山东大学 软件 学院

面向对象 课程实验报告

|  |  |
| --- | --- |
| 实验题目：媒体播放器 | |
| 实验学时：16 | 实验日期： 2024.5.18 |
| 实验目的：  《面向对象技术》是计算机科学与技术、软件工程专业学生的一门必修课程。面向对象技术是当前最重要的程序设计思想与方法，利用抽象性、封装性、继承性、多态性等特性帮助人们开发出模块化、复用性强、易修改、可扩展的软件系统和程序。本实验通过若干个备选的相对复杂、独立的题目，使学生在实验过程中全面、系统的熟悉面向对象技术的基本概念和设计技术，分析、设计一个相对完整的应用，并使用面向对象编程语言进行实现，通过实验掌握使用面向对象技术进行程序设计、开发的思想和技能 | |
| 硬件环境：  CPU 2GHz、内存4GB、存储空间5GB。 | |
| 软件环境：  IntelliJ IDEA、OpenJDK-17 | |
| 实验步骤与内容：  1.实验要求：  媒体播放器：模拟实现一个多功能媒体播放器，它能播放音频（如mp3歌曲）、视频（选作，不属于基本要求）。媒体播放器界面，有进度条，可展示总播放时间、当前播放时间，进度条可拖动，有播放、暂停键，有快进、快退键。  **2.** **功能与目标分析**  音频播放功能  支持常见的音频格式，如mp3、wav等。  能够加载并播放音频文件。  提供播放、暂停、停止控制。  支持快进、快退操作，以调整播放进度。  视频播放功能（选作）  支持常见的视频格式，如mp4、avi等。  能够加载并播放视频文件。  提供与音频播放相同的播放、暂停、停止控制。  支持快进、快退操作。  可选功能：支持视频全屏播放、视频窗口大小调整等。  进度条与播放时间显示  显示当前播放进度，通过进度条展示。  进度条支持拖动，以直接跳转到指定播放位置。  显示总播放时间以及当前播放时间。  用户界面  设计直观易用的播放器界面。  布局合理，方便用户进行各种操作。  提供清晰的播放状态指示，如播放/暂停图标。  其他功能  支持音量调节。  支持播放列表管理，可添加、删除、编辑播放列表。  支持循环播放、随机播放等播放模式。  可选功能：支持歌词同步显示（仅限音频播放）。  **3.** **设计思路**  使用面向对象的思想来设计一个多功能媒体播放器，可以将不同的功能划分到不同的类中。以下是一个简化的类结构和每个类的属性和方法示例：  1. MediaPlayerController 类  这个类作为控制器，负责协调各个组件之间的交互。  属性：  MediaPlayer mediaPlayer: 播放媒体的对象。  MediaView mediaView: 用于显示视频的视图（可选）。  Slider progressSlider: 进度条。  Label currentTimeLabel: 显示当前播放时间的标签。  Label totalTimeLabel: 显示总播放时间的标签。  方法：  play(): 播放媒体。  pause(): 暂停媒体。  stop(): 停止媒体。  seek(double value): 跳转到指定进度。  updateProgressSlider(): 更新进度条的值。  updateTimeLabels(): 更新当前时间和总时间的显示。  handlePlayButtonAction(): 处理播放按钮的点击事件。  handlePauseButtonAction(): 处理暂停按钮的点击事件。  handleFastForwardButtonAction(): 处理快进按钮的点击事件。  handleRewindButtonAction(): 处理快退按钮的点击事件。  2. MediaPlayer 类  这个类封装了媒体播放的逻辑。  属性：  Media media: 要播放的媒体。  MediaPlayer mediaPlayer: JavaFX 提供的媒体播放器实例。  方法：  play(): 播放媒体。  pause(): 暂停媒体。  stop(): 停止媒体。  seek(Duration position): 跳转到指定位置。  getCurrentTime(): 获取当前播放时间。  getTotalDuration(): 获取媒体的总时长。  3. MediaPlayerView 类  这个类负责构建和更新播放器的用户界面。  属性：  MediaView mediaView: 用于显示视频的视图（可选）。  Slider progressSlider: 进度条。  Label currentTimeLabel: 显示当前播放时间的标签。  Label totalTimeLabel: 显示总播放时间的标签。  方法：  initializeUI(): 初始化用户界面组件。  updateProgressSlider(double value): 更新进度条的值。  updateTimeLabels(Duration currentTime, Duration totalTime): 更新当前时间和总时间的显示。  4. MainApp 类  这个类是JavaFX应用程序的入口点。  方法：  start(Stage primaryStage): 应用程序的启动方法，用于设置舞台和显示播放器界面。**UML图**    **4.实验步骤：**  设计一个使用JavaFX实现的多功能媒体播放器实验的思路可以遵循以下几个步骤：  1. 确定需求和功能  首先，明确媒体播放器的需求和功能。这包括音频播放、视频播放（可选）、进度条控制、播放时间显示、播放/暂停/快进/快退控制等。  2. 选择JavaFX组件  JavaFX提供了一系列的用户界面(UI)组件，如Button、Slider、Label等，用于构建播放器的界面。你需要根据功能需求选择相应的组件。  3. 设计用户界面  使用JavaFX的布局容器（如VBox、HBox、BorderPane等）和控件（如Button、Slider等）来设计播放器的用户界面。确保界面直观易用，并且布局合理。  4. 实现音频播放功能  使用JavaFX的Media和MediaPlayer类来实现音频播放功能。你需要加载音频文件到Media对象中，然后创建一个MediaPlayer实例来播放音频。  5. 实现视频播放功能（可选）  如果你选择实现视频播放功能，你可以使用JavaFX的MediaView类来显示视频。与音频播放类似，你需要加载视频文件到Media对象中，并使用MediaPlayer和MediaView来播放和显示视频。  6. 实现进度条和时间显示  使用JavaFX的Slider组件来实现进度条。你可以将Slider的value属性与MediaPlayer的当前播放时间绑定，以便用户可以通过拖动进度条来跳转到指定位置。同时，使用Label组件来显示总播放时间和当前播放时间。  7. 实现播放控制  使用Button控件来实现播放、暂停、快进、快退等控制功能。为这些按钮添加事件处理程序，以便在用户点击时执行相应的操作。  8. 整合和测试  将以上各个部分整合到一起，形成一个完整的媒体播放器应用。然后，对播放器进行详细的测试，确保所有功能都能正常工作，并且界面布局合理、操作流畅。  9. 优化和扩展  根据测试结果和用户反馈，对播放器进行优化和扩展。例如，你可以添加音量控制、播放列表管理、播放模式选择等功能。你还可以使用CSS来美化播放器的外观和风格。   1. **实现界面：**   如下图所示：  （1）半透明界面，背景自己决定    （2）    （3）    （4） | |
| 结论分析与体会：  通过这次媒体播放器的面向对象实验，我深刻体会到了面向对象设计的魅力和实用性。它让我学会了如何将复杂的问题分解为简单的部分，并通过类和对象来组织和管理这些部分。同时，我也认识到了面向对象设计在软件开发中的重要性，它不仅能够帮助我们提高代码的质量和效率，还能够使我们的系统更加灵活、易于扩展和维护。在未来的学习和工作中，我将继续深入学习面向对象设计的相关知识，并将其应用到实际项目中，不断提高自己的编程能力和解决问题的能力。 | |

