(String dir) {**

**return dir.endsWith("mp3") || dir.endsWith("wav");**

**}**

**void addOneMusicToFullList(String absolutePath)**

**{**

**try {**

**AudioFile f = AudioFileIO.read(new File(absolutePath));**

**Tag tag = f.getTag();**

**String artist = tag.getFirst(FieldKey.ARTIST);**

**String albumName = tag.getFirst(FieldKey.ALBUM);**

**int finalSlashIndex = absolutePath.lastIndexOf("/");**

**String title = absolutePath.substring(finalSlashIndex + 1, absolutePath.length() - 4);**

**int time = f.getAudioHeader().getTrackLength();**

**int minute = time / 60;**

**int sec = time % 60;**

**String timeStr = "" + minute + ":" + sec;**

**musicFullList.add(new MusicIdentifier(new MusicMetaInfo(title, artist, albumName, timeStr), absolutePath));**

**} catch (Exception e) {}**

**}**

**void generatePaneDirList(ArrayList<String> folderDirs)**

**{**

**ArrayList<String> dirs = new ArrayList<>(folderDirs);**

**for(String elem : dirs)**

**{**

**File baseFile = new File(elem);**

**ListView<String> listView = new ListView<>();**

**listView.setOnMouseClicked(new EventHandler<MouseEvent>() {**

**@Override**

**public void handle(MouseEvent mouseEvent) {**

**makePlayListAndShow();**

**}**

**});**

**File files[] = baseFile.listFiles();**

**if(files.length != 0)**

**{**

**ObservableList<String> ob = FXCollections.observableArrayList();**

**for (File file : files)**

**{**

**if (file.isDirectory() || !validateDirectory(file.getPath()))**

**{**

**;**

**} else {**

**ob.add(file.getName());**

**addOneMusicToFullList(file.getAbsolutePath());**

**}**

**}**

**listView.setItems(ob);**

**TitledPane pane = new TitledPane(elem, listView);**

**fileAccordion.getPanes().add(pane);**

**}**

**}**

**}**

**//Make background playlist, then show it in the tableView**

**void makePlayListAndShow()**

**{**

**//Make**

**backgroundPlayList.clear();**

**TitledPane currentExpandedPane = fileAccordion.getExpandedPane();**

**ListView<String> expandedList = (ListView<String>)currentExpandedPane.getContent();**

**ObservableList<String> fkList = expandedList.getItems();**

**for(String elem : fkList)**

**{**

**int resIndex = 0;**

**for(resIndex = 0; resIndex < musicFullList.size(); resIndex++)**

**{**

**String curTitle = musicFullList.get(resIndex).getInfo().getTitle();**

**if(curTitle.equals(elem.substring(0, elem.length() - 4)))**

**{**

**break;**

**}**

**}**

**backgroundPlayList.add(musicFullList.get(resIndex));**

**}**

**currentPlayingMusicIndexInPlayList = 0;**

**//Show**

**playListTable.getItems().clear();**

**for(int i = 0; i < backgroundPlayList.size(); i++)**

**{**

**MusicMetaInfo info = backgroundPlayList.get(i).getInfo();**

**playListTable.getItems().add(new MusicMetaInfo(info.getTitle(), info.getAuthor(), info.getAlbum(), info.getTimeLen()));**

