

REAL TIME MUSIC PLAYER USING CLIP INTERFACE IN JAVA

Report submitted to the
SASTRA Deemed to be
University as the
requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

BHARATHI S

(Reg. No.: 123015018, B. Tech. - Information
Technology)



SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU,
INDIA - 613401



SCHOOL OF COMPUTING
THANJAVUR, TAMIL NADU,
INDIA - 613401

Bonafide Certificate

This is to certify that the report titled “**Real Time Music Player using Clip Interface in java**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Smt. Bharathi.S(Reg.No.123015018, Information Technology)** during the academic year 2021-22, in the School of Computing.

Project Based Work *Viva voce* held on

Examiner 1

Examiner 2

ACKNOWLEDGEMENTS

First of all, I would like to thank God Almighty for his endless blessings. I would like to express my sincere gratitude to Dr S. Vaidyasubramaniam, Vice-Chancellor for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank Dr A.Uma Maheshwari, Dean, School of Computing and R. Chandramouli, Registrar for their overwhelming support provided during my course span in SASTRA Deemed University.

I am extremely grateful to Dr. Shankar Sriram, Associate Dean, School of Computing for his constant support, motivation and academic help extended for the past three years of my life in School of Computing.

I would specially thank and express my gratitude to N.Sasikala Devi, Associate Professor, School of Computing for providing me an opportunity to do this project and for her guidance and support to successfully complete the project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project and for successful completion of all courses during my academic life at SASTRA Deemed University.

Finally, I thank my parents and all others who help me acquire this interest in project and aided me in completing it within the deadline without much struggle

ABSTRACT

Socket Programming in Java enables communication between Client and Server. Music player is one of the applications which make use of a few concepts of Computer Networks. Socket Programming here plays an import role in sharing the needed music file from the Server to the Client based on Client's choice of genre.

The code makes use of a few modules which helps in transferring images, music files and the lyrics files from the Server to the Client.

Implementation of playing music files is done using the Clip interface in Java. On a click of the choice of genre by the Client, a new frame displays the songs available in almost 3 languages in the selected genre.

To be more precise the concept of Socket programming here, is to establish a connection between the Client and the Server and get the Client's choice and make the Server respond accordingly.

A few options for playing the song files are pausing the song, resuming from where it was left before, restarting the song and so on.

As the application of Music player utilizes file transfer, transport of messages to and from the Server and Client, This project has implemented the FTP in the Application layer over TCP in the Transport Layer.

This project consists of the following 4 modules:

- **Audio Transfer Module:** This module helps in transferring the audio file from the Server to the Client using the `AudioInputStream` class. Here, the audio format of the music file is extracted from the Server and fed to the Client which further creates a new file of the music file in the Client's File Directory. This transfer of music is done by incorporating a connection through the Socket and a Server Socket using Socket Programming in Java.
- **Image Transfer Module:** This module helps in sending the posters of each and every song asked by the Client to the Server using the `BufferedImage` class which reads the image from the specified directory converts it into bytes using the `ByteArrayOutputStream` class. The width and height of the image are also sent in order to create the exact same image in the Client's Directory.
- **Audio Play Module:** This module uses a `Clip` object to play an audio file. Here, the `AudioInputStream` class helps to enable this functionality. The play function in the `Clip` interface helps to play a WAV file.
- **GUI Main Module:** This module initializes all GUI Components required to construct the necessary GUI frame for a specific Genre. Once the genre button is clicked, the respective GUI is opened.

This project has a lot of future improvements that can be done like adding the Client's favorite playlist, playing the songs in a queue and adding variety of songs.

TABLE OF CONTENTS

S.no	TOPIC	Page Number
1.	Introduction	1
2.	Source code – Main Server	6
3.	Source code – Main Client	7
4.	Source code – Genre GUI	10
5.	Source code – Play Song	18
6.	Source code – Transfer audio file	19
7.	Source code – Transfer image file	22
8.	References	26
9.	Conclusion and Future works	27

INTRODUCTION - BASIC CONCEPTS

Socket programming helps develop a connection between the Client and the Server using TCP or UDP. The connection established can be either connection oriented or connection less based on the protocol used. For developing a connection between two IP Addresses, we need a port number and an IP Address to connect with an application running on a different JRE.

We use a Socket class to create a Socket at the Client's end and a ServerSocket class to create a serversocket at the Server's end. The socket class has an Application Programming Interface (API).

