

# Technical report

## Digital Innovations driving media entertainment

### 1. Software Installation

#### 1.1 Java Installation

Step 1: Open your internet browser

Step 2: Search for the latest version of Java JDK, JRE and download it

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

Step 3: Install the package of java (JDK and JRE)

JDK is installed in this path "C:\Program Files\Java\jdk-10.0.{x}", where {x} is the java version

JRE is installed in this path "C:\Program Files\Java\jre-10.0.{x}".

Step 4: Set the Environment variables

- Go to control panel → System and Security → System → Advanced System settings → Environment variables → New variables → In "Variable Name", enter "JAVA\_HOME" ⇒ In "Variable Value", enter your JDK installed directory path → ok → ok
- Here path of the JDK means "c:\Program Files\Java\jdk-10.0.xx", where xx is the version number

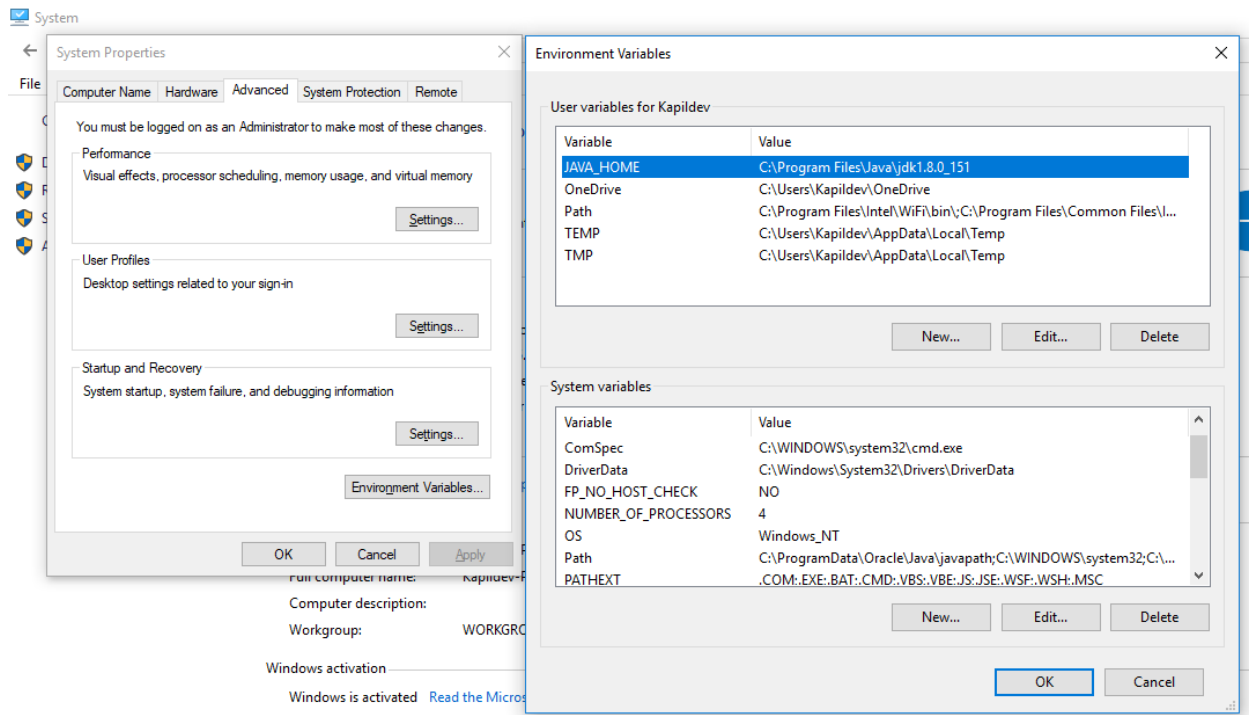


Fig 1 : To set environmental variables

Step 5: To verify environment variables and java is installed correctly

- Restart the system and open command prompt and check java is correctly installed or not by typing java and press enter.

