课程设计——Java 程序设计

音乐播放器

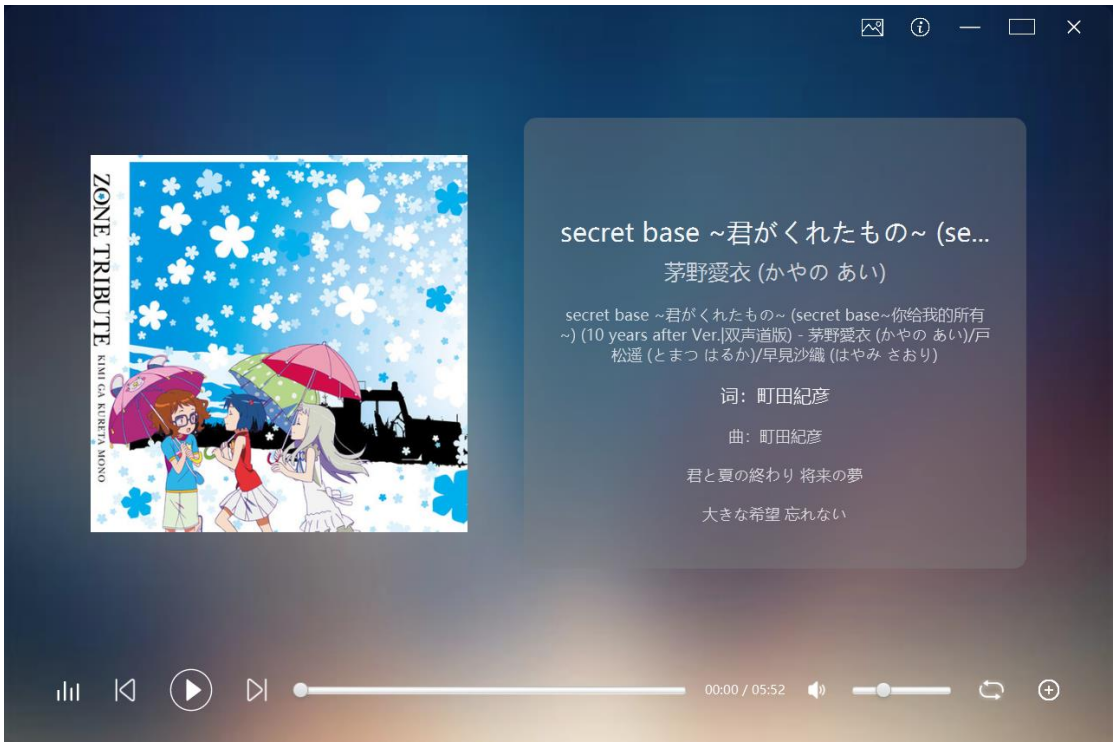
程序设计报告

Soplayer

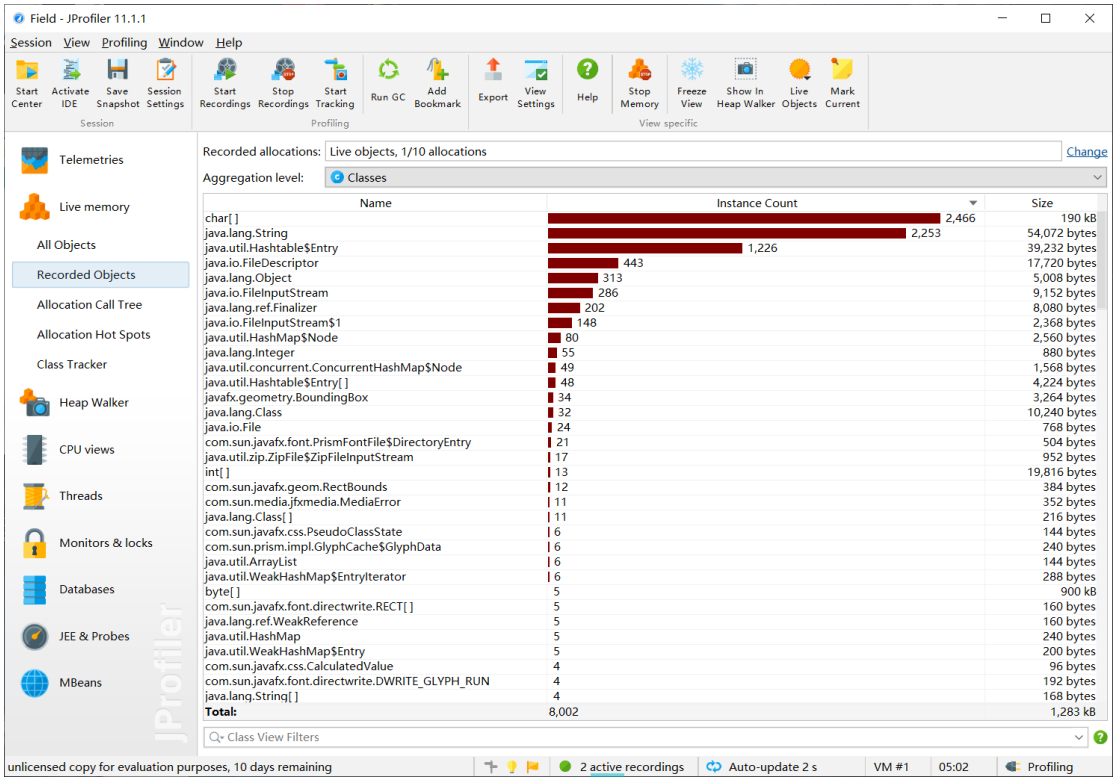
Name： Cap

Email: littlegreedy@qq.com

Project Time: 2020.02.14-2020.03.04



IDEA JProfiler 性能分析插件，图表为程序运行时对象的数量（1/10 样本）



目录

音乐播放器

一、概述

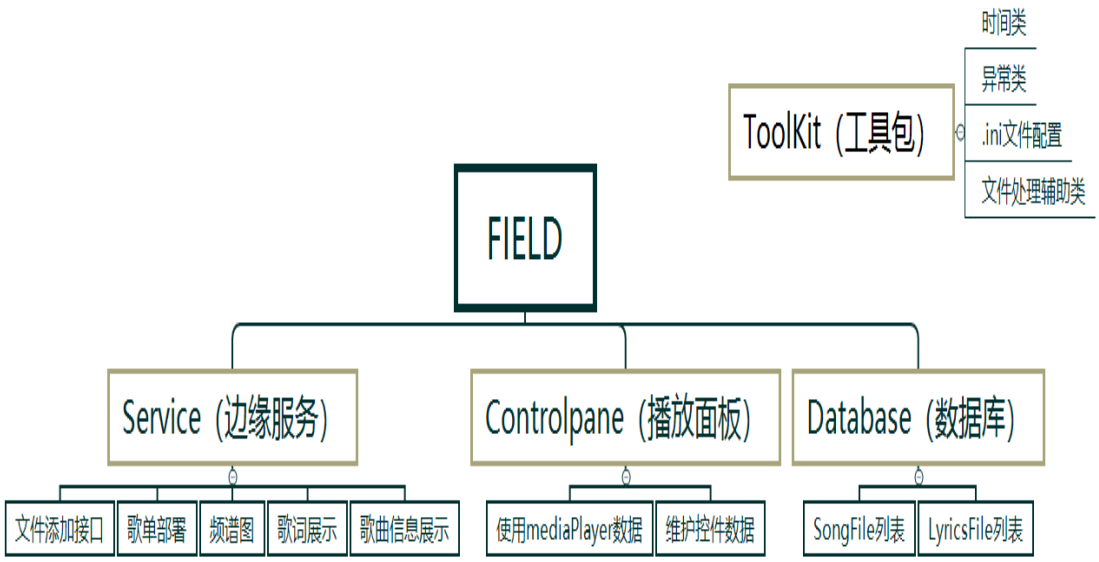