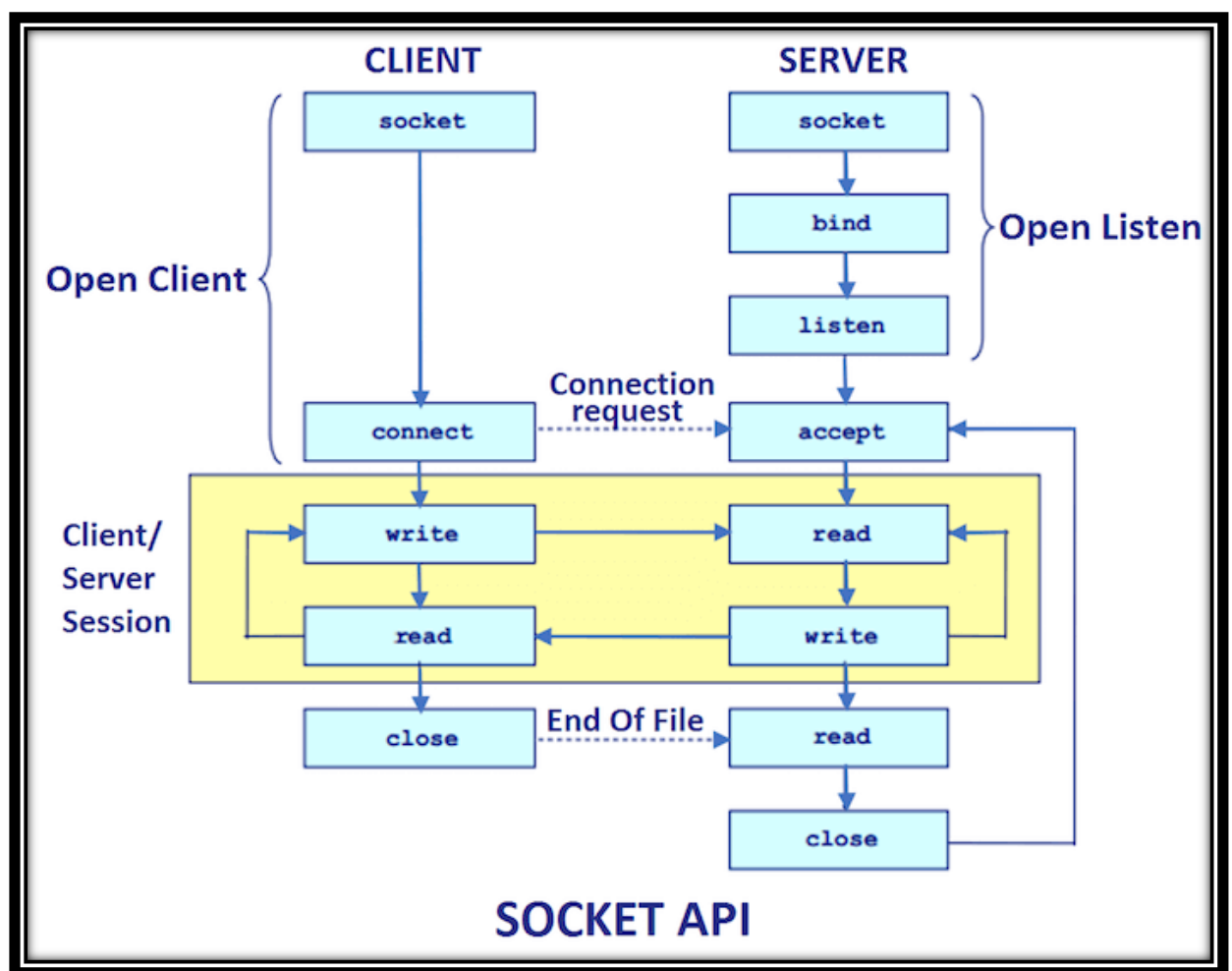
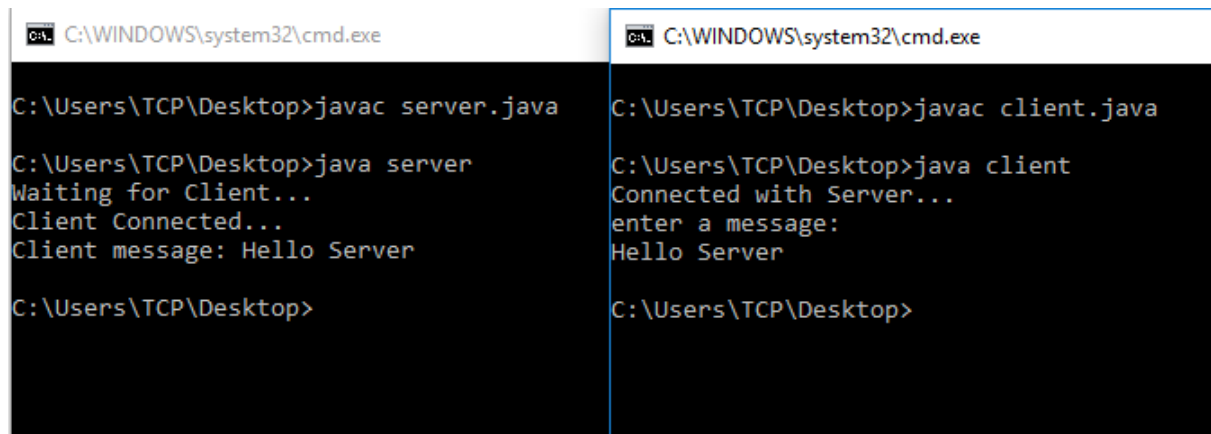


Figure 1: Socket Class's API

In the ServerSocket Class, the accept () function accepts the nearby sockets available. Like the establishment of Bluetooth connection between two devices, the serversocket waits for a socket to be connected and on establishment of

connection the accept function is invoked.

The java.net package helps in accessing the Socket and ServerSocket class objects throughout the code. Here's an example of how we should compile the code in two different command prompts. We first have to compile the Server side as the server socket only has to be waiting for the socket.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\TCP\Desktop>javac server.java
C:\Users\TCP\Desktop>java server
Waiting for Client...
Client Connected...
Client message: Hello Server
C:\Users\TCP\Desktop>
```