**}**

**}**

**void playMusicWithPath(String absolutePath)**

**{**

**if(mainMedia != null)**

**{**

**mainPlayer.stop();**

**mainPlayer.currentTimeProperty().removeListener(progressListener);**

**}**

**mainMedia = new Media("file://" + absolutePath);**

**mainPlayer = new MediaPlayer(mainMedia);**

**mainPlayer.autoPlayProperty().addListener((observable) -> {**

**if(mainPlayer != null && mainPlayer.getStatus() == MediaPlayer.Status.STOPPED)**

**{**

**if(loopBtn.isSelected())**

**{**

**onForwardButtonClicked();**

**}**

**else if (shuffleBtn.isSelected())**

**{**

**Random fuckingRandom = new Random(System.currentTimeMillis());**

**currentPlayingMusicIndexInPlayList = fuckingRandom.nextInt(backgroundPlayList.size());**

**playMusicWithPath(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath());**

**}**

**else**

**{**

**;**

**}**

**}**

**});**

**mainPlayer.setAudioSpectrumListener((double timestamp, double duration, float[] magnitudes, float[] phases) -> {**

**vizContainer.getChildren().clear();**

**int i = 0;**

**int x = 125;**

**double y = contentPane.getHeight() / 2;**

**Random rand = new Random(System.currentTimeMillis());**

**for(float phase : phases)**

**{**

**int red = rand.nextInt(255);**

**int green = rand.nextInt(255);**

**int blue = rand.nextInt(255);**

**Circle circle = new Circle(10);**

**circle.setCenterX(rand.nextInt(860));**

**circle.setCenterY(rand.nextInt(632));**

**circle.setFill(Color.rgb(red, green, blue, 0.2));**

**vizContainer.getChildren().add(circle);**

**i += 5;**

**}**

**});**

**mainPlayer.currentTimeProperty().addListener(progressListener);**

**mainPlayer.setOnReady(() -> {**

**timeSlide.setValue(0);**

**timeSlide.setMax(mainPlayer.getMedia().getDuration().toSeconds());**

**});**

**setMetaInfoDisplay();**

**setTotalTimeDisplay();**

**mainPlayer.play();**

**setPlayButtonStatus();**

**setLrcFileIfExists();**

**}**

**void setLrcFileIfExists()**

**{**

**String songPath = backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath();**

**String prefix = backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getTitle();**

**String lrcName = prefix + ".lrc";**

**int lastSlashIndex = songPath.lastIndexOf("/");**

**songPath = songPath.substring(0, lastSlashIndex + 1);**

**File lrcFile = new File(songPath + lrcName);**

**if(!lrcFile.exists())**

**return;**

**try {**

**FileReader rder = new FileReader(lrcFile);**

**BufferedReader reader = new BufferedReader(rder);**

**String tmp;**

**lrcContainer = new ArrayList<>();**

**while((tmp = reader.readLine()) != null)**

**{**

**if(tmp.indexOf("www") != -1 || tmp.indexOf("com") != -1)**

**continue;**

**else**

**{**

**// Mess encoding while reading lrc file in Chinese website**

**byte[] bs = tmp.getBytes();**

**String appender = new String(bs, "UTF-8");**

**lrcContainer.add(appender);**

**}**

**}**

**} catch(Exception e) {}**

**}**

**void setPlayButtonStatus()**

**{**

**if(mainPlayer != null)**

**{**

**if(mainPlayer.getStatus() != MediaPlayer.Status.PLAYING)**

**{**

**playBtn.setText("||");**

**}**

**else**

**{**

**playBtn.setText("▷");**

**}**

**}**

**}**

**void setTotalTimeDisplay()**

**{**

**timeLabel.setText("0:00 / " + backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getTimeLen());**

**}**

**void setMetaInfoDisplay()**

**{**

**albumLabel.setText(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getAlbum());**

**singerLabel.setText(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getAuthor());**

**titleField.setText(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getTitle());**