## 1.2 Eclipse Installation

Step 1: Go to Internet browser and type

<http://www.eclipse.org/downloads/packages/release/luna/r/eclipse-ide-java-developers>

Step 2: Download eclipse according to the operating system type (Windows 32 bit or Windows 64 bit)

Step 3: Start the Eclipse installer executable file

Step 4: Select the package to install "Eclipse IDE for Java Developers"

Step 5: Select your installation folder and click on next-next

Step 6: Launch eclipse

## 2. Data Files and its Directory

Step 1: Create a folder in which music files are stored

Example-E:\MUSIC CITY\prgmMusic

This path will be used in the programming code to find the location of music files

Step 2: Music files should be in .wav extension because Java doesn't supports .mp3 file

Step 3: Convert the song in .wav file online by clicking following site:-

<https://audio.online-convert.com/convert-to-wav>

- Choose the .mp3 files that needs to be converted in .wav files and upload it online
- Click on start conversion and once it is done, download the files and store in the above folder you have already created.

- \

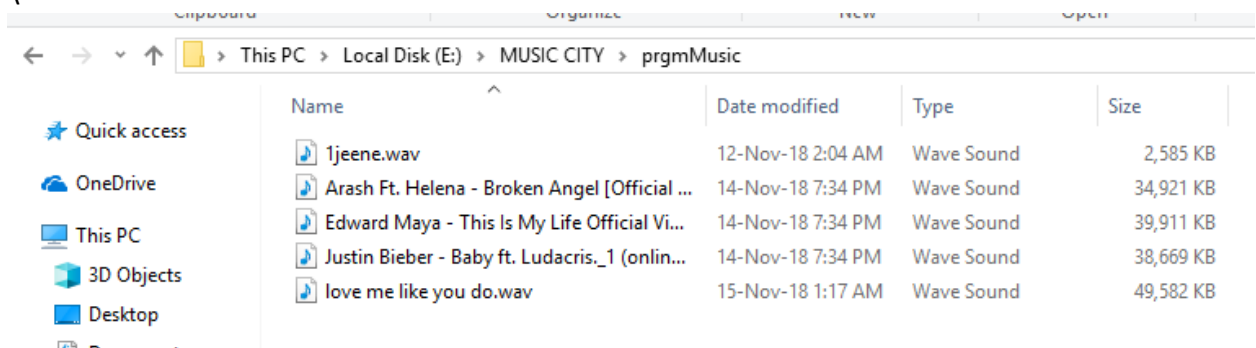


Fig 2 : Music files in .wav format

## 3. Steps for writing and executing java program in Eclipse

Step 1: Launch Eclipse

- Run "eclipse.exe" file which will be in eclipse installed directory
- Select workspace directory where you want to save the programming files
- If the "Welcome" screen appears close it by clicking on "cross" button which is next to the "Welcome" title

Step 2: Create a new Java Project

- Go to "File" menu → "New" → "Java project" (or "File" → "New" → "Project" → "Java project").

