SPOTIFY INDIA

JAVA PROJECT

Vaibhav Saini 378/Co/14
Vaibhav Singal 381/Co/14
Dipanker Singh 397/Co/14

Table of Contents

INTRODUCTION

Music player is a very integral part of our lives. Whenever we want to listen to any audio we require a music player. Keeping this in mind we came up with an idea of creating a music player using the concepts taught to us in our winter training of JAVA. The SPOTIFY INDIA music player performs operations of play, pause, resume, stop.

FEATURES

The music player we have created performs the following operations:

1. **File Selection**
   To play any file we need a file chooser. For that we have used swing class *Jfilechooser* which only selects file of specific extension. In our case it was "*.MP3, mp3, mpeg3" which was implemented using *FileNameExtensionFilter*. So one can easily play any mp3 file using our music player. The name of the song is displayed as it converts the name of the file into a string which can be displayed easily. This function also ensures that the new thread does not overlap with the previous running thread whenever we want to play a new song while the previous song is still running i.e. the first thread is stopped before running the next thread.

2. **Play**
   After selection of a file the music player buffers the complete song once before playing so that there are no lags while streaming. This was implemented using *BufferInputStream* class. It uses an internal buffer to store data. It adds more efficiency than to write data directly into the Stream. So, it makes the performance fast. As soon as the song is selected the music starts playing. For the playing we need to create a new thread because we want the music to basically play in the background. This is done so that we can still click on buttons stop, pause, resume etc. We can still access everything in the main thread but has the music playing in the separate thread.

3. **Pause**
   Pause button performs the operations of pausing the song at a specific point so that it could be played from the same position again. This was implemented as the location (where it was paused) is stored in the variable *pauseLocation*. This *pauseLocation* is determined by using the *available* function of the *FileInputStream* which gives the unprocessed length of the input stream (here song). This is done as we do not want to play from beginning. Also this

pauseLocation variable is set as public so that other methods such as *Resume* can use that. After updating all the required variables we stop the thread.

4. **Stop**
   Stop is basically a special case of pause function with the basic difference that the pauseLocation here will be set to NULL here i.e. the available length unprocessed would now be the complete song.

5. **Resume**
   The resume function is executed only if pause or stop has been executed before. Now since the previous thread was stopped in the functions pause or stop we have to again start the song by using a new thread. Resume function makes use of the variable *pauseLocation* which stored the unprocessed length of the input stream. This function further uses the *Skip* function of the *FileInputStream* class to skip the location of the song to where we paused (or from where we have to begin).