**authorField.setText(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getAuthor());**

**if(albumLabel.getText() == "" || singerLabel.getText() == "")**

**{**

**albumLabel.setText("Modified music");**

**singerLabel.setText("Huabin Wang");**

**titleField.setText("No title");**

**authorField.setText("Huabin Wang");**

**String curAbsPath = backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath();**

**titleField.setText(curAbsPath.substring(curAbsPath.lastIndexOf("/") + 1, curAbsPath.indexOf(".")));**

**}**

**}**

**@FXML**

**private void addOneDirectory()**

**{**

**DirectoryChooser directoryChooser = new DirectoryChooser();**

**File newdir = directoryChooser.showDialog(null);**

**if(newdir != null)**

**{**

**String chosenDir = newdir.getPath();**

**File baseFile = new File(chosenDir);**

**ListView<String> listView = new ListView<>();**

**listView.setOnMouseClicked(new EventHandler<MouseEvent>() {**

**@Override**

**public void handle(MouseEvent mouseEvent) {**

**makePlayListAndShow();**

**}**

**});**

**File files[] = baseFile.listFiles();**

**if(files.length != 0)**

**{**

**ObservableList<String> ob = FXCollections.observableArrayList();**

**for (File file : files)**

**{**

**if (file.isDirectory() || !validateDirectory(file.getPath()))**

**{**

**;**

**} else {**

**ob.add(file.getName());**

**addOneMusicToFullList(file.getAbsolutePath());**

**}**

**}**

**listView.setItems(ob);**

**TitledPane pane = new TitledPane(chosenDir, listView);**

**fileAccordion.getPanes().add(pane);**

**}**

**}**

**}**

**@FXML**

**private void removeOneDirectory()**

**{**

**TitledPane paneForRemove = fileAccordion.getExpandedPane();**

**ObservableList<TitledPane> oblist = fileAccordion.getPanes();**

**oblist.remove(paneForRemove);**

**}**

**@FXML**

**private void clickTableRowForPlay()**

**{**

**ObservableList<TablePosition> selectedRowCells = playListTable.getSelectionModel().getSelectedCells();**

**if(selectedRowCells.size() != 0)**

**{**

**int selectedIndexInCurrentPlayList = selectedRowCells.get(0).getRow();**

**currentPlayingMusicIndexInPlayList = selectedIndexInCurrentPlayList;**

**String selectedMusicPath = backgroundPlayList.get(selectedIndexInCurrentPlayList).getAbsolutePath();**

**playMusicWithPath(selectedMusicPath);**

**}**

**}**

**@FXML**

**private void onPlayButtonClicked()**

**{**

**if(mainPlayer != null)**

**{**

**if(mainPlayer.getStatus() == MediaPlayer.Status.STOPPED || mainPlayer.getStatus() == MediaPlayer.Status.PAUSED)**

**{**

**if(mainPlayer.getStatus() == MediaPlayer.Status.PAUSED)**

**mainPlayer.play();**

**else**

**{**

**String firstMusicPath = backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath();**

**playMusicWithPath(firstMusicPath);**

**}**

**}**

**else**

**{**

**mainPlayer.pause();**

**}**

**setPlayButtonStatus();**

**}**

**else**

**{**

**playListTable.getFocusModel().focus(0);**

**String firstMusicPath = backgroundPlayList.get(0).getAbsolutePath();**

**playMusicWithPath(firstMusicPath);**

**}**

**}**

**@FXML**

**private void onForwardButtonClicked()**

**{**

**if(mainPlayer == null)**

**return;**

**mainPlayer.stop();**

**currentPlayingMusicIndexInPlayList++;**

**if(currentPlayingMusicIndexInPlayList >= backgroundPlayList.size())**

**currentPlayingMusicIndexInPlayList = 0;**

**playMusicWithPath(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath());**

**setPlayButtonStatus();**

**}**

**@FXML**

**private void onBackButtonClicked()**

**{**

**if(mainPlayer == null)**

**return;**

**mainPlayer.stop();**

**currentPlayingMusicIndexInPlayList--;**

