MUSIC PLAYLIST USING QUEUE DATA STRUCTURE

TABLE OF CONTENTS

ABSTRACT

To solve the problem of complex functions and large required memory (RAM) for music players in the current market, a new music player of simple, convenient, less required memory as well as user-friendly is developed. Based on the current requirements, using the Java language and Eclipse programming tools lead to the design and coding of the music player. The new design mainly realizes a few core functions including the main play interface, playlists, menus, settings and song search. This player has the merits of high performance, simple operation, and runs independently on any device. At the same time, the player can also browse and access files from that particular device.

INTRODUCTION

In modern society, people live a fast-paced life, and pressure is constantly present in their lives. Due to the wide use of mobile phones, music has become the daily essential spiritual food and has become an essential application in a person’s mobile. An application like MP3 music players is used to balance stress and happiness. It accompanies people anytime, anywhere and anyplace even when people are taking the bus and exercising.

The mobile MP3 music player application is designed to allow users to listen to music in a more convenient and comfortable way without too much restriction. Moreover, it can play music properly without interference from advertisements and offline.

Since many developers realize that modern humans are living in a stressful situation, they have captured the commercial opportunity, therefore many similar applications have emerged in the market like Spotify, Gaana, Youtube Music, Amazon music, etc. These applications have easy-to-use interfaces and features that make the user experience better.

However, these existing music players blindly pursue fancy appearance and huge features, resulting in the high utilization rate of users' mobile phones, such as CPU and memory. Whereas, for most normal users, these features are meaningless. Therefore, this project is designed to dedicate to the MP3 music player to optimize performance and simplify to meet user needs.

OBJECTIVE

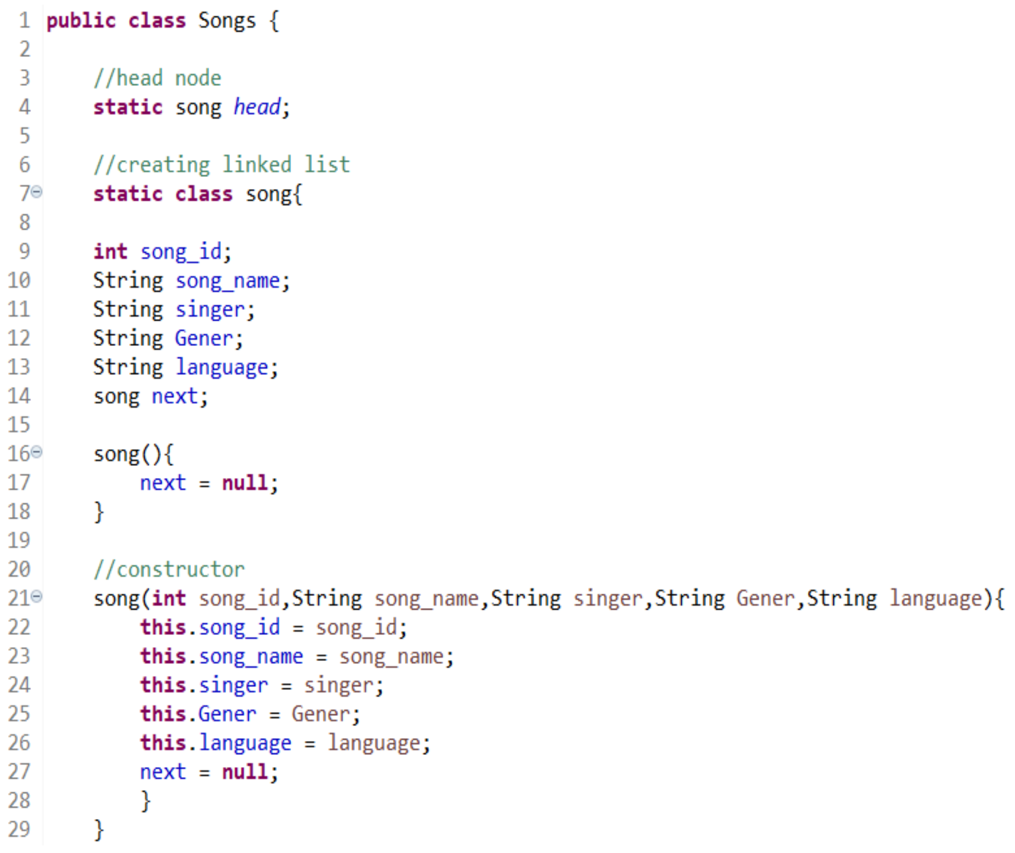
This program is aimed to be a user-friendly application that allows the user to create music playlists according to their choice.

PROPOSED APPROACH

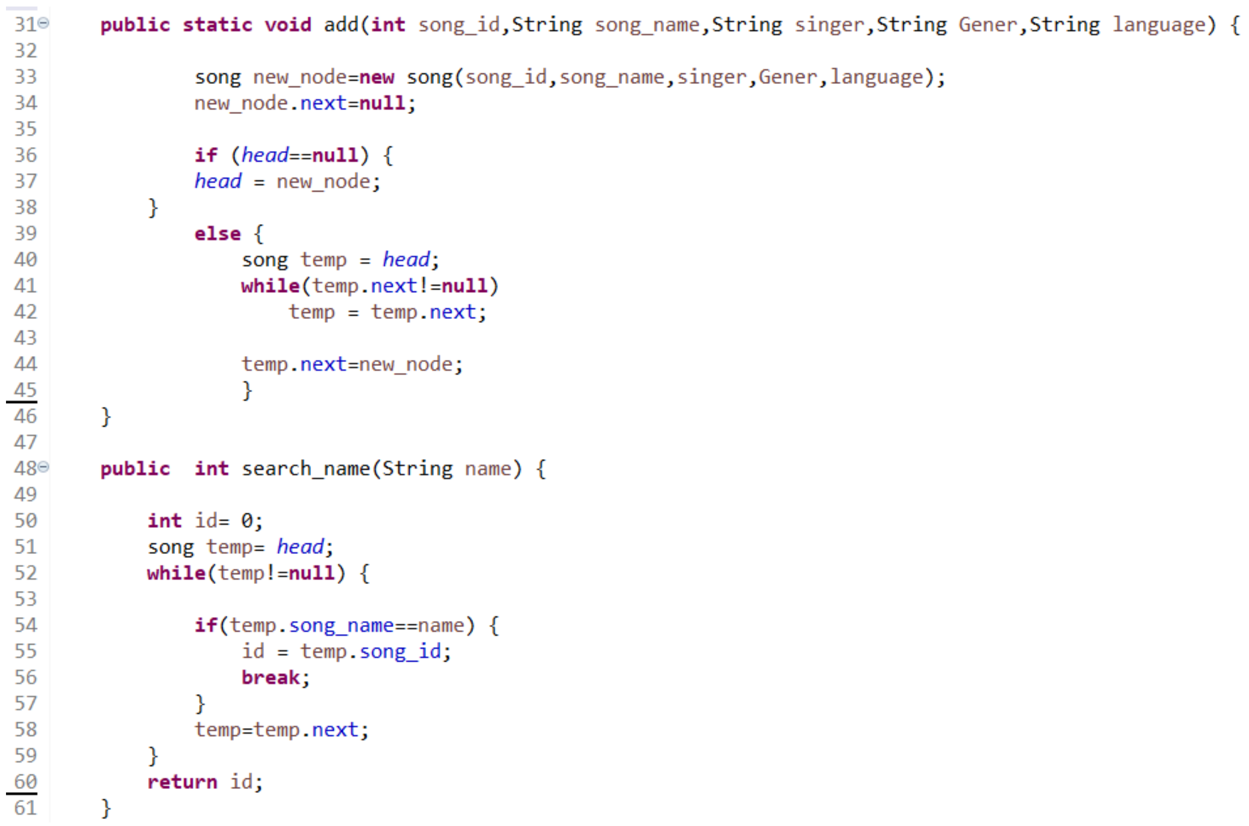
This proposed approach of music player uses Linked List and queue concept to build a playlist and queue for the application. This program is designed in such a way that it allows the user to create any number of playlists and play the songs. LinkedList data structure serves the purpose of storing the playlist and the queue data structure used helps us in temporarily storing the songs that are to be played in a queue.

Songs.java Class

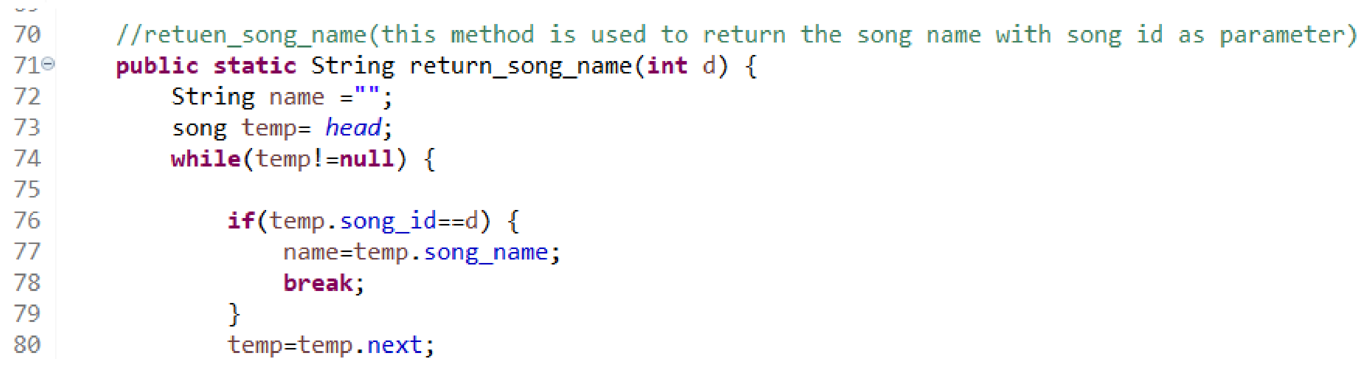
CODE



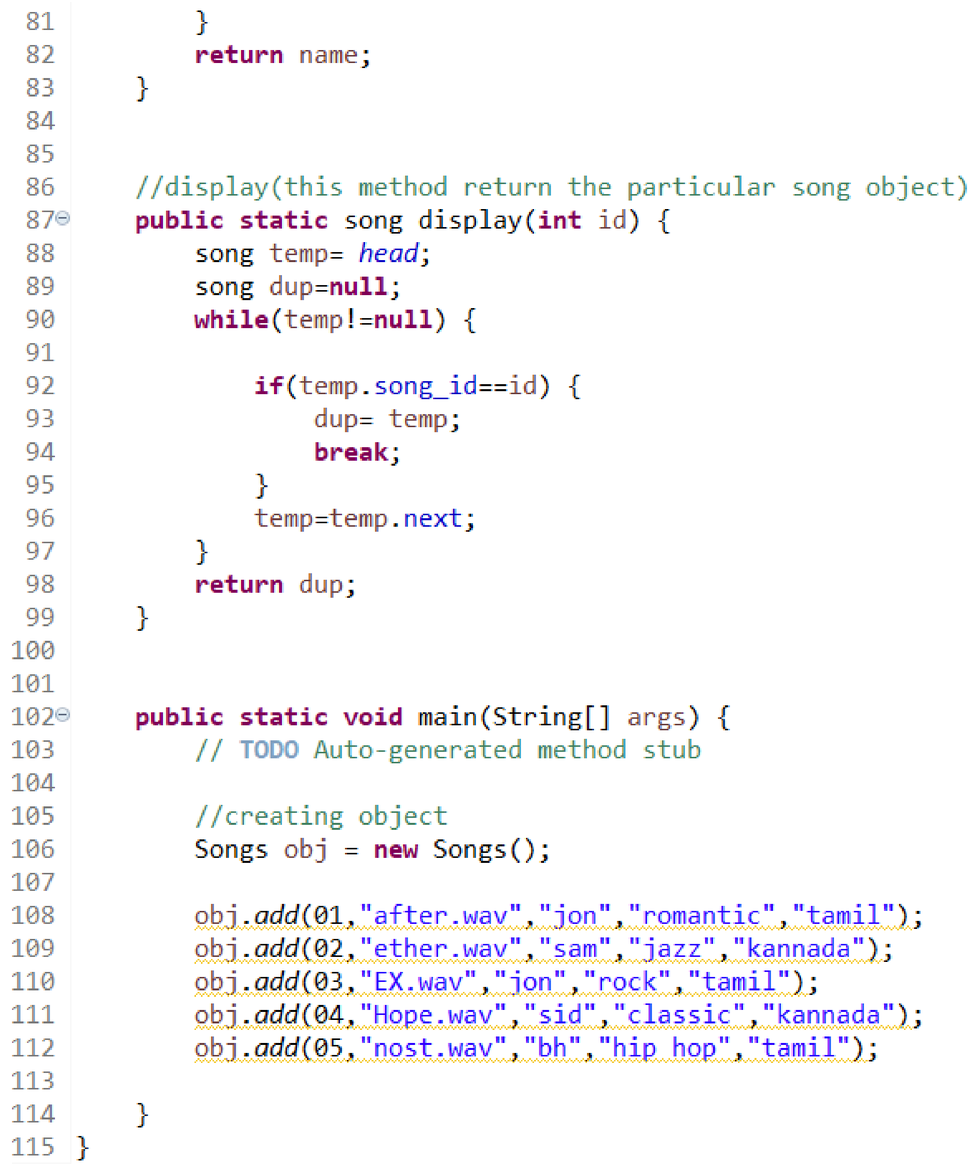
```java
public class Songs {

    //head node
    static song head;

    //creating linked list
    static class song{

    int song_id;
    String song_name;
    String singer;
    String Gener;
    String language;
    song next;

    song(){
        next = null;
    }

    //constructor
    song(int song_id,String song_name,String singer,String Gener,String language){
        this.song_id = song_id;
        this.song_name = song_name;
        this.singer = singer;
        this.Gener = Gener;
        this.language = language;
        next = null;
        }
    }
```



```
public static void add(int song_id,String song_name,String singer,String Gener,String language) {

        song new_node=new song(song_id,song_name,singer,Gener,language);
        new_node.next=null;

        if (head==null) {
        head = new_node;
    }
        else {
            song temp = head;
            while(temp.next!=null)
                temp = temp.next;

            temp.next=new_node;
            }
}

public  int search_name(String name) {

    int id= 0;
    song temp= head;
    while(temp!=null) {

        if(temp.song_name==name) {
            id = temp.song_id;
            break;
        }
        temp=temp.next;
    }
    return id;
}
```



```
//retuen_song_name(this method is used to return the song name with song id as parameter)
public static String return_song_name(int d) {
    String name ="";
    song temp= head;
    while(temp!=null) {

        if(temp.song_id==d) {
            name=temp.song_name;
            break;
        }
        temp=temp.next;
```



```
        }
        return name;
    }


    //display(this method return the particular song object)
    public static song display(int id) {
        song temp= head;
        song dup=null;
        while(temp!=null) {

            if(temp.song_id==id) {
                dup= temp;
                break;
            }
            temp=temp.next;
        }
        return dup;
    }


    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //creating object
        Songs obj = new Songs();

        obj.add(01,"after.wav","jon","romantic","tamil");
        obj.add(02,"ether.wav","sam","jazz","kannada");
        obj.add(03,"EX.wav","jon","rock","tamil");
        obj.add(04,"Hope.wav","sid","classic","kannada");
        obj.add(05,"nost.wav","bh","hip hop","tamil");

    }
}
```

WORKING

This class contains the most important variables which are used to store crucial information about the song in a static class called ***song***. Information like ***song id***, ***song name***, ***singer***, ***song genre*** and ***song language*** are stored. The node comprises all these variables and can be accessed using the dot operator. This class is a linked list with a static song class as the node.

