JAVA MUSIC PLAYER USING CLIP INTERFACE AND SOCKET PROGRAMMING

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

CSE302: COMPUTER NETWORKS

Submitted by

DIVYA NARASHIMHAN
(Reg. No.: 123003061, CSE)

February 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING
THANJAVUR – 613 401

<u>Bonafide Certificate</u>

This is to certify that the report titled "**Java Music Player Using Clip Interface and Socket Programming**" submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Ms. Divya Narashimhan (Reg. No.123003061,CSE**) during the academic year 2020-21, in the School of Computing.

Project Based Work *Viva voc*e 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.Umamakeswri, 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 also thank him for providing me an opportunity to do this project and for his guidance 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.

ABBREVIATIONS

ACK	Acknowledgment
CLI	Command Line Interface
TCP	Transmission Control Protocol
DNS	Domain Name System
FTP	File Transfer Protocol

ABSTRACT

Java Music Player, a simple Wav music player, is one of the classic applications of various layers of computer networks that explore various operations and dimensions from playing a music like playing selected mp3 music files, pausing the music, resuming the music, and stopping the music. With an added Client Server Architecture, the typical transfer of image and audio files from server to client (Player to Listener) here. Music helps users to create a fresh mind, inspire life, and also boost the mind of the user.

In this project, FTP is implemented when transfer of files happen, and the socket Programming in itself is the implementation of TCP, so Application layer FTP over Transport layer TCP is implemented here.

Java Music Player Functionalities
- Play the mp3 music.
- Pause / resume the music.
- Stop the music.
- Transfer of images
- Transfer of Audio files
- A well designed GUI

KEY WORDS - Socket Programming, Image Transfer, File Transfer, FTP, TCP, Client, Server, Application layer, Socket layer

TABLE OF CONTENTS

INTRODUCTION - BASIC CONCEPTS

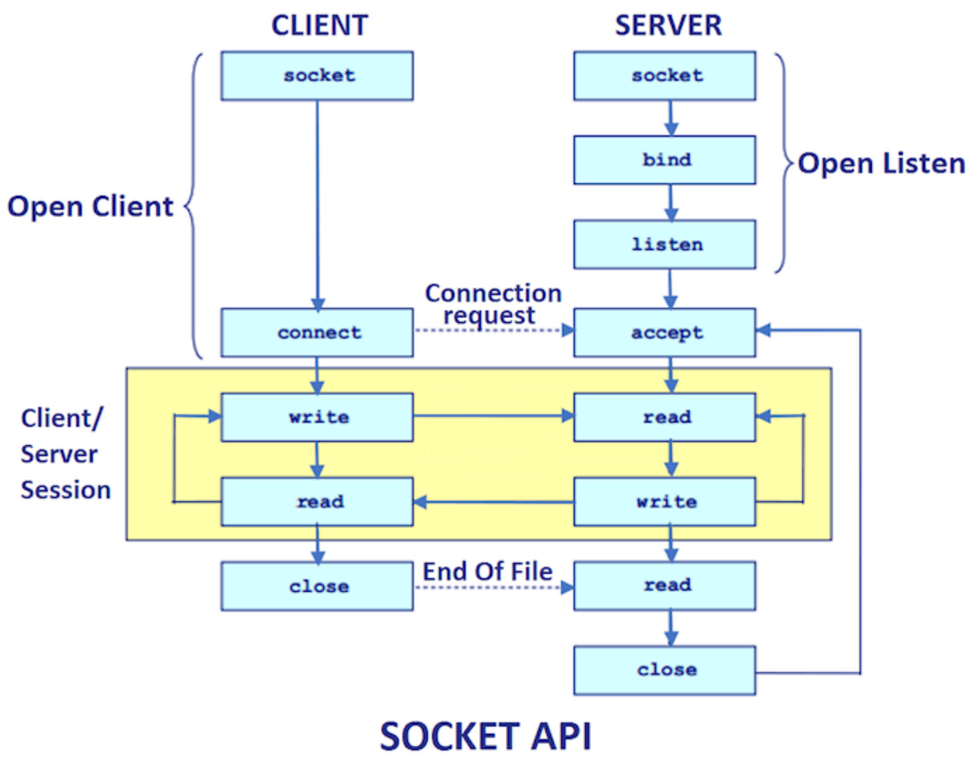
SOCKET PROGRAMMING

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket(“127.0.0.1”, 5000)

- The first argument – IP address of Server. (127.0.0.1 is the IP address of localhost,where code will run on the single stand-alone machine).
- The second argument – TCP Port. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

