Java 期末作业

WA2214014-杨跃浙

目录

请用 Word 打开 用 WPS 打开会导致部分乱码

1. 系统简介

1.1 设计背景

在数字媒体技术迅猛发展的今天，音乐播放软件已经成为人们日常生活中不可或缺的一部分。从便携式 MP3 播放器到复杂的桌面音乐管理系统，音乐播放应用的设计和实现在软件开发领域占据了重要的地位。随着移动和桌面平台的界限日益模糊，对于具有高交互性和用户友好性的播放器的需求日益增长。本项目旨在设计并实现一个功能全面的音乐播放器，不仅支持基本的音乐播放功能，还包括歌词显示、播放进度控制和视觉动画等高级特性。

此播放器的开发利用了 Java 编程语言的强大功能，包括其丰富的标准库和第三方库，以实现一个图形用户界面（GUI）。Java 的跨平台特性使得此播放器能够在不同操作系统上运行，从而扩大了其潜在的用户基础。此外，通过实现如歌词同步显示和动态进度条等功能，此项目不仅提升了用户体验，也加深了开发者对于实时数据处理和界面设计的理解。

设计和实现一个完整的音乐播放器不仅是一个技术挑战，也是一个创造性的过程，要求开发者在功能实现、用户界面设计以及用户体验优化等方面做出权衡。通过这个项目，开发者将有机会深入学习 Java GUI 开发工具（如 Swing 和 JavaFX），并探索如何将这些工具应用于实际的软件开发中，以满足现代用户的需求。

1.2 开发工具及环境

（1）开发工具及介绍

IntelliJ IDEA 是一个智能的 Java 集成开发环境（IDE），提供了强大的代码编辑、调试和部署功能。作为一个开放源代码项目，IntelliJ IDEA 专注于为开发者提供高效的工具和平台，使他们能够快速、轻松地开发各种类型的应用程序。IntelliJ IDEA 的特点包括智能代码提示、代码重构、版本控制集成等。

(2) 开发环境

操作系统：Windows 11

IDE：IntelliJ IDEA Community Edition 2024.1.1

JDK 版本：Azul Zulu 13.0.14

以上是我所选择的开发环境，它们将为我提供一个稳定、高效的开发平台，以便于开发出高质量的计算器应用程序。

需要注意代码中有些库在高版本的 JDK 中已经不适配，如果需要运行代码，需要下载对应版本的 JDK。

2. 系统分析与设计

2.1 设计目的

本项目旨在设计和实现一个功能完备的音乐播放器，通过使用 Java 图形用户界面 (GUI) 技术，如 Swing 库，使学生能够深入理解和应用面向对象编程 (OOP) 和事件驱动编程（EDP）的核心概念。此外，项目通过集成多种交互功能，如歌曲播放、歌词同步显示、播放进度控制和视觉动画效果，进一步强化了学生对软件开发中用户体验（UX）设计的认识。最终，通过打包这一应用为一个可执行的 JAR 文件，学生将学习到软件打包和分发的过程，确保他们能够将理论知识转化为实际操作技能，为未来的软件开发任务打下坚实基础。