In this class methods are declared to do a few basic operations like adding a song, searching for a song etc.

add() -This method adds new song elements that create objects.

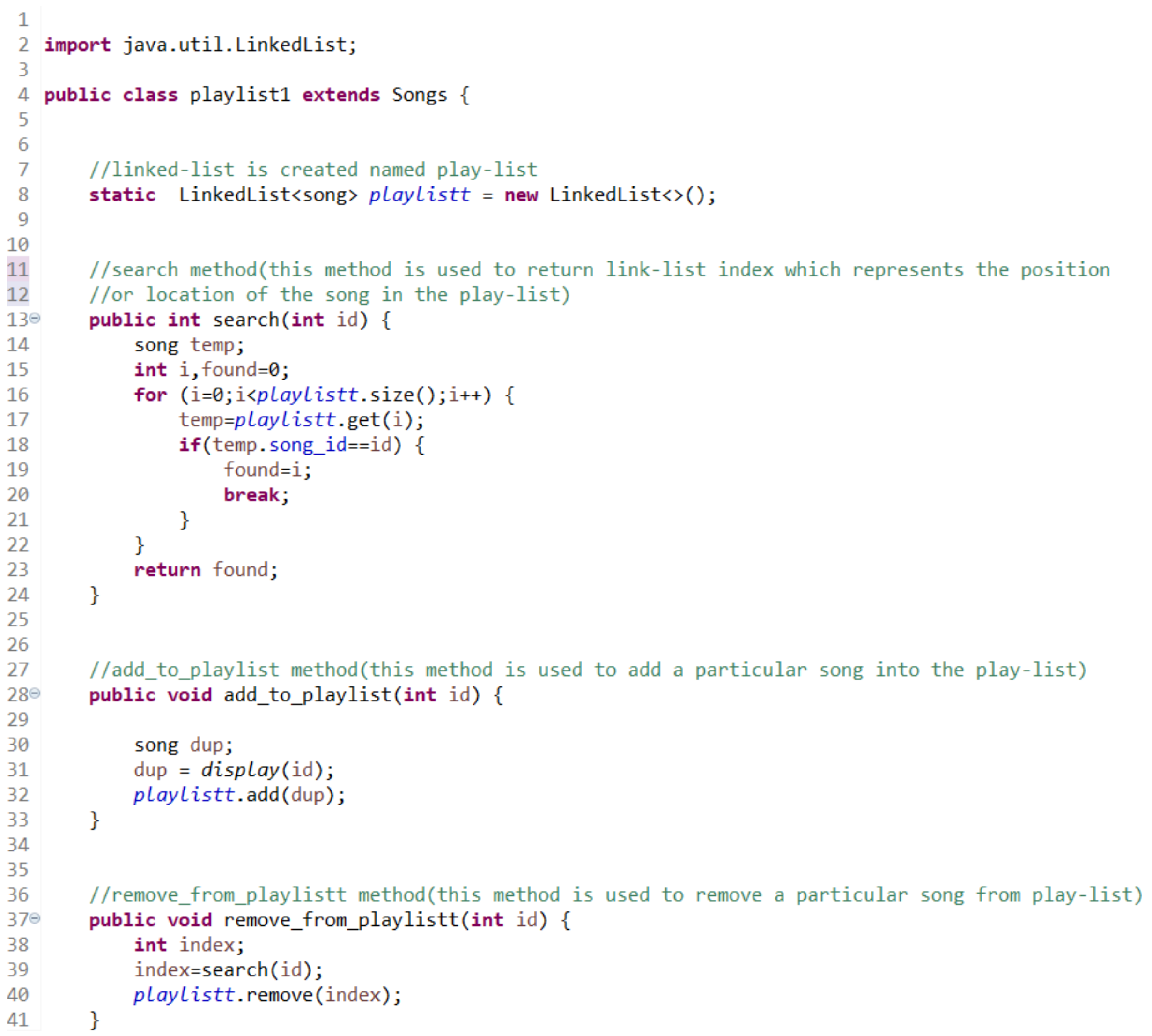
search_name()- This method takes the song name as the parameter and returns the song id.

return_song_name() - This method returns the song id for a input song name which is passed as a parameter(used in operation class to get the song name for the passed song object).

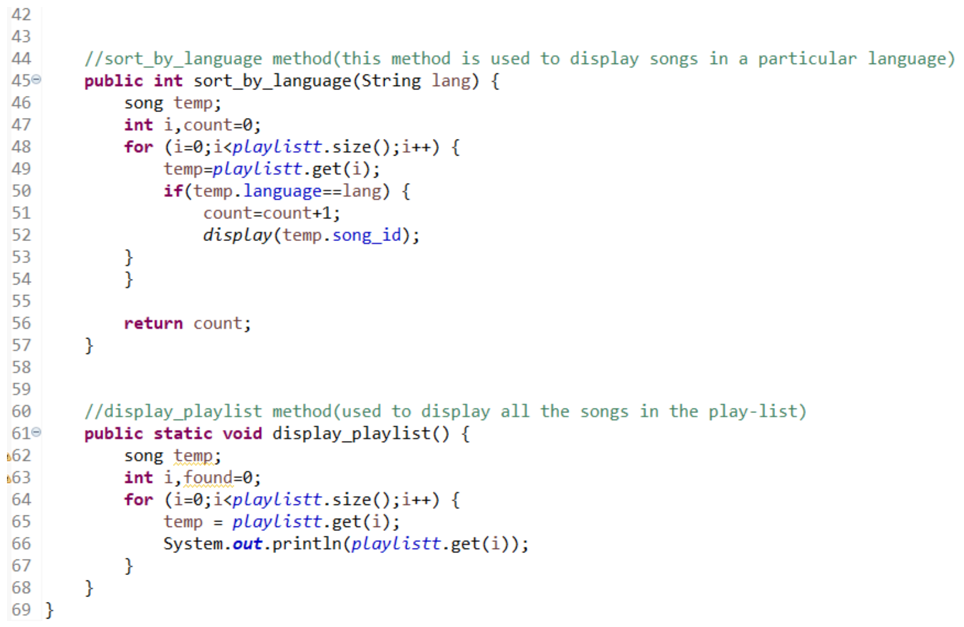
song display()- this method returns a song node for the particular song id passed as the parameter.

<u>playlist1.java Class</u>

<u>CODE</u>



```java

import java.util.LinkedList;

public class playlist1 extends Songs {


    //linked-list is created named play-list
    static  LinkedList<song> playlistt = new LinkedList<>();


    //search method(this method is used to return link-list index which represents the position
    //or location of the song in the play-list)
    public int search(int id) {
        song temp;
        int i,found=0;
        for (i=0;i<playlistt.size();i++) {
            temp=playlistt.get(i);
            if(temp.song_id==id) {
                found=i;
                break;
            }
        }
        return found;
    }


    //add_to_playlist method(this method is used to add a particular song into the play-list)
    public void add_to_playlist(int id) {

        song dup;
        dup = display(id);
        playlistt.add(dup);
    }


    //remove_from_playlistt method(this method is used to remove a particular song from play-list)
    public void remove_from_playlistt(int id) {
        int index;
        index=search(id);
        playlistt.remove(index);
    }
```



```


    //sort_by_language method(this method is used to display songs in a particular language)
    public int sort_by_language(String lang) {
        song temp;
        int i,count=0;
        for (i=0;i<playlistt.size();i++) {
            temp=playlistt.get(i);
            if(temp.language==lang) {
                count=count+1;
                display(temp.song_id);
        }
        }

        return count;
    }


    //display_playlist method(used to display all the songs in the play-list)
    public static void display_playlist() {
        song temp;
        int i,found=0;
        for (i=0;i<playlistt.size();i++) {
            temp = playlistt.get(i);
            System.out.println(playlistt.get(i));
        }
    }
}
```

<u>WORKING</u>

This whole class inherits all the properties from the *Songs* class. It consists of a LinkedList of song type which is used to store songs in a playlist. It also contains few methods that perform operations like adding a song to a playlist, removing a song etc.

search() - It is used to return the index of the song that is passed in the parameter in the linked list which represents the location of the song in the playlist.

add_to_playlist() - This method adds a particular song to the playlist.

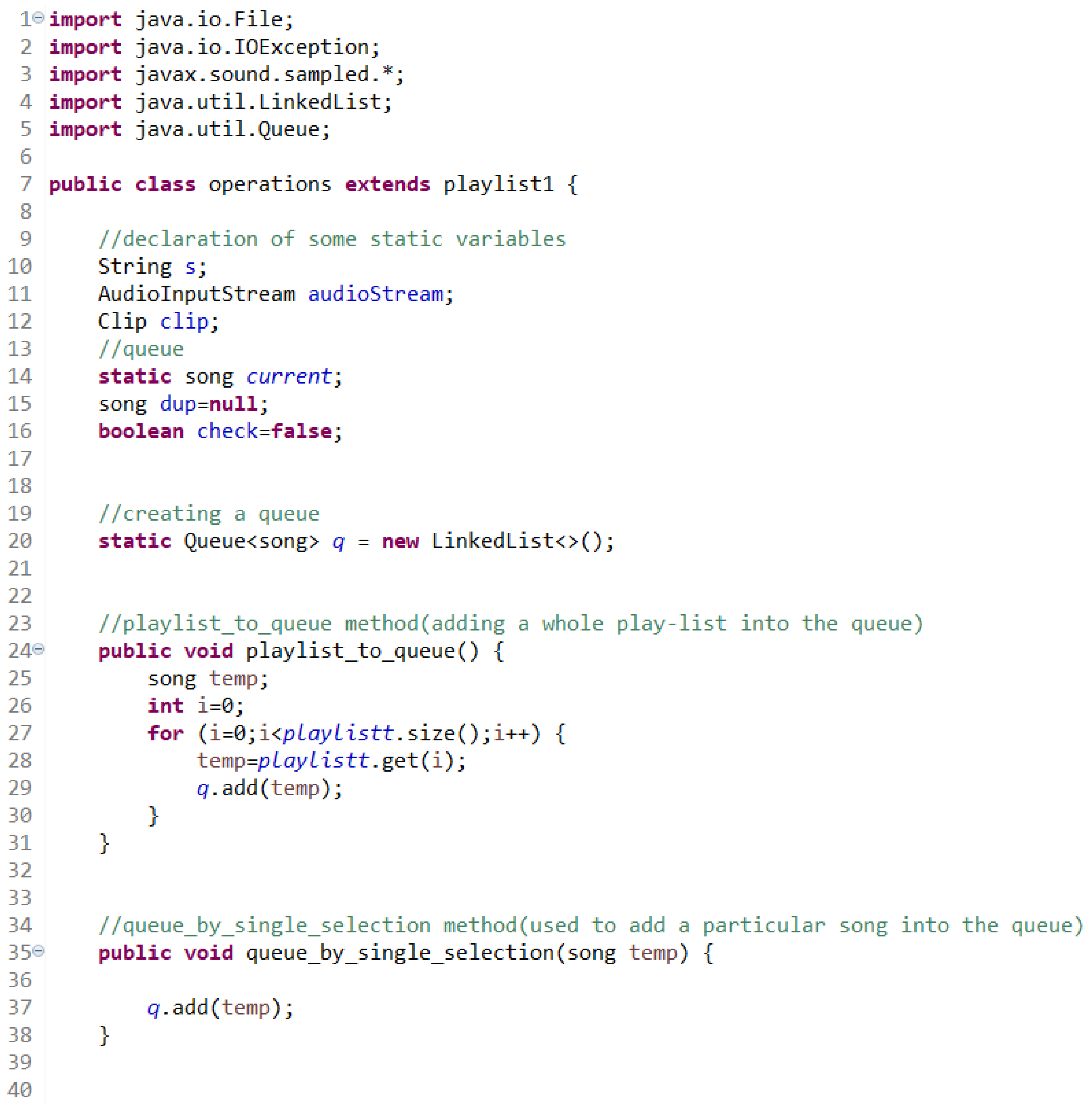
remove_from_playlistt() - It removes or deletes a particular song from the playlist.

sort_by_language() - This method sorts and displays all the songs in a language in that particular playlist.