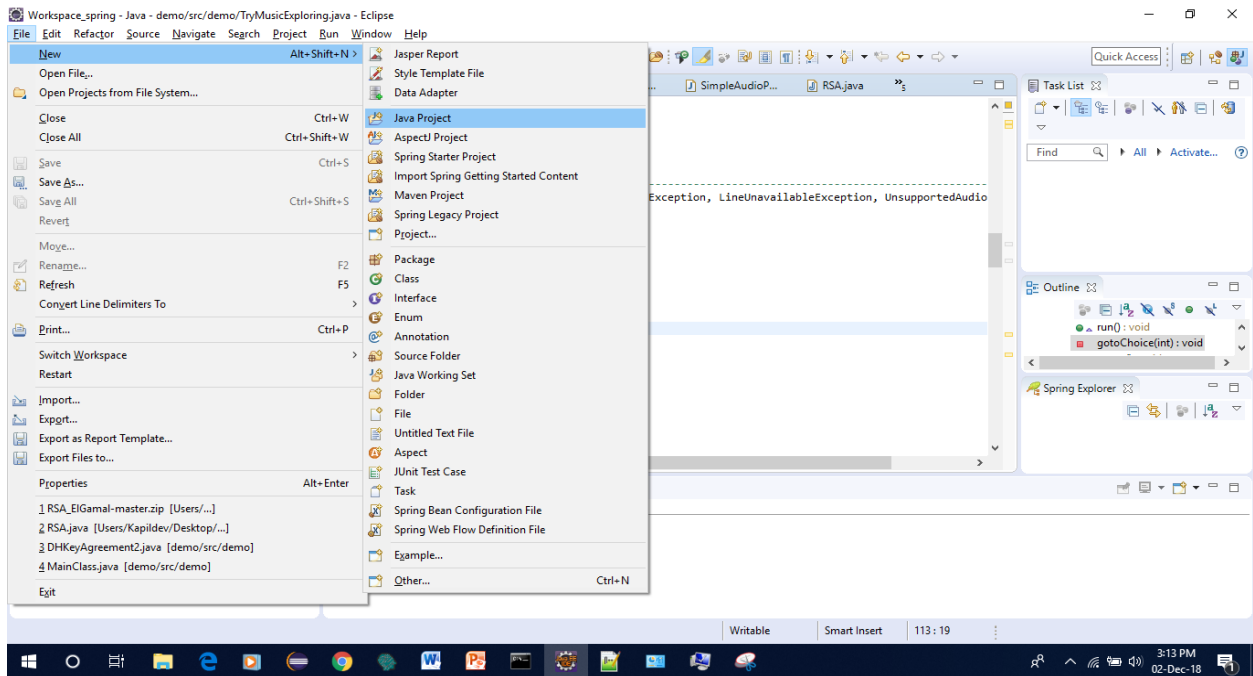


Fig 3: How to create Java project

- The "New Java Project" dialog pops up will appear.
  1. In "Project name", enter "demo".
  2. Check "Use default location".
  3. In "JRE", select "Use default JRE. Make sure that your JDK is 1.8 and above and then click on "Next" button.
  4. Finish.

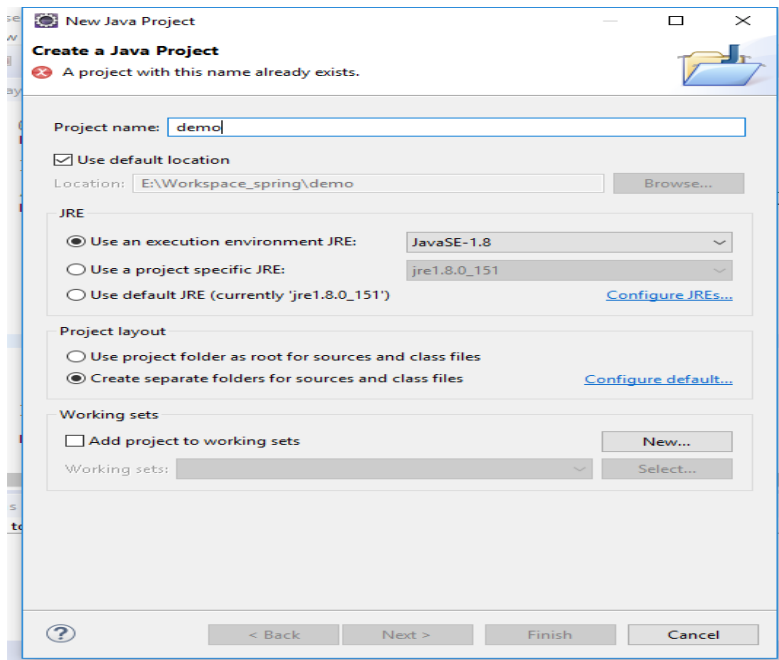


Fig 4: Writing first java program

### Step 3: Writing Java Program

- In the "Package explorer" right click on "demo" folder → new → class
- New java class dialog box will appear
  1. In "Source folder", write "demo".
  2. In "Package", write "demo".
  3. In "Name", enter "TryMusicExploring". This is our main class.
  4. Select "public static void main(String[] args)" to make this class main class.
  5. Click on Finish button
  6. Copy the source code and save it