```
C:\WINDOWS\system32\cmd.exe
C:\Users\TCP\Desktop>javac client.java
C:\Users\TCP\Desktop>java client
Connected with Server...
enter a message:
Hello Server
C:\Users\TCP\Desktop>
```

Figure 2: Execution of the simple code at both ends

A server can connect to more than one client using socket programming which is also called as Multithreaded Socket Programming. Here, every Client needs to run on a unique thread. There is a fact that maximum of 50 clients can only connect to a common server at that instant.

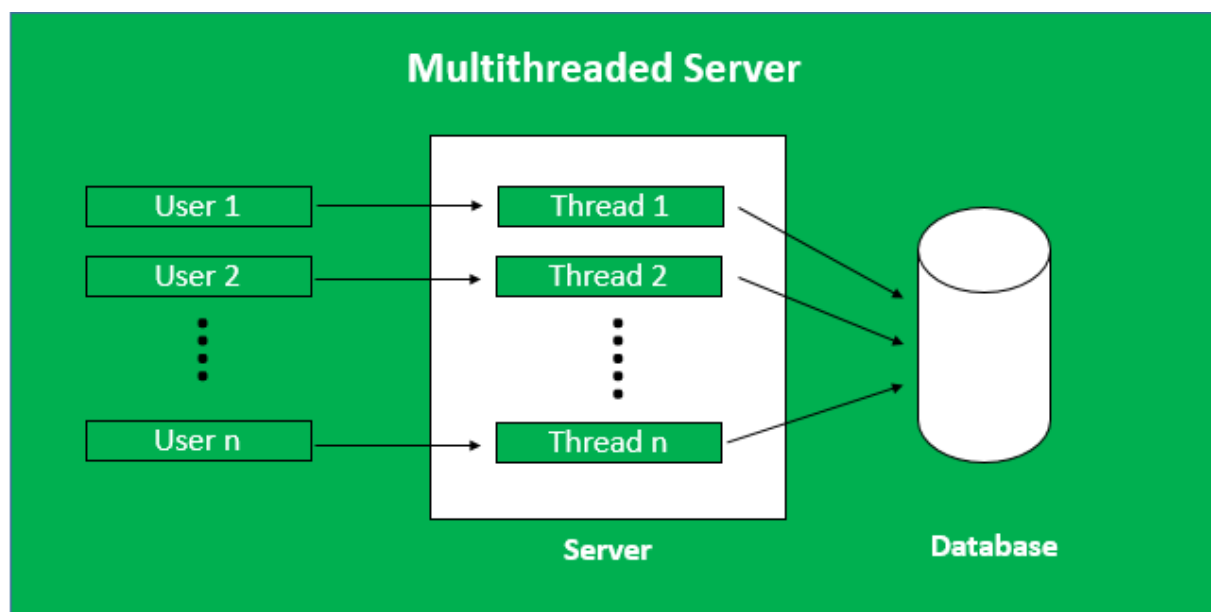


Figure 3: Multi-Threaded Socket Programming using Multi-Threaded Server

GUI Design Module

The project presents to you a GUI allowing the Client choose his/her favorite genre. After selecting the genre, a new frame will be displayed containing the list of songs in the specified genre.

The frame will consist of Poster of the song as ImageIcon, the name of the song as label, the play button which plays the specified song and finally the download button which downloads the song and saves it in the Listener's directory.

If the song is played, another frame will open which will play the song clicked by the Client. This frame contains different features a song file will generally consist such as play, pause, resume, stop and restart.

The project has also used the readUTF and writeUTF functions to send and receive string messages under socket connection. The music files (.wav), the poster image files (.jpeg/.jpg) and the lyrics text file (.txt) will also be sent from the Server to the Client based on their need.

Here is the sample of how the last frame playing the music will look like.

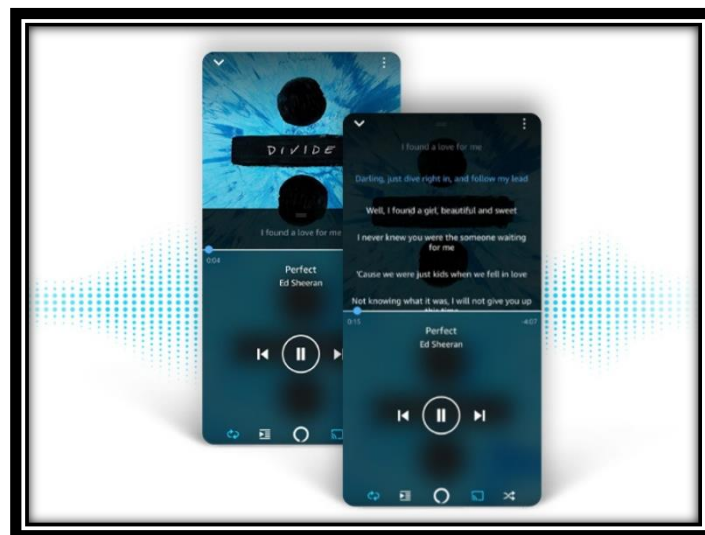


Figure 4: Sample of the Audio player

This project takes its creativity from the trending Music player platforms like Spotify and Amazon Music.

Here are a few clips of how the frame after selecting a genre will look like. We will be displaying the popular Tamil, English and Telugu songs under the specified

genre along with a play and download button for each song.

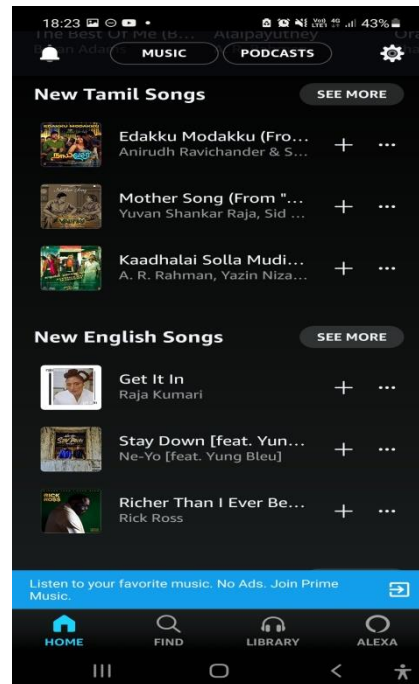


Figure 5: Sample Frame after selecting the genre

Now talking in detail about the various types of file transfer incorporated in our project, the BufferedImage class helps read and image file, get the height and width of the Image, and sent it to the Client with the exact same clarity. Similarly, The AudioInputStream class helped to read the format of an audio file, its contents and then transfer the entire audio file in its compressed form. The use of Clip Interface also played an important role in the transfer of music files from the Server to the Client.

For transferring the music files from the Server to the Client, the following packages need to be imported.

- import javax.sound.sampled.AudioFileFormat;
- import javax.sound.sampled.AudioFormat;
- import javax.sound.sampled.AudioInputStream;
- import javax.sound.sampled.AudioSystem;
- import java.io.File;
- import java.io.OutputStream;
- import java.net.InetSocketAddress;
- import java.net.ServerSocket;
- import java.net.Socket

For transferring the image (poster) files from the Server to the Client, the following packages need to be imported.

- `import javax.imageio.ImageIO;`
- `import javax.swing.*;`
- `import java.awt.*;`
- `import java.io.*;`
- `import java.awt.image.BufferedImage;`
- `import java.net.ServerSocket;`
- `import java.net.Socket;`
- `import java.nio.ByteBuffer;`

Later, for playing a Music file, the project implements a Clip interface which imports the following packages.

- `import javax.sound.sampled.AudioInputStream;`
- `import javax.sound.sampled.AudioSystem;`
- `import javax.sound.sampled.Clip;`
- `import javax.sound.sampled.LineUnavailableException;`
- `import javax.sound.sampled.UnsupportedAudioFileException;`

An audio input stream is an input stream with a specified audio format and length. The length is expressed in sample frames, not bytes. Several methods are provided for reading a certain number of bytes from the stream, or an unspecified number of bytes. The audio input stream keeps track of the last byte that was read. You can skip over an arbitrary number of bytes to get to a later position for reading. An audio input stream may support marks. When you set a mark, the current position is remembered so that you can return to it later.

The `AudioSystem` class includes many methods that manipulate `AudioInputStream` objects. For example, the methods let you:

- obtain an audio input stream from an external audio file, stream, or URL
- write an external file from an audio input stream
- convert an audio input stream to a different audio format

So, this concludes the introduction and we hope this Real Time Music player developed will be user friendly. A few improvements that can be accomplished in the future would be creating a playlist of the Client's choice, playing songs in a queue, looping an entire playlist, adding songs onto a temporary queue, looping a particular song etc.

SOURCE CODE

Server.java