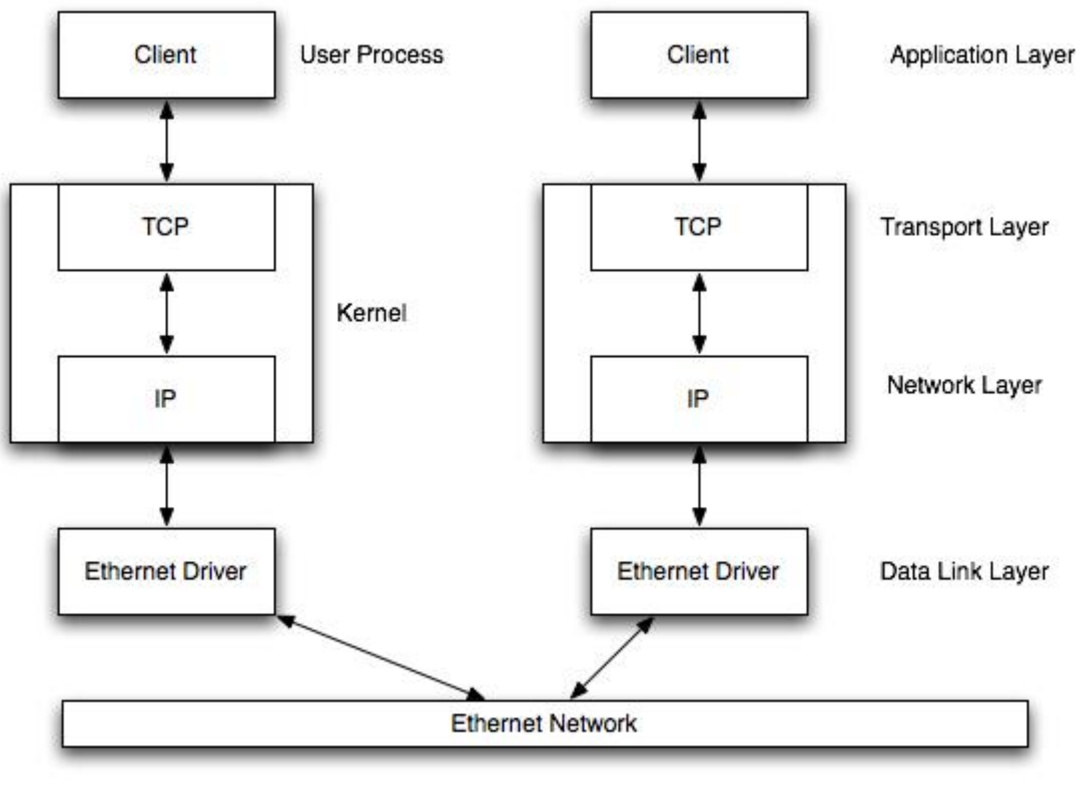
To communicate over a socket connection, streams are used to both input and output the data.



The client-server model

The client-server model is one of the most used communication paradigms in networked systems. Clients normally communicates with one server at a time. From a server's perspective, at any point in time, it is not unusual for a server to be communicating with multiple clients. Client need to know of the existence of and the address of the server, but the server does not need to know the address of (or even the existence of) the client prior to the connection being established

Client and servers communicate by means of multiple layers of network protocols



SERVER

Creating Server:

To create the server application, we create the instance of ServerSocket class. We use a port number for the communication between the client and server. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

Example :

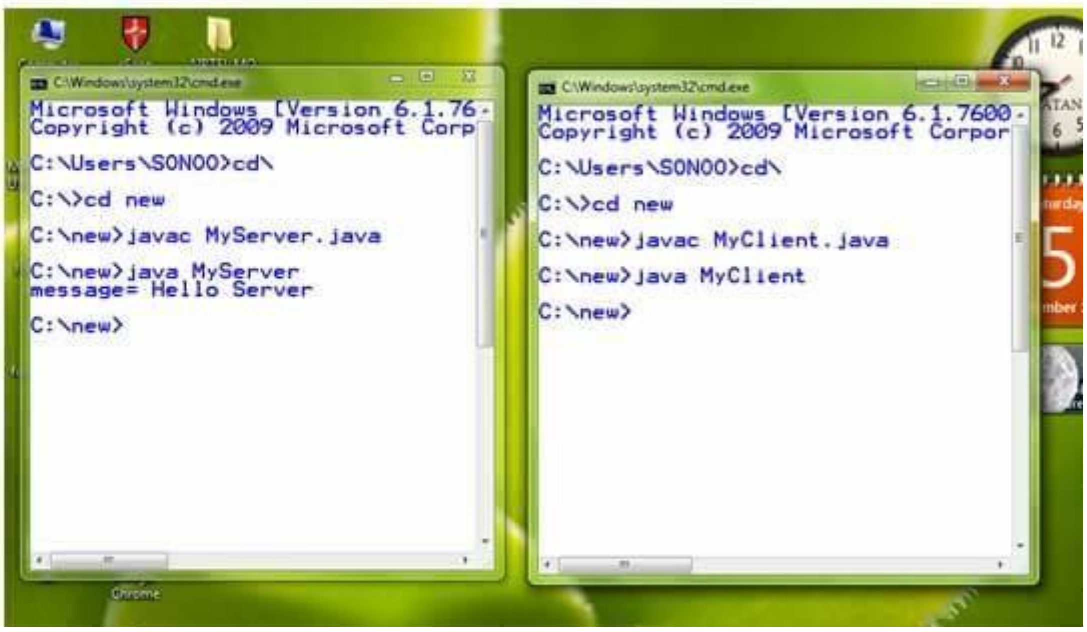
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client

Creating Client:

To create the client application, we need to create the instance of Socket class. we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

Socket s=new Socket("localhost",6666);

Running both the client and server code simultaneously, we get the required output i.e the connection is established.



```
Microsoft Windows [Version 6.1.76
Copyright (c) 2009 Microsoft Corp

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyServer.java

C:\new>java MyServer
message= Hello Server

C:\new>
```

```
Microsoft Windows [Version 6.1.7600
Copyright (c) 2009 Microsoft Corpor

C:\Users\SONOO>cd\

C:\>cd new

C:\new>javac MyClient.java

C:\new>java MyClient

C:\new>
```