**实验代码：**

package com.example.player;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.layout.AnchorPane;  
import javafx.scene.media.Media;  
import javafx.scene.media.MediaPlayer;  
import javafx.scene.media.MediaView;  
import javafx.scene.paint.Color;  
import javafx.stage.Stage;  
import javafx.stage.StageStyle;  
  
import java.io.File;  
import java.io.IOException;  
  
public class HelloApplication extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 stage.initStyle(StageStyle.*TRANSPARENT*);  
 FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 scene.setFill(Color.*rgb*(0, 0, 0, 0.1));  
 stage.setTitle("Simple Player!");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

package com.example.player;  
  
import javafx.beans.InvalidationListener;  
import javafx.beans.Observable;  
import javafx.beans.value.ChangeListener;  
import javafx.beans.value.ObservableValue;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.Slider;  
import javafx.scene.layout.Pane;  
import javafx.scene.media.Media;  
import javafx.scene.media.MediaPlayer;  
import javafx.scene.media.MediaView;  
import javafx.stage.FileChooser;  
import javafx.util.Duration;  
  
import java.io.File;  
import java.net.MalformedURLException;  
  
public class HelloController {  
 @FXML  
 private MediaView mediaView;  
 private MediaPlayer player;  
 @FXML  
 private Button operationBtn;  
 @FXML  
 private Slider timeSlider;  
 @FXML  
 private Label timelabal;  
 @FXML  
 private Pane mediaBorder;  
 @FXML  
 private Button kuaitui;  
 @FXML  
 private Button kuaijin;  
  
 private String total;  
 private int totalTime;  
 @FXML  
 public void initialize()  
 {  
 timeSlider.valueProperty().addListener(new ChangeListener<Number>() {  
 @Override  
 public void changed(ObservableValue<? extends Number> observableValue, Number number, Number t1) {  
 double value=t1.doubleValue();  
 if(value==player.getCurrentTime().toSeconds())  
 {  
  