1）主要内容

采用 JavaFX 组件完成的一款小巧、界面精美的本地音乐播放器，支持添加本地音乐、歌曲及歌词文件解析、歌词滚动、歌单列表操作、频谱图展示、歌词海报显示、自定义背景、系统托盘控制等。同时使用.ini 文件记录应用设置信息。

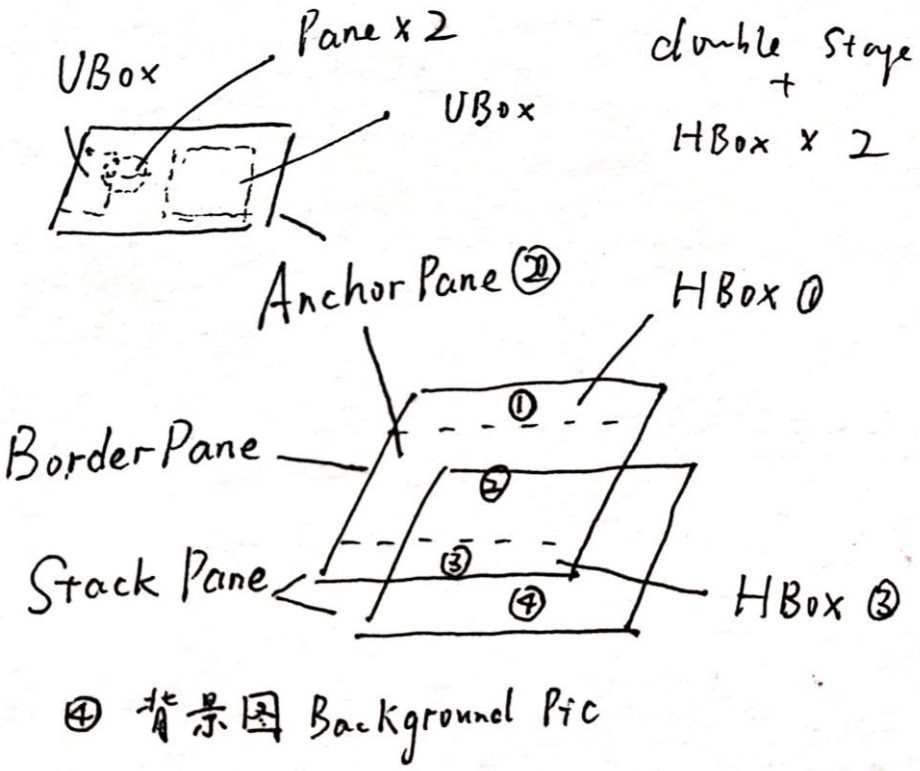
2）已实现的目标

- 界面简洁而精美且支持自定义背景
- 支持播放的音乐格式：.mp3 文件、.wav 文件、.aac 文件
- 支持解析歌词并展示（.lrc 文件）
- 支持解析歌词文件（缩略图、专辑、时长等）
- 支持频谱图动效
- 支持拖动添加文件
- 支持系统托盘控制
- 支持歌单列表控制