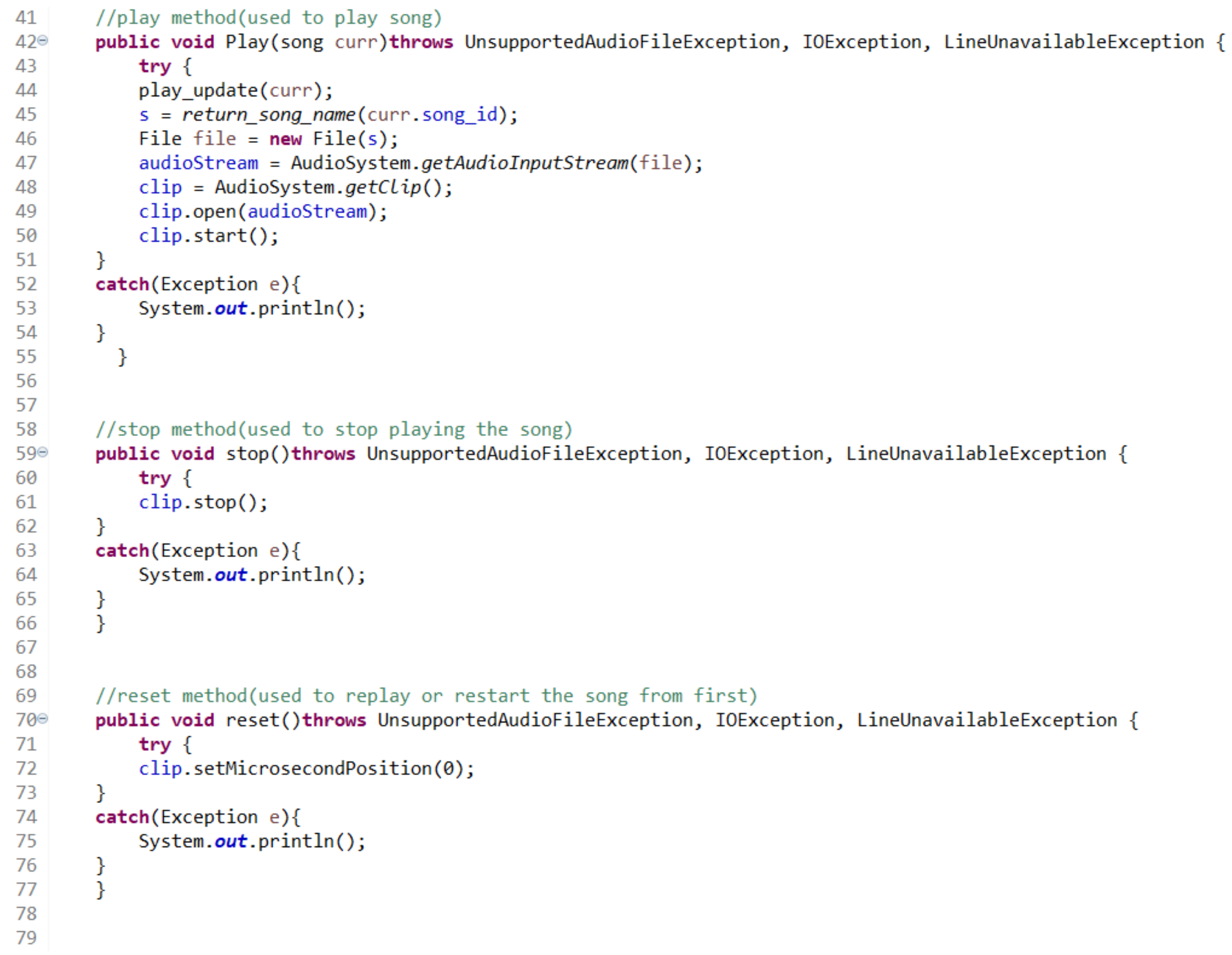
display_playlist() - This method displays all the songs in the playlist.

Operations.java Class

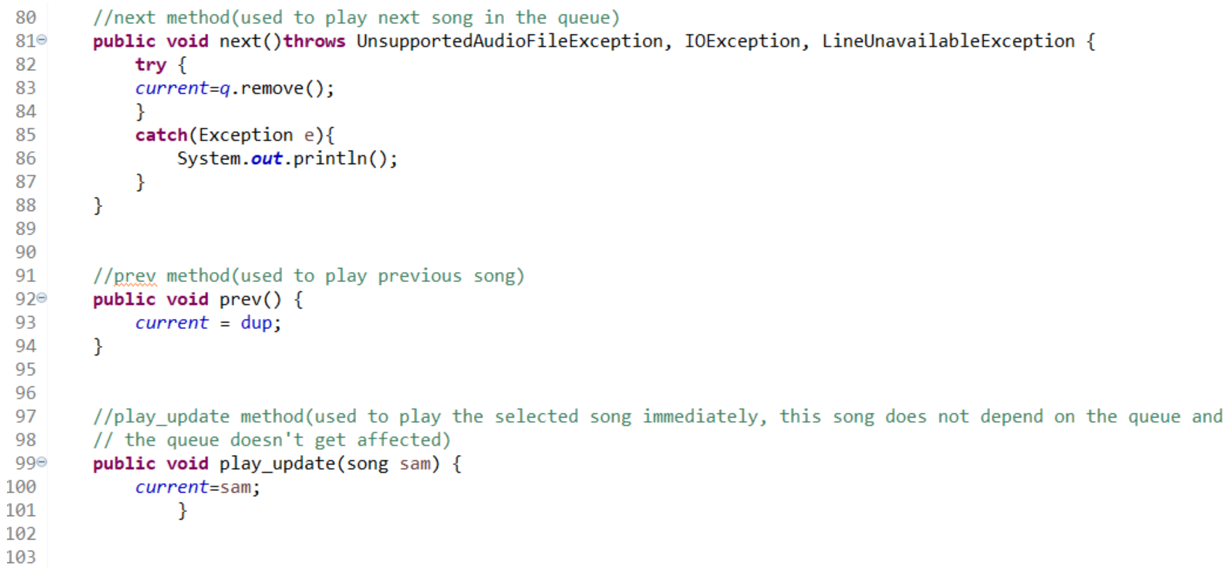
CODE



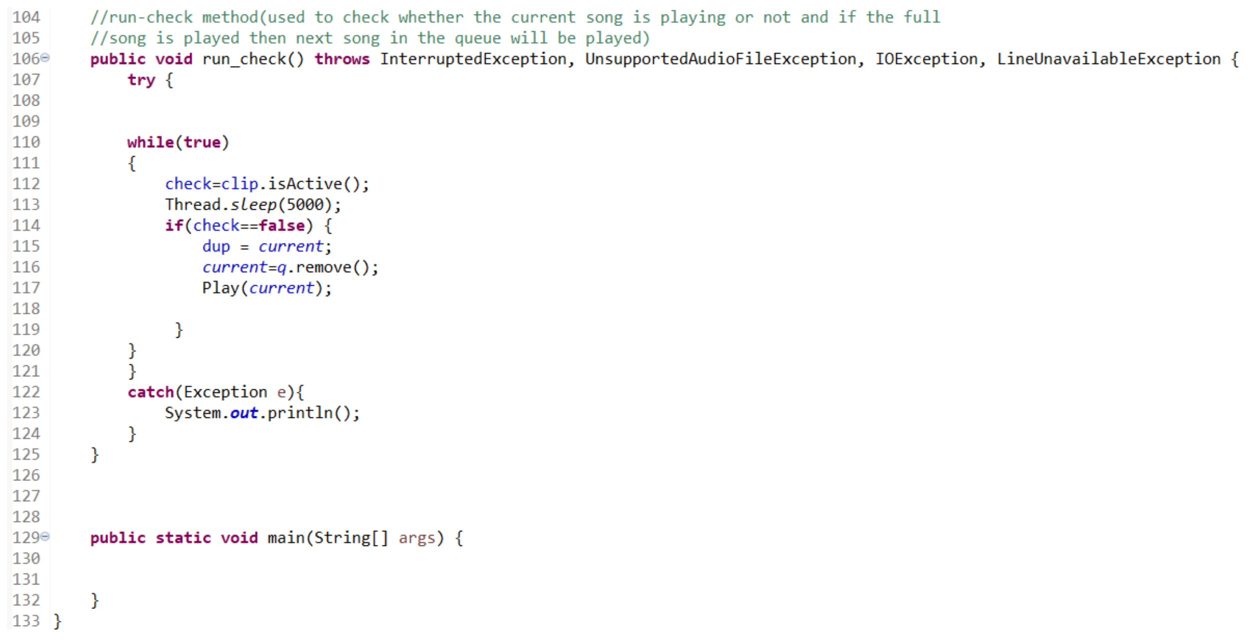
```java
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.*;
import java.util.LinkedList;
import java.util.Queue;

public class operations extends playlist1 {

    //declaration of some static variables
    String s;
    AudioInputStream audioStream;
    Clip clip;
    //queue
    static song current;
    song dup=null;
    boolean check=false;


    //creating a queue
    static Queue<song> q = new LinkedList<>();


    //playlist_to_queue method(adding a whole play-list into the queue)
    public void playlist_to_queue() {
        song temp;
        int i=0;
        for (i=0;i<playlistt.size();i++) {
            temp=playlistt.get(i);
            q.add(temp);
        }
    }


    //queue_by_single_selection method(used to add a particular song into the queue)
    public void queue_by_single_selection(song temp) {

        q.add(temp);
    }


```



```java
    //play method(used to play song)
    public void Play(song curr)throws UnsupportedAudioFileException, IOException, LineUnavailableException {
        try {
        play_update(curr);
        s = return_song_name(curr.song_id);
        File file = new File(s);
        audioStream = AudioSystem.getAudioInputStream(file);
        clip = AudioSystem.getClip();
        clip.open(audioStream);
        clip.start();
    }
    catch(Exception e){
        System.out.println();
    }
      }


    //stop method(used to stop playing the song)
    public void stop()throws UnsupportedAudioFileException, IOException, LineUnavailableException {
        try {
        clip.stop();
    }
    catch(Exception e){
        System.out.println();
    }
    }


    //reset method(used to replay or restart the song from first)
    public void reset()throws UnsupportedAudioFileException, IOException, LineUnavailableException {
        try {
        clip.setMicrosecondPosition(0);
    }
    catch(Exception e){
        System.out.println();
    }
    }

```



```java
    //next method(used to play next song in the queue)
    public void next()throws UnsupportedAudioFileException, IOException, LineUnavailableException {
        try {
        current=q.remove();
        }
        catch(Exception e){
            System.out.println();
        }
    }


    //prev method(used to play previous song)
    public void prev() {
        current = dup;
    }


    //play_update method(used to play the selected song immediately, this song does not depend on the queue and
    // the queue doesn't get affected)
    public void play_update(song sam) {
        current=sam;
            }


```



```
    //run-check method(used to check whether the current song is playing or not and if the full
    //song is played then next song in the queue will be played)
    public void run_check() throws InterruptedException, UnsupportedAudioFileException, IOException, LineUnavailableException {
        try {


        while(true)
        {
            check=clip.isActive();
            Thread.sleep(5000);
            if(check==false) {
                dup = current;
                current=q.remove();
                Play(current);

             }
        }
        }
        catch(Exception e){
            System.out.println();
        }
    }



    public static void main(String[] args) {


    }
}
```

WORKING:

This class is fundamentally used to perform operations like play, stop the song, play the next or previous song etc. We have declared a queue data structure with ***song*** as its type and many local variables for operation purpose. This queue data structure is used to store songs that are supposed to be played in a queue in the player.

The methods defined in this for the operations for play, stop, play previous or play next song etc.

playlist_to_queue method() - This method adds all the songs in the playlist to the current queue.

queue_by_single_selection method() - This method add a particular song to the queue.

Play() - This method plays the song which is passed as the parameter.

stop() - It stops the song that is currently being played.

reset() - This resets the current playing song i.e. it starts playing the currently played song from the start.

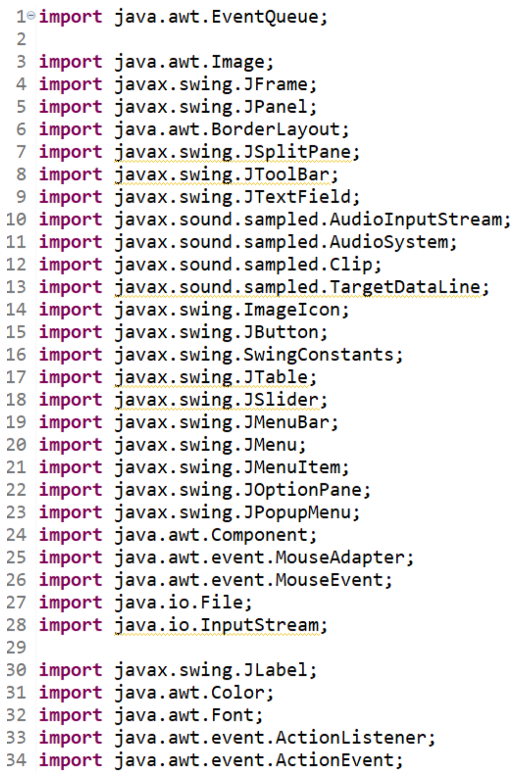
next() - This method when called removes the currently played song and automatically makes the *Play()* method to play the next song.

prev() - It sets the currently played song with the previous song so the *Play()* plays the previous song.