```
package Clipplayer;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server
{
    Server() throws Exception
    {
        ServerSocket serverSocket = new ServerSocket(13085);
        Socket socket = serverSocket.accept();
        System.out.println("Connection Established !");
        //OutputStream outputStream = socket.getOutputStream();
        DataInputStream dis=new DataInputStream(socket.getInputStream());
        String selected = dis.readUTF();
        System.out.println(selected);
        socket.close();
        serverSocket.close();
    }
    public static void main(String args[])
    {
        try
        {
            @SuppressWarnings("unused")
            Server s = new Server();
        }
        catch(Exception e)
        {
            System.out.println("exp");
        }
    }
};
```

Client.java

```
package Clipplayer;
import javax.swing.*;
```

```

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
public class Client implements ActionListener
{
    JFrame frmMusicPlayer;
    final ButtonGroup buttonGroup = new ButtonGroup();
    JLabel lblNewLabel;
    JRadioButton melody,folkmusic,popmusic;
    JPanel panel;
    Socket socket;
    DataOutputStream dos;
    OutputStream outputStream;
    Client() throws Exception
    {
        socket = new Socket("localhost",13085);
        //InputStream inputStream = socket.getInputStream();
        outputStream = socket.getOutputStream();
        dos = new DataOutputStream(outputStream);
        //DataInputStream dis = new DataInputStream(inputStream);
        frmMusicPlayer = new JFrame();
        frmMusicPlayer.setVisible(true);
        frmMusicPlayer.getContentPane().setBackground(new Color(255, 255,
255));
        frmMusicPlayer.getContentPane().setForeground(new Color(255, 255,
255));
        frmMusicPlayer.setBackground(new Color(255, 255, 255));
        frmMusicPlayer.setTitle("Music Player");
        frmMusicPlayer.setBounds(100, 100, 450, 300);
        frmMusicPlayer.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frmMusicPlayer.getContentPane().setLayout(null);

        panel = new JPanel();
        panel.setBounds(40, 98, 374, 122);
        frmMusicPlayer.getContentPane().add(panel);
        panel.setLayout(null);

        lblNewLabel = new JLabel("Choose your favourite Genre !");
        lblNewLabel.setFont(new Font("Arial Black", Font.BOLD, 14));
        lblNewLabel.setBounds(103, 30, 245, 50);
        frmMusicPlayer.getContentPane().add(lblNewLabel);

        folkmusic = new JRadioButton("Folk Music ");

```

```

        folkmusic.setBounds(40, 22, 83, 70);
        panel.add(folkmusic);
        buttonGroup.add(folkmusic);
        folkmusic.setBackground(new Color(255, 255, 204));
        folkmusic.addActionListener(this);

        popmusic = new JRadioButton("Pop Music");
        popmusic.setBounds(148, 22, 83, 70);
        panel.add(popmusic);
        buttonGroup.add(popmusic);
        popmusic.setBackground(new Color(255, 255, 204));
        popmusic.addActionListener(this);

        melody = new JRadioButton("Melody");
        melody.setBounds(257, 22, 83, 70);
        panel.add(melody);
        buttonGroup.add(melody);
        melody.setBackground(new Color(255, 255, 204));
        melody.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e)
    {
        try
        {
            if (e.getSource() == popmusic)
            {
                System.out.println("Pop Music");
                String selected = "Pop Music";
                dos.writeUTF(selected);
            }
            else if(e.getSource() == melody)
            {
                System.out.println("Melody");
                String selected = "Melody";
                dos.writeUTF(selected);
            }
            else if(e.getSource() == folkmusic)
            {
                System.out.println("Folk Music");
                String selected = "Folk Music";
                Clipplayer.FolkGUI.createGUI();
                dos.writeUTF(selected);
            }
        }
    }
}

```