TCP

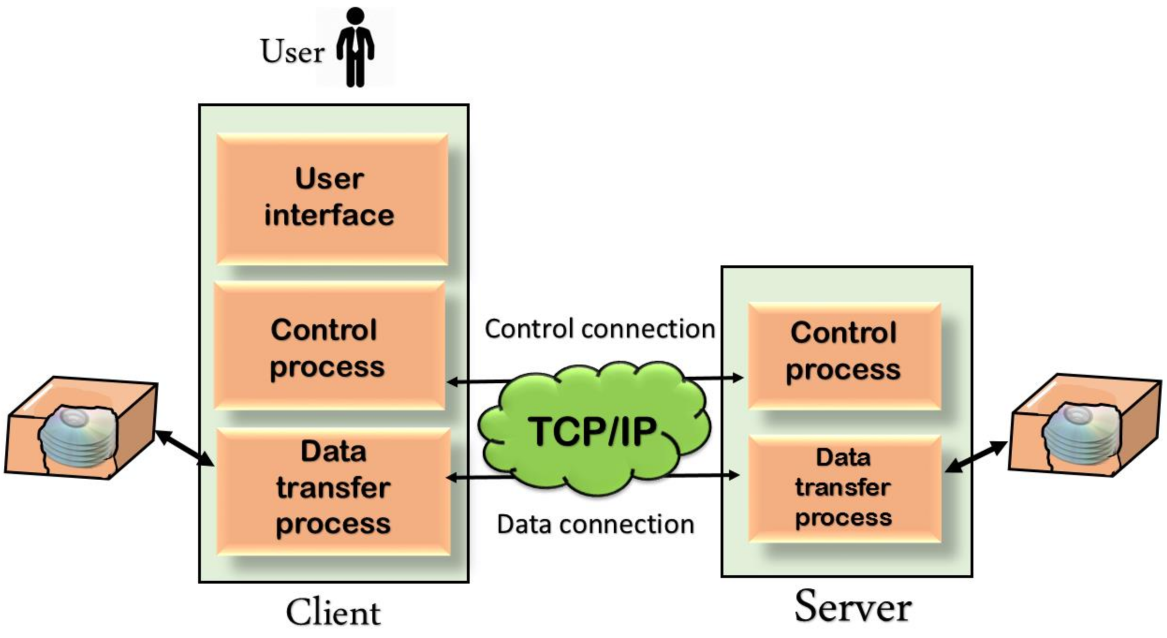
The **Transmission Control Protocol** (TCP) is a transport protocol that is used on top of IP to ensure reliable transmission of packets. TCP includes mechanisms to solve many of the problems that arise from packet-based messaging, such as lost packets, out of order packets, duplicate packets, and corrupted packets.

TCP sockets provide an open bi-directional connection between two endpoints. Each connection is uniquely identified using the combination of the client socket and server socket, which in turn contains four elements: the client IP address and port, and the server IP address and port. We call this a TCP socket pair.

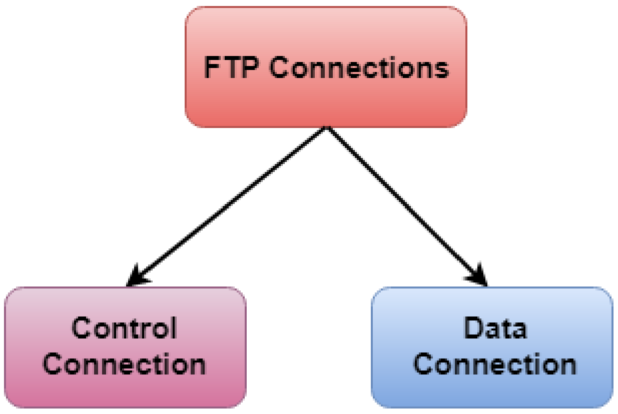
FTP

The File Transfer Protocol is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server.

Although transferring files from one system to another is very simple and straightforward, but sometimes it can cause problems. For example, two systems may have different file conventions. Two systems may have different ways to represent text and data. Two systems may have different directory structures. FTP protocol overcomes these problems by establishing two connections between hosts. One connection is used for data transfer, and another connection is used for the control connection.



There are two types of connections in FTP:



- **Control Connection:** The control connection uses very simple rb ules for communication. Through control connection, we can transfer a line of command or line of response at a time. The control connection is made between the control processes. The control connection remains connected during the entire interactive FTP session.
- **Data Connection:** The Data Connection uses very complex rules as data types may vary. The data connection is made between data transfer processes. The data connection opens when a command comes for transferring the files and closes when the file is transferred.

PREMILINARIES

3.1 JAVA JDK

The Java Development Kit (JDK) is a cross-plat formed software development environment that offers a collection of tools and libraries necessary for developing Java-based software applications and applets. It is a core package used in Java, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment).

JDK contains:

- Java Runtime Environment (JRE),
- An interpreter/loader (Java),
- A compiler (javac),
- An archiver (jar) and many more.

The Java Runtime Environment in JDK is usually called Private Runtime because it is separated from the regular JRE and has extra contents. The Private Runtime in JDK contains a JVM and all the class libraries present in the production environment, as well as additional libraries useful to developers, e.g., internationalization libraries and the IDL libraries.

3.2 ECLIPSE IDE

Developers are always on the lookout for the best IDE for their work. The Eclipse IDE has been around since the inception of the Eclipse project. Since its release in 2000, the Eclipse IDE is widely used by developers around the world.

The Eclipse IDE is an open-source and free IDE with a modular architecture. It is written in Java and is considered to be the most popular Java IDE. It works on all main platforms including Linux, Windows, Mac OS, etc. and has powerful features that can be used to carry out full-fledged projects. It also provides documentation and modeling support and offers UML, OCL, SysML, implementation tools. Besides that, it provides support for Git, Apache Maven, Gradle, etc.

3.1 JAVA PACKAGES

The Java packages used for the corresponding purposes are mentioned below:

1. **java.io**: Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations. We can perform file handling in Java by Java I/O API.

2. **java.util:** Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string

tokenizer, a random-number generator, and a bit array).

3. **javax.sound:** Main subpackages of javax.sound namely javax.sound.sampled and javax.sound.sampled.spi Provides interfaces and classes for capture, processing, and playback of sampled audio data and Supplies abstract classes for service providers to subclass when offering new audio devices, sound file readers and writers, or audio format converters respectively

4. javax.net : Provides **the classes necessary to create an applet** and the classes an applet uses to communicate with its applet context. Contains all of the classes for creating user interfaces and for painting graphics and images.

