MUSIC PLAYER USING DOUBLY LINKED LIST

A PROJECT REPORT

Submitted by

BL.EN.U4AIE21121	Srinidhi Kannan
BL.EN.U4AIE21126	Suryamritha M
BL.EN.U4AIE21139	Varshini Balaji

in fulfillment of Project Phase -2

BACHELOR OF TECHNOLOGY

IN

Computer Science with Artificial Intelligence E

AMRITA SCHOOL OF ENGINEERING, BANGALORE
AMRITA VISHWA VIDYAPEETHAM
BANGALORE 560 035
July-2022

ABSTRACT

Music makes everyone happy and having all your favorites in one makes us even happier. A music player is an application wherein digital audio files are played. In this project we have implemented a Music Player with Linked List with various functions like insert front, insert back, delete front, delete back, pause, play, stop, and many more. This project this done in Java language and Eclipse programming tools leading to the design and coding of music players. This project uses various libraries such as

- java.util.Scanner
- java.io.File
- javax.sound.sampled.AudioInputStream
- javax.sound.sampled.AudioSystem
- javax.sound.sampled.Clip
- javax. swing. JOptionPane.

INDEX

CHAPTER 1

INTRODUCTION

1.1 Music Player

Music Player plays digital audio files. This is a very organized way to listen to your favorite songs. In this project, we have used a Doubly Linked list, which is a data Structure in Java for the implementation of this project. This project involves various functions which include inserting front, inserting back, deleting front, deleting back, playing/pausing, displaying the songs in the Playlist, and stopping the song. It also includes various Audio libraries in java for playing the song needed. The libraries included are

- java.util.Scanner
- java.io.File
- javax.sound.sampled.AudioInputStream
- javax.sound.sampled.AudioSystem
- javax.sound.sampled.Clip
- javax. swing. JOptionPane.

We have also created various classes wherein the audio files and be played. One of them is the scanner class for obtaining the input primitive types like int, double, string, etc. The others are user-defined classes like music, Node dl, musicStuff, and many more for the implementation of the doubly linked list. There are various objects created too. in this Music Player by entering the file path one can easily access the required song.

CHAPTER 2
LITERATURE SURVEY

Pan Yong-Cai, Liu Wen-chao, & Li Xiao. (2010). *Development and Research of Music Player Application Based on Android. 2010 International Conference on Communications and Intelligence Information Security.* doi:10.1109/icciis.2010.28

- Music player is a kind of common mobile terminal application. This paper takes a music player as an example to illustrate how to develop Android applications.
- This player has a friendly interface, convenient operation, good expansibility and maintainability. After testing, the player can satisfy the basic demand of users and run stable.

Qing, Z., Ying, L., Yuan, P. G., & Sheng, L. Z. (2015). *Music Player Based on the Cordova Cross-Platform. 2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence.* doi:10.1109/acit-csi.2015.85

- Cordova is one of the popular cross-platform mobile application development frameworks currently, with open source, free and at the same time, it supports multiple mobile operating system, etc.
- This paper implements a simple music player quickly based on the Cordova framework using HTML + CSS + JavaScript, and developed my own Cordova plug-in, reduced secondary development, and adaptation for different platforms, and the instance provides a reference for the development of cross-platform mobile applications.

CHAPTER 3

FUNCTIONS USED

- **Insert front**

  Inserts songs at the front

- **Insert back**

  Insert songs at the end

- **Delete front**

  Deletes songs at the front

- **Delete back**

  Deletes songs at the back

- **Playmusic**

  Plays the specified song in a loop and pauses the played song and continues back when asked

- **Display**

  Show all the available songs on the Playlist

CHAPTER 4
LIBRARIES USED

- **java.util.Scanner**
  The Scanner class is used to get user input, and it is found in java. util package

- **java.io.File**
  The Java.io.File class is an abstract representation of file and directory pathnames

- **javax.sound.sampled.AudioInputStream**
  An audio input stream is an input stream with a specified audio format and length. The length is expressed in sample frames, not bytes.