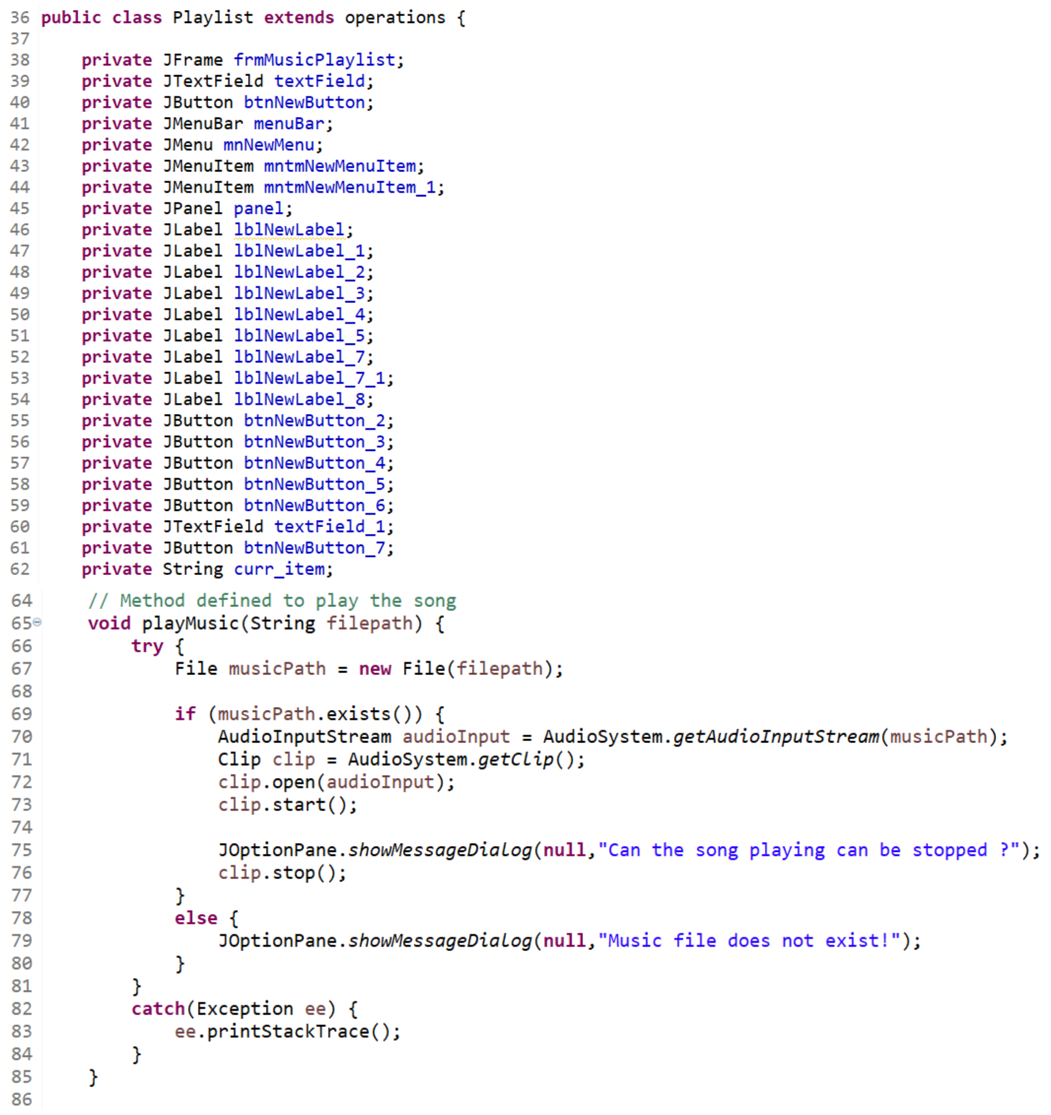
play_update() - It is used to play the selected song at that instance. This method doesn't depend on the queue and does not affect the existing queue in any way.

run_check() - It is used to check if a song is currently being played, this is set to check for every 5 seconds. If it is found that the song has ended it updates the *Play()* method to play the next song.

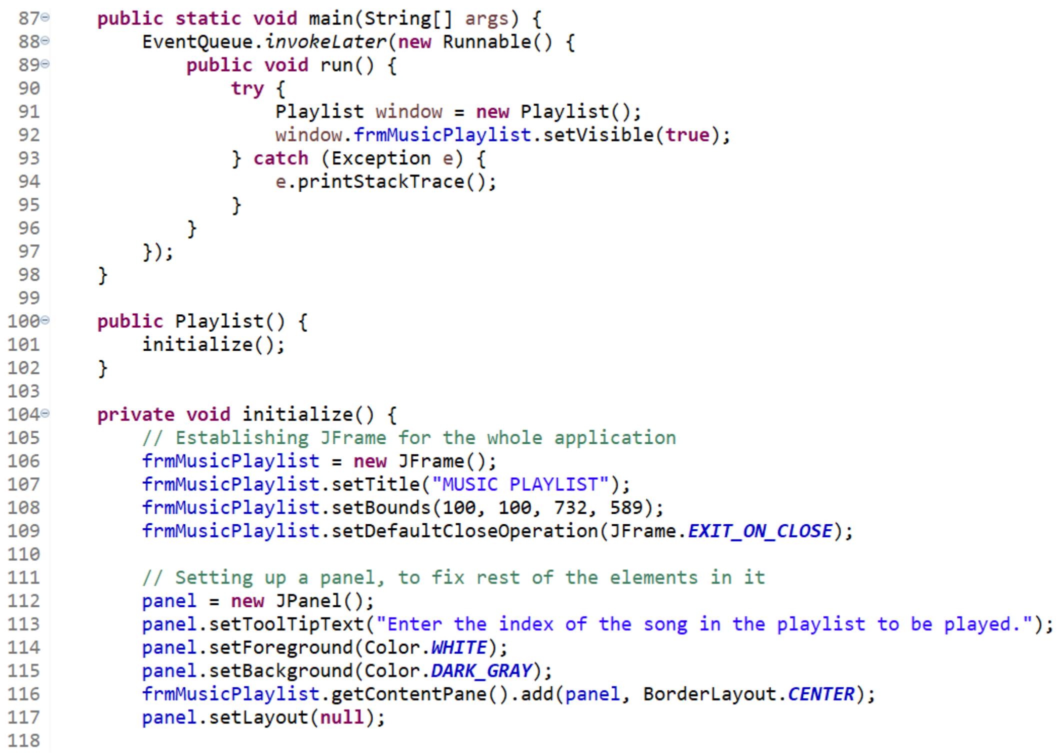
Playlist.java (GUI Class)



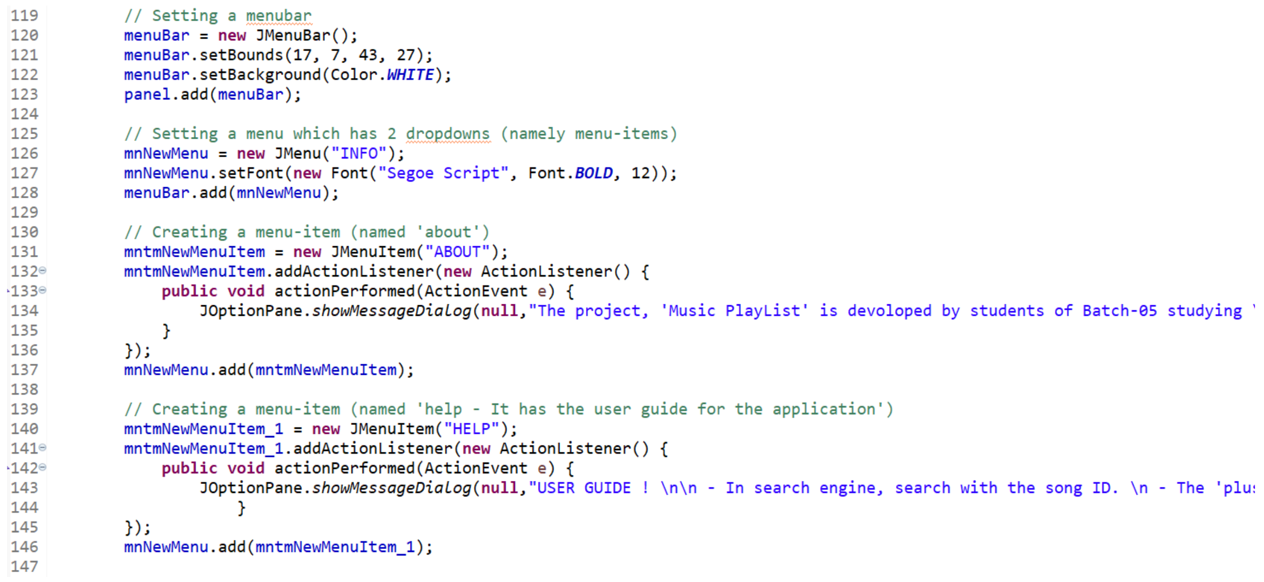
```java
import java.awt.EventQueue;

import java.awt.Image;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import javax.swing.JSplitPane;
import javax.swing.JToolBar;
import javax.swing.JTextField;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.TargetDataLine;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.SwingConstants;
import javax.swing.JTable;
import javax.swing.JSlider;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPopupMenu;
import java.awt.Component;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.InputStream;

import javax.swing.JLabel;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```



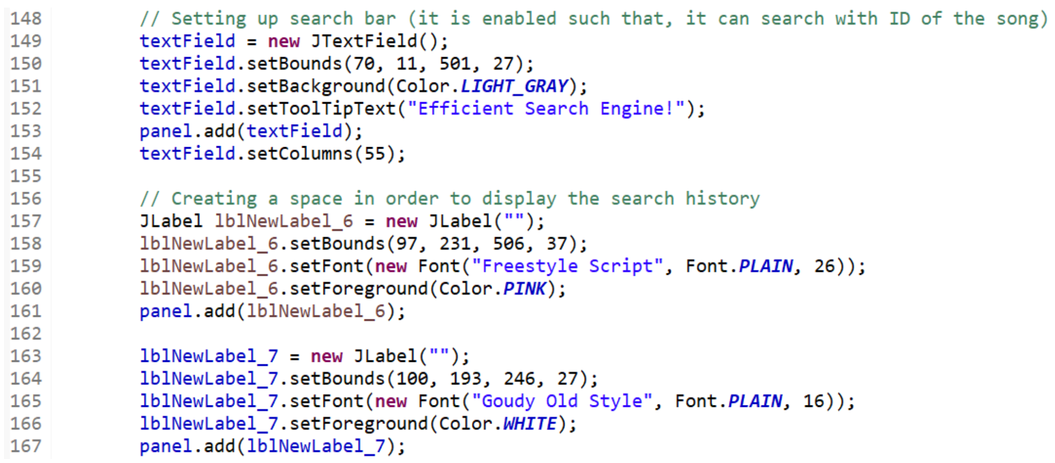
```java
public class Playlist extends operations {

    private JFrame frmMusicPlaylist;
    private JTextField textField;
    private JButton btnNewButton;
    private JMenuBar menuBar;
    private JMenu mnNewMenu;
    private JMenuItem mntmNewMenuItem;
    private JMenuItem mntmNewMenuItem_1;
    private JPanel panel;
    private JLabel lblNewLabel;
    private JLabel lblNewLabel_1;
    private JLabel lblNewLabel_2;
    private JLabel lblNewLabel_3;
    private JLabel lblNewLabel_4;
    private JLabel lblNewLabel_5;
    private JLabel lblNewLabel_7;
    private JLabel lblNewLabel_7_1;
    private JLabel lblNewLabel_8;
    private JButton btnNewButton_2;
    private JButton btnNewButton_3;
    private JButton btnNewButton_4;
    private JButton btnNewButton_5;
    private JButton btnNewButton_6;
    private JTextField textField_1;
    private JButton btnNewButton_7;
    private String curr_item;
     // Method defined to play the song
     void playMusic(String filepath) {
         try {
             File musicPath = new File(filepath);

             if (musicPath.exists()) {
                 AudioInputStream audioInput = AudioSystem.getAudioInputStream(musicPath);
                 Clip clip = AudioSystem.getClip();
                 clip.open(audioInput);
                 clip.start();

                 JOptionPane.showMessageDialog(null,"Can the song playing can be stopped ?");
                 clip.stop();
             }
             else {
                 JOptionPane.showMessageDialog(null,"Music file does not exist!");
             }
         }
         catch(Exception ee) {
             ee.printStackTrace();
         }
     }

```



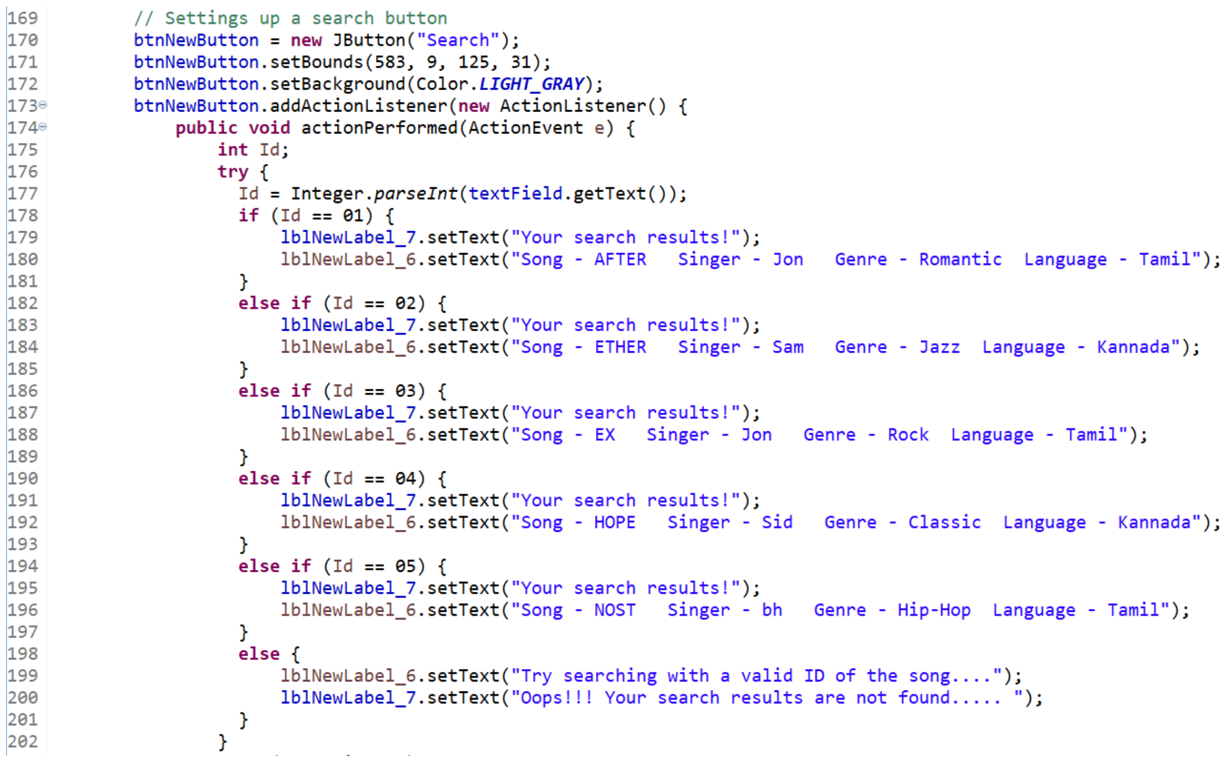
```java
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Playlist window = new Playlist();
                    window.frmMusicPlaylist.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public Playlist() {
        initialize();
    }