```

        catch(Exception ae)
        {
            ae.printStackTrace();
        }
    }
    public static void main(String args[])
    {
        try
        {
            @SuppressWarnings("unused")
            Client c = new Client();
        }
        catch(Exception e)
        {
            System.out.println("exp");
        }
    }
};

```

These are the source codes for the main Client and Server. When the Client chooses a genre, he/she will get another frame of that particular genre.

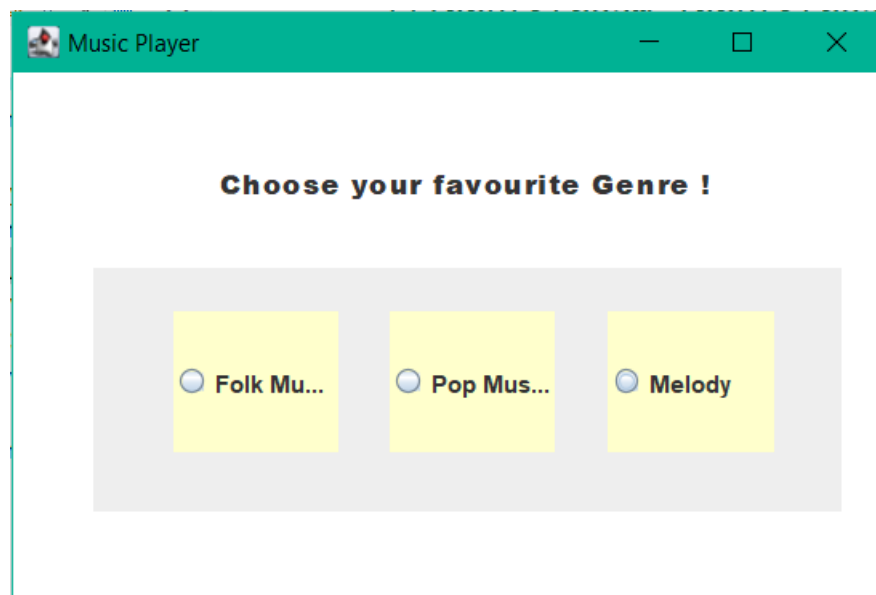


Figure 6: GUI asking Client to choose their Favorite genre

When the Client chooses Folk Music, then the files from the Server of genre Folk Music are sent to the Client and the new frame opens as shown below.

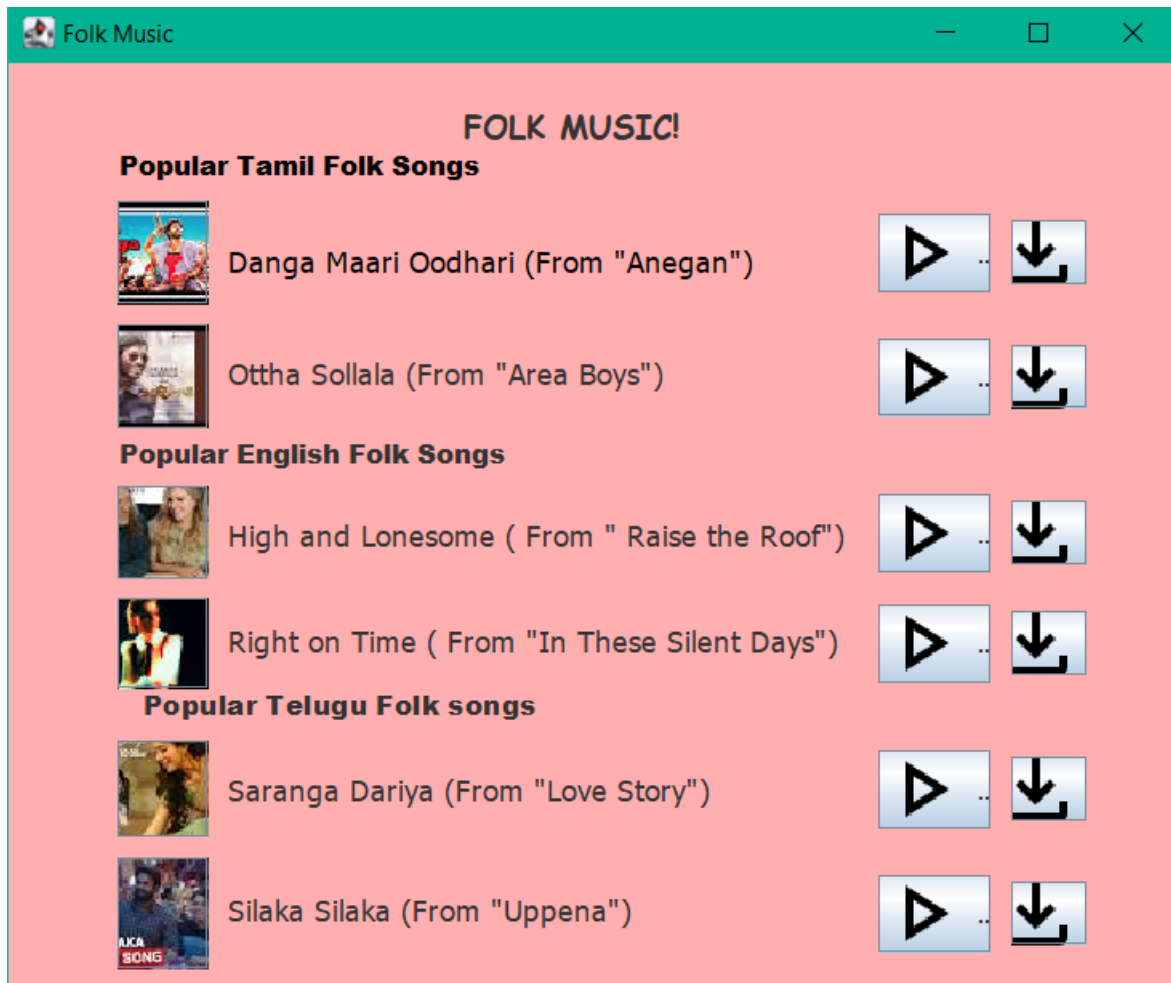


Figure 7: GUI Frame – if selected Folk Music Genre

FolkGUI.java

```
package Clipplayer;
```

```
import java.awt.EventQueue;
```

```
import javax.swing.JFrame;
```

```
import javax.swing.JPanel;
```

```
import javax.swing.JLabel;
```

```
import java.awt.Font;
```

```
import javax.swing.JButton;
```

```
import javax.swing.ImageIcon;
```

```
import java.awt.Dimension;
```

```
import java.awt.event.ActionListener;
```

```
import java.awt.event.ActionEvent;
```

```
import java.awt.Color;
```

```

public class FolkGUI
{
    JFrame frmFolkMusic;
    public static void main(String[] args)
    {
        EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
                try
                {
                    FolkGUI window = new FolkGUI();
                    window.frmFolkMusic.setVisible(true);
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }
            }
        });
    }
    public FolkGUI()
    {
        createGUI();
    }
    static void createGUI()
    {
        initialize();
    }
    static void initialize()
    {
        JFrame frmFolkMusic;
        frmFolkMusic = new JFrame();
        frmFolkMusic.getContentPane().setBackground(Color.PINK);
        frmFolkMusic.setForeground(Color.YELLOW);
        frmFolkMusic.setBackground(Color.BLACK);
        frmFolkMusic.setTitle("Folk Music");
        frmFolkMusic.setMinimumSize(new Dimension(600, 500));
        frmFolkMusic.setSize(new Dimension(1214, 636));
        frmFolkMusic.setSize(631,430);
        frmFolkMusic.setBounds(100, 100, 450, 300);
        frmFolkMusic.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frmFolkMusic.getContentPane().setLayout(null);
    }
}

```



```

JPanel panel = new JPanel();
panel.setBackground(Color.PINK);
panel.setFont(new Font("Arial Black", Font.PLAIN, 13));
panel.setMinimumSize(new Dimension(100, 100));
panel.setBounds(34, 10, 515, 453);
frmFolkMusic.getContentPane().add(panel);
panel.setLayout(null);

JLabel lblNewLabel = new JLabel("FOLK MUSIC! ");
lblNewLabel.setBounds(193, 0, 128, 43);
lblNewLabel.setFont(new Font("Comic Sans MS", Font.BOLD, 16));
panel.add(lblNewLabel);

JLabel lblNewLabel_1 = new JLabel("Popular Tamil Folk Songs");
lblNewLabel_1.setForeground(new Color(0, 0, 0));
lblNewLabel_1.setFont(new Font("Arial Black", Font.PLAIN, 13));
lblNewLabel_1.setBounds(21, 32, 206, 17);
panel.add(lblNewLabel_1);

JButton btnNewButton = new JButton("New button");
btnNewButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
    }
});
btnNewButton.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\danga
mari.jpeg"));
btnNewButton.setBounds(21, 59, 45, 52);
panel.add(btnNewButton);

JButton btnNewButton_1 = new JButton("New button");
btnNewButton_1.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\otha solala.jpeg"));
btnNewButton_1.setBounds(21, 121, 45, 52);
panel.add(btnNewButton_1);

JLabel lblNewLabel_2 = new JLabel("Popular English Folk Songs");
lblNewLabel_2.setFont(new Font("Arial Black", Font.PLAIN, 13));
lblNewLabel_2.setBounds(21, 168, 206, 32);
panel.add(lblNewLabel_2);