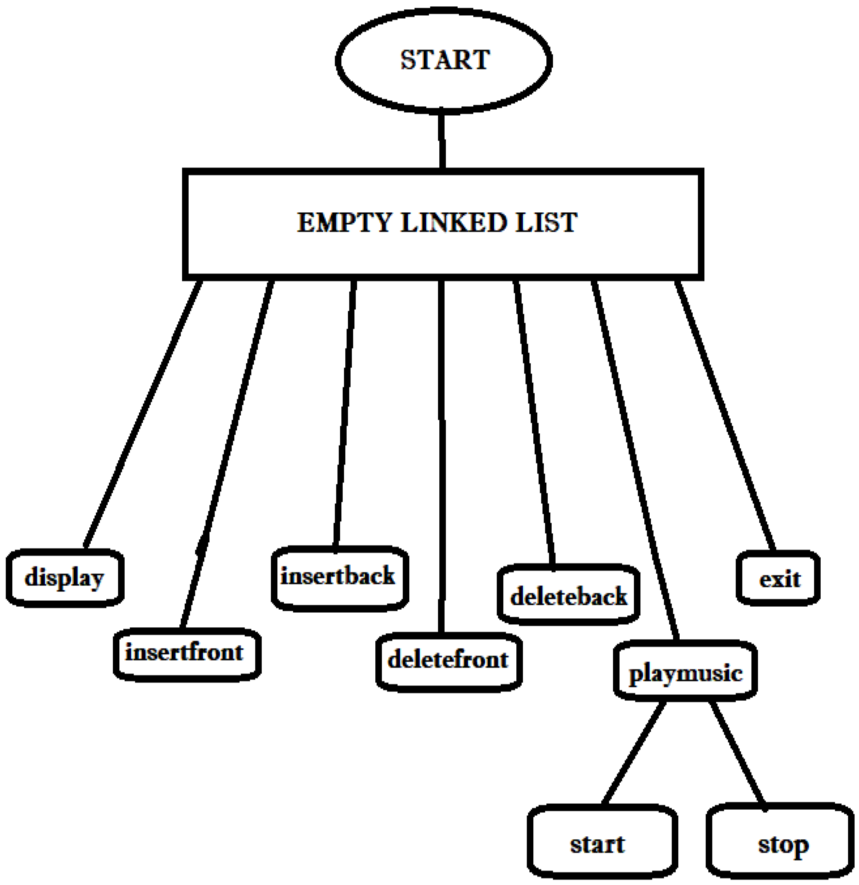
- **javax.sound.sampled.AudioSystem**
  The AudioSystem class acts as the entry point to the sampled-audio system resources. This class lets you query and access the mixers that are installed on the system.

- **javax.sound.sampled.Clip**
  The Clip interface represents a special kind of data line whose audio data can be loaded before playback, instead of being streamed in real-time.

- **javax. swing. JOptionPane.**
  JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something.

CHAPTER 5

FLOW DIAGRAM



CHAPTER 6

CODE

The project consists of two files:

music.java

```
package Playingmusic;
import java.util.Scanner;




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

		String filepath3="Chellama.wav";
		String filepath4="Dandelions.wav";
		String filepath5="Pasoori.wav";
		String filepath6="katy_perry_roar.wav";
		String filepath7="Halmithi.wav";
		String filepath8="Uyire.wav";
		String filepath9="Adada.wav";
		String filepath10="Pavizha.wav";
		musicStuff musicobject=new musicStuff();


		System.out.println(" insertion front---");
		musicobject.insertfront(filepath3);
		System.out.println("");
		musicobject.display();
		System.out.println("");
		musicobject.playmusic(filepath3);
		System.out.println("");
		System.out.println("after deletion front---");
		musicobject.delete_front();
		System.out.println("");
		musicobject.display();
		System.out.println("");

		/*System.out.println(" insertion front---");
		musicobject.insertfront(filepath1);
		System.out.println("");
		System.out.println(" insertion front---");
		musicobject.insertfront(filepath2);
		System.out.println("");
		System.out.println(" insertion back---");
```

```
            musicobject.insertback(filepath3);
            System.out.println("");
            musicobject.display();



            System.out.println("");
            System.out.println("after deletion front---");
            musicobject.delete_front();
            System.out.println("");
            musicobject.display();

            musicobject.playmusic(filepath2);

            System.out.println("");
            System.out.println("after deletion back---");
            musicobject.delete_back();
            musicobject.display();


            System.out.println(" insertion front---");
            musicobject.insertfront(filepath4);
            System.out.println("");
            System.out.println(" insertion front---");
            musicobject.insertfront(filepath5);



            System.out.println("");
            System.out.println("after deletion back---");
            System.out.println("");
            musicobject.delete_back();
            musicobject.display();*/

            System.out.println("******************************");
            System.out.println("----------SVS PLAYER---------");
            System.out.println("******************************");
            Scanner in=new Scanner(System.in);
            int choice=0;
            while(choice!=7) {
            System.out.println(" ");
            System.out.println("1.Display 2.inserfront  3.insertback 
    4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit");
            choice=in.nextInt();
            String add;
            add=in.nextLine();
            switch(choice) {
            case 1:
            musicobject.display();
            break;
            case 2:System.out.println("Enter address to insert front
");
            musicobject.insertfront(in.nextLine());
            break;
            case 3:System.out.println("Enter address to insert
back");
            musicobject.insertback(in.nextLine());
            break;
            case 4:
            musicobject.delete_front();
            break;
            case 5:
            musicobject.delete_back();
            break;
            case 6:System.out.println("song u want to listen");
            musicobject.playmusic(in.nextLine());
            break;
            case 7:System.out.println("GOOD BYE");
            break;


            }
            }


}
}
```

musicStuff.java

```
package Playingmusic;

import java.io.File;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.swing.JOptionPane;



import java.util.Scanner;
class Node_dl{
Node_dl next;
Node_dl prev;
String data;

Node_dl(){
data=null;  // starting data and null=0
prev=null;
```