    private void initialize() {
        // Establishing JFrame for the whole application
        frmMusicPlaylist = new JFrame();
        frmMusicPlaylist.setTitle("MUSIC PLAYLIST");
        frmMusicPlaylist.setBounds(100, 100, 732, 589);
        frmMusicPlaylist.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Setting up a panel, to fix rest of the elements in it
        panel = new JPanel();
        panel.setToolTipText("Enter the index of the song in the playlist to be played.");
        panel.setForeground(Color.WHITE);
        panel.setBackground(Color.DARK_GRAY);
        frmMusicPlaylist.getContentPane().add(panel, BorderLayout.CENTER);
        panel.setLayout(null);

```



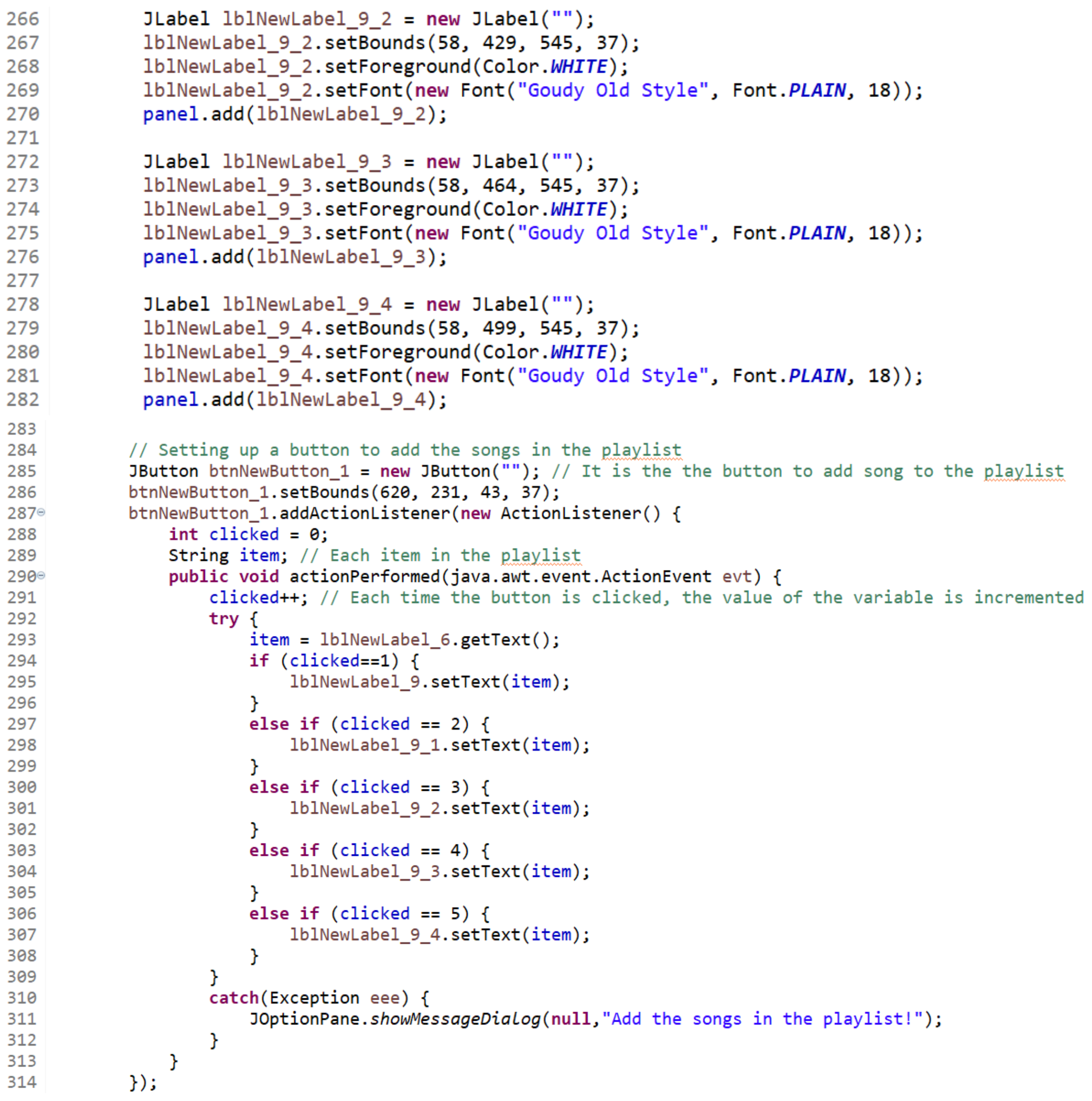
```java
        // Setting a menubar
        menuBar = new JMenuBar();
        menuBar.setBounds(17, 7, 43, 27);
        menuBar.setBackground(Color.WHITE);
        panel.add(menuBar);

        // Setting a menu which has 2 dropdowns (namely menu-items)
        mnNewMenu = new JMenu("INFO");
        mnNewMenu.setFont(new Font("Segoe Script", Font.BOLD, 12));
        menuBar.add(mnNewMenu);

        // Creating a menu-item (named 'about')
        mntmNewMenuItem = new JMenuItem("ABOUT");
        mntmNewMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null,"The project, 'Music PlayList' is devoloped by students of Batch-05 studying
            }
        });
        mnNewMenu.add(mntmNewMenuItem);

        // Creating a menu-item (named 'help - It has the user guide for the application')
        mntmNewMenuItem_1 = new JMenuItem("HELP");
        mntmNewMenuItem_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null,"USER GUIDE ! \n\n - In search engine, search with the song ID. \n - The 'plu
                }
        });
        mnNewMenu.add(mntmNewMenuItem_1);

```



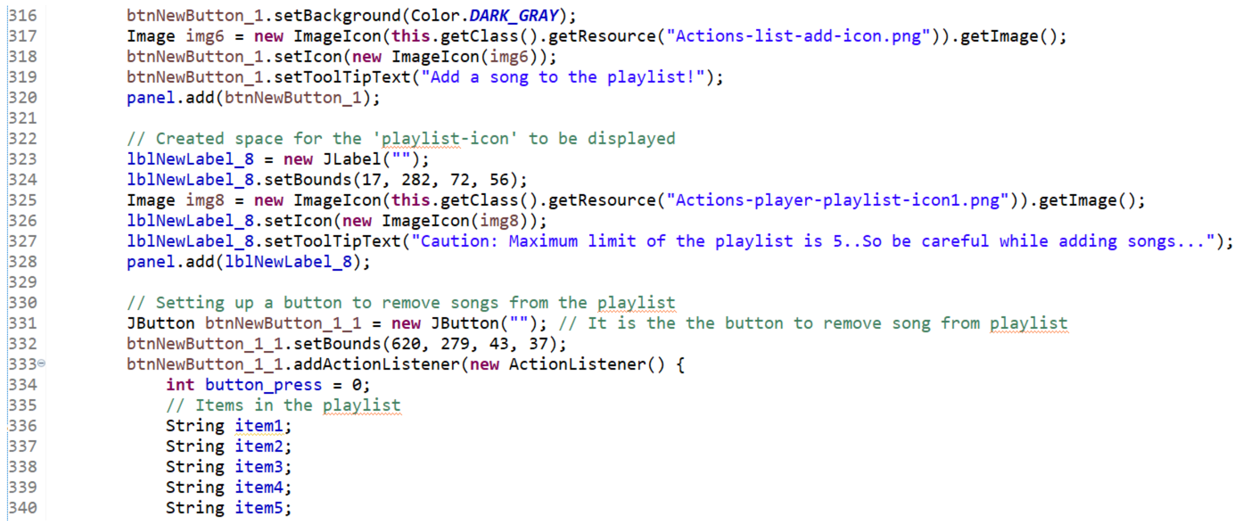
```java
        // Setting up search bar (it is enabled such that, it can search with ID of the song)
        textField = new JTextField();
        textField.setBounds(70, 11, 501, 27);
        textField.setBackground(Color.LIGHT_GRAY);
        textField.setToolTipText("Efficient Search Engine!");
        panel.add(textField);
        textField.setColumns(55);

        // Creating a space in order to display the search history
        JLabel lblNewLabel_6 = new JLabel("");
        lblNewLabel_6.setBounds(97, 231, 506, 37);
        lblNewLabel_6.setFont(new Font("Freestyle Script", Font.PLAIN, 26));
        lblNewLabel_6.setForeground(Color.PINK);
        panel.add(lblNewLabel_6);

        lblNewLabel_7 = new JLabel("");
        lblNewLabel_7.setBounds(100, 193, 246, 27);
        lblNewLabel_7.setFont(new Font("Goudy Old Style", Font.PLAIN, 16));
        lblNewLabel_7.setForeground(Color.WHITE);
        panel.add(lblNewLabel_7);
```



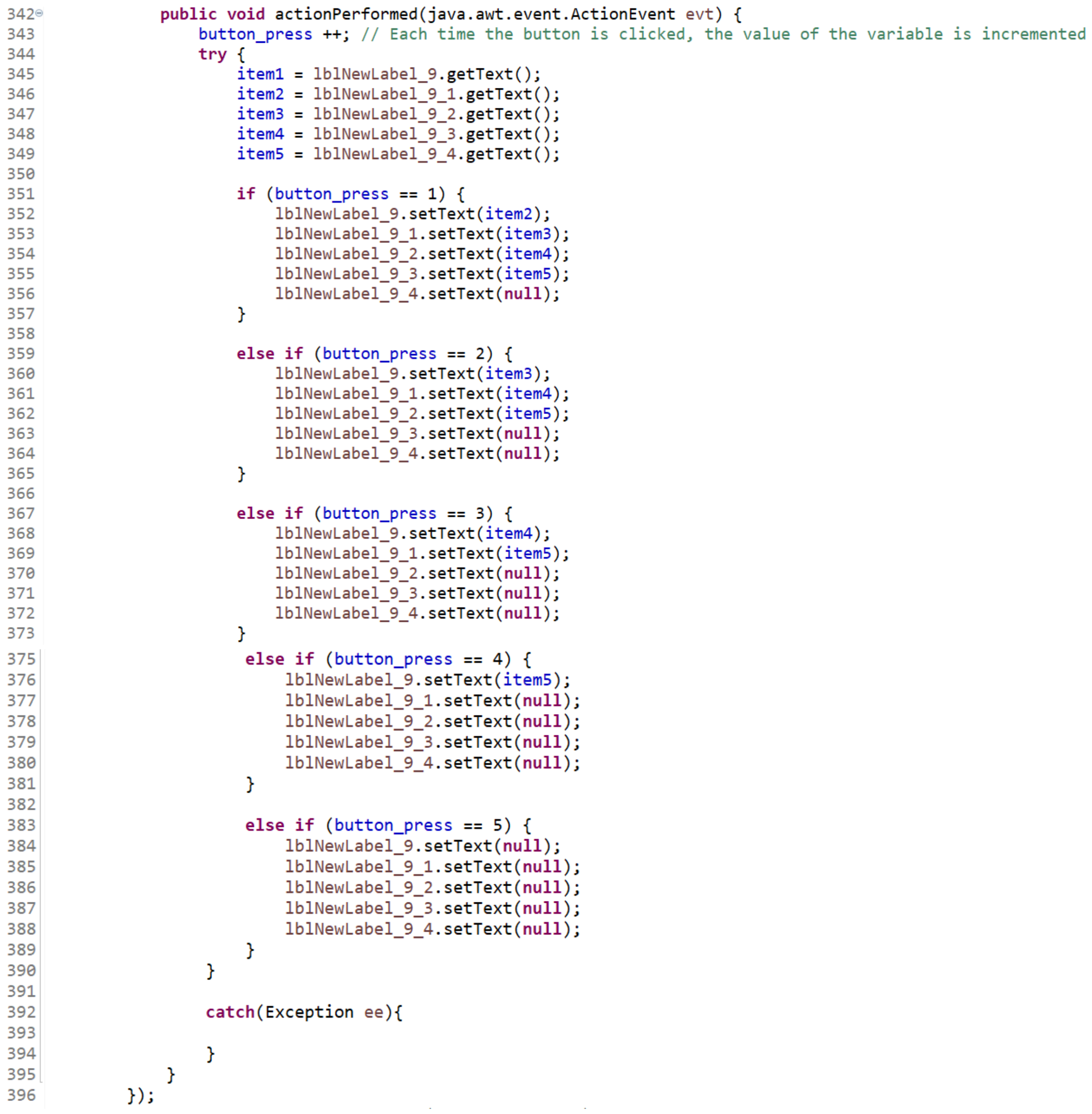
```java
        // Settings up a search button
        btnNewButton = new JButton("Search");
        btnNewButton.setBounds(583, 9, 125, 31);
        btnNewButton.setBackground(Color.LIGHT_GRAY);
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int Id;
                try {
                  Id = Integer.parseInt(textField.getText());
                  if (Id == 01) {
                      lblNewLabel_7.setText("Your search results!");
                      lblNewLabel_6.setText("Song - AFTER   Singer - Jon   Genre - Romantic  Language - Tamil");
                  }
                  else if (Id == 02) {
                      lblNewLabel_7.setText("Your search results!");
                      lblNewLabel_6.setText("Song - ETHER   Singer - Sam   Genre - Jazz  Language - Kannada");
                  }
                  else if (Id == 03) {
                      lblNewLabel_7.setText("Your search results!");
                      lblNewLabel_6.setText("Song - EX   Singer - Jon   Genre - Rock  Language - Tamil");
                  }
                  else if (Id == 04) {
                      lblNewLabel_7.setText("Your search results!");
                      lblNewLabel_6.setText("Song - HOPE   Singer - Sid   Genre - Classic  Language - Kannada");
                  }
                  else if (Id == 05) {
                      lblNewLabel_7.setText("Your search results!");
                      lblNewLabel_6.setText("Song - NOST   Singer - bh   Genre - Hip-Hop  Language - Tamil");
                  }
                  else {
                      lblNewLabel_6.setText("Try searching with a valid ID of the song....");
                      lblNewLabel_7.setText("Oops!!! Your search results are not found..... ");
                  }
                }
                catch(Exception ee){
                    JOptionPane.showMessageDialog(null,"Your search is unable to find! Please enter a valid song ID!");
                }
            }
        });
        btnNewButton.setFont(new Font("Segoe Print", Font.PLAIN, 12));
        Image img1 = new ImageIcon(this.getClass().getResource("zoom.png")).getImage();
        btnNewButton.setIcon(new ImageIcon(img1));
        panel.add(btnNewButton);