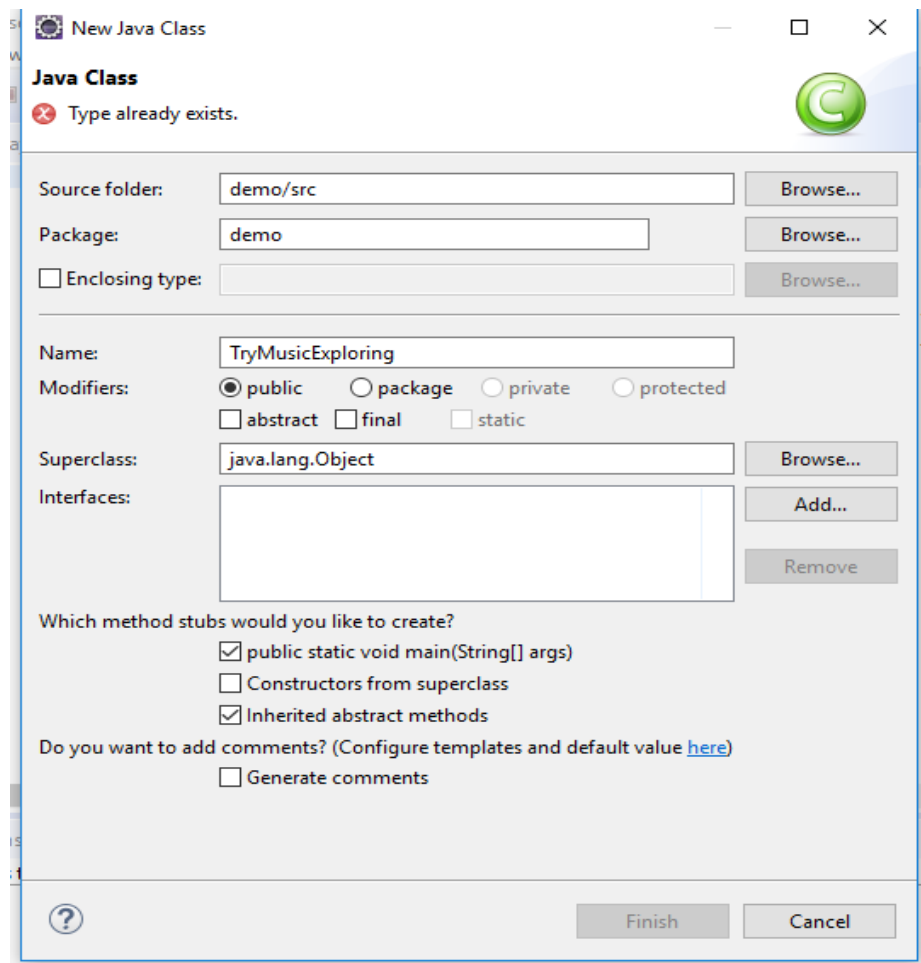


Fig 5: Giving Class name to java program

(Note: In screenshot, error means already file is created with the same name and two file cannot have same name in the same folder)

#### Step 4: Executing java program

1. Right click on your java main class file
2. Select "Run as"
3. Then click on Java Application
4. Program will start execution if there is no error or else it will throw exception or error

