Java实验一设计文档：音乐播放器设计

王一洲 2013220201009

实现了播放、暂停、停止、前一首、后一首、添加/删除音乐，播放时可以显示歌曲信息，循环列表播放、动态更新音乐列表。

一、设计播放器界面：

这次音乐播放器实验我使用的是swing来设计的，swing的优点我感觉是可用的方法多，而且布局比较容易。但是在开始设计布局的时候，我首先使用了BorderLayout，但是这个布局只有四个方向，而播放器的控件有很多并不适合，之后使用了GridLayout，这个布局可以用但是从设计的角度来说就不是很适合了。所以我没有使用布局，通过每一个控件的setBounds来确定位置，写出自己想要的布局。

这里有注意的问题就是，JFrame的对象在最后要设置setVisible来显示，而且经过多次实践我发现这个方法必须写在所有控件声明的最后面，也就是布局代码的结尾才能显示出来。还有关于JLabel，在设计确定位置的时候必须要用setOpaque这个方法使其不透明才能调整。

关于按钮的美化问题，由于使用了swing，所以按钮可以使用setIcon方法用一张图片替代按钮，这样就对按钮做了美化，但是有一个新的问题就是使用了图片替代按钮后原来按钮依然存在，所以要用setContentAreaFilled和setBorderPainted这两个方法设置false参数来使按钮不显示，就完成了图片按钮的功能。

还有一个细节问题，在实例化按钮对象的时候习惯写成new JButton(“btn”)，这样写看似没有问题，可是如果使用图片代替按钮之后，这个btn还依然存在，而在后面的时间响应是通过btn来判断是不是改按钮，那么这个时候就要使用JButton的setActionCommand方法来设置该按钮的响应命令，这样图片按钮就不会出现文字了。

二、事件响应与按钮逻辑：

布局设计好只是静态的，不能做出任何响应，此时就要给按钮添加点击事件了，addActionlistener将每一个按钮添加监听，actionPerformed来写按钮的响应，getActionCommand方法获得之前设置的按钮响应命令。

播放：点击播放之后按钮就会变成暂停的样子，那么一开始暂停按钮和播放按钮重叠，而且暂停按钮是不可见的，点击了播放按钮之后，通过setVisible方法设置播放按钮不可见而暂停按钮可见，就实现了动态过程。然后创建播放的线程，如果不使用线程就会卡死，同时创建更新音乐列表的线程，此时有一个全局变量f4来判断这是不是第一次更新列表，如果不是的话就重新获得列表显示在JTextarea中，如果是第一次的话直接开始线程(具体的更新方法后面解释)

停止：点击之后将暂停按钮隐藏，将播放按钮显示。

暂停：点击之后将暂停按钮隐藏，将恢复播放按钮显示，一开始这两个按钮重叠放置。

恢复：点击之后将恢复按钮隐藏，将暂停按钮显示，一开始这两个按钮重叠放置。

添加、前一首、后一首、删除：创建对应功能的线程。

三、具体功能实现：

1. 播放：使用外部库JLayer的AdvancedPlayer类的play方法，首先，通过java io流创建一个要播放的音乐的流文件，使用FileInputStream实例化，将这个流的对象作为AdvancedPlayer类对象的参数，然后使用play方法播放音乐。同时，再引入Mp3Fenge库，这个库可以分割歌曲信息，首先FIle类的对象作为Mo3Fenge的参数，然后调用Mp3Info方法得到的音乐的信息，之后有getTitle等获得具体音乐信息的方法，获取信息后通过setText方法显示在JLabel上，就可以看到歌曲信息了。

2. **循环播放**：因为不知道要播放多少首歌曲，所以将所有歌曲保存在 ArrayList<String> 中，设置全局变量flag2，初始为0，当play方法结束后没也就是播放结束后判断，如果此时flag2等于arraylist的长度也就是播放完了，赋值flag2为0，创建播放线程从头开始播放，如果不是队列的最后一个，那么flag2加1，继续创建播放线程播放。

3. **停止**：调用JLayer的AdvancedPlayer类的close方法，停止当前正在播放的音乐文件。

4. **添加音乐**：FileDialog类可以提供一个文件选择窗口，

```
FileDialog fi1 = new FileDialog(frame, "ADD", FileDialog.LOAD);
```

通过这样设置就可以添加音乐，再通过getDirectory和getFile方法获得文件的路径和名称，再通过arraylist的add方法添加到队列里，设置去全局变量f3表示这不是第一次更新，创建更新线程完成动态更新音乐列表。