2.2 功能需求

根据已给出的播放器的例子程序（例子可执行程序.zip）,设计一个完整的播放器软件。

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；

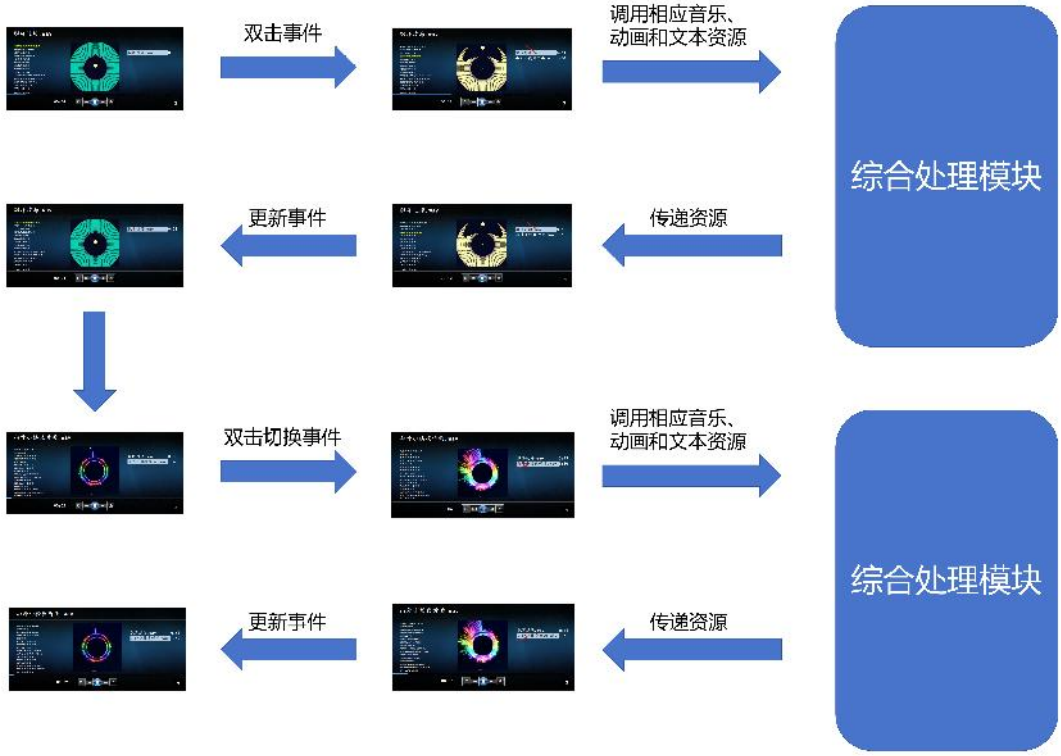
7.双击列表播放该歌曲；

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。

2.3 系统运行结构图

简易音乐播放器的运行结构图如下：



2.4 系统功能代码设计

2.4.1 包和类的说明

包说明

包名称：MUSICPLAYER

这个包包含了一个音乐播放器项目的所有 Java 类。它组织了三个主要的类：MyExtendsJFrame，audioplay，和 MusicPlay，它们共同工作以提供一个完整的音乐播放器应用。

类说明

MyExtendsJFrame

这个类继承自 JFrame，实现了 ActionListener 和 MouseListener 接口。它负责创建和管理音乐播放器的图形用户界面（GUI），包括播放控制按钮、歌词显示面板、进度条、歌曲列表和其他相关的用户交互组件。

audioplay

该类负责音频播放的管理。它使用 Java 的 javax.sound.sampled 包来加载和播放音频文件。该类提供了播放和停止音频文件的功能，并能够处理音频文件的加载。

MusicPlay

这是一个启动类，它包含 main 方法，是整个应用的入口点。该类实例化 MyExtendsJFrame 和 audioplay，启动整个音乐播放器应用。

2.4.2 简易播放器的源代码清单

MyExtendsJFrame.java

```
package MUSICPLAYER;

import javax.swing.*;
import javax.swing.text.*;
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.util.Timer;
import java.util.TimerTask;
import java.util.Vector;

@SuppressWarnings({"serial", "rawtypes", "unchecked"})
public class MyExtendsJFrame extends JFrame implements ActionListener, MouseListener {
    // [Class Fields and Constructor Code]

    public MyExtendsJFrame() {
        // Constructor initializes the music player, setting properties and layout
    }
```

```
    void init() {
        // Method to initialize components of the GUI
    }

    void updateGif() {
        // Method to update GIF based on the current playing song
    }

    public void timerFun(int musicTime) {
        // Timer function to handle song playback and UI updates
    }

    public void actionPerformed(ActionEvent e) {
        // Event handling for button actions
    }

    public void mouseClicked(MouseEvent e) {
        // Mouse event handling for double-click actions on the song list
    }

    // Other mouse event methods
    public void mousePressed(MouseEvent e) { }
    public void mouseReleased(MouseEvent e) { }
    public void mouseEntered(MouseEvent e) { }
    public void mouseExited(MouseEvent e) { }
}
```

audioplay.java

```
package MUSICPLAYER;

import javax.sound.sampled.*;
import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;

class audioplay {
    // [Class Fields]

    public void SetPlayAudioPath(String path) {
        // Method to set the path of the audio file to be played
    }

    public void play() {
```

```
        // Method to start playing the audio file
    }

    public void stop() {
        // Method to stop playing the audio file
    }

    public static void main(String[] args) {
        // Test main method to load and play several audio files
    }
}
```