5. Javax.swing : **Java Swing tutorial** is a part of Java Foundation Classes (JFC) that *is* used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

BASIC MODULES:

1. Basic GUI Module for the Wav Player :

   Using the basic java swing function, 3 basic templates one for the basic music frame of displaying all the albums available , one for diplaying the list of songs in a particular album

2. Socket Connection and Transfer of Image Files:

   Using simple socket programming and javax.net and javax.awt.image package transfer of files from player to listener happens here.

3. Socket Connection and Transfer of Audio Files

   With the use of javax.sound package , and javax.net package audio file transfer from player to listener happens.

4. The clip interface to Perform various operations of music

   Clip interface is used to play music in java. With the use of inbuild player class and play() function we can play a song. Similarly stop, pause and resume functions are also being implemented.

SOURCE CODE

Listener class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import java.awt.Color;
import javax.swing.JButton;
import javax.swing.JTextArea;
import javax.swing.ImageIcon;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;


public class Listener impliments ActionListener
{
	Socket so = null;
	DataOutputStream dos = null;
	DataInputStream dis= null;
	JFrame frmMusicPlayer;
	private JTextField txtWelcomeToThe;
	private JTextField txtHaveALook;



	public Listener() throws Exception
	{
		initialize();
		so = new Socket("localhost", 1999);
		dos = new DataOutputStream(so.getOutputStream());
		dis=new DataInputStream(so.getInputStream());
		System.out.println(dis.readUTF());

	}


	void initialize()
	{
		frmMusicPlayer = new JFrame();
		frmMusicPlayer.setVisible(true);
		frmMusicPlayer.setTitle("Music Player");
		frmMusicPlayer.getContentPane().setBackground(new
Color(255, 255, 204));
		frmMusicPlayer.setBounds(100, 100, 450, 300);
	
	frmMusicPlayer.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frmMusicPlayer.getContentPane().setLayout(null);
		
		JPanel panel = new JPanel();
		panel.setBackground(new Color(204, 204, 255));
		panel.setBounds(10, 11, 414, 239);
		frmMusicPlayer.getContentPane().add(panel);
		panel.setLayout(null);
		
		txtWelcomeToThe = new JTextField();
		txtWelcomeToThe.setBounds(92, 11, 234, 20);
		txtWelcomeToThe.setFont(new Font("Calibri", Font.PLAIN,
13));
		txtWelcomeToThe.setBackground(new Color(175, 238, 238));
	
	txtWelcomeToThe.setHorizontalAlignment(SwingConstants.CENTER);
		txtWelcomeToThe.setText("     Welcome to the Music
Player!");
		panel.add(txtWelcomeToThe);
		txtWelcomeToThe.setColumns(10);
		
		txtHaveALook = new JTextField();
		txtHaveALook.setBounds(92, 38, 234, 20);
		txtHaveALook.setFont(new Font("Calibri", Font.PLAIN, 13));
		txtHaveALook.setBackground(new Color(175, 238, 238));
	
	txtHaveALook.setHorizontalAlignment(SwingConstants.CENTER);
		txtHaveALook.setText("Have a look at your fav albums :)");
		txtHaveALook.setColumns(10);
		panel.add(txtHaveALook);
		
		JButton btnNewButton = new JButton("New button");
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				
			}
		});
		
		btnNewButton.setBounds(22, 81, 101, 92);
		btnNewButton.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\Reputation - Taylor
Swift.jpg"));
		panel.add(btnNewButton);
		
		JButton btnNewButton_1 = new JButton("New button");
		btnNewButton_1.setBounds(155, 81, 101, 92);
		btnNewButton_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\SOUR- Olivia_Rodrigo.png"));
		panel.add(btnNewButton_1);
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				GUI.SourGUI.GUI2();
			}
		});
		
		JButton btnNewButton_2 = new JButton("New button");
		btnNewButton_2.setBounds(288, 81, 101, 92);
		btnNewButton_2.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\Red( Taylor's Version) - Taylor
Swift.jpg"));
		panel.add(btnNewButton_2);
		
		JTextArea txtrReputationTaylorSwift = new JTextArea();
		txtrReputationTaylorSwift.setBounds(22, 183, 101, 34);
		txtrReputationTaylorSwift.setFont(new Font("Calibri",
Font.PLAIN, 13));
		txtrReputationTaylorSwift.setText(" Reputation\r\nTaylor
Swift\r\n");
		txtrReputationTaylorSwift.setBackground(new Color(255,
255, 204));
		panel.add(txtrReputationTaylorSwift);
		
		JTextArea txtrSourOliviaRodrigo = new JTextArea();
		txtrSourOliviaRodrigo.setBounds(155, 183, 101, 34);
		txtrSourOliviaRodrigo.setFont(new Font("Calibri",
Font.PLAIN, 13));
		txtrSourOliviaRodrigo.setText("Sour\r\nOlivia
Rodrigo\r\n");
		txtrSourOliviaRodrigo.setBackground(new Color(204, 255,
255));
		panel.add(txtrSourOliviaRodrigo);
		
		JTextArea txtrRedTaylorSwift = new JTextArea();
		txtrRedTaylorSwift.setBounds(288, 184, 101, 34);
		txtrRedTaylorSwift.setLineWrap(true);
		txtrRedTaylorSwift.setWrapStyleWord(true);
		txtrRedTaylorSwift.setFont(new Font("Calibri", Font.PLAIN, 
13));
		txtrRedTaylorSwift.setText("Red\r\nTaylor Swift\r\n");
		txtrRedTaylorSwift.setBackground(new Color(255, 255, 
204));
		panel.add(txtrRedTaylorSwift);
	}
	public static void main(String[] args)
	{
		/*EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try {
					Listener window = new Listener();
					window.frmMusicPlayer.setVisible(true);
				} 
				catch (Exception e) 
				{
					e.printStackTrace();
				}
			}
		});*/
	try
		{
			Listener l= new Listener();
		}
		catch (Exception ie)
		{
			ie.printStackTrace();
		}
		