5. 播放前、后一首： 播放下一首时，如果flag2此时是队列末尾，那么赋值flag2为0，创建播放线程从头播放，如果不是flag2加1，创建线程继续播放下一首。播放前一首时，flag2如果为0，设置flag2位了、队列对打长度减1，因为队列从0开始，然后创建线程播放，如果不是，flag2减1，继续创建线程播放。

6. 更新音乐列表：如果全局变量f3=0，说明这是第一次更新音乐列表，循环创建Mp3Fenge的对象和Mp3Info方法来获得音乐信息append在JTextarea上，知道循环到和arraylist长度一样停。如果f3不为0，那么先清空当先JTextarea，在执行循环，更新音乐列表。

7. 暂停： 暂停功能的实现比较复杂，我经过了多次反复的实践得到了一种还算是行得通的方法。首先，jlayer中有Player类，这个类有play方法，可以播放音乐。还有个AdvancedPlayer类，这个类的play方法也可以播放。

在完成暂停功能的时候，我是这样想的，用Player类的getPosition()方法获得当前的位置，然后再用AdvancedPlayer的play(int start ,int end)来继续播放，这个end变量要用Mp3Fenge的getInfo里的TrackLenth方法获得长度赋值给end。但是有个问题，就是AdvancedPlayer中并没有getPosition方法，也就是说只能暂停一次，因为第二次播放用的是AdvancedPlayer的对象，而不是Player的对象。

所以我换了一种播放方法，就是一帧一帧的播放音乐，这里又有问题，就是比如说AdvancedPlayer的对象是p,然后

```
while(true){
p.play(count,count+1);
count++;
}
```

这样一帧一帧播放，结果是断断续续。然后我又把end固定，也就是

```
while(true){
p.play(count,len);//len就是整首歌的长度
count++;
}
```

这样可以播放了，但是我测试了count的值一直都是0.也就是说count没有自增，我又分析了原因:

```
p3 = new AdvancedPlayer(new FileInputStream("test.mp3"));
p3.play(1,10);
p3.play(11,20);
```

结果是只播放了第一个play，第二个没有播放，

如果:

```
p3 = new AdvancedPlayer(new FileInputStream("test.mp3"));
p3.play(1,10);
p3 = new AdvancedPlayer(new FileInputStream("test.mp3"));
p3.play(11,20);
```

那么此时就可以播放了。

所以我猜测这个play方法播放之后就会关闭FileInputStream这个流，假设我暂停了，那么要再次播放就只能重新创建流，但是之前播放的位置并没有记录下来，就不能继续播放。

如果不能使用一帧一帧的播放，那么就不能得到当前位置，那么暂停功能也就不能实现，所以我这里采用了另一种方法:

在第一次播放的时候，创建FileInputStream的对象，打开音频文件，之前我还声明了全局变量long total。当点击按钮进行第一次播放的时候对FileInputStream对象f1使用available()方法，代码如下:

```
public long total,left;

f1 = new FileInputStream("test3.mp3");
total=f1.available();
bis = new BufferedInputStream(f1);
p3=new AdvancedPlayer(bis);
```

```
p3.play();
```
当然bis p3都是之前声明好的全局变量，f1.available()的值赋给了total，等于total此时记录了f1对象当前全部可用的字节数。

但我点击暂停按钮，再次调用available方法获得此时可用的字节数，赋值给left，代码如下：

```
left=f1.available();
p3.close();
```

当我再次点击播放的时候，total是全部的水，left是剩下的水，那么total减去left就是已经喝掉的水，也就是已经播放的字节，那么就获得了当前的position 这样就避免了AdvancedPlayer类没有getPosition方法的问题，可以直接获得位置。在播放的时候使用skip方法跳过已经播放了的字节，等于是接着上一次的播放。同时，这个方法也不需要在播放的时候一帧一帧循环的去播放，也就是play（int ，int）的方法。
我觉得这算是一个比较好的解决方案了。。代码如下：

```
f1 = new FileInputStream("test3.mp3");
f1.skip(total-left);
bis = new BufferedInputStream(f1);
p3=new AdvancedPlayer(f1);
p3.play();
```

每一次暂停都会获得余下的字节数，所以只需用第一次的total减就可以了。
以上的方法就解决了暂停的问题。

8. 删除并播放下一首： 在删除的时候arraylist类有一个remove方法可以删掉队列中一个指定位置的变量，flag2记录了当前播放歌曲的位置，那么就可以删除掉当前正在播放的歌曲。然后使用AdvancedPlayer类中的close方法关闭当前的播放，设定f3=1表示不是第一次更新列表，然后创建更新音乐列表线程和播放线程完成该功能。