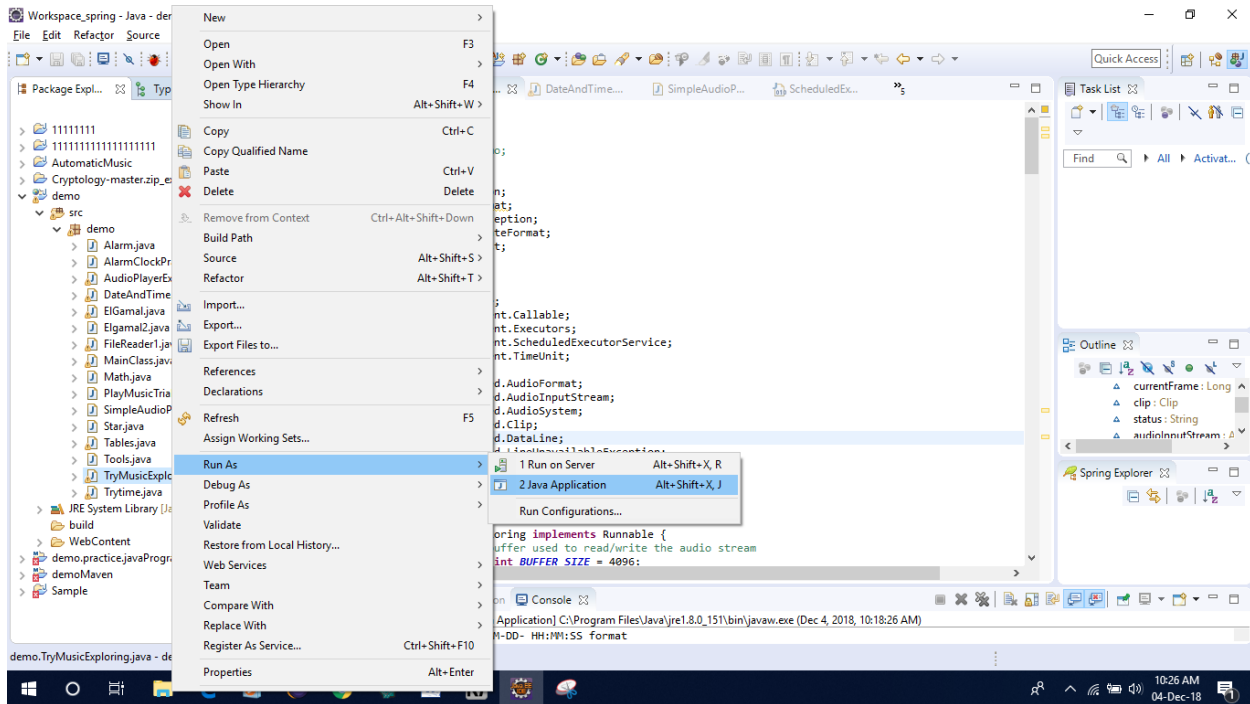
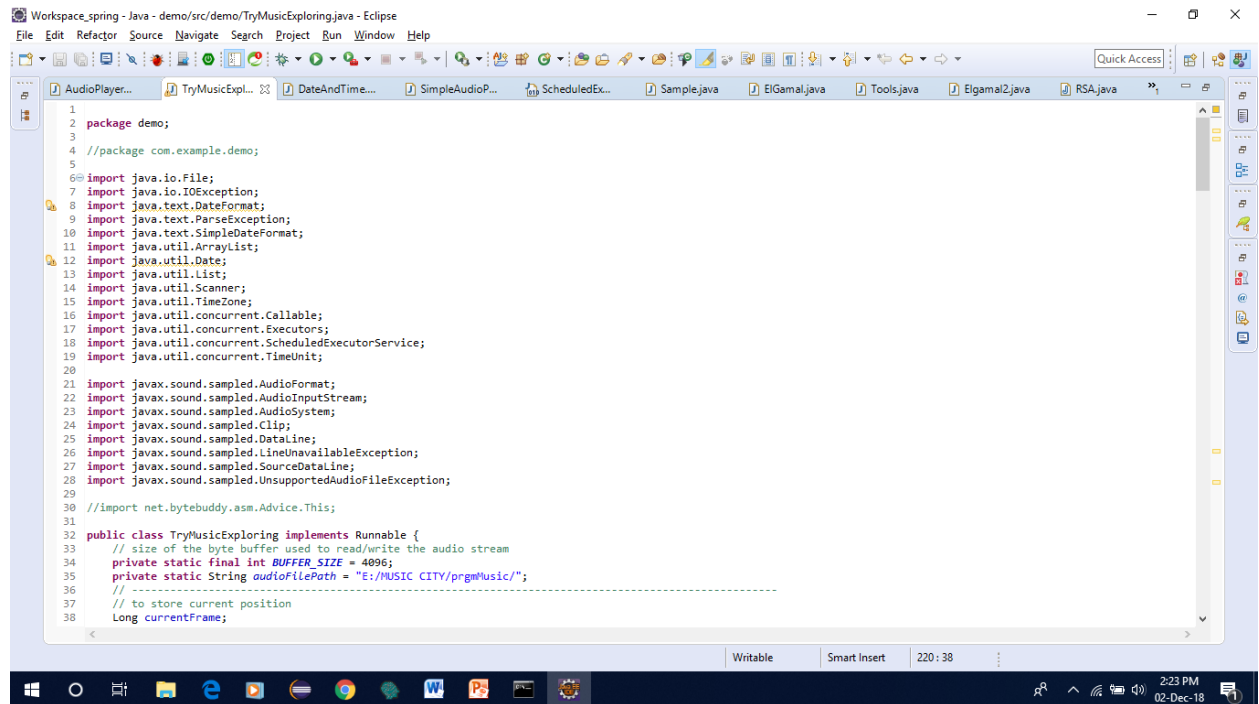