3）项目功能架构图、主要功能流程图



4）布局示意图



二、主要类的设计

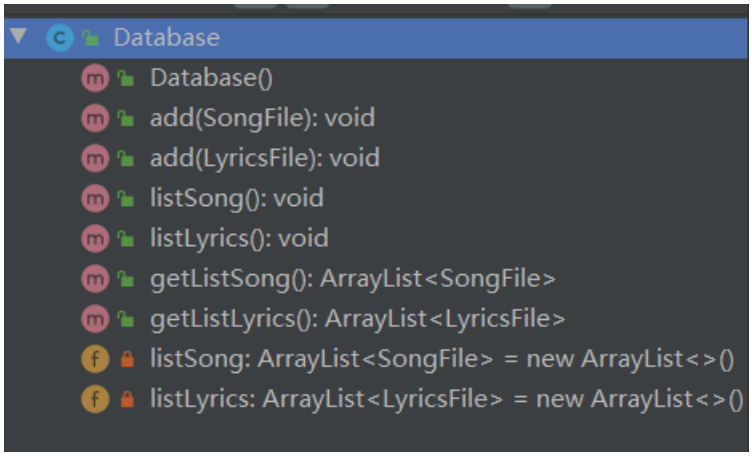
1）项目设计综括

包 database 目录下包含图片资源和歌曲、歌词文件资源，作为 model 存放数据；而播放器视图及控件联系并不紧密，则将 **view** 和 **controller** 两部分合并。包 controlpane 和包 layout 为播放器主面板及布局文件。包 service 为功能实现模块，主要分为文件添加、菜单服务、可视化服务（关于、歌曲信息的文本及图像展示）。包 toolKit 为封装好了的小工具包。

2）依次介绍各个主要类的设计

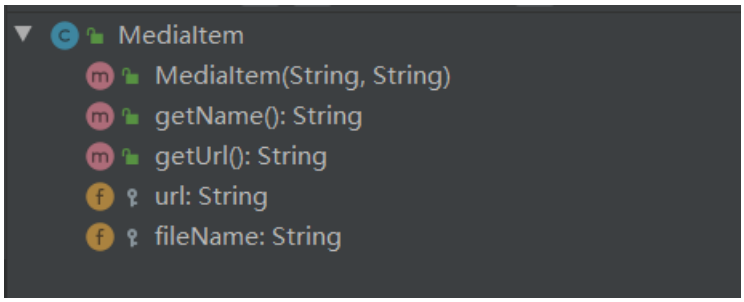
a） Database 类

主要负责存储及更新歌曲文件类 SongFile 的列表、歌词文件类 LyricsFile 的列表，是播放器的数据池。Database 类将歌曲和歌词文件组织为一个一维结构，便于管理。它并未包含对文件列表进行管理的相关方法，只负责提供数据，业务逻辑层和数据层进行分离是为了方便维护，降低了耦合度。如果迭代过程中需新增管理方法，可直接继承处理即可，这样增加了可拓展性，同时也为日后增加数据库支持提供接口。

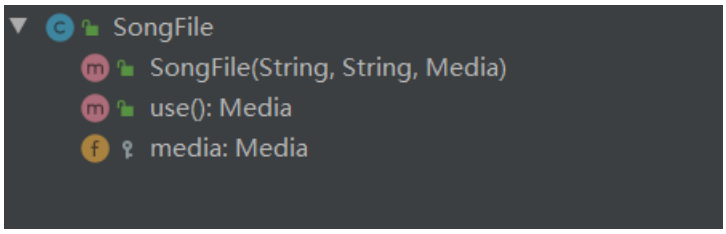


b) MediaItem 类、SongFile 类和 LyricsFile 类

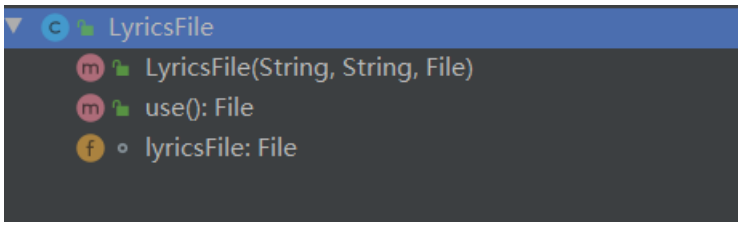
MediaItem 类是歌曲文件类 SongFile、歌词文件类 LyricsFile 的父类，维护共同含有的文件名及文件路径 url，其中 url 是文件重要参数，程序可以通过 url 获取文件也文件名。鉴于 media、Mediaplayer 媒体应用的特点，可在维持原功能不变且改动极少的情况下，增加 MediaItem 类的视频子类，使实现简单的音、视频播放器成为可能。



SongFile 类存储 media 媒体文件，考虑到 Mediaplayer 类的构造方法及简化程序的目标，这样使用可以将歌曲文件列表看成一个 media 列表。