	}
}
```

GUI2:

```
package GUI
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JTextArea;
import javax.swing.ImageIcon;
import javax.swing.SwingConstants;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;
import java.awt.Color;

public class SourGUI
{

	JFrame frmSourOliviaRodrigo;


	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					SourGUI window = new SourGUI();

	window.frmSourOliviaRodrigo.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}


	public SourGUI()
	{
	void GUI2()
		{
		initialize();
		}
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frmSourOliviaRodrigo = new JFrame();
		frmSourOliviaRodrigo.setTitle("SOUR- Olivia Rodrigo");
		frmSourOliviaRodrigo.setBackground(new Color(169, 169, 
169));
		frmSourOliviaRodrigo.setBounds(100, 100, 508, 300);
	
	frmSourOliviaRodrigo.setDefaultCloseOperation(JFrame.EXIT_ON_CLO
SE);
		frmSourOliviaRodrigo.getContentPane().setLayout(null);
		
		JPanel panel = new JPanel();
		panel.setBackground(new Color(220, 220, 220));
		panel.setBounds(0, 0, 492, 261);
		frmSourOliviaRodrigo.getContentPane().add(panel);
		panel.setLayout(null);
		
		JButton btnNewButton_1 = new JButton("New button");
		btnNewButton_1.setIcon(new 
ImageIcon("C:\\Users\\Acer\\Pictures\\Good For You.jpg"));
		btnNewButton_1.setBounds(10, 64, 47, 42);
		panel.add(btnNewButton_1);
		
		JButton btnNewButton_2 = new JButton("New button");
		btnNewButton_2.setIcon(new 
ImageIcon("C:\\Users\\Acer\\Pictures\\Driver's Liscence.jpg"));
		btnNewButton_2.setBounds(10, 116, 47, 42);
		panel.add(btnNewButton_2);
		
		JButton btnNewButton_3 = new JButton("New button");
		btnNewButton_3.setIcon(new 
ImageIcon("C:\\Users\\Acer\\Pictures\\Traitor.jpg"));
		btnNewButton_3.setBounds(10, 173, 47, 42);
		panel.add(btnNewButton_3);
		
		JTextArea txtrStepForward = new JTextArea();
		txtrStepForward.setBackground(new Color(216, 191, 216));
		txtrStepForward.setFont(new Font("Calibri", Font.PLAIN, 
13));
		txtrStepForward.setText("1 Step forward and 3 steps 
back");
		txtrStepForward.setBounds(81, 20, 180, 22);
		panel.add(txtrStepForward);
		
		JButton btnNewButton_4_1_1 = new JButton("New button");
		btnNewButton_4_1_1.setBackground(new Color(255, 255, 
255));
	
	btnNewButton_4_1_1.setHorizontalAlignment(SwingConstants.LEFT);
```

```
		btnNewButton_4_1_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\play-button.png"));
		btnNewButton_4_1_1.setBounds(291, 21, 30, 32);
		panel.add(btnNewButton_4_1_1);
		
		JButton btnNewButton_4_1_2 = new JButton("New button");
		
	btnNewButton_4_1_2.setHorizontalAlignment(SwingConstants.LEFT);
		btnNewButton_4_1_2.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\resume-button.png"));
		btnNewButton_4_1_2.setBounds(380, 21, 30, 32);
		panel.add(btnNewButton_4_1_2);
		
		JButton btnNewButton_4_1_2_1 = new JButton("New button");
		
	btnNewButton_4_1_2_1.setHorizontalAlignment(SwingConstants.LEFT)
;
		btnNewButton_4_1_2_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\stop-button.png"));
		btnNewButton_4_1_2_1.setBounds(420, 21, 30, 32);
		panel.add(btnNewButton_4_1_2_1);
		
		JButton btnNewButton_4_1_1_1 = new JButton("New button");
		btnNewButton_4_1_1_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\pause-button.png"));
		btnNewButton_4_1_1_1.setBounds(340, 21, 30, 32);
		panel.add(btnNewButton_4_1_1_1);
		
		JButton btnNewButton_4_1_1_2 = new JButton("New button");
		btnNewButton_4_1_1_2.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\play-button.png"));
		
	btnNewButton_4_1_1_2.setHorizontalAlignment(SwingConstants.LEFT)
;
		btnNewButton_4_1_1_2.setBounds(291, 74, 30, 32);
		panel.add(btnNewButton_4_1_1_2);
		
		JButton btnNewButton_4_1_1_2_1 = new JButton("New
button");
		btnNewButton_4_1_1_2_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\pause-button.png"));
		btnNewButton_4_1_1_2_1.setBounds(340, 74, 30, 32);
		panel.add(btnNewButton_4_1_1_2_1);
		
		JButton btnNewButton_4_1_1_2_2 = new JButton("New
button");
		btnNewButton_4_1_1_2_2.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\resume-button.png"));
```

```
	btnNewButton_4_1_1_2_2.setHorizontalAlignment(SwingConstants.LEF
T);
		btnNewButton_4_1_1_2_2.setBounds(380, 74, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2);

		JButton btnNewButton_4_1_1_2_2_1 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_1.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\stop-button.png"));

	btnNewButton_4_1_1_2_2_1.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_1.setBounds(420, 74, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_1);

		JButton btnNewButton_4_1_1_2_2_2 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_2.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\play-button.png"));

	btnNewButton_4_1_1_2_2_2.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_2.setBounds(291, 126, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_2);

		JButton btnNewButton_4_1_1_2_2_3 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_3.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\pause-button.png"));
		btnNewButton_4_1_1_2_2_3.setBounds(340, 126, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_3);