 }  
 else {  
 player.seek(Duration.*seconds*(value));  
 }  
 kuaijin.setOnAction(event -> {  
 player.seek(Duration.*seconds*((double) player.getCurrentTime().toSeconds()+5));  
 });  
 kuaitui.setOnAction(event -> {  
 player.seek(Duration.*seconds*((double) player.getCurrentTime().toSeconds()-5));  
 });  
 }  
 });  
 }  
  
 @FXML  
 public void selectVideo() throws MalformedURLException {  
 FileChooser fileChooser=new FileChooser();  
 fileChooser.setSelectedExtensionFilter(new FileChooser.ExtensionFilter("mp4 file","mp4" ));  
 File file=fileChooser.showOpenDialog(null);  
 if (player!=null)  
 {  
 player.pause();  
 operationBtn.setText("播放");  
 timeSlider.setValue(0);  
 }  
 if(file!=null)  
 {  
 //System.out.println(file.getAbsolutePath());  
  
 Media media = new Media(file.toURI().toURL().toString());  
  
 player=new MediaPlayer(media);  
 player.setOnReady(new Runnable() {  
 @Override  
 public void run() {  
 Duration duration=player.getMedia().getDuration();  
  
 timeSlider.setMax(duration.toSeconds());  
 totalTime=(int)duration.toSeconds();  
 timelabal.setText("00:00/"+timeFormat((int)duration.toSeconds()));  
 }  
 });  
  
 player.currentTimeProperty().addListener(new InvalidationListener() {  
 @Override  
 public void invalidated(Observable observable) {  
 timeSlider.setValue(player.getCurrentTime().toSeconds());  
  
 timelabal.setText(timeFormat((int)player.getCurrentTime().toSeconds())+"/"+timeFormat(totalTime));  
 }  
 });  
 mediaView.setMediaPlayer(player);  
 }  
 }  
  
  
 @FXML  
 public void play()  
 {  
 if(player==null)  
 {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*);  
 alert.setHeaderText("暂未加载任何视频");  
 alert.setContentText("请先选择需要播放的视频");  
 alert.show();  
 }  
 else {  
 if(operationBtn.getText().equals("播放")) {  
 player.play();  
 operationBtn.setText("暂停");  
 }  
 else {  
 player.pause();  
 operationBtn.setText("播放");  
  
 }  
 }  
 }  
 private String timeFormat(int seconds)  
 {  
 int minute=seconds/60;  
 int second=seconds%60;  
 String minuteStr="";  
 String secondStr="";  
 if(minute<10)  
 {  
 minuteStr="0"+minute;  
 }  
 else {  
 minuteStr=""+minute;  
 }  
 if(second<10)  
 {  
 secondStr="0"+second;  
 }  
 else {  
 secondStr=""+second;  
 }  
 return minuteStr+":"+secondStr;  
 }  
  
 public void close(ActionEvent event) {  
 System.*exit*(0);  
 }  
}

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.Slider?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.media.MediaView?>  
  
<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="650.0" prefWidth="934.0" style="-fx-background-color: rgba(0, 0, 0, 0.5);" stylesheets="@css/ElementFX.css" xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.player.HelloController">  
 <children>  
 <Button layoutX="185.0" layoutY="614.0" mnemonicParsing="false" onAction="#selectVideo" text="选择视频文件" />  
 <Button fx:id="operationBtn" layoutX="474.0" layoutY="614.0" mnemonicParsing="false" onAction="#play" text="播放" />  
 <Pane fx:id="mediaBorder" layoutX="117.0" layoutY="74.0" prefHeight="402.0" prefWidth="706.0" style="-fx-border-color: black;">  
 <children>  
 <MediaView fx:id="mediaView" fitHeight="400.0" fitWidth="800.0" layoutX="-5.0" layoutY="1.0" />  
 </children></Pane>  
 <Slider fx:id="timeSlider" layoutX="72.0" layoutY="520.0" prefHeight="14.0" prefWidth="790.0" />  
 <Label fx:id="timelabal" layoutX="433.0" layoutY="549.0" text="00:00/00:00" />  
 <Button fx:id="kuaitui" layoutX="72.0" layoutY="557.0" mnemonicParsing="false" text="快退" />  
 <Button fx:id="kuaijin" layoutX="807.0" layoutY="557.0" mnemonicParsing="false" text="快进" />  
 <Button layoutX="889.0" layoutY="14.0" mnemonicParsing="false" onAction="#close" text="X" />  
 </children>  
</Pane>