JButton btnNewButton_2 = new JButton("New button");

```

```

        btnNewButton_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            }
        });
        btnNewButton_2.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\high and
lonesome.jpeg"));
        btnNewButton_2.setBounds(21, 202, 45, 46);
        panel.add(btnNewButton_2);

        JButton btnNewButton_3 = new JButton("New button");
        btnNewButton_3.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\right on
time.jpeg"));
        btnNewButton_3.setBounds(21, 258, 45, 45);
        panel.add(btnNewButton_3);

        JLabel lblNewLabel_3 = new JLabel("Popular Telugu Folk songs");
        lblNewLabel_3.setFont(new Font("Arial Black", Font.BOLD, 13));
        lblNewLabel_3.setBounds(33, 302, 206, 17);
        panel.add(lblNewLabel_3);

        JButton btnNewButton_4 = new JButton("New button");
        btnNewButton_4.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            }
        });
        btnNewButton_4.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\silaka
silaka.jpeg"));
        btnNewButton_4.setBounds(21, 387, 45, 56);
        panel.add(btnNewButton_4);

        JButton btnNewButton_5 = new JButton("New button");
        btnNewButton_5.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music player\\saranga
dariya.jpeg"));
        btnNewButton_5.setBounds(21, 329, 45, 48);
        panel.add(btnNewButton_5);

        JLabel lblNewLabel_4 = new JLabel("Danga Maari Oodhari (From
\\'Anegan\\'"));
        lblNewLabel_4.setForeground(Color.BLACK);
        lblNewLabel_4.setFont(new Font("Tahoma", Font.PLAIN, 15));

```

```

lblNewLabel_4.setBounds(76, 68, 265, 43);
panel.add(lblNewLabel_4);

JLabel lblNewLabel_5 = new JLabel("Ottha Sollala (From \"Area
Boys\")");
lblNewLabel_5.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel_5.setBounds(76, 124, 221, 43);
panel.add(lblNewLabel_5);

JLabel lblNewLabel_6 = new JLabel("Right on Time ( From \"In These
Silent Days\")");
lblNewLabel_6.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel_6.setBounds(76, 263, 315, 32);
panel.add(lblNewLabel_6);

JLabel lblNewLabel_7 = new JLabel("High and Lonesome ( From \"
Raise the Roof\")");
lblNewLabel_7.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel_7.setBounds(76, 210, 330, 32);
panel.add(lblNewLabel_7);

JLabel lblNewLabel_8 = new JLabel("Saranga Dariya (From \"Love
Story\")");
lblNewLabel_8.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel_8.setBounds(76, 329, 265, 48);
panel.add(lblNewLabel_8);

JLabel lblNewLabel_9 = new JLabel("Silaka Silaka (From
\"Uppena\")");
lblNewLabel_9.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel_9.setBounds(76, 396, 233, 34);
panel.add(lblNewLabel_9);

JButton btnNewButton_6 = new JButton("New button");
btnNewButton_6.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
btnNewButton_6.setBounds(467, 69, 38, 32);
panel.add(btnNewButton_6);

JButton btnNewButton_7 = new JButton("New button");
btnNewButton_7.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)

```

```

        {
            String fname = "Danga Maari Oodhari";
            try
            {
                Clipplayer.PlaySong.playsong(fname);
            }
            catch(Exception da)
            {
                da.printStackTrace();
            }
        }
    });
    btnNewButton_7.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
    btnNewButton_7.setBounds(401, 66, 56, 39);
    panel.add(btnNewButton_7);

    JButton btnNewButton_7_1 = new JButton("New button");
    btnNewButton_7_1.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            String fname = "Otha Sollala";
            try
            {
                Clipplayer.PlaySong.playsong(fname);
            }
            catch(Exception da)
            {
                da.printStackTrace();
            }
        }
    });
    btnNewButton_7_1.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
    btnNewButton_7_1.setBounds(401, 128, 56, 39);
    panel.add(btnNewButton_7_1);

    JButton btnNewButton_7_2 = new JButton("New button");
    btnNewButton_7_2.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {

```

```

        String fname = "High And Lonesome";
        try
        {
            Clipplayer.PlaySong.playsong(fname);
        }
        catch(Exception da)
        {
            da.printStackTrace();
        }
    }
});
btnNewButton_7_2.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
btnNewButton_7_2.setBounds(401, 206, 56, 39);
panel.add(btnNewButton_7_2);

JButton btnNewButton_7_3 = new JButton("New button");
btnNewButton_7_3.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String fname = "Right On Time";
        try
        {
            Clipplayer.PlaySong.playsong(fname);
        }
        catch(Exception da)
        {
            da.printStackTrace();
        }
    }
});
btnNewButton_7_3.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
btnNewButton_7_3.setBounds(401, 261, 56, 39);
panel.add(btnNewButton_7_3);

JButton btnNewButton_7_4 = new JButton("New button");
btnNewButton_7_4.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String fname = "saranga dariya";
        try

```

```

        {
            Clipplayer.PlaySong.playsong(fname);
        }
        catch(Exception da)
        {
            da.printStackTrace();
        }
    }

});
btnNewButton_7_4.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
btnNewButton_7_4.setBounds(401, 334, 56, 39);
panel.add(btnNewButton_7_4);

JButton btnNewButton_7_5 = new JButton("New button");
btnNewButton_7_5.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        String fname = "Silaka Silaka";
        try
        {
            Clipplayer.PlaySong.playsong(fname);
        }
        catch(Exception da)
        {
            da.printStackTrace();
        }
    }
});
btnNewButton_7_5.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\PLAYBLACK.png"));
btnNewButton_7_5.setBounds(401, 396, 56, 39);
panel.add(btnNewButton_7_5);

JButton btnNewButton_6_1 = new JButton("New button");
btnNewButton_6_1.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
btnNewButton_6_1.setBounds(467, 209, 38, 32);
panel.add(btnNewButton_6_1);

JButton btnNewButton_6_2 = new JButton("New button");
btnNewButton_6_2.setIcon(new

```

```

ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
    btnNewButton_6_2.setBounds(467, 131, 38, 32);
    panel.add(btnNewButton_6_2);

    JButton btnNewButton_6_3 = new JButton("New button");
    btnNewButton_6_3.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
    btnNewButton_6_3.setBounds(467, 264, 38, 32);
    panel.add(btnNewButton_6_3);

    JButton btnNewButton_6_4 = new JButton("New button");
    btnNewButton_6_4.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
    btnNewButton_6_4.setBounds(467, 337, 38, 32);
    panel.add(btnNewButton_6_4);

    JButton btnNewButton_6_5 = new JButton("New button");
    btnNewButton_6_5.setIcon(new
ImageIcon("C:\\Users\\Welcome\\Downloads\\Images music
player\\outline_file_download_black_24dp.png"));
    btnNewButton_6_5.setBounds(467, 399, 38, 32);
    panel.add(btnNewButton_6_5);
}
}

```

The below code helps us to play a song using the Clip interface.