Fig 6: How to execute Java program

## Code Explanation

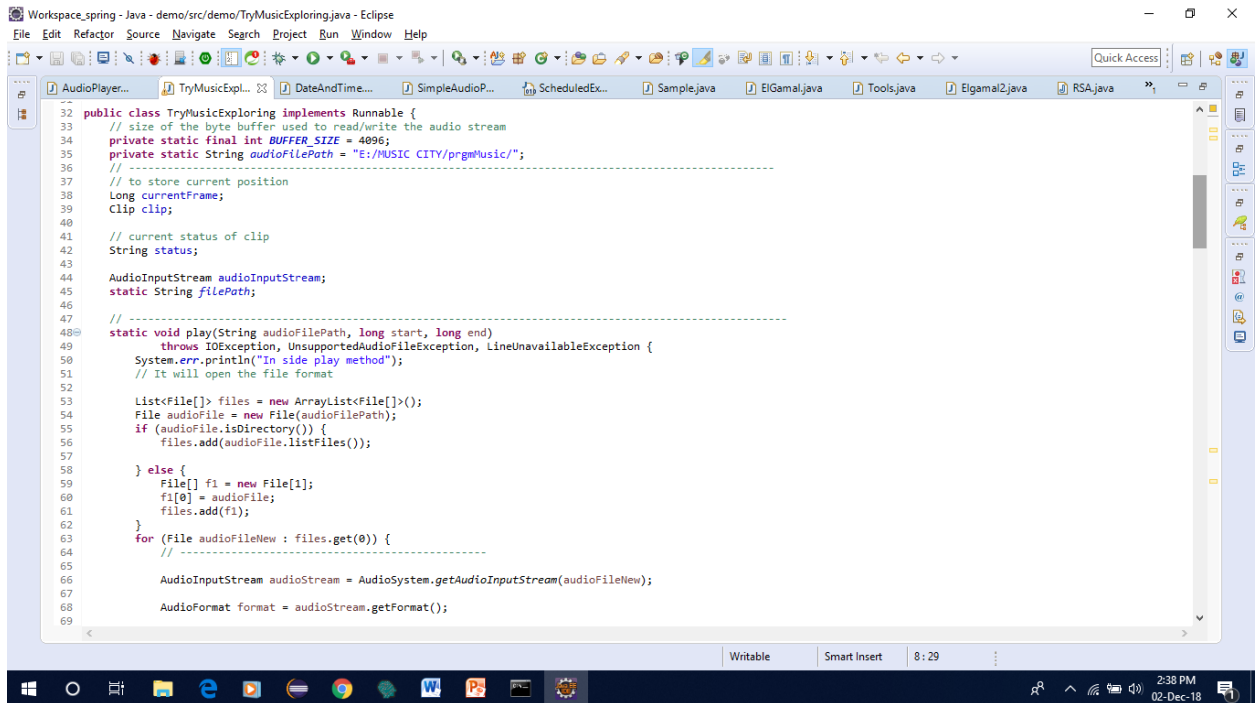
- First we need to import the java libraries or packages which are required for the project. These packages provide a number of functionalities like input output file, exception, date format, arraylist, multithreading, music clip method, scanner for user input and output, unsupported audio file exception etc.



```
1 package demo;
2
3 //package com.example.demo;
4
5
6 import java.io.File;
7 import java.io.IOException;
8 import java.text.DateFormat;
9 import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.ArrayList;
12 import java.util.Date;
13 import java.util.List;
14 import java.util.Scanner;
15 import java.util.TimeZone;
16 import java.util.concurrent.Callable;
17 import java.util.concurrent.Executors;
18 import java.util.concurrent.ScheduledExecutorService;
19 import java.util.concurrent.TimeUnit;
20
21 import javax.sound.sampled.AudioFormat;
22 import javax.sound.sampled.AudioInputStream;
23 import javax.sound.sampled.AudioSystem;
24 import javax.sound.sampled.Clip;
25 import javax.sound.sampled.DataLine;
26 import javax.sound.sampled.LineUnavailableException;
27 import javax.sound.sampled.SourceDataLine;
28 import javax.sound.sampled.UnsupportedAudioFileException;
29
30 //import net.bytebuddy.asm.Advice.This;
31
32 public class TryMusicExploring implements Runnable {
33     // size of the byte buffer used to read/write the audio stream
34     private static final int BUFFER_SIZE = 4096;
35     private static String audioFilePath = "E:/MUSIC CITY/prgmMusic/";
36     // -----
37     // to store current position
38     Long currentFrame;
```

Fig 7: Java libraries and package

- Here we have implemented multithreading concept in our main class i.e TryMusicExploring class
- We have provided the path of the music folder in which songs are stored in .wav format
- Play method has been created in which three parameter is passed mainly file path, user starting time and user ending time respectively.
- Array-list has been created to play all songs one by one in a sequence