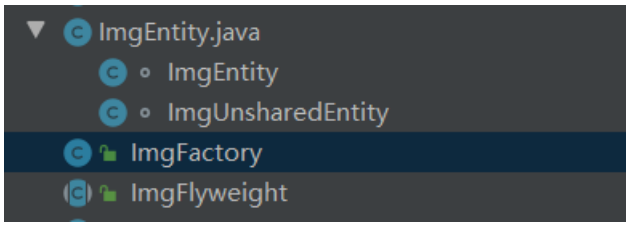
LyricsFile 类存储歌词文件



C) 图片引入的四种类。

随着可视化效果的进一步多元化，播放器可能需要更多的引用同种图片，而这些图片仅

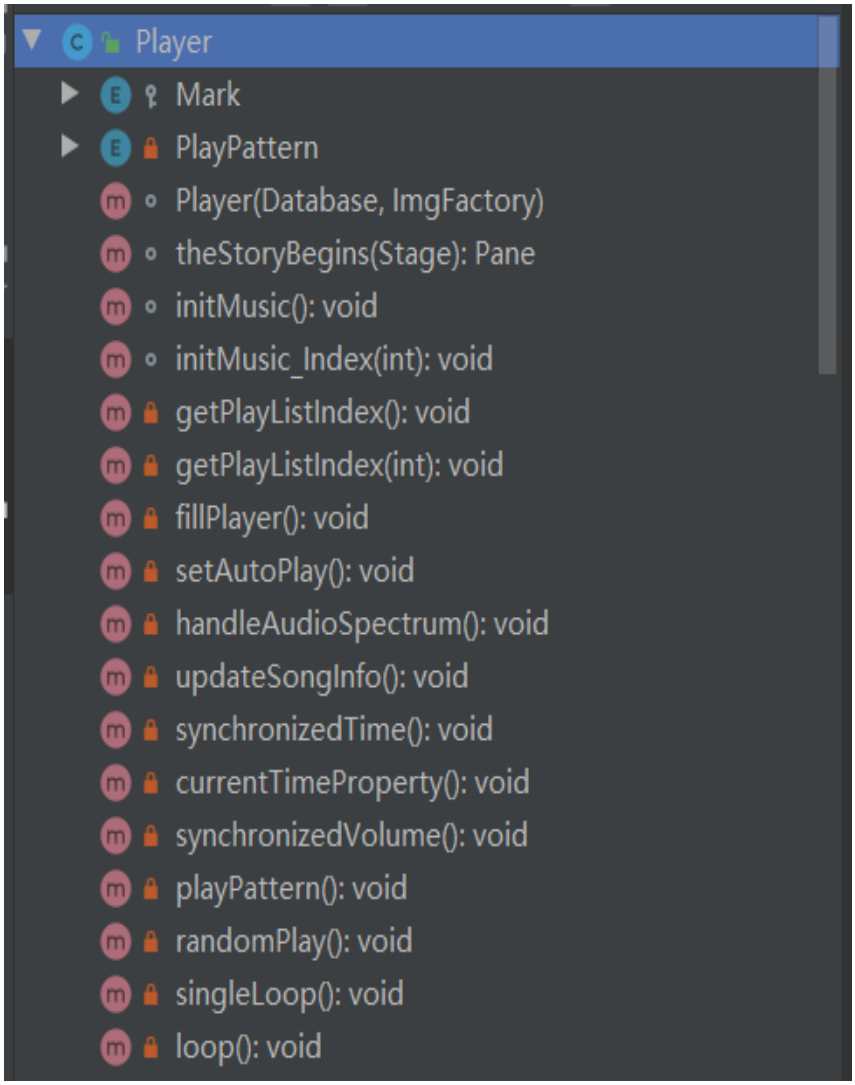
仅是大小和锚定位置不同。为了减少创建对象的数量，减少内存占用和提高性能，此处试用享元模式。



c) Player 类

Player 类作为播放器的核心，其它类皆围绕此类的外部运作。Player 类主要负责调度播放类 Mediaplayer 和收集播放文件数据、初始化布局界面和非模块化的所有简单控件。它集中了控制代码部分和视图控件部分，只预留启动、关闭两类接口。