**if(currentPlayingMusicIndexInPlayList < 0)**

**currentPlayingMusicIndexInPlayList = backgroundPlayList.size() - 1;**

**playMusicWithPath(backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getAbsolutePath());**

**setPlayButtonStatus();**

**}**

**@FXML**

**private void onLoopBtnClicked()**

**{**

**if(!loopBtn.isSelected())**

**{**

**loopBtn.setSelected(false);**

**}**

**else**

**{**

**loopBtn.setSelected(true);**

**if(shuffleBtn.isSelected())**

**shuffleBtn.setSelected(false);**

**}**

**}**

**@FXML**

**private void onRandBtnClicked()**

**{**

**if(!shuffleBtn.isSelected())**

**{**

**shuffleBtn.setSelected(false);**

**}**

**else**

**{**

**shuffleBtn.setSelected(true);**

**if(loopBtn.isSelected())**

**loopBtn.setSelected(false);**

**}**

**}**

**@FXML**

**private void onPlayListBtnClicked()**

**{**

**if(!playListBtn.isSelected())**

**{**

**playListBtn.setSelected(true);**

**}**

**else**

**{**

**playListBtn.setSelected(true);**

**songBtn.setSelected(false);**

**playListTable.setVisible(true);**

**vizContainer.setVisible(false);**

**authorField.setVisible(false);**

**titleField.setVisible(false);**

**lyricField.setVisible(false);**

**}**

**}**

**@FXML**

**private void onSongDisplayBtnClicked()**

**{**

**if(!songBtn.isSelected())**

**{**

**songBtn.setSelected(true);**

**}**

**else**

**{**

**songBtn.setSelected(true);**

**playListBtn.setSelected(false);**

**playListTable.setVisible(false);**

**vizContainer.setVisible(true);**

**authorField.setVisible(true);**

**titleField.setVisible(true);**

**lyricField.setVisible(true);**

**}**

**}**

**@FXML**

**public void initialize(URL location, ResourceBundle resources)**

**{**

**generatePaneDirList(readTempFile());**

**titleCol.setCellValueFactory(new PropertyValueFactory<>("title"));**

**authorCol.setCellValueFactory(new PropertyValueFactory<>("author"));**

**albumCol.setCellValueFactory(new PropertyValueFactory<>("album"));**

**timeCol.setCellValueFactory(new PropertyValueFactory<>("timeLen"));**

**titleCol.setPrefWidth(344);**

**authorCol.setPrefWidth(86);**

**albumCol.setPrefWidth(344);**

**timeCol.setPrefWidth(86);**

**titleCol.setResizable(false);**

**authorCol.setResizable(false);**

**albumCol.setResizable(false);**

**timeCol.setResizable(false);**

**titleCol.setStyle("-fx-alignment: CENTER");**

**authorCol.setStyle("-fx-alignment: CENTER");**

**albumCol.setStyle("-fx-alignment: CENTER");**

**timeCol.setStyle("-fx-alignment: CENTER");**

**playListTable.getColumns().add(titleCol);**

**playListTable.getColumns().add(authorCol);**

**playListTable.getColumns().add(albumCol);**

**playListTable.getColumns().add(timeCol);**

**timeSlide.valueProperty().addListener((observable) -> {**

**if(mainPlayer != null && mainPlayer.getStatus() == MediaPlayer.Status.PAUSED)**

**{**

**double dur = timeSlide.getValue() \* 1000;**

**mainPlayer.seek(Duration.millis(dur));**

**}**

**String totTime;**

**if(currentPlayingMusicIndexInPlayList == -1)**

**totTime = "0:00";**

**else**

**totTime = backgroundPlayList.get(currentPlayingMusicIndexInPlayList).getInfo().getTimeLen();**

**String currentTime;**

**Duration curDur = mainPlayer.getCurrentTime();**

**int curSec = (int)curDur.toSeconds();**

**int curMin = curSec / 60;**

**curSec = curSec % 60;**

**currentTime = "" + curMin + ":" + curSec;**

**timeLabel.setText(currentTime + " / " + totTime);**

**if(mainPlayer != null && mainPlayer.getStatus() == MediaPlayer.Status.PLAYING && curSec >= 1)**

**{**

**String fitTime = String.format("%02d", curMin) + ":" + String.format("%02d", curSec);**

**if(lrcContainer != null)**

**{**