		JButton btnNewButton_4_1_1_2_2_4 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_4.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\resume-button.png"));

	btnNewButton_4_1_1_2_2_4.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_4.setBounds(380, 126, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_4);

		JButton btnNewButton_4_1_1_2_2_5 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_5.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\stop-button.png"));
```

```
	btnNewButton_4_1_1_2_2_5.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_5.setBounds(420, 126, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_5);

		JButton btnNewButton_4_1_1_2_2_6 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_6.setBackground(new Color(255, 255,
255));
		btnNewButton_4_1_1_2_2_6.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\play-button.png"));

	btnNewButton_4_1_1_2_2_6.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_6.setBounds(291, 170, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_6);

		JButton btnNewButton_4_1_1_2_2_7 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_7.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\pause-button.png"));
		btnNewButton_4_1_1_2_2_7.setBounds(340, 170, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_7);

		JButton btnNewButton_4_1_1_2_2_8 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_8.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\resume-button.png"));

	btnNewButton_4_1_1_2_2_8.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_8.setBounds(380, 170, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_8);

		JButton btnNewButton_4_1_1_2_2_9 = new JButton("New
button");
		btnNewButton_4_1_1_2_2_9.setIcon(new
ImageIcon("C:\\Users\\Acer\\Pictures\\stop-button.png"));

	btnNewButton_4_1_1_2_2_9.setHorizontalAlignment(SwingConstants.L
EFT);
		btnNewButton_4_1_1_2_2_9.setBounds(420, 173, 30, 32);
		panel.add(btnNewButton_4_1_1_2_2_9);

		JTextArea txtrGoodForYou = new JTextArea();
		txtrGoodForYou.setBackground(new Color(216, 191, 216));
		txtrGoodForYou.setFont(new Font("Calibri", Font.PLAIN,
13));
```

```
		txtrGoodForYou.setText("Good for you ");
		txtrGoodForYou.setBounds(81, 73, 180, 22);
		panel.add(txtrGoodForYou);

		JTextArea txtrDriversLiscence = new JTextArea();
		txtrDriversLiscence.setBackground(new Color(216, 191, 
216));
		txtrDriversLiscence.setFont(new Font("Calibri", 
Font.PLAIN, 13));
		txtrDriversLiscence.setText("Drivers Liscence ");
		txtrDriversLiscence.setBounds(81, 125, 180, 22);
		panel.add(txtrDriversLiscence);

		JTextArea txtrTraitorOliviaRodrigo = new JTextArea();
		txtrTraitorOliviaRodrigo.setBackground(new Color(216, 191, 
216));
		txtrTraitorOliviaRodrigo.setWrapStyleWord(true);
		txtrTraitorOliviaRodrigo.setFont(new Font("Calibri", 
Font.PLAIN, 13));
		txtrTraitorOliviaRodrigo.setText("Traitor");
		txtrTraitorOliviaRodrigo.setBounds(81, 182, 180, 22);
		panel.add(txtrTraitorOliviaRodrigo);

		JButton btnNewButton_1_1 = new JButton("New button");
		btnNewButton_1_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
			}
		});
		btnNewButton_1_1.setIcon(new 
ImageIcon("C:\\Users\\Acer\\Pictures\\1 Step Forward And 3 Steps 
Back.jpg"));
		btnNewButton_1_1.setBounds(10, 11, 47, 42);
		panel.add(btnNewButton_1_1);
	}
}
```

Player class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.net.*;

public class player
{
	Socket so = null;
	DataOutputStream dos = null;
```

```
          DataInputStream dis= null;

          player() throws Exception
          {

                    ServerSocket ss = new ServerSocket(1999);
                    so = ss.accept();
                    dis=new DataInputStream(so.getInputStream());
                    dos = new DataOutputStream(so.getOutputStream());
                    dos.writeUTF("Socket Connection Successful");
          }



          public static void main(String[] args)
          {
               try
               {
                    player p = new player();
               }
               catch (Exception e)
               {
                    e.printStackTrace();
               }

          }


};
```

Image Send Class

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.awt.image.BufferedImage;
public class Imagesend
{

      public static void main(String[] args) throws Exception
          {
             Socket socket = new Socket("localhost", 13085);
             OutputStream outputStream = socket.getOutputStream();
```

```
                BufferedImage image = ImageIO.read(new
File("C:\\Users\\Acer\\Documents\\CN Project\\Images\\Olivia -
1D.jpg"));

                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();
            }

}
```

Image Receive Class

```
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.nio.ByteBuffer;
import java.awt.image.BufferedImage;
public class Imagereceive
{

        public static void main(String[] args) throws Exception {
            ServerSocket serverSocket = new ServerSocket(13085);
            Socket socket = serverSocket.accept();
            InputStream inputStream = socket.getInputStream();

            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\\Acer\\Documents\\CN
Project\\Images\\Olivia2 - 1D.jpg"));