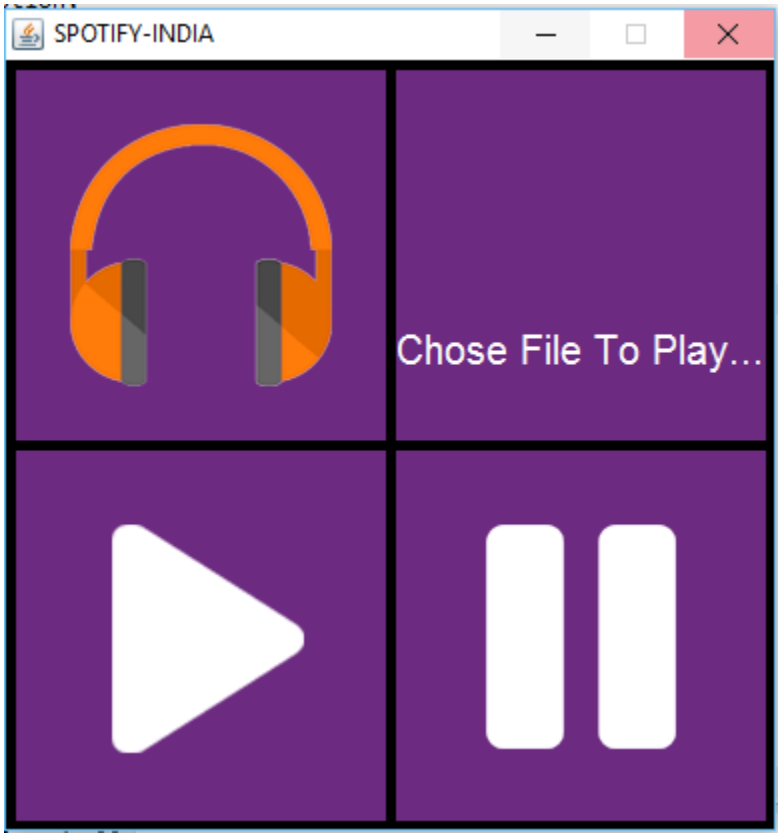
TECHNOLOGIES USED

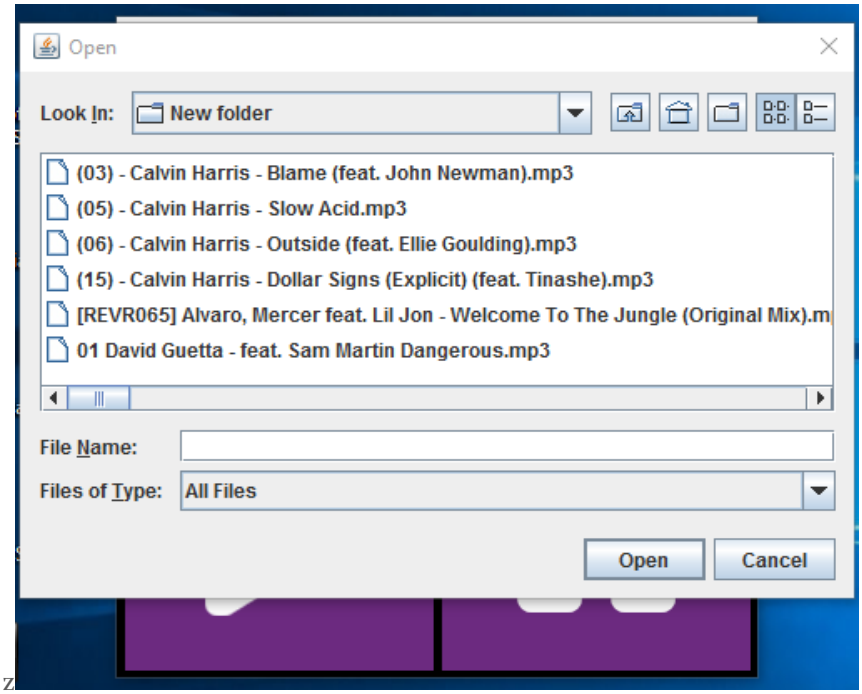
1. *JavaZoom* library for decoding and performing various operations on the audio file.
2. Threading – For playing the song in the background and to overcome overlapping.
3. Streams – for streaming of the audio file