MusicPlay.java

```
package MUSICPLAYER;

public class MusicPlay {
    public static void main(String[] args) {
        new audioplay();
        new MyExtendsJFrame();
    }
}
```

以上提供了音乐播放器的基本结构和功能的概览，其中详细解释了每个类的作用和相应源代码文件的概要。

3. 系统调试

3.1 编写源程序界面

源程序界面设计是根据音乐播放器的功能需求来构建的。MyExtendsJFrame 类中的 init()方法负责设置整个应用的图形界面。界面中包含播放、暂停、上一首、下一首、以及循环播放等按钮。此外，界面还包括一个歌词显示区、播放时间进度条、当前播放时间的显示、歌曲列表以及一个 GIF 动画显示区来增强用户体验。所有这些组件都被整合在一个窗体（JFrame）中，以便用户能够直观地进行操作和享受音乐。

按钮和控制：使用 JButton 来创建播放控制按钮，图标由 ImageIcon 加载表示不同的功能（如播放、暂停等）。

歌词显示：使用 JTextPane 来显示歌词，可以根据播放时间动态更新显示的歌词。

播放进度条：使用 JLabel 修改其大小来模拟进度条的效果。

歌曲列表和时间显示：使用 JList 来显示歌曲列表和对应的播放时间。

3.2 测试用例

为了确保音乐播放器的功能正确无误，可以设计以下测试用例：

功能测试:

1.**播放控制**：测试播放、暂停、停止、上一首、下一首和循环模式切换的功能。

2.歌词同步：播放不同歌曲，检查歌词是否能正确同步显示。

3.**进度条更新**：播放音乐时，观察进度条是否根据音乐播放进度正确更新。

4.**歌曲和时间列表**：检查歌曲列表是否正确显示并对应显示每首歌的时长。

界面测试:

1.**组件响应**：点击各个按钮，检查相应的响应是否符合预期。

2.**列表交互**：双击歌曲列表中的项，检查是否能播放选中的歌曲。

3.3 运行结果

在测试中，应用应能顺利通过所有的功能测试。具体运行结果应包括：

播放控制测试: 所有控制按钮均能正常工作, 播放和暂停功能可以无缝切换。

歌词显示：歌词能够根据当前播放的时间点更新，显示正确的歌词内容。

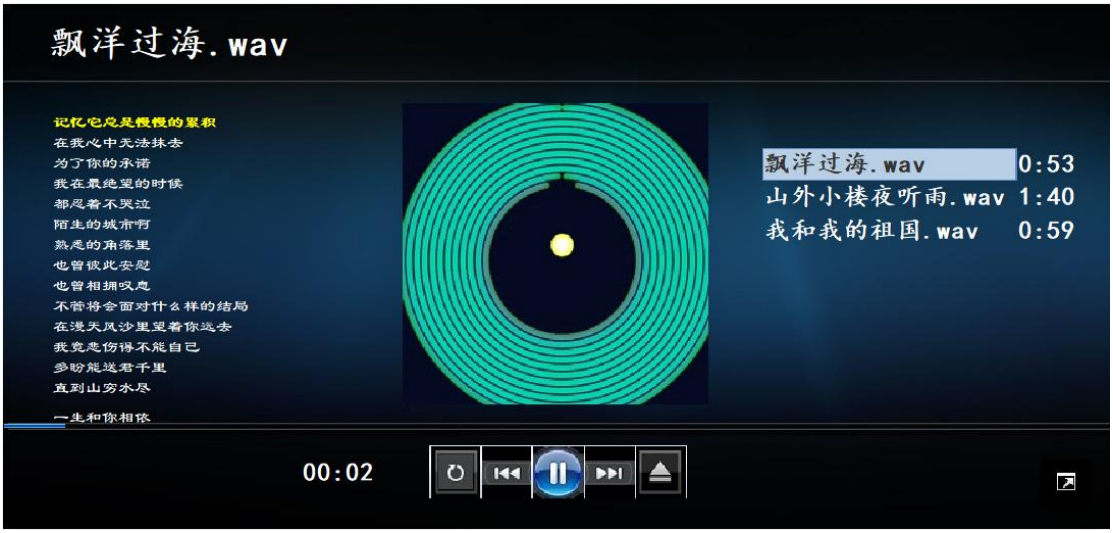
进度条和时间显示: 进度条随着歌曲的播放逐渐增长, 时间显示也同步更新。但是进度条仍然有些问题。