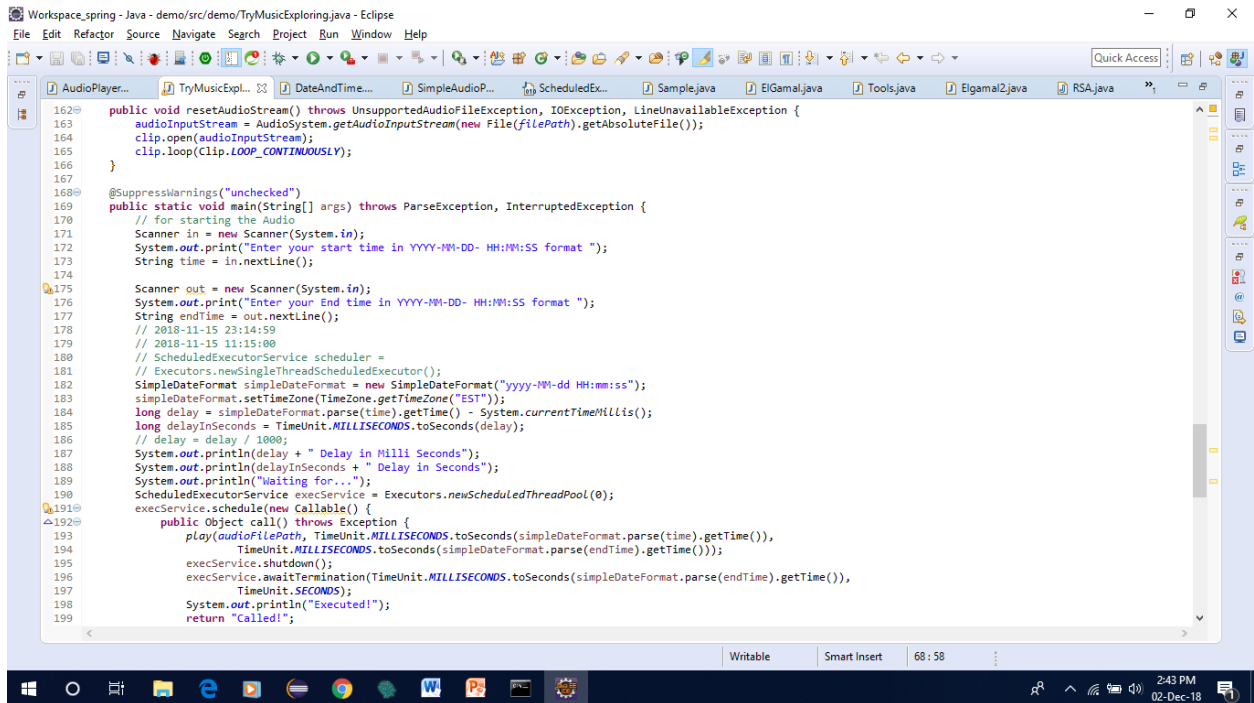


```
32 public class TryMusicExploring implements Runnable {
33     // size of the byte buffer used to read/write the audio stream
34     private static final int BUFFER_SIZE = 4096;
35     private static String audioFilePath = "E:/MUSIC CITY/prgmMusic/";
36     // -----
37     // to store current position
38     Long currentFrame;
39     Clip clip;
40
41     // current status of clip
42     String status;
43
44     AudioInputStream audioInputStream;
45     static String filePath;
46
47     // -----
48     static void play(String audioFilePath, long start, long end)
49         throws IOException, UnsupportedOperationException, LineUnavailableException {
50         System.err.println("In side play method");
51         // It will open the file format
52
53         List<File> files = new ArrayList<File>();
54         File audioFile = new File(audioFilePath);
55         if (audioFile.isDirectory()) {
56             files.add(audioFile.listFiles());
57
58         } else {
59             File[] fl = new File[1];
60             fl[0] = audioFile;
61             files.add(fl);
62         }
63         for (File audioFileNew : files.get(0)) {
64             // -----
65
66             AudioInputStream audioStream = AudioSystem.getAudioInputStream(audioFileNew);
67
68             AudioFormat format = audioStream.getFormat();
69 }
```