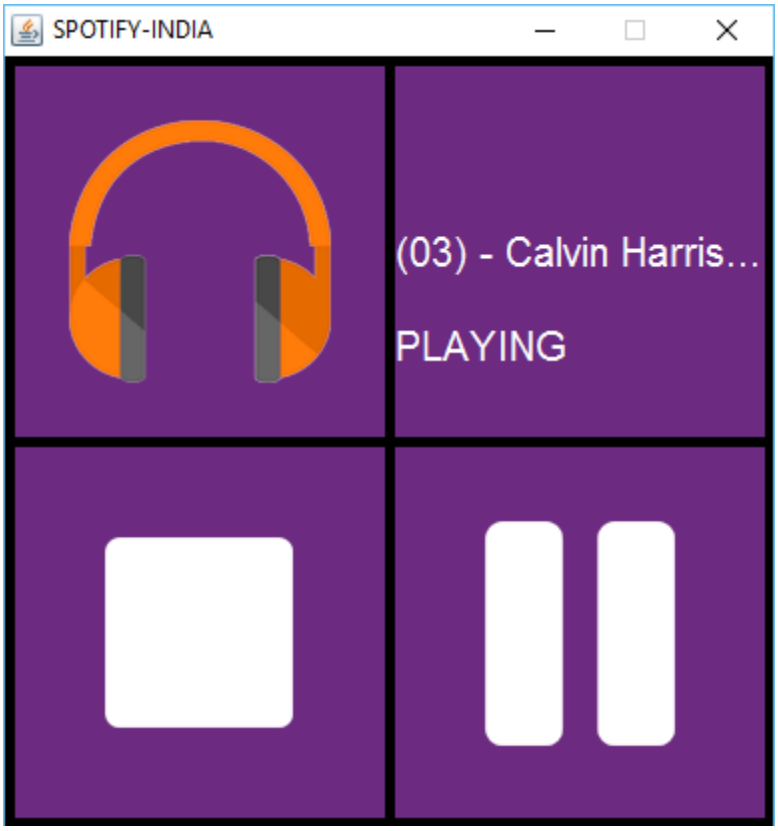
SCOPE OF DEVELOPMENT

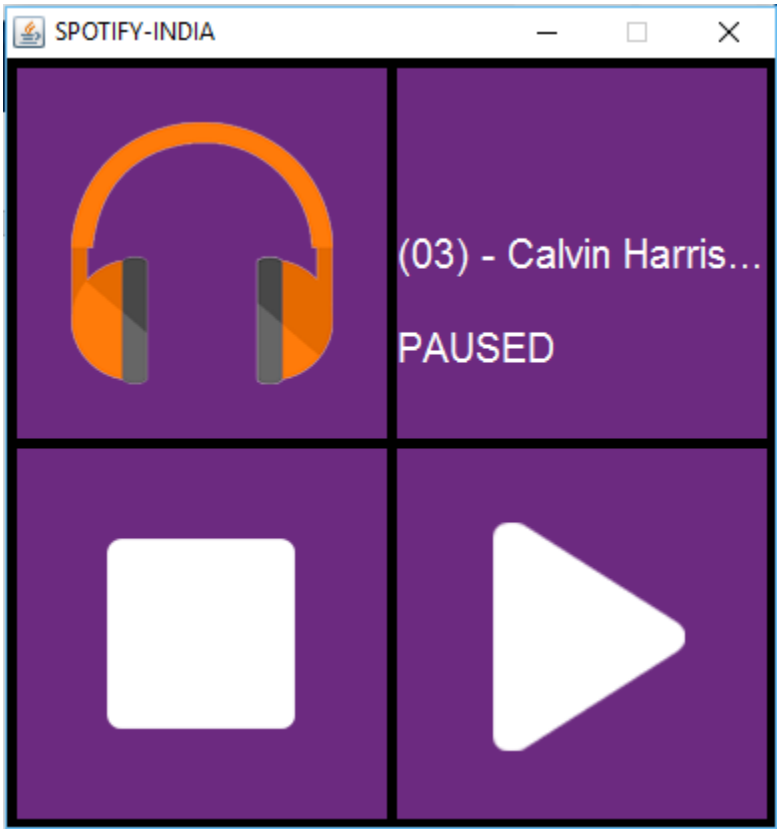
1. To add a loop function ability:
   Implement this using method isComplete which is in the javazoom library. It basically checks weather a song is completed or not.
2. Adding a playlist.
3. Adding features like shuffle, repeat, skip etc.