                serverSocket.close();
             }

}
```

Audio Send class

```
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 AudioSend
{
	public static void main(String[] args)
	{
	try
	{
		ServerSocket serverSocker = new ServerSocket();
		Socket client = null;
		serverSocker.bind(new InetSocketAddress(6666));
		if (serverSocker.isBound())
		{
			client = serverSocker.accept();
			OutputStream out = client.getOutputStream();
			while (true)
			{
				AudioInputStream ain = testPlay("Olivia.wav");
				if (ain != null)
				{
					AudioSystem.write(ain, AudioFileFormat.Type.WAVE,
out);
				}
			}
		}
```

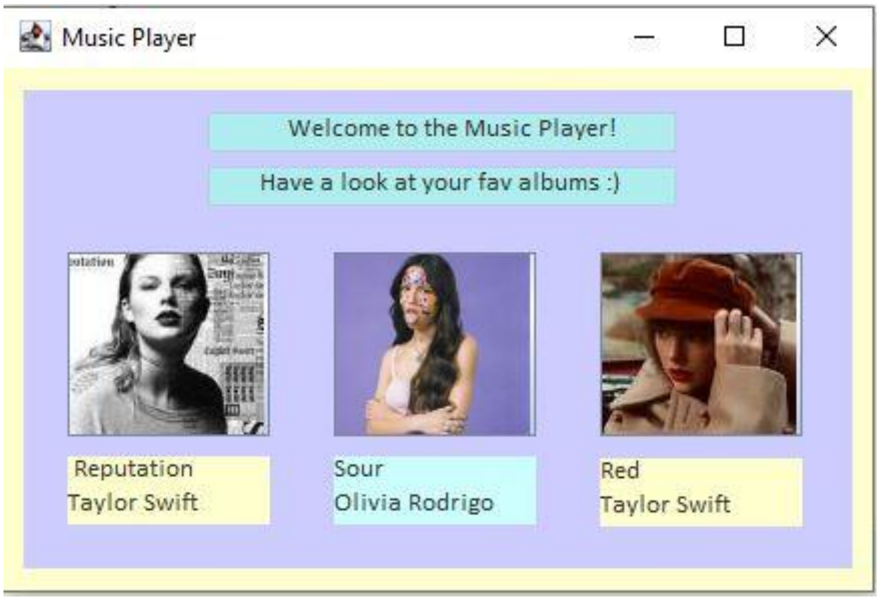
```
			serverSocker.close();
	}
		catch (Exception e)
		{
		e.printStackTrace();
	    }
	}
		public static AudioInputStream testPlay(String filename)
		{
			AudioInputStream din = null;
			try
			{
				File file = new File(filename);
				AudioInputStream in = AudioSystem.getAudioInputStream(file);
				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;
		}
}
```

Audio Receive class

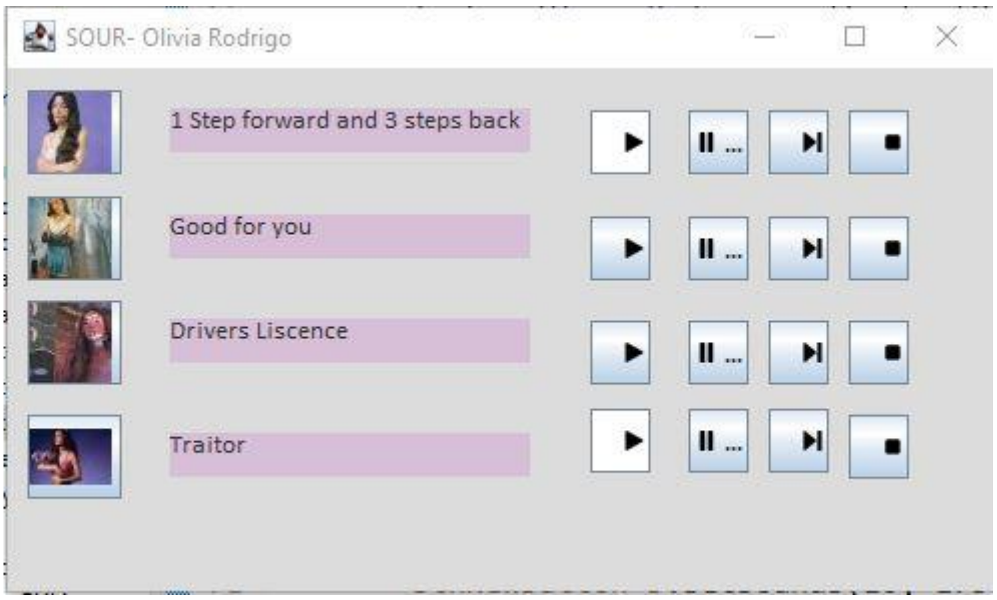
```
import javax.sound.sampled.*;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.IOException;
import java.net.Socket;