```

package Clipplayer;
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class PlaySong
{
    public PlaySong()
    {

```

```

    }
    static void playsong(String fname) throws UnsupportedAudioFileException,
IOException, LineUnavailableException
    {
        String filePath;
        Clip clip;
        AudioInputStream audioInputStream;
        File f = new File(fname);
        filePath = f.getAbsolutePath();
        audioInputStream = AudioSystem.getAudioInputStream(new
File(filePath).getAbsolutePath());
        clip = AudioSystem.getClip();
        clip.open(audioInputStream);
        clip.loop(Clip.LOOP_CONTINUOUSLY);
        clip.start();
    }
    public static void main(String[] args)
    {
        try
        {
            @SuppressWarnings("unused")
            PlaySong ps = new PlaySong();
        }
        catch(Exception e)
        {
            System.out.println("Can't play audio !");
            e.printStackTrace();
        }
    }
}

```

The below code is to Transfer Image files between Server and the Client.

Server1.java -> to send an audio file from the Server to the Client.

```

import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import java.io.File;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class Server1
{
    public static void main(String[] args)

```



```

{
try
{
    ServerSocket serverSocket = new ServerSocket();
    Socket client = null;
    serverSocket.bind(new InetSocketAddress(6666));
    if (serverSocket.isBound())
    {
        client = serverSocket.accept();
        OutputStream out = client.getOutputStream();
        while (true)
        {
            AudioInputStream ain = testPlay("Kutty Pattas.wav");
            if (ain != null)
            {
                AudioSystem.write(ain, AudioFileFormat.Type.WAVE,
out);
            }
        }
        serverSocket.close();
    }
}
catch (Exception e)
{
    e.printStackTrace();
}
}

public static AudioInputStream testPlay(String filename)
{
    AudioInputStream din = null;
    try
    {
        File file = new File(filename);
        AudioInputStream in = AudioSystem.getAudioInputStream(file);
        System.out.println("Before :: " + in.available());
        AudioFormat baseFormat = in.getFormat();
        AudioFormat decodedFormat = new
AudioFormat(AudioFormat.Encoding.PCM_UNSIGNED, baseFormat.getSampleRate(),8,
baseFormat.getChannels(), baseFormat.getChannels(),baseFormat.getSampleRate(), false);
        din = AudioSystem.getAudioInputStream(decodedFormat, in);
        System.out.println("After :: " + din.available());
        return din;
    }
    catch (Exception e)
    {
        // Handle exception.
        e.printStackTrace();
    }
    return din;
}
}

```

```
}
```

Client1.java

```
import javax.sound.sampled.*;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.IOException;
import java.net.Socket;

public class Client1
{
    private static Socket socket;
    private static BufferedInputStream inputStream;
    public static void main(String[] args) throws LineUnavailableException
    {
        try
        {
            socket = new Socket("127.0.0.1", 6666);
            if (socket.isConnected())
            {
                inputStream = new BufferedInputStream(socket.getInputStream());
                AudioInputStream ais =
AudioSystem.getAudioInputStream(inputStream);
                try
                {
                    File f = new File("test.wav");
                    AudioSystem.write(ais, AudioFileFormat.Type.WAVE, f);
                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }
                /*// IF YOU WANT TO PLAY SOUND DIRECTLY FROM SPEAKERS
COMMENT OUT THE TRY CATCH BLOCK ABOVE
// AND UNCOMMENT THE BELOW SECTION
Clip clip = AudioSystem.getClip();
clip.open(ais);
clip.start();

while (inputStream != null) {
    if (clip.isActive()) {

        System.out.println("***** Buffred *****" +
inputStream.available());

    }
}*/
            }
        }
    }
}
```

```

        catch (IOException | UnsupportedOperationException e)
        {
            System.err.println(e);
        }
    }
}

```

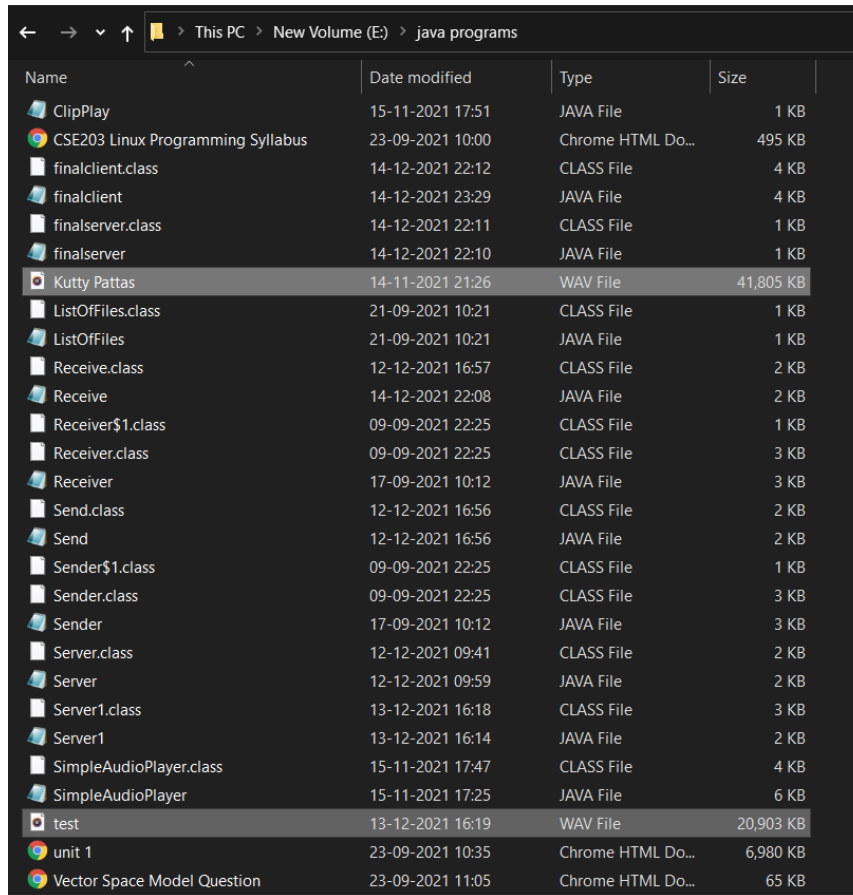


Figure 8: Copy of song Kutty Pattas.wav into test.wav

Server.java -> to transfer an Image from Server to Client