**for(String elem : lrcContainer)**

**{**

**if(elem.contains(fitTime))**

**{**

**int sentenceStartIndex = elem.indexOf("]") + 1;**

**String displayStr = elem.substring(sentenceStartIndex, elem.length() - 1);**

**lyricField.setText(displayStr);**

**break;**

**}**

**}**

**}**

**}**

**});**

**progressListener = (observable, oldValue, newValue) -> timeSlide.setValue(newValue.toSeconds());**

**muteGroup = new ToggleGroup();**

**muteGroup.selectedToggleProperty().addListener((ov, toggle, new\_toggle) -> {**

**if(mainPlayer != null)**

**{**

**if (new\_toggle == null)**

**mainPlayer.setMute(false);**

**else**

**mainPlayer.setMute(true);**

**}**

**});**

**muteBtn.setToggleGroup(muteGroup);**

**playListBtn.setSelected(true);**

**playListTable.setVisible(true);**

**authorField.setVisible(false);**

**titleField.setVisible(false);**

**lyricField.setVisible(false);**

**vizContainer = new javafx.scene.Group();**

**vizContainer.resize(860, 632);**

**vizContainer.setVisible(false);**

**contentPane.getChildren().add(vizContainer);**

**}**

**}**

**mediaUI.fxml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<?import javafx.scene.text.\*?>**

**<?import javafx.scene.image.\*?>**

**<?import javafx.scene.control.\*?>**

**<?import javafx.scene.layout.\*?>**

**<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="841.0" prefWidth="1200.0" xmlns="http://javafx.com/javafx/8" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">**

**<children>**

**<HBox layoutX="-1.0" layoutY="70.0" prefHeight="632.0" prefWidth="1200.0">**

**<children>**

**<Accordion fx:id="fileAccordion" layoutY="77.0" prefHeight="632.0" prefWidth="340.0" />**

**<Pane fx:id="contentPane" layoutX="313.0" layoutY="77.0" prefHeight="632.0" prefWidth="860.0">**

**<children>**

**<TableView fx:id="playListTable" onMouseClicked="#clickTableRowForPlay" prefHeight="632.0" prefWidth="860.0" visible="false" />**

**<VBox layoutX="294.0" layoutY="189.0">**

**<children>**

**<TextField fx:id="titleField" alignment="CENTER" editable="false" layoutX="332.0" layoutY="138.0" prefHeight="25.0" prefWidth="273.0" text="Awesome Shit" />**

**<TextField fx:id="authorField" alignment="CENTER" editable="false" layoutX="383.0" layoutY="163.0" text="Damn Author" />**

**<Separator layoutX="469.0" layoutY="197.0" opacity="0.0" orientation="VERTICAL" prefHeight="119.0" prefWidth="4.0" />**

**<TextField fx:id="lyricField" alignment="CENTER" editable="false" layoutX="383.0" layoutY="372.0" text="No LRC loaded yet" />**

**</children>**

**</VBox>**

**</children></Pane>**

**</children>**

**</HBox>**

**<HBox fx:id="playControlBox" layoutX="60.0" layoutY="774.0">**

**<children>**

**<Button fx:id="backBtn" layoutX="14.0" layoutY="579.0" mnemonicParsing="false" onMouseClicked="#onBackButtonClicked" prefHeight="40.0" prefWidth="40.0" text="◀◀" />**

**<Button fx:id="playBtn" layoutX="54.0" layoutY="579.0" mnemonicParsing="false" onMouseClicked="#onPlayButtonClicked" prefHeight="40.0" prefWidth="40.0" text="▷">**

**<font>**

**<Font name="System Bold" size="15.0" />**

**</font></Button>**

**<Button fx:id="forwardBtn" layoutX="94.0" layoutY="579.0" mnemonicParsing="false" onMouseClicked="#onForwardButtonClicked" prefHeight="40.0" prefWidth="40.0" text="►►" />**

**</children>**

**</HBox>**

**<VBox fx:id="songInfoBox" alignment="CENTER" layoutX="287.0" layoutY="774.0" prefHeight="40.0" prefWidth="179.0">**

**<children>**

**<Label fx:id="singerLabel" alignment="CENTER" contentDisplay="CENTER" layoutX="241.0" layoutY="581.0" prefHeight="15.0" prefWidth="366.0" text="Awesome Shit" textAlignment="CENTER" />**