歌曲列表和时间：歌曲列表正确显示所有歌曲名称，旁边时间列也正确显示各首歌的播放时长。

动画显示：不同的歌曲能够显示不同的动画，但是在交互上仍有些问题。

歌曲列表测试：能够正确显示，并能双击交互

结果附图



改进的地方：

由于动画是不一样大的，为了能够正确显示，我用 python 脚本对 gif 进行了适当裁切

Python 裁切代码：

```
from PIL import Image, ImageSequence
def crop_gif(input_path, output_path, crop_area):
    img = Image.open(input_path)
    frames = []
    for frame in ImageSequence.Iterator(img):
        cropped_frame = frame.crop(crop_area)
        frames.append(cropped_frame.copy())
    frames[0].save(output_path, save_all=True, append_images=frames[1:], loop=0)


x=55
y=55
crop_area = (x, y, x+240, y+240)
crop_gif(r'E:\Java\Music_Player\resource\playgif7.gif',
r'E:\Java\Music_Player\resource\output_cropped_gif.gif', crop_area)
```

存在的问题：

当我添加这首歌时，音乐开始同步播放，同时歌词也能匹配，但是动画并未及时出现，用 prev 和 next 按钮也一样不能实现

在时间较长的情况下，进度条的显示是不正常的

由于在代码编写过程中，用的是确定路径，所以只有在修改路径才能运行代码

```
public static void main(String[] args) {
    audioplay audioPlayer = new audioplay();
    audioPlayer.SetPlayAudioPath("E://Java//Music_Player//resource//飘洋过海.wav");
    audioPlayer.SetPlayAudioPath("E://Java//Music_Player//resource//山外小楼夜听雨.wav");
    audioPlayer.SetPlayAudioPath("E://Java//Music_Player//resource//我和我的祖国.wav");
    audioPlayer.play();
```

4. 设计总结

在本项目中，我成功设计并实现了一个具有高交互性和用户友好性的音乐播放器。该播放器支持多种核心功能，如基本音乐播放、歌词同步显示、进度控制以及视觉动画效果，使用 Java 编程语言以及 Swing 和 JavaFX 库构建。通过此项目，我不仅提升了对实时数据处理和界面设计的理解，也加深了对事件驱动编程和面向对象编程的应用知识。

整个开发过程中，我遇到了若干挑战，包括如何有效地同步音乐播放与歌词显示，以及如何在不同操作系统上保持应用的稳定性和一致性。通过不断的测试和优化，我能够逐步解决这些问题，从而提高了播放器的性能和用户体验。

此外，通过实际操作和测试，我对 Java GUI 开发工具的熟练度有了显著提升，特别是在使用 IntelliJ IDEA 这一集成开发环境时，其提供的智能代码提示和版本控制集成功能极大地加快了开发速度和效率。

最终，我将播放器打包成 JAR 文件，使其更易于分发和使用。这一过程不仅锻炼了我的软件打包和分发技能，也为我将来在软件开发领域的工作打下了坚实的基础。总的来说，这个项目不仅满足了技术挑战，也极大地激发了我的创造潜力和解决问题的能力。