BasicPlayer myMusicPlayer=new BasicPlayer();

BasicController playerController =(BasicController)myMusicPlayer;

String filePath="D:/SexyTracks/Just\_dance.mp3";

File file=new File(filePath);

try

{

playerController.open(file);

playerController.play();

}catch(Exception ex){}

try

{

//choice can be any boolean,I just included it to show the pause & stop

if(choice)

playerController.pause();

else

playerController.stop();

}catch(Exception ex){}

/\*The setGain() method takes double within 0.0 to 1.0.

1 is for full sound(100%)\*/

playerController.setGain(0.85);

class MyMusicPlayer implements BasicPlayerListener

/\*\*

\* Open callback, stream is ready to play.

\*

\* properties map includes audio format dependant features such as

\* bitrate, duration, frequency, channels, number of frames, vbr flag, ...

\*

\* @param stream could be File, URL or InputStream

\* @param properties audio stream properties.

\*/

public void opened(Object stream, Map properties)

{

}

/\*\* \* Progress callback while playing.

\*

\* This method is called severals time per seconds while playing.

\* properties map includes audio format features such as

\* instant bitrate, microseconds position, current frame number, ...

\*

\* @param bytesread from encoded stream.

\* @param microseconds elapsed (reseted after a seek !).

\* @param pcmdata PCM samples.

\* @param properties audio stream parameters.

\*/

public void progress(int bytesread, long microseconds, byte[] pcmdata, Map properties)

{

/\*shows you how much seconds of the song has been played\*/

long time=microseconds/(1000\*1000);//time in seconds

System.out.println(time);

// Pay attention to properties. It depends on underlying JavaSound SPI

// MP3SPI provides mp3.equalizer.

}