**<Separator opacity="0.0" orientation="VERTICAL" prefHeight="0.0" prefWidth="5.0" />**

**<Label fx:id="albumLabel" alignment="CENTER" contentDisplay="CENTER" layoutX="241.0" layoutY="601.0" prefHeight="15.0" prefWidth="261.0" text="Damn Author" textAlignment="CENTER" />**

**</children>**

**</VBox>**

**<Slider fx:id="timeSlide" blockIncrement="1.0" layoutX="498.0" layoutY="786.0" prefHeight="15.0" prefWidth="383.0" />**

**<Label fx:id="timeLabel" layoutX="896.0" layoutY="787.0" text="0:00 / 0:00" textAlignment="CENTER" />**

**<HBox fx:id="playModeBox" layoutX="1052.0" layoutY="774.0" prefHeight="40.0" prefWidth="120.0">**

**<children>**

**<ToggleButton fx:id="loopBtn" mnemonicParsing="false" onMouseClicked="#onLoopBtnClicked" prefHeight="46.0" prefWidth="60.0" text="Loop" />**

**<ToggleButton fx:id="shuffleBtn" mnemonicParsing="false" onMouseClicked="#onRandBtnClicked" prefHeight="40.0" prefWidth="57.0" text="Rand" />**

**</children>**

**</HBox>**

**<ImageView fx:id="albumImgView" fitHeight="40.0" fitWidth="40.0" layoutX="217.0" layoutY="774.0" pickOnBounds="true" preserveRatio="true">**

**<image>**

**<Image url="@defaultAlbumPic.png" />**

**</image>**

**</ImageView>**

**<ToggleButton fx:id="muteBtn" layoutX="986.0" layoutY="781.0" mnemonicParsing="false" prefHeight="25.0" prefWidth="52.0" text="Mute" />**

**<HBox fx:id="contentSwitchBox" layoutX="520.0" layoutY="23.0">**

**<children>**

**<ToggleButton fx:id="playListBtn" mnemonicParsing="false" onMouseClicked="#onPlayListBtnClicked" prefHeight="25.0" prefWidth="81.0" text="Play List">**

**<font>**

**<Font name="System Bold" size="13.0" />**

**</font></ToggleButton>**

**<ToggleButton fx:id="songBtn" mnemonicParsing="false" onMouseClicked="#onSongDisplayBtnClicked" prefHeight="25.0" prefWidth="74.0" text="Song">**

**<font>**

**<Font name="System Bold" size="13.0" />**

**</font></ToggleButton>**

**</children>**

**</HBox>**

**<HBox fx:id="addRemoveBox" layoutX="60.0" layoutY="724.0">**

**<children>**

**<Button fx:id="addListBtn" layoutX="60.0" layoutY="729.0" mnemonicParsing="false" onMouseClicked="#addOneDirectory" prefHeight="30.0" prefWidth="30.0" text="+" />**

**<Button fx:id="removeBtn" layoutX="89.0" layoutY="729.0" mnemonicParsing="false" onMouseClicked="#removeOneDirectory" prefHeight="30.0" prefWidth="30.0" text="—" />**

**</children>**

**</HBox>**

**</children>**

**</Pane>**

**Main.java**

**package sample;**

**import javafx.application.Application;**

**import javafx.fxml.FXMLLoader;**

**import javafx.scene.Parent;**

**import javafx.scene.Scene;**

**import javafx.stage.Stage;**

**public class Main extends Application {**

**@Override**

**public void start(Stage primaryStage) throws Exception{**

**Parent root = FXMLLoader.load(getClass().getResource("mediaUI.fxml"));**

**primaryStage.setTitle("MediaPlayer");**

**primaryStage.setResizable(false);**

**primaryStage.setScene(new Scene(root));**

**primaryStage.show();**

**}**

**public static void main(String[] args) {**

**launch(args);**

**}**

**}**

【小结或讨论】

通过本次实验，对Java的GUI开发，以及带用户图形界面程序开发的一般套路有所了解，为之后的课程内容打好了坚实的基础。