        // Setting up spaces for user-attractive icons - Also can be extended in future
        lblNewLabel_1 = new JLabel("");
        lblNewLabel_1.setBounds(97, 63, 96, 96);
        Image img_1 = new ImageIcon(this.getClass().getResource("music-icon.png")).getImage();
        lblNewLabel_1.setIcon(new ImageIcon(img_1));
        lblNewLabel_1.setToolTipText("MUSIC");
        panel.add(lblNewLabel_1);

        lblNewLabel_2 = new JLabel("");
        lblNewLabel_2.setBounds(195, 63, 96, 96);
        Image img_2 = new ImageIcon(this.getClass().getResource("finder-icon.png")).getImage();
        lblNewLabel_2.setIcon(new ImageIcon(img_2));
        lblNewLabel_2.setToolTipText("Finder");
        panel.add(lblNewLabel_2);

        lblNewLabel_3 = new JLabel("");
        lblNewLabel_3.setBounds(296, 63, 96, 96);
        Image img_3 = new ImageIcon(this.getClass().getResource("clock-icon.png")).getImage();
        lblNewLabel_3.setIcon(new ImageIcon(img_3));
        lblNewLabel_3.setToolTipText("Time");
        panel.add(lblNewLabel_3);

        lblNewLabel_4 = new JLabel("");
        lblNewLabel_4.setBounds(397, 63, 96, 96);
        Image img_4 = new ImageIcon(this.getClass().getResource("Apps-system-software-update-icon.png")).getImage();
        lblNewLabel_4.setIcon(new ImageIcon(img_4));
        lblNewLabel_4.setToolTipText("Settings");
        panel.add(lblNewLabel_4);

        lblNewLabel_5 = new JLabel("");
        lblNewLabel_5.setBounds(507, 63, 96, 96);
        Image img_5 = new ImageIcon(this.getClass().getResource("global-icon.png")).getImage();
        lblNewLabel_5.setIcon(new ImageIcon(img_5));
        lblNewLabel_5.setToolTipText("Global Users of the App");
        panel.add(lblNewLabel_5);

        lblNewLabel_7_1 = new JLabel("");
        lblNewLabel_7_1.setBounds(100, 193, 246, 27);
        panel.add(lblNewLabel_7_1);

        // Creating space for the playlist - in order to add the songs
        JLabel lblNewLabel_9 = new JLabel("");
        lblNewLabel_9.setBounds(58, 349, 545, 37);
        lblNewLabel_9.setForeground(Color.WHITE);
        lblNewLabel_9.setFont(new Font("Goudy Old Style", Font.PLAIN, 18));
        panel.add(lblNewLabel_9);

        JLabel lblNewLabel_9_1 = new JLabel("");
        lblNewLabel_9_1.setBounds(58, 394, 545, 37);
        lblNewLabel_9_1.setFont(new Font("Goudy Old Style", Font.PLAIN, 18));
        lblNewLabel_9_1.setForeground(Color.WHITE);
        panel.add(lblNewLabel_9_1);

```



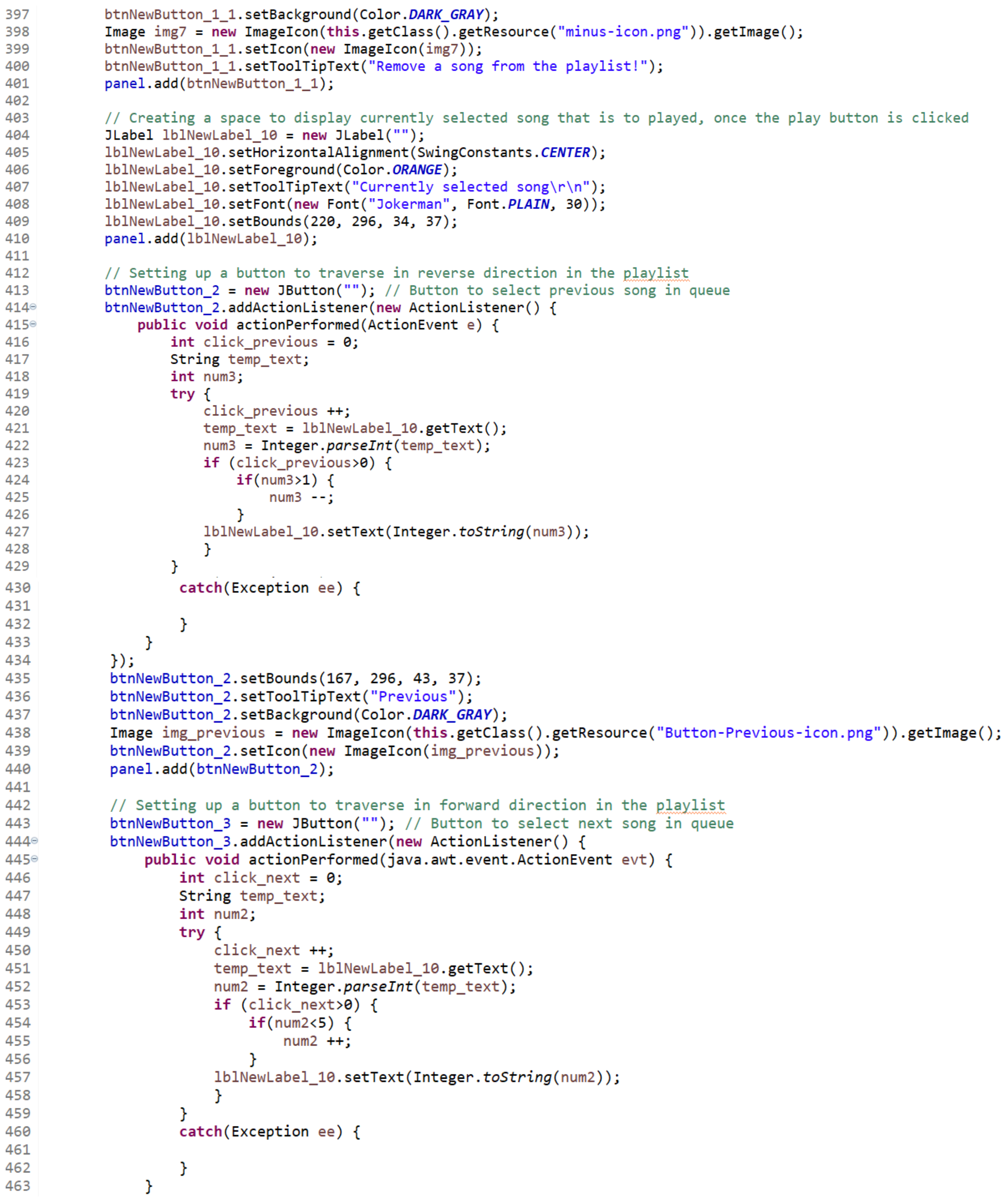
```java
        JLabel lblNewLabel_9_2 = new JLabel("");
        lblNewLabel_9_2.setBounds(58, 429, 545, 37);
        lblNewLabel_9_2.setForeground(Color.WHITE);
        lblNewLabel_9_2.setFont(new Font("Goudy Old Style", Font.PLAIN, 18));
        panel.add(lblNewLabel_9_2);

        JLabel lblNewLabel_9_3 = new JLabel("");
        lblNewLabel_9_3.setBounds(58, 464, 545, 37);
        lblNewLabel_9_3.setForeground(Color.WHITE);
        lblNewLabel_9_3.setFont(new Font("Goudy Old Style", Font.PLAIN, 18));
        panel.add(lblNewLabel_9_3);

        JLabel lblNewLabel_9_4 = new JLabel("");
        lblNewLabel_9_4.setBounds(58, 499, 545, 37);
        lblNewLabel_9_4.setForeground(Color.WHITE);
        lblNewLabel_9_4.setFont(new Font("Goudy Old Style", Font.PLAIN, 18));
        panel.add(lblNewLabel_9_4);

        // Setting up a button to add the songs in the playlist
        JButton btnNewButton_1 = new JButton(""); // It is the the button to add song to the playlist
        btnNewButton_1.setBounds(620, 231, 43, 37);
        btnNewButton_1.addActionListener(new ActionListener() {
            int clicked = 0;
            String item; // Each item in the playlist
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clicked++; // Each time the button is clicked, the value of the variable is incremented
                try {
                    item = lblNewLabel_6.getText();
                    if (clicked==1) {
                        lblNewLabel_9.setText(item);
                    }
                    else if (clicked == 2) {
                        lblNewLabel_9_1.setText(item);
                    }
                    else if (clicked == 3) {
                        lblNewLabel_9_2.setText(item);
                    }
                    else if (clicked == 4) {
                        lblNewLabel_9_3.setText(item);
                    }
                    else if (clicked == 5) {
                        lblNewLabel_9_4.setText(item);
                    }
                }
                catch(Exception eee) {
                    JOptionPane.showMessageDialog(null,"Add the songs in the playlist!");
                }
            }
        });
```



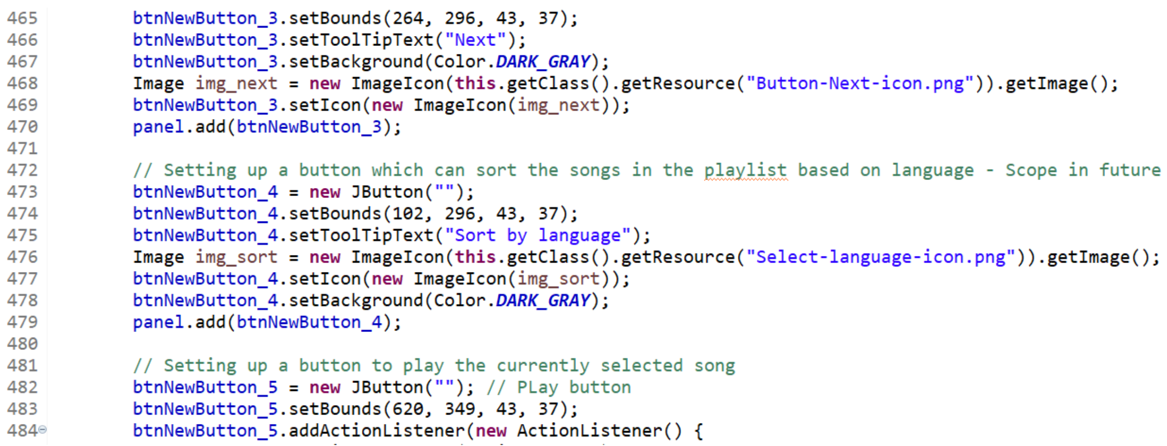
```java
        btnNewButton_1.setBackground(Color.DARK_GRAY);
        Image img6 = new ImageIcon(this.getClass().getResource("Actions-list-add-icon.png")).getImage();
        btnNewButton_1.setIcon(new ImageIcon(img6));
        btnNewButton_1.setToolTipText("Add a song to the playlist!");
        panel.add(btnNewButton_1);

        // Created space for the 'playlist-icon' to be displayed
        lblNewLabel_8 = new JLabel("");
        lblNewLabel_8.setBounds(17, 282, 72, 56);
        Image img8 = new ImageIcon(this.getClass().getResource("Actions-player-playlist-icon1.png")).getImage();
        lblNewLabel_8.setIcon(new ImageIcon(img8));
        lblNewLabel_8.setToolTipText("Caution: Maximum limit of the playlist is 5..So be careful while adding songs...");
        panel.add(lblNewLabel_8);