Fig 8: Play method and array list for files

- Here resetAudioStream method checks whether the music file is in .wav extension or not. If it is not in the proper format then it will throw exception
- In main method, we take starting time, ending time and date from user in the following format "YYYY-MM-DD- HH:MM:SS"
- System will calculate delay time which is  
Delay=User start time – current system time  
Which will be calculated in millisecond then we will convert it into second and delay time is shown to the user.
- This delay time is passed to the scheduler. System will wait for the delay time and it will automatically start playing song at scheduled time





```
162 public void resetAudioStream() throws UnsupportedAudioFileException, IOException, LineUnavailableException {
163     audioInputStream = AudioSystem.getAudioInputStream(new File(filePath).getAbsolutePath());
164     clip.open(audioInputStream);
165     clip.loop(Clip.LOOP_CONTINUOUSLY);
166 }
167
168 @SuppressWarnings("unchecked")
169 public static void main(String[] args) throws ParseException, InterruptedException {
170     // for starting the Audio
171     Scanner in = new Scanner(System.in);
172     System.out.print("Enter your start time in YYYY-MM-DD- HH:MM:SS format ");
173     String time = in.nextLine();
174
175     Scanner out = new Scanner(System.in);
176     System.out.print("Enter your End time in YYYY-MM-DD- HH:MM:SS format ");
177     String endTime = out.nextLine();
178     // 2018-11-15 23:14:59
179     // ScheduledExecutorService scheduler =
180     // Executors.newSingleThreadScheduledExecutor();
181     SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
182     simpleDateFormat.setTimeZone(TimeZone.getTimeZone("EST"));
183     long delay = simpleDateFormat.parse(time).getTime() - System.currentTimeMillis();
184     long delayInSeconds = TimeUnit.MILLISECONDS.toSeconds(delay);
185     // delay = delay / 1000;
186     System.out.println(delay + " Delay in Milli Seconds");
187     System.out.println(delayInSeconds + " Delay in Seconds");
188     System.out.println("Waiting for...");
189     ScheduledExecutorService execService = Executors.newScheduledThreadPool(0);
190     execService.schedule(new Callable() {
191         public Object call() throws Exception {
192             play(audioFilePath, TimeUnit.MILLISECONDS.toSeconds(simpleDateFormat.parse(time).getTime()),
193                 TimeUnit.MILLISECONDS.toSeconds(simpleDateFormat.parse(endTime).getTime()));
194             execService.shutdown();
195             execService.awaitTermination(TimeUnit.MILLISECONDS.toSeconds(simpleDateFormat.parse(endTime).getTime()),
196                 TimeUnit.SECONDS);
197             System.out.println("Executed!");
198             return "Called!";
199         }
200     }, delayInSeconds, TimeUnit.SECONDS);
201 }
```

Fig 9: Showing main method and how input is taken from the user

Here three methods has been created pause, resumeAudio and restart. Pause method is used for pausing the video by the user when he presses specified key. Song will be resumed if the user play specified key for the resume. Restart method is used to restart the song id user presses specified key. In all three method Clip class method has been imported.

```

120 public void pause() {
121     System.out.println("music paused");
122
123     if (status.equals("paused"))
124     {
125         System.out.println("audio is already paused"); return;
126     }
127     this.currentFrame = this.clip.getMicrosecondPosition(); clip.stop();
128     status = "paused";
129     System.out.println("music paused");
130 }
131
132 // Method to resume the audio
133 public void resumeAudio() throws UnsupportedOperationException, IOException, LineUnavailableException {
134
135     if (status.equals("play"))
136     {
137         System.out.println("Audio is already " + "being played");
138         return;
139     }
140     clip.close(); resetAudioStream();
141     clip.setMicrosecondPosition(currentFrame); this.run();
142     System.out.println("music is resuming.");
143 }
144
145 // Method to restart the audio
146 public void restart() throws IOException, LineUnavailableException, UnsupportedOperationException {
147
148     clip.stop();
149     clip.close();
150     resetAudioStream();
151     currentFrame = 0L;
152     clip.setMicrosecondPosition(0);
153 }

```

Fig 10: Method for pause, reset and restart music song

Below screenshot shows the method for taking input from the user when he presses specified key. While loop has been created so that it repeatedly asks questions to the user. For selecting specified key switch and case statement is used.

```

103 private void gotoChoice(int c) throws IOException, LineUnavailableException, UnsupportedOperationException {
104     switch (c) {
105     case 1:
106         pause();
107         break;
108     case 2:
109         resumeAudio();
110         break;
111     case 3:
112         restart();
113         break;
114     case 4:
115         //stop();
116         break;
117     }
118 }
119

```

Fig 11: User pressing specified key

```

243
244     while (true) {
245         System.out.println("1. pause");
246         System.out.println("2. resume");
247         System.out.println("3. restart");
248         //System.out.println("4. stop");
249
250         int c = in.nextInt();
251         player.gotoChoice(c);
252         if (c == 4)
253             break;
254     }
255     in.close();
256 }
257
258 catch (Exception ex) {
259     System.out.println("Error with playing sound.");
260     // ex.printStackTrace();
261 }
262

```

Fig 12: For displaying options in the screen