```

import java.io.*;
import java.net.Socket;
import java.nio.ByteBuffer;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;

public class Server
{
    Server() //throws Exception
    {
        ServerSocket serverSocket = new ServerSocket(13085);
        Socket socket = serverSocket.accept();
        System.out.println("Connection Established !");
        OutputStream outputStream = socket.getOutputStream();
    }
}

```

```

        DataInputStream dis=new DataInputStream(socket.getInputStream());
        String selected = dis.readUTF();
        System.out.println(selected);
        if(selected == "Pop Music")
        {
            BufferedImage image = ImageIO.read(new
File("C:\\Users\\Welcome\\Downloads\\Images music player\\cute ponnu.jpeg"));
            ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
            ImageIO.write(image, "jpg", byteArrayOutputStream);
            byte[] size =
ByteBuffer.allocate(4).putInt(byteArrayOutputStream.size()).array();
            outputStream.write(size);
            outputStream.write(byteArrayOutputStream.toByteArray());
            outputStream.flush();
            System.out.println("Flushed: " + System.currentTimeMillis());
            Thread.sleep(120000);
            System.out.println("Closing: " + System.currentTimeMillis());
            socket.close();
        }
    }
    public static void main(String args[])
    {
        //try
        //{
            Server s = new Server();
        //}
        /*catch(Exception e)
        {
            System.out.println("exp");
        }*/
    }
};

```

Client.java

```

package Clipplayer;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.io.*;
import java.awt.image.BufferedImage;
import java.net.ServerSocket;
import java.net.Socket;
import java.nio.ByteBuffer;
public class Client
{

```

```

        Socket socket = new Socket("localhost", 13085);
        InputStream inputStream = socket.getInputStream();
        OutputStream outputStream = socket.getOutputStream();
        DataOutputStream dos = new DataOutputStream(outputStream);
Client() throws Exception
{
    JFrame f = new JFrame("Music Player");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel select = new JLabel("Choose your favourite genre ! ");
    Choice c = new Choice();
    c.add("Pop Music");
    c.add("Melody");
    c.add("Jazz");
    c.add("Folk Music");
    f.add(select);
    f.add(c);
    f.setVisible(true);
    String selected = c.getSelectedItem();
    dos.writeUTF(selected);
    System.out.println("Reading: " + System.currentTimeMillis());
    byte[] sizeAr = new byte[4];
    inputStream.read(sizeAr);
    int size = ByteBuffer.wrap(sizeAr).asIntBuffer().get();
    byte[] imageAr = new byte[size];
    inputStream.read(imageAr);
    BufferedImage image = ImageIO.read(new ByteArrayInputStream(imageAr));
    System.out.println("Received " + image.getHeight() + "x" + image.getWidth() + ": " +
System.currentTimeMillis());
    ImageIO.write(image, "jpg", new File("C:\\Users\\Welcome\\Downloads\\Images music
player"));
    serverSocket.close();
}
public static void main(String args[])
{
    try
    {
        Client c =new Client();
    }
    catch(Exception e)
    {
        System.out.println("exp");
    }
}
};

```

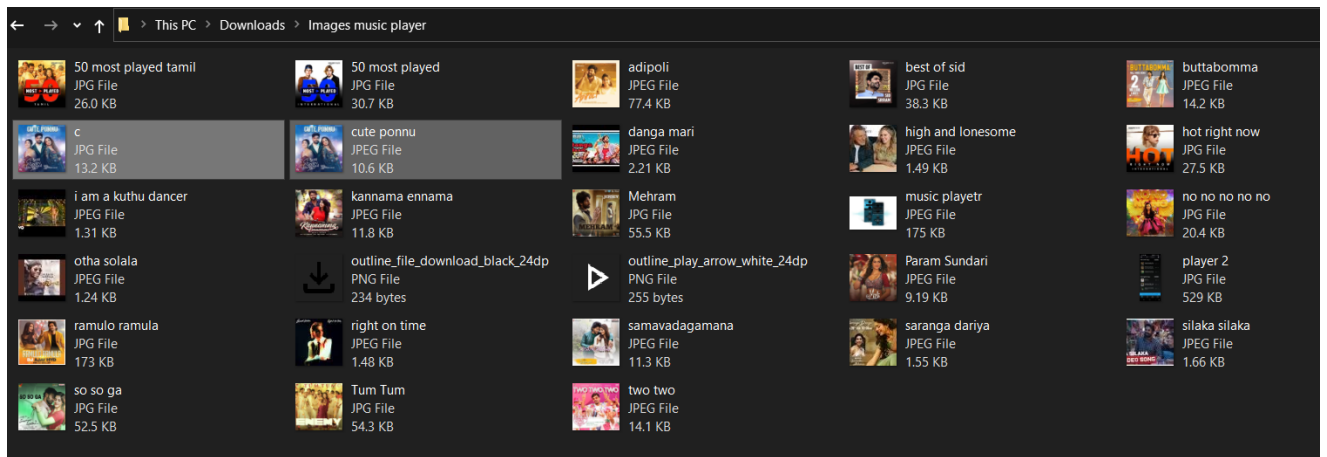


Figure 9: Copy of image cute ponnu.jpeg to c.jpeg

REFERENCES

1. <https://www.geeksforgeeks.org/play-audio-file-using-java/>
2. <https://stackoverflow.com/questions/22454354/transfer-image-file-through-socket-programming-in-java>
3. <https://stackoverflow.com/questions/27117567/java-transfer-a-file-from-server-to-client-and-from-client-to-server/27117668>
4. <https://www.javatpoint.com/socket-programming>

CONCLUSION & FUTURE WORKS

A real time Music player is developed using the Clip interface in java which helped the play, resume, pause, stop operations on the Music player.

Socket programming made the application easily transfer the music, image, text files from the server to the Client based on the Client's choice of choosing their favourite genre. JRE(Java Runtime Environment) provided an all-featured Swing package which helped in developing the Music player as a user-friendly one. Sockets played a very important role in communicating the Client's choice efficiently to the Server which in return made the Server to quickly respond and present to the Client the required window application.

This project can further be improvised by including the following ideas:

- Broadcasting the music – one Server and many Clients.
- Continuous looping of a song and a playlist.
- Creation of user's favourite playlist based on previous plays of the user.
- Creation of a temporary queue at that instance.