        // Setting up a button to remove songs from the playlist
        JButton btnNewButton_1_1 = new JButton(""); // It is the the button to remove song from playlist
        btnNewButton_1_1.setBounds(620, 279, 43, 37);
        btnNewButton_1_1.addActionListener(new ActionListener() {
            int button_press = 0;
            // Items in the playlist
            String item1;
            String item2;
            String item3;
            String item4;
            String item5;
```



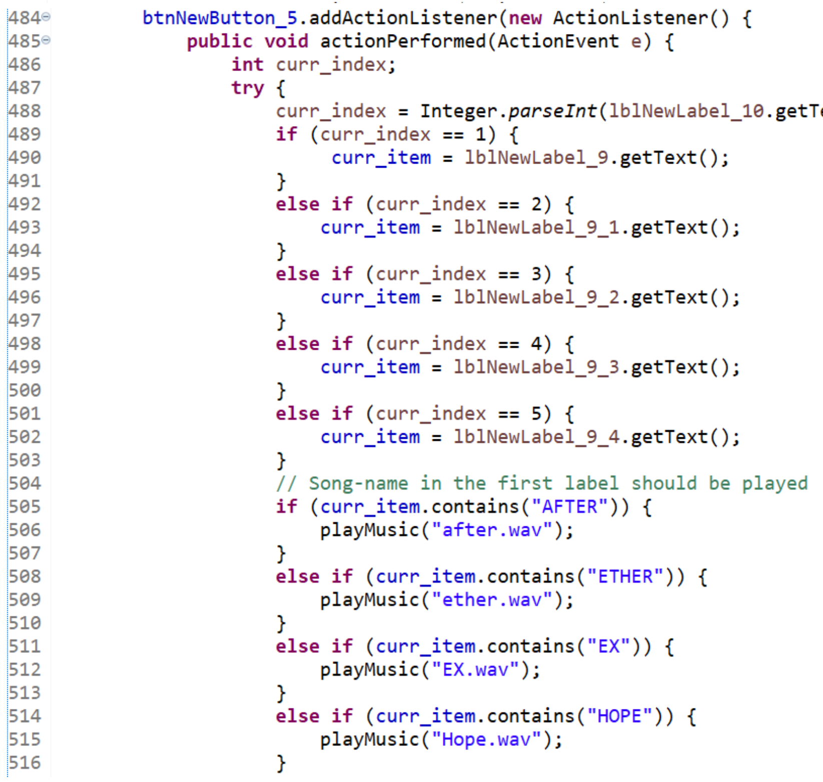
```java
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            button_press ++; // Each time the button is clicked, the value of the variable is incremented
            try {
                item1 = lblNewLabel_9.getText();
                item2 = lblNewLabel_9_1.getText();
                item3 = lblNewLabel_9_2.getText();
                item4 = lblNewLabel_9_3.getText();
                item5 = lblNewLabel_9_4.getText();

                if (button_press == 1) {
                    lblNewLabel_9.setText(item2);
                    lblNewLabel_9_1.setText(item3);
                    lblNewLabel_9_2.setText(item4);
                    lblNewLabel_9_3.setText(item5);
                    lblNewLabel_9_4.setText(null);
                }

                else if (button_press == 2) {
                    lblNewLabel_9.setText(item3);
                    lblNewLabel_9_1.setText(item4);
                    lblNewLabel_9_2.setText(item5);
                    lblNewLabel_9_3.setText(null);
                    lblNewLabel_9_4.setText(null);
                }

                else if (button_press == 3) {
                    lblNewLabel_9.setText(item4);
                    lblNewLabel_9_1.setText(item5);
                    lblNewLabel_9_2.setText(null);
                    lblNewLabel_9_3.setText(null);
                    lblNewLabel_9_4.setText(null);
                }
                else if (button_press == 4) {
                    lblNewLabel_9.setText(item5);
                    lblNewLabel_9_1.setText(null);
                    lblNewLabel_9_2.setText(null);
                    lblNewLabel_9_3.setText(null);
                    lblNewLabel_9_4.setText(null);
                }

                else if (button_press == 5) {
                    lblNewLabel_9.setText(null);
                    lblNewLabel_9_1.setText(null);
                    lblNewLabel_9_2.setText(null);
                    lblNewLabel_9_3.setText(null);
                    lblNewLabel_9_4.setText(null);
                }
            }

            catch(Exception ee){

            }
        }
    });
```



```java
        btnNewButton_1_1.setBackground(Color.DARK_GRAY);
        Image img7 = new ImageIcon(this.getClass().getResource("minus-icon.png")).getImage();
        btnNewButton_1_1.setIcon(new ImageIcon(img7));
        btnNewButton_1_1.setToolTipText("Remove a song from the playlist!");
        panel.add(btnNewButton_1_1);

        // Creating a space to display currently selected song that is to played, once the play button is clicked
        JLabel lblNewLabel_10 = new JLabel("");
        lblNewLabel_10.setHorizontalAlignment(SwingConstants.CENTER);
        lblNewLabel_10.setForeground(Color.ORANGE);
        lblNewLabel_10.setToolTipText("Currently selected song\r\n");
        lblNewLabel_10.setFont(new Font("Jokerman", Font.PLAIN, 30));
        lblNewLabel_10.setBounds(220, 296, 34, 37);
        panel.add(lblNewLabel_10);

        // Setting up a button to traverse in reverse direction in the playlist
        btnNewButton_2 = new JButton(""); // Button to select previous song in queue
        btnNewButton_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int click_previous = 0;
                String temp_text;
                int num3;
                try {
                    click_previous ++;
                    temp_text = lblNewLabel_10.getText();
                    num3 = Integer.parseInt(temp_text);
                    if (click_previous>0) {
                        if(num3>1) {
                            num3 --;
                        }
                    lblNewLabel_10.setText(Integer.toString(num3));
                    }
                }
                catch(Exception ee) {

                }
            }
        });
        btnNewButton_2.setBounds(167, 296, 43, 37);
        btnNewButton_2.setToolTipText("Previous");
        btnNewButton_2.setBackground(Color.DARK_GRAY);
        Image img_previous = new ImageIcon(this.getClass().getResource("Button-Previous-icon.png")).getImage();
        btnNewButton_2.setIcon(new ImageIcon(img_previous));
        panel.add(btnNewButton_2);

        // Setting up a button to traverse in forward direction in the playlist
        btnNewButton_3 = new JButton(""); // Button to select next song in queue
        btnNewButton_3.addActionListener(new ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                int click_next = 0;
                String temp_text;
                int num2;
                try {
                    click_next ++;
                    temp_text = lblNewLabel_10.getText();
                    num2 = Integer.parseInt(temp_text);
                    if (click_next>0) {
                        if(num2<5) {
                            num2 ++;
                        }
                    lblNewLabel_10.setText(Integer.toString(num2));
                    }
                }
                catch(Exception ee) {

                }
            }
```

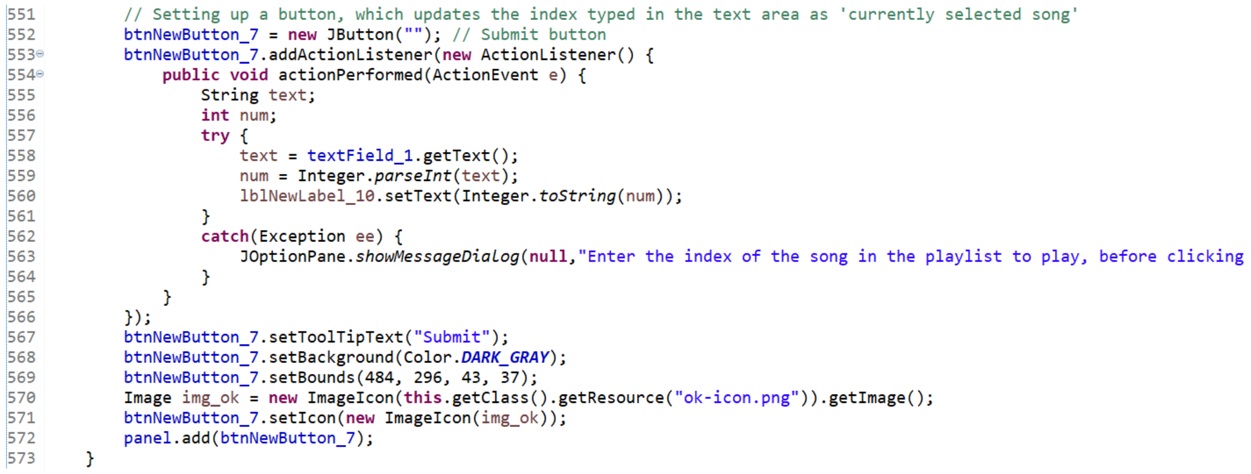


```java
        btnNewButton_3.setBounds(264, 296, 43, 37);
        btnNewButton_3.setToolTipText("Next");
        btnNewButton_3.setBackground(Color.DARK_GRAY);
        Image img_next = new ImageIcon(this.getClass().getResource("Button-Next-icon.png")).getImage();
        btnNewButton_3.setIcon(new ImageIcon(img_next));
        panel.add(btnNewButton_3);

        // Setting up a button which can sort the songs in the playlist based on language - Scope in future
        btnNewButton_4 = new JButton("");
        btnNewButton_4.setBounds(102, 296, 43, 37);
        btnNewButton_4.setToolTipText("Sort by language");
        Image img_sort = new ImageIcon(this.getClass().getResource("Select-language-icon.png")).getImage();
        btnNewButton_4.setIcon(new ImageIcon(img_sort));
        btnNewButton_4.setBackground(Color.DARK_GRAY);
        panel.add(btnNewButton_4);

        // Setting up a button to play the currently selected song
        btnNewButton_5 = new JButton(""); // PLay button
        btnNewButton_5.setBounds(620, 349, 43, 37);
        btnNewButton_5.addActionListener(new ActionListener() {
```

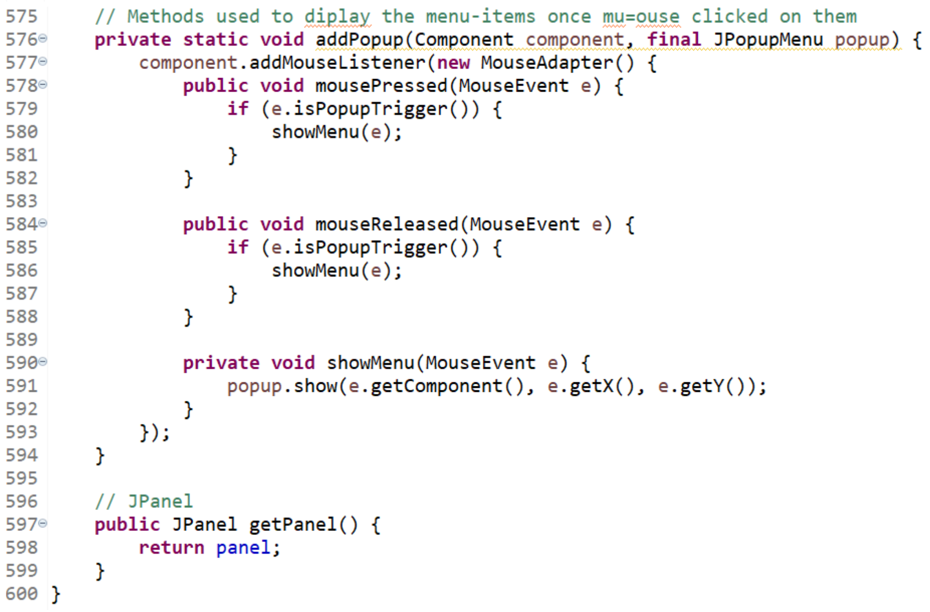


```java
btnNewButton_5.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int curr_index;
        try {
            curr_index = Integer.parseInt(lblNewLabel_10.getText());
            if (curr_index == 1) {
                 curr_item = lblNewLabel_9.getText();
            }
            else if (curr_index == 2) {
                curr_item = lblNewLabel_9_1.getText();
            }
            else if (curr_index == 3) {
                curr_item = lblNewLabel_9_2.getText();
            }
            else if (curr_index == 4) {
                curr_item = lblNewLabel_9_3.getText();
            }
            else if (curr_index == 5) {
                curr_item = lblNewLabel_9_4.getText();
            }
            // Song-name in the first label should be played
            if (curr_item.contains("AFTER")) {
                playMusic("after.wav");
            }
            else if (curr_item.contains("ETHER")) {
                playMusic("ether.wav");
            }
            else if (curr_item.contains("EX")) {
                playMusic("EX.wav");
            }
            else if (curr_item.contains("HOPE")) {
                playMusic("Hope.wav");
            }
```

```java
                    else if (curr_item.contains("NOST")) {
                        playMusic("nost.wav");
                    }
                }
                catch(Exception eee) {

                }
            }
        });
        btnNewButton_5.setToolTipText("Play");
        btnNewButton_5.setBackground(Color.DARK_GRAY);
        Image img_play = new ImageIcon(this.getClass().getResource("Button-Play-icon.png")).getImage();
        btnNewButton_5.setIcon(new ImageIcon(img_play));
        panel.add(btnNewButton_5);

        // Setting up a button which can stop the currently playing song
        btnNewButton_6 = new JButton(""); // Stop button
        btnNewButton_6.setBounds(665, 349, 43, 37);
        btnNewButton_6.setToolTipText("Stop");
        Image img_stop = new ImageIcon(this.getClass().getResource("Button-Stop-icon.png")).getImage();
        btnNewButton_6.setIcon(new ImageIcon(img_stop));
        btnNewButton_6.setBackground(Color.DARK_GRAY);
        panel.add(btnNewButton_6);

        // A text-area used to type the songs which we need to play
        textField_1 = new JTextField();
        textField_1.setHorizontalAlignment(SwingConstants.CENTER);
        textField_1.setBounds(438, 296, 36, 37);
        textField_1.setForeground(Color.WHITE);
        textField_1.setBackground(Color.GRAY);
        textField_1.setFont(new Font("Segoe UI Emoji", Font.PLAIN, 26));
        panel.add(textField_1);
        textField_1.setColumns(10);
```

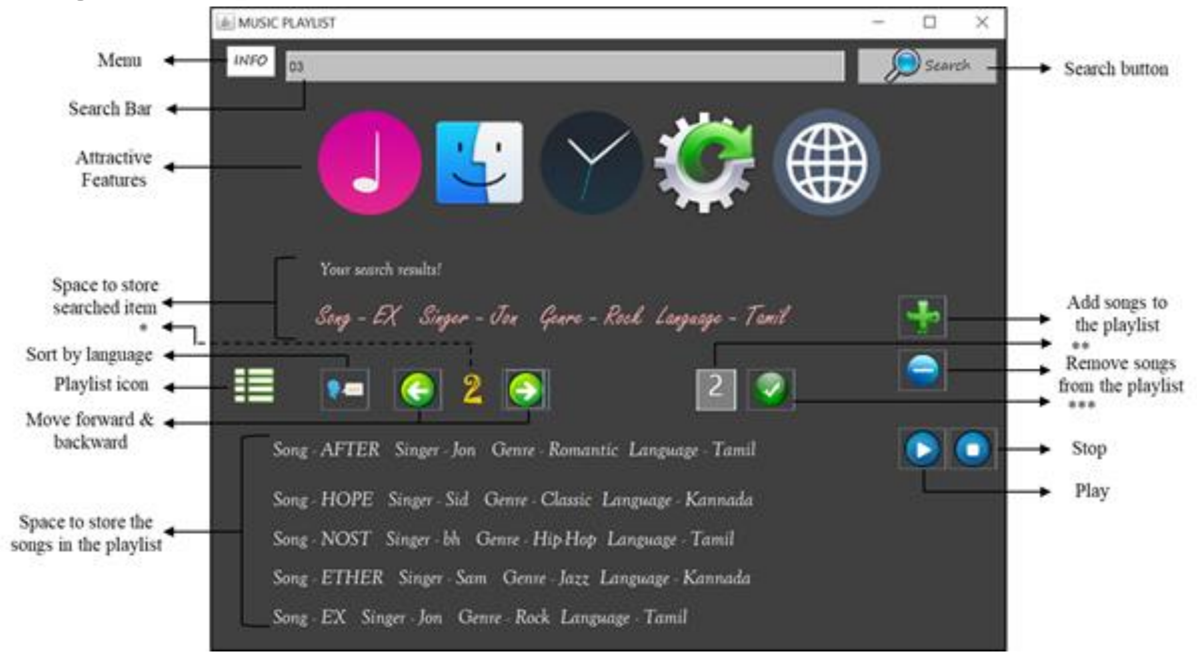