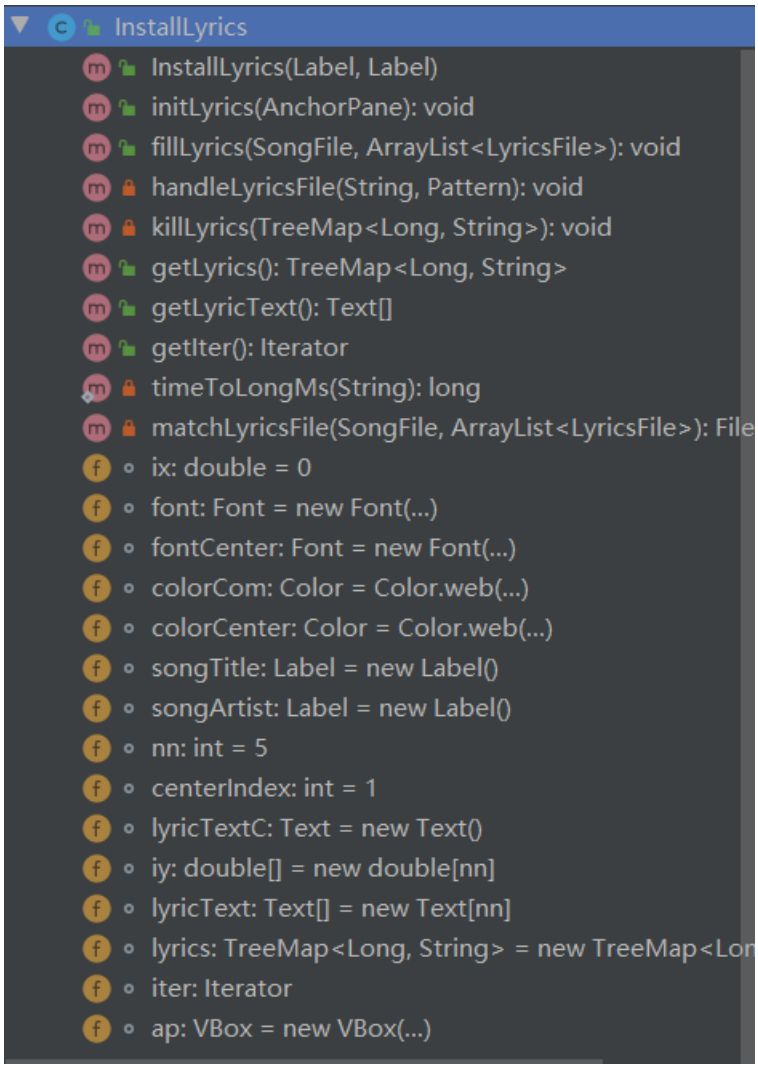
此处显示其所有的成员方法：

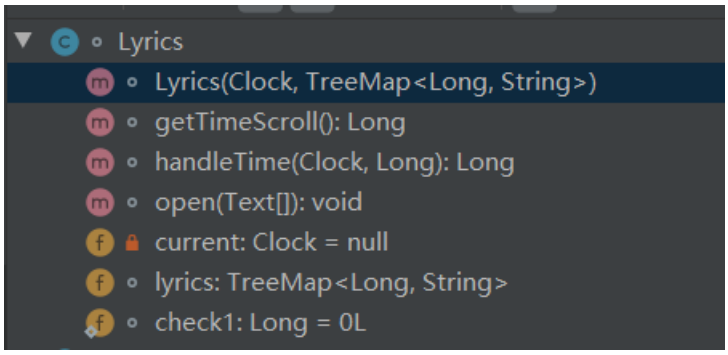


d) InstallLyrics 类、Lyrics 类

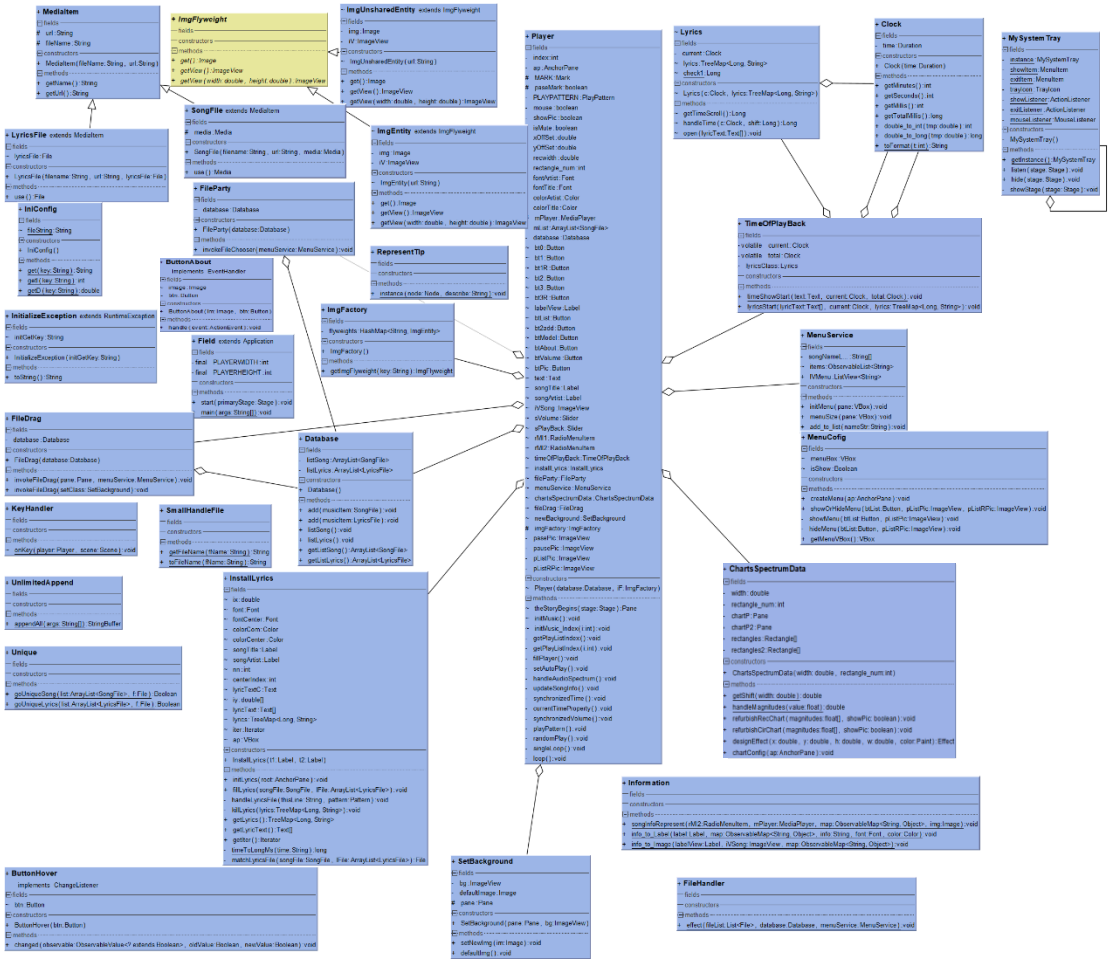
InstallLyrics，顾名思义，是用来初始化安装歌词的，它将在两种情况下启用：一、播放器打开时初始化布局（由构造器和 initLyrics 方法完成）；二、切歌时匹配并解析将播放的歌词（由 fillLyrics 方法与四个私有方法协助完成）。歌词从获取、处理到显示是本课题的一个小难点，于是在此用两个类将问题分层解决，前者完成静态地获取和初步显示，后者则处理 TreeMap 数据实时更新歌词并显示。