public class AudioReceive
{
		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 | UnsupportedAudioFileException e)
                    {
          System.err.println(e);
       }
    }
}
```

OUTPUT SNAPSHOTS



The main frame of The Java Music player showing the albums in the player.



One of the buttons when clicked (sour here) a new frame with the songs in the album is opened

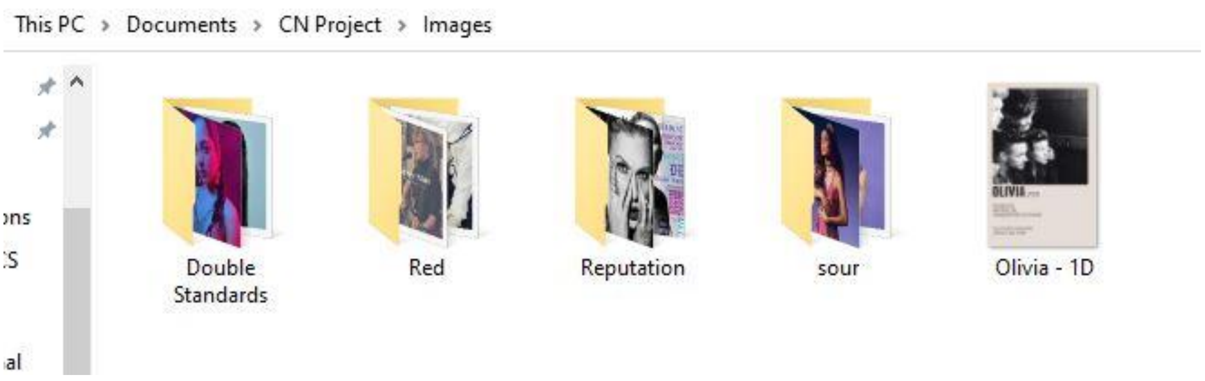


Illustration before the image transfer

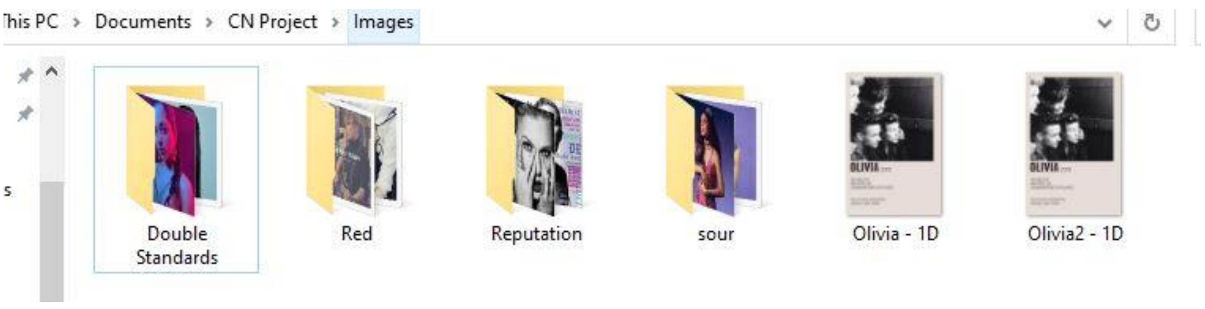
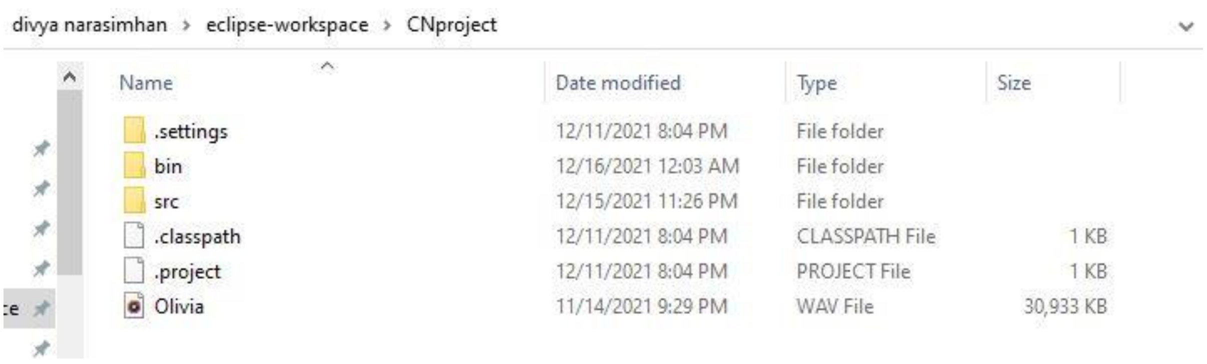
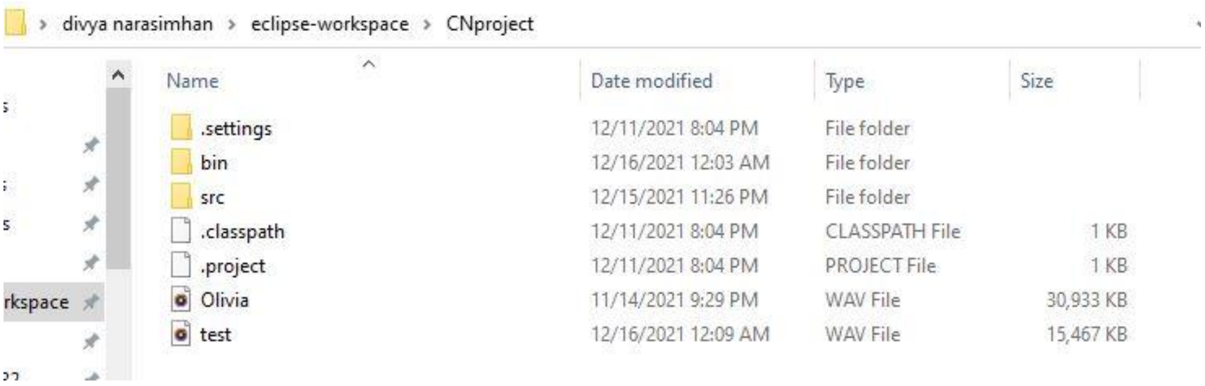


Illustration after the image transfer


divya narasimhan › eclipse-workspace › CNproject

Name	Date modified	Type	Size
.settings	12/11/2021 8:04 PM	File folder	
bin	12/16/2021 12:03 AM	File folder	
src	12/15/2021 11:26 PM	File folder	
.classpath	12/11/2021 8:04 PM	CLASSPATH File	1 KB
.project	12/11/2021 8:04 PM	PROJECT File	1 KB
Olivia	11/14/2021 9:29 PM	WAV File	30,933 KB

View of the folder before the audio file transfer


› divya narasimhan › eclipse-workspace › CNproject

Name	Date modified	Type	Size
.settings	12/11/2021 8:04 PM	File folder	
bin	12/16/2021 12:03 AM	File folder	
src	12/15/2021 11:26 PM	File folder	
.classpath	12/11/2021 8:04 PM	CLASSPATH File	1 KB
.project	12/11/2021 8:04 PM	PROJECT File	1 KB
Olivia	11/14/2021 9:29 PM	WAV File	30,933 KB
test	12/16/2021 12:09 AM	WAV File	15,467 KB

View of the folder after the audio file transfer

CONCLUSION

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.

FUTURE WORKS:

- This project can be further extended by creating a better integration between different java programs.
- A better GUI can be developed and Real time Music playing can be added
- A group listening feature or group song broadcasting can be implemented with more clients in the socket
- Additional Features like Queue and playlists can be created.

REFRENCES

- Wikipedia
- Geeks for Geeks
- Stack Overflow
- Java Network Programming by Elliot Rusty Harold
- Learning Network Programming with java by Richard M. Reese