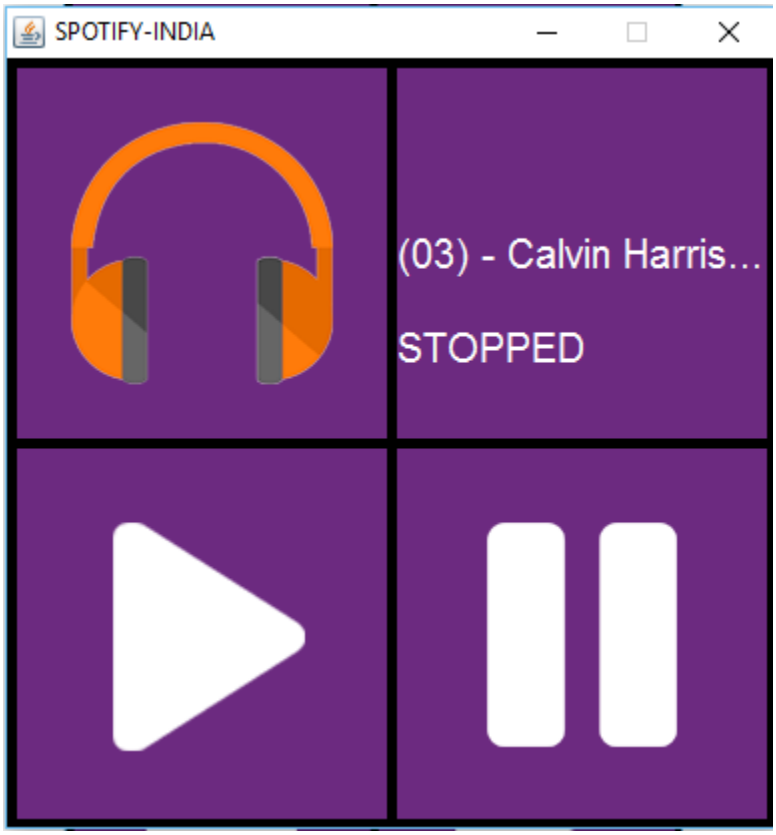
SCREENSHOTS

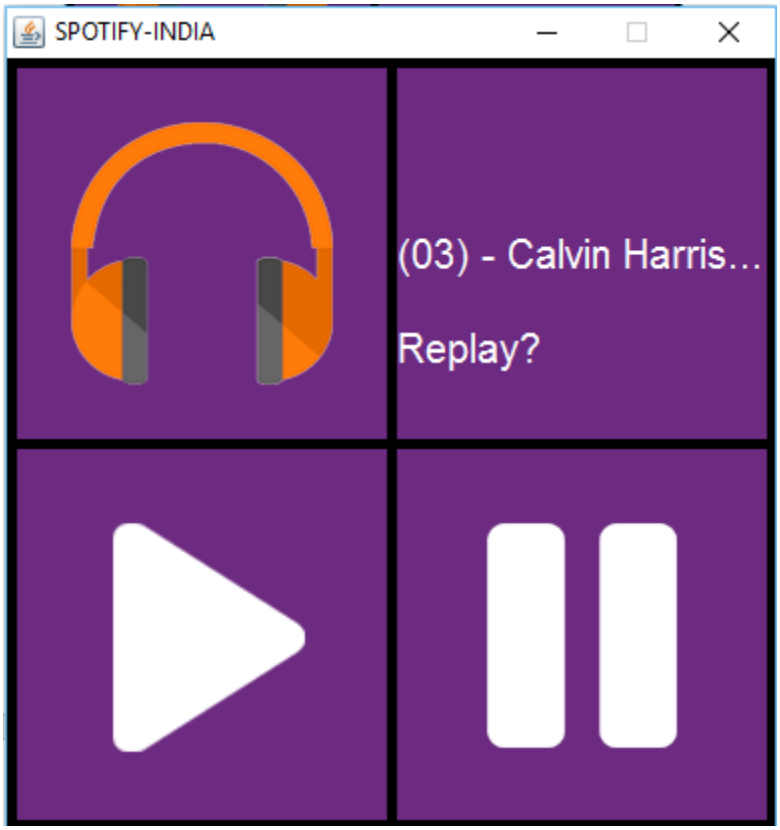





SPOTIFY-INDIA
(03) - Calvin Harris...
PLAYING


SPOTIFY-INDIA
(03) - Calvin Harris...
PAUSED


SPOTIFY-INDIA
(03) - Calvin Harris...
STOPPED


SPOTIFY-INDIA
(03) - Calvin Harris...
Replay?