此处显示其所有的成员：





3）综上所述，各个类之间的关系用 UML 图显示



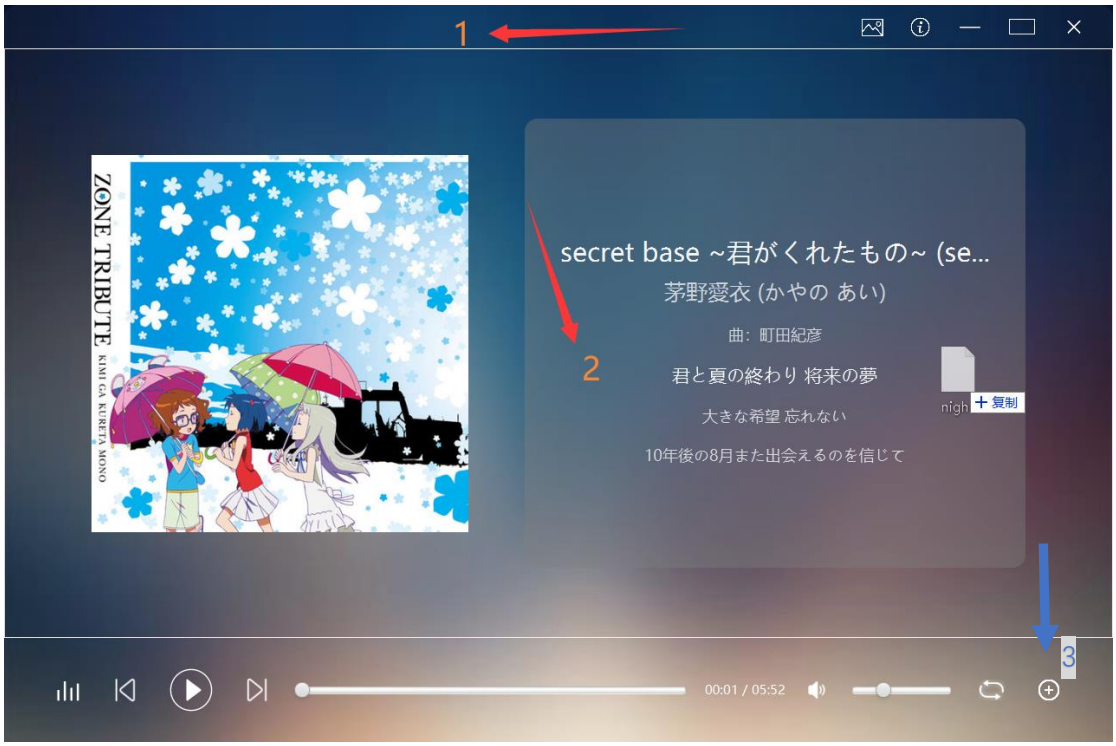
三、程序的功能特点和运行操作方法

在 Java 虚拟机环境下，通过执行 .jar 文件启动。

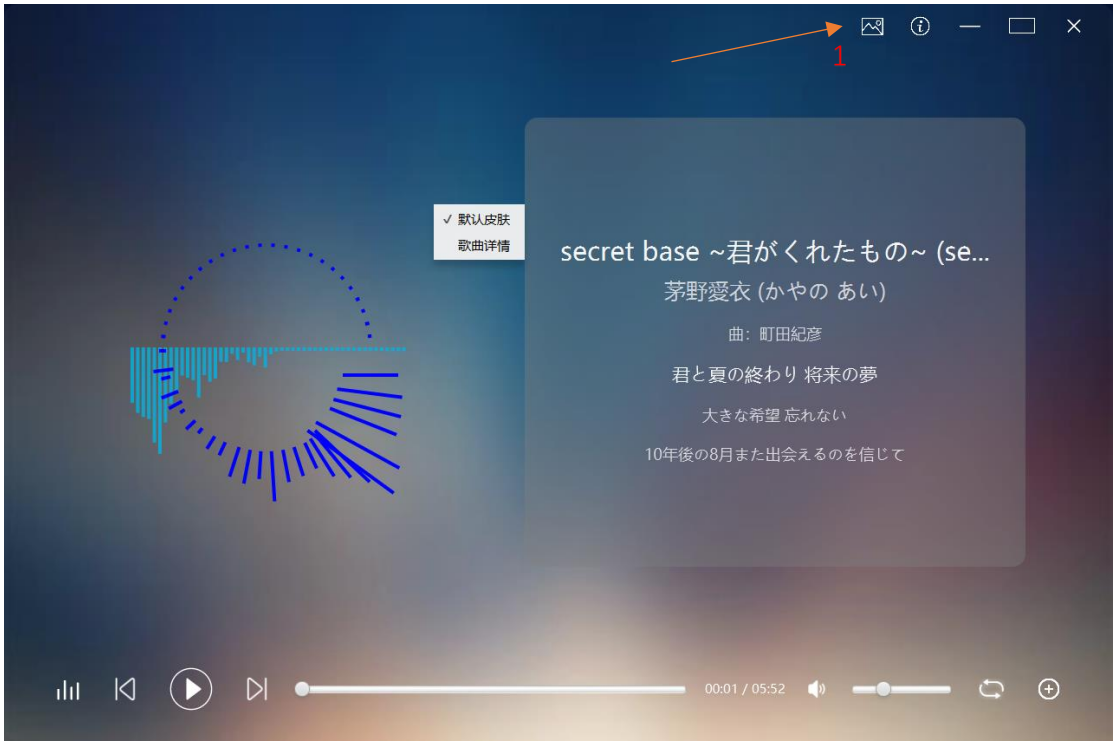
开发版本为 java version "1.8.0_241"

建议运行环境版本不低于 Java(TM) SE Runtime Environment (build 1.8.0_241-b07)