```java
        // Setting up a button, which updates the index typed in the text area as 'currently selected song'
        btnNewButton_7 = new JButton(""); // Submit button
        btnNewButton_7.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String text;
                int num;
                try {
                    text = textField_1.getText();
                    num = Integer.parseInt(text);
                    lblNewLabel_10.setText(Integer.toString(num));
                }
                catch(Exception ee) {
                    JOptionPane.showMessageDialog(null,"Enter the index of the song in the playlist to play, before clicking
                }
            }
        });
        btnNewButton_7.setToolTipText("Submit");
        btnNewButton_7.setBackground(Color.DARK_GRAY);
        btnNewButton_7.setBounds(484, 296, 43, 37);
        Image img_ok = new ImageIcon(this.getClass().getResource("ok-icon.png")).getImage();
        btnNewButton_7.setIcon(new ImageIcon(img_ok));
        panel.add(btnNewButton_7);
    }
```



```java
    // Methods used to diplay the menu-items once mu=ouse clicked on them
    private static void addPopup(Component component, final JPopupMenu popup) {
        component.addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                if (e.isPopupTrigger()) {
                    showMenu(e);
                }
            }

            public void mouseReleased(MouseEvent e) {
                if (e.isPopupTrigger()) {
                    showMenu(e);
                }
            }

            private void showMenu(MouseEvent e) {
                popup.show(e.getComponent(), e.getX(), e.getY());
            }
        });
    }

    // JPanel
    public JPanel getPanel() {
        return panel;
    }
}
```

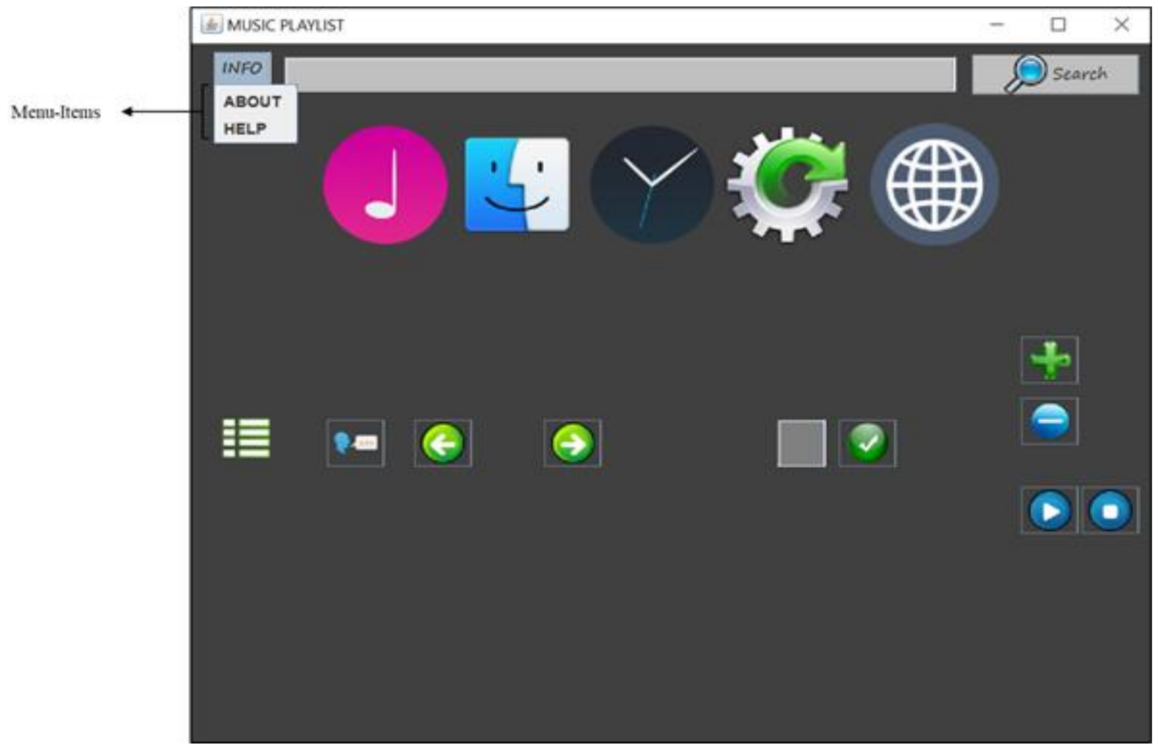
Design of Graphical User Interface (GUI)


MUSIC PLAYLIST
Menu
INFO
Search Bar
Search
Search button
Attractive Features
Your search results!
Space to store searched item
Song - EX Singer - Jon Genre - Rock Language - Tamil
*
Sort by language
Playlist icon
Move forward & backward
Add songs to the playlist
**
Remove songs from the playlist
***
Stop
Play
Space to store the songs in the playlist
Song - AFTER Singer - Jon Genre - Romantic Language - Tamil
Song - HOPE Singer - Sid Genre - Classic Language - Kannada
Song - NOST Singer - bh Genre - Hip Hop Language - Tamil
Song - ETHER Singer - Sam Genre - Jazz Language - Kannada
Song - EX Singer - Jon Genre - Rock Language - Tamil

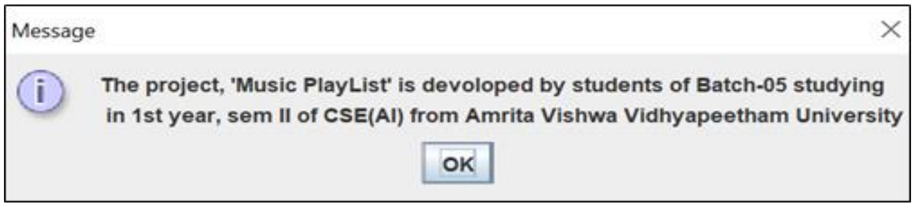
*Sort with respect to language

**Text-field to type the song index (with respect to playlist)

***Submit button (Updates the typed index as currently selected song)

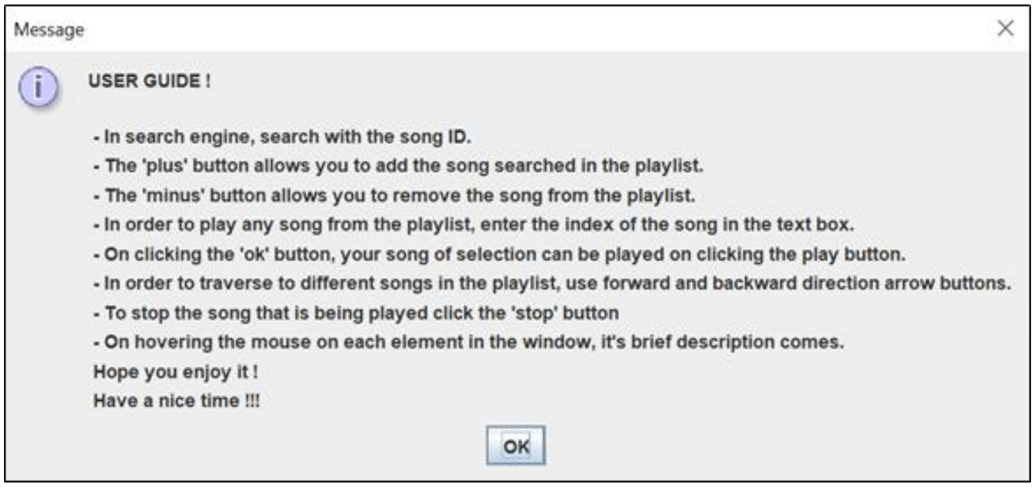


<u>Menu-Item 'ABOUT'</u>



The project, 'Music PlayList' is devoloped by students of Batch-05 studying in 1st year, sem II of CSE(AI) from Amrita Vishwa Vidhyapeetham University

<u>Menu-Item 'HELP'</u>



USER GUIDE !

- In search engine, search with the song ID.
- The 'plus' button allows you to add the song searched in the playlist.
- The 'minus' button allows you to remove the song from the playlist.
- In order to play any song from the playlist, enter the index of the song in the text box.
- On clicking the 'ok' button, your song of selection can be played on clicking the play button.
- In order to traverse to different songs in the playlist, use forward and backward direction arrow buttons.
- To stop the song that is being played click the 'stop' button
- On hovering the mouse on each element in the window, it's brief description comes.

Hope you enjoy it !
Have a nice time !!!

Click here to view the User-Guide Video of the application

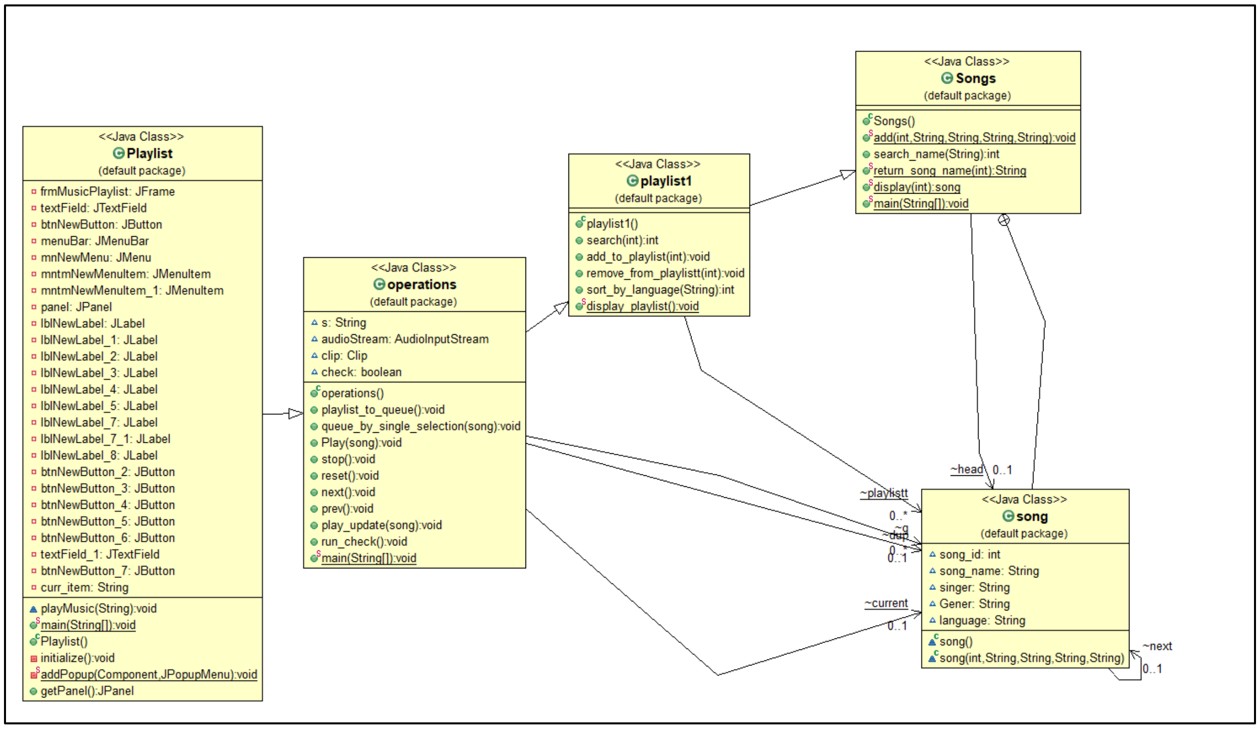
<u>Working of the Interface</u>

The search bar is designed in such a way that a song can be searched with its ID. Then the song that is searched is displayed along with the name of the singer, genre, and language of the song. The song searched, could be added to the playlist if wanted. The songs in the playlist could be removed from the playlist as per the wish of the user. As it is a playlist created using Queue Data Structure, on removing the song from the playlist, the first added gets removed.

There is also a user-efficient way created. The user can enter the index of the song in the playlist in the textField. On clicking the 'tick' button, the song in that index, gets updated as the current song and would be played on clicking 'play' button. All the songs in the playlist can be accessed by the forward and backward traversing buttons. On clicking the 'stop' button, the currently playing song gets stopped.

There is a user-guide provided in the 'HELP' section in the menu 'INFO'.

UML DIAGRAM



CHALLENGES FACED

There are a few challenges faced by our Music Playlist Application.

- Graphically, it finds it tedious to sort the songs based on language, genre, singer’s name and many other such types.
- The size of the playlist is small when implemented in a graphical interface.
- The search bar finds it difficult to search the song with its name and it is not quite powerful.
- There isn’t any voice search of the song
- The songs in the playlist cannot be saved

CONCLUSION

There are a lot of areas in this music playlist, where advancements could be made in future. It is possible to expand this project to another level and make it a universally accepted model.

Future scopes

- The data of the songs could be made available in online databases. Then the songs can be accessed from there and could be played without storing it in our device. A lot of storage space could be saved as a result of it.
- Various sorting algorithms could be used to sort the songs list based on different themes like song-name, singer-name, language, genre.
- With respect to the graphical interface, the size of the playlist could be kept as unfixed. This would enhance the quality of the playlist.
- Searching the songs with its name too could be enabled.
- A voice search too could be enabled.

As a result, in the future, there is scope to overcome all the challenges faced and there is a chance to improve this model into a more advanced one.

REFERENCE

Linked List Data Structure - GeeksforGeeks
Data Structure and Algorithms - Linked List - Tutorialspoint
Queue - Linked List Implementation - GeeksforGeeks
Linked List Implementation of Queue - javatpoint
Data Structures in The Real World — Linked List | by Christopher Webb |
Music Player with Doubly Linked List in C: algorithms (reddit.com)