鼠标在按钮上悬停会显示说明。

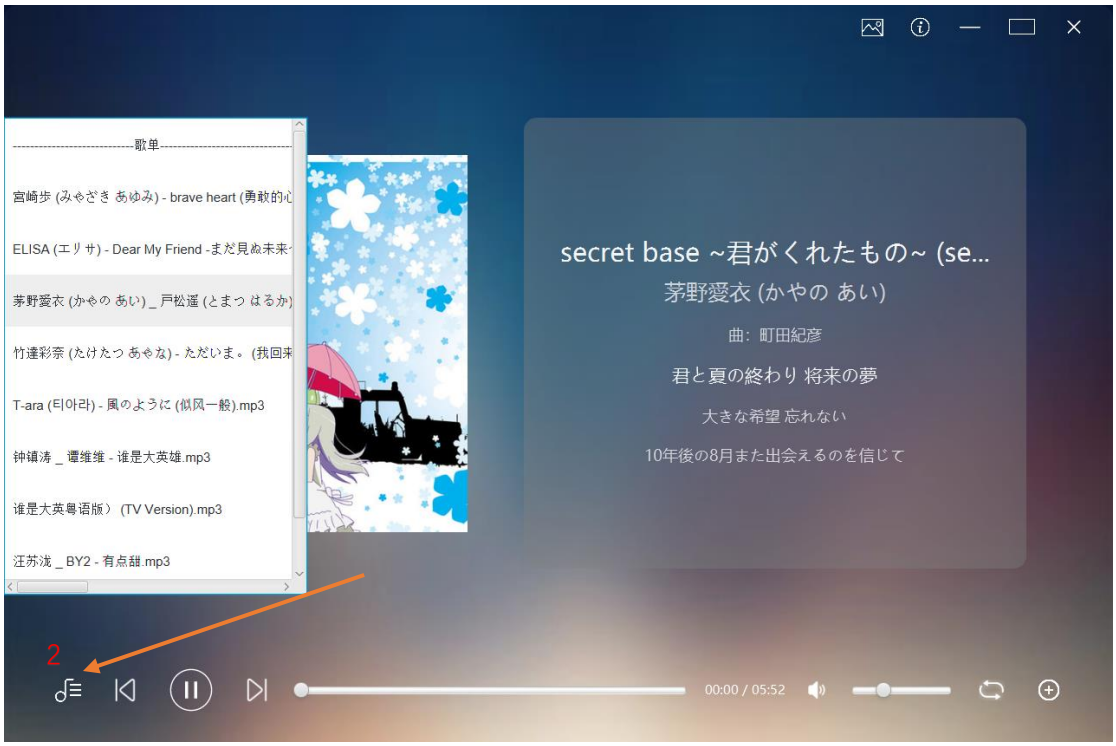


添加歌曲和歌词，可通过拖动文件入中央布局（数字 2 所在的整块区域），也可通过鼠标单击加号图标（数字 3）或快捷键（Ctrl+0），拖动图片入顶端布局（数字 1 所在的整块区域）即可设置背景

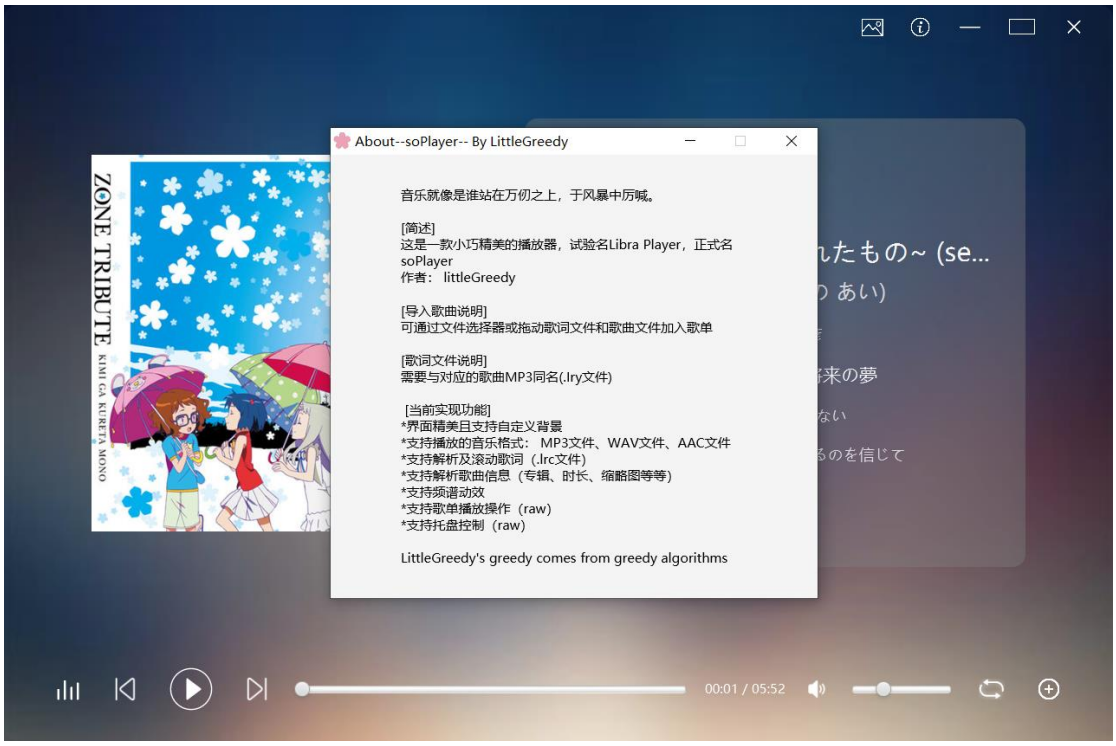


右键小菜单设默认背景+查看当前歌曲详情。歌词海报可以通过标号 1 位置切换是否显示，频谱动效与海报互斥显示。

标号 2 为歌曲列表菜单：



About 显示：



托盘菜单如下：

点击关闭程序时，程序并未结束运行，而是最小化托盘显示。

四、实现中值得一提的地方

1）歌词展示

通过遍历去匹配歌曲的合适歌词文件，以 GBK 编码格式读取歌词文件，用正则处理歌词并存储入 TreeMap（按键值升序排列的映射）

```
FileInputStream fis = new FileInputStream(f);
InputStreamReader isr = new InputStreamReader(fis, charsetName: "gbk"); //指定以gbk编码读入
BufferedReader br = new BufferedReader(isr);
```

显示歌词则比对 key 和播放时间即可。

这里为获取歌词放送时间的步骤：

因为 currentTime 刷新周期为 100ms，故设置偏移量提前获取，保持词曲近似同步。

```
//获得下一句歌词的时间
Long getTimeScroll() throws Exception{
    //xx:xx:xx
    @Nullable Long check2 = lyrics.floorKey(handleTime(current , shift: 100L)); //偏移量
    if(check2==null) return null;
    try {
        //确保歌词映射的每句歌词不被重复显示
        if (check1 == check2 || check2 == 0L) {
            return -10L;    //0L不可
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    check1 = check2;
    return check2;
}
```

这里为刷新歌词文本显示的步骤：

```
69      void  open(Text[] lyricText) throws  Exception{
70          Long key=this.getTimeScroll();
71          if(key==null || !lyrics.containsKey(key) )  //key重复或未查询到key, 则退出函数
72              return ;
73          String center = lyrics.get(key);   //value
74          lyricText[1].setText(center);  //初始化高亮核心层
75          if (lyrics.lowerKey(key) != null) {      //刷新高亮层的上一层的text
76                  lyricText[0].setText(lyrics.get(lyrics.lowerKey(key)));
77          }
78          if (lyrics.higherKey(key) != null) {    //刷新高亮层的下三层的text
79              try {
80                  SortedMap<Long, String> map = lyrics.tailMap(lyrics.higherKey(key));   //剩余未播放的歌词填入排序map
81                  //遍历刷新三个text
82                  int index = 2;
83                  int limit=map.size();   //剩余未播放的歌词数量
84                  for (Map.Entry entry : map.entrySet()) {
85                      if (index > 4) break;      //只能刷新2、3、4三层
86                      String value = (String) entry.getValue();
87                      if(limit>=index) {
88                          lyricText[index].setText(value);
89                      }
90                      else {    //拒绝剩余刷新歌词  因为foreach循环遍历 map.entrySet()
91                          lyricText[index].setText("");
92                      }
93                         //                  int key = (int) entry.getKey();
94                      index++;
95                  }
96              }catch (ArrayIndexOutOfBoundsException e){
97                  e.printStackTrace();
98              }catch (Exception e){
99                  e.printStackTrace();
100             }
```

2）图片转换

使用 JavaFX 实现拖拽过程中，剪贴板所含**文件列表**无法转换成 javafx.scene.image.Image 的对象，仅返回 null 值。首先推测剪贴板未获取数据，查询文档、多次 debug 后排除猜想，确定问题出在转换。尝试过调用转换类、符号强制转换，无一例外结果都为 null 。

```
db.getContent(DataFormat.IMAGE);
Image image=(Image)O;
```

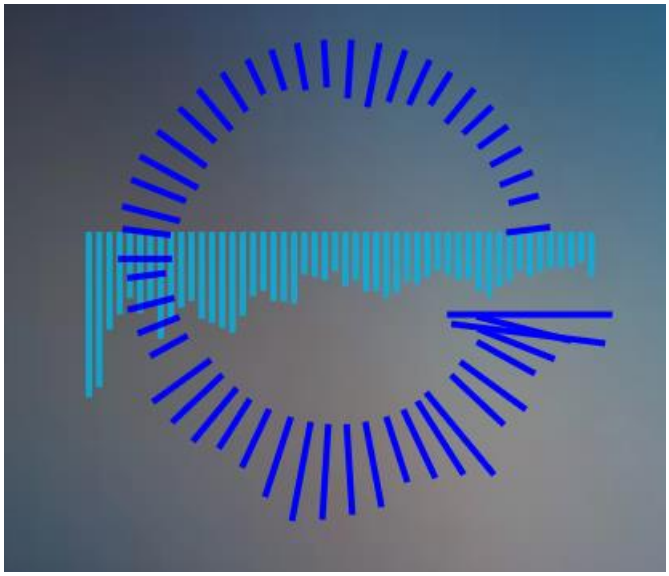
最后在 stackOverflow 找到了答案，采用文件操作类逐一读取列表图片，并转换成 FX 可读图片格式。

```
InputStream is = new FileInputStream(f);
BufferedImage bi = ImageIO.read(is);
Image image = SwingFXUtils.toFXImage(bi,  wimg: null);
```

3）频谱动效

Mediaplayer 播放时是可以解析到当前播放时间的，再加上平时使用的本地播放器都能将歌曲信息能完好的显示出来，此时我猜想歌曲的名字、图片、甚至是频谱都被存储在歌曲文件中，而 Java 中必定有相应的方法进行解析。　　在官方文档及其他相关资料中我发现了 AudioSpectrum 这个组合词，这更加验证了我的想法。

首先利用 Mediaplayer 的 AudioSpectrumListener 监听频谱数据，同步获取经处理后实时绘制小矩形。



4）自定异常类设计

继承运行时异常，目的为保证配置文件的绝对正确性。

```
public class InitializeException extends RuntimeException{
    String initGetKey;
    public InitializeException(String initGetKey) { this.initGetKey=initGetKey; }
    //重写object方法，输出异常文字信息
    public String toString(){
//          System.out.println(UnlimitedAppend.appendAll("配置文件读取异常\n请检查.ini配置文件：",in
//          return UnlimitedAppend.appendAll("配置文件读取异常\n请检查.ini配置文件：",initGetKey,"部
        return null;
    }
}
```

该类的对象在读取.ini 文件的操作类方法中被投出：

```java
public static String get (String key) throws InitializeException{
    Properties ini = null;
    File file=new File(fileString);
    try
    {
        ini = new Properties ();
        ini.load (new FileInputStream(file));
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
    }

    if(!ini.containsKey (key))
    {
        throw  new InitializeException(key);
    }
    return ini.get(key).toString();
}
```

五、程序设计反思与总结

1） 界面 ui 色调统一，色度主要由透明度来控制，如此显得美观简洁。
2）多采用 CSS 方式以简化代码美化界面。
3）StringBuffer 在需要进行大量字符串拼接的场合表现奇佳，经测试可以有效降低程序的内存占用。行文不断重复 append 显得不够美观，而本身的 final 限定符无法重写其方法，故在工具类写了一个 UnlimitedAppend 类配合不定参数达到如同 addAll()般较好的简洁效果。
4）由于歌单得维持有序、且时常增删查的操作，写程序后期打算采用 B+树数据结构，但考虑到将添加事务型数据库的支持，遂暂时数组简化处理。
5）设计来源于生活，留心生活。在网易云音乐中点击随机播放，显示单词：shuffle，后知后觉果真有 shuffle 算法 ，位于 Java 的 Collections 中。
6）程序不只是写，而是设计。

全文完，感谢您的阅读~