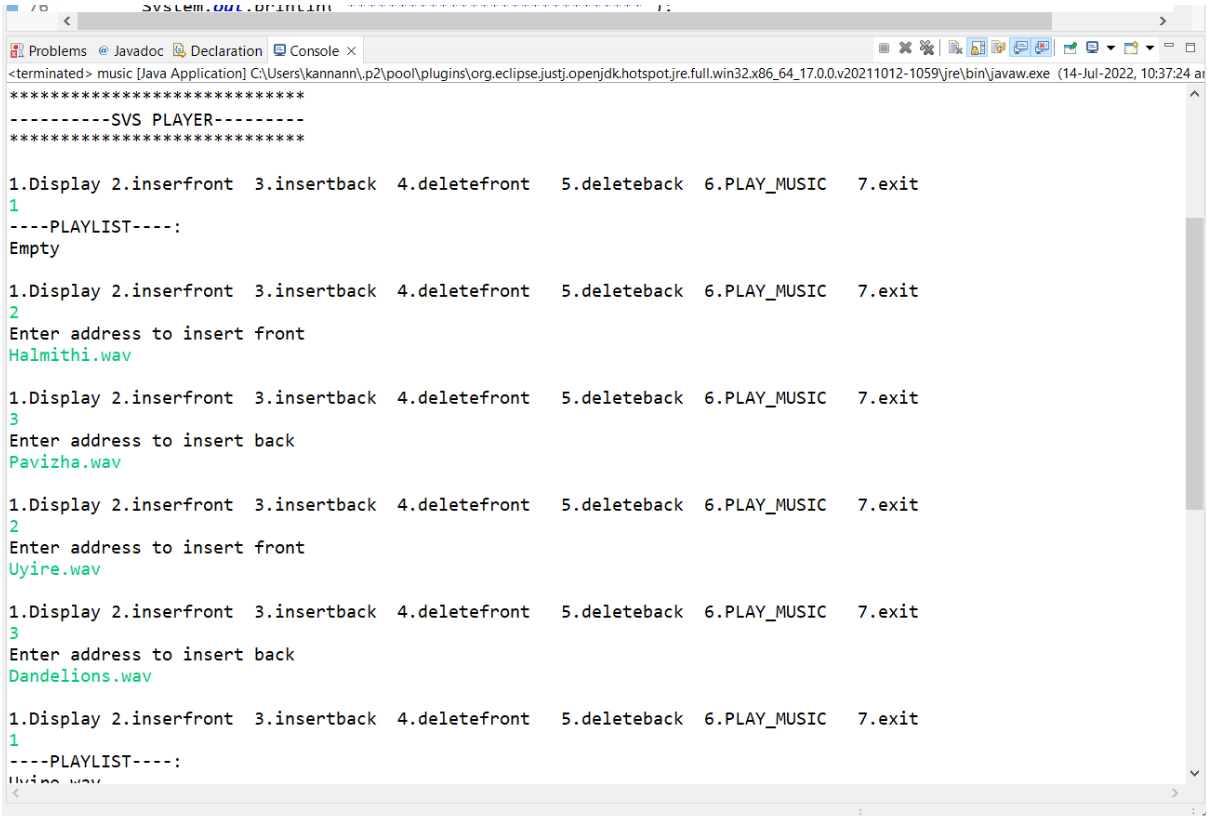
```
next=null;
}

Node_dl(String a){
prev=null;
data=a;   // when we add some data 1st ele data=a,next=null;
next=null;
}
}





public class musicStuff {

            Node_dl head=null; //obj1ect with value null;
            // initially head = 0;
            musicStuff(){
                  head=null;
            }
            musicStuff(String a){
                  head=new Node_dl(a);  // obj1ect for class Node_dl
      // Node_dl with parameter
      // new Node_dl will be created
            }
      void insertfront(String musicLocation) {
            if (head==null) {
                              head=new Node_dl(musicLocation); //
create a new Node_dl with data as x...1st Node_dl is head now
                        }
            else {
                              Node_dl temp=new
Node_dl(musicLocation);// another variable temp
                              temp.next=head;// newly added
                              head.prev=temp;
                              head=temp;
                        }
                  }
      void insertback(String musicLocation) {
            if(head==null) {
                   head=new Node_dl(musicLocation);
            }
            else {
                  Node_dl temp=head;
                  while(temp.next!=null) {
                        temp=temp.next;
                  }
```

```
                        Node_dl t=new Node_dl(musicLocation);
                        temp.next=t;
                        t.prev=temp;
                        }
                }
        void delete_front() {
                if(head!=null) {
                        head=head.next;
                        System.out.println("Deletd front");
                }
                else {
                        System.out.println("EMPTY");
                }
        }
        void delete_back() {
                //System.out.println("deleted item from back:");
                if(head ==null) {
                        System.out.println("EMPTY");
                }

                else {
                        if(head.next==null)
                                head=null;

                        else{Node_dl p=head,temp=head;    // p is to have
prev node's address of temp.
                    while(temp.next!=null)
                    {p=temp;                // current temp's address in p
                    temp=temp.next;}        // now temp is moved to next node

                    p.next=null;
                   // once temp has reached the last node,
                                       //p will have last but one node's
address
              }
                         System.out.println("Deletd Back");
                }
}

        void display() {
                System.out.println("----PLAYLIST----:");
                if(head==null) {
                System.out.println("Empty");
                }
                else {
                        Node_dl temp=head;
                        while(temp!=null){
                                System.out.println(temp.data);
                                temp=temp.next;
```

```
			}
		}
	}

	void playmusic(String musicLocation)
	{
		System.out.println("Playing song----------- ");
		try {
			File musicPath= new File(musicLocation);
			if(musicPath.exists()) {
				AudioInputStream
audioInput=AudioSystem.getAudioInputStream(musicPath);
				Clip clip=AudioSystem.getClip();
				clip.open(audioInput);
				clip.start();
				clip.loop(Clip.LOOP_CONTINUOUSLY);
				JOptionPane.showMessageDialog(null,"Press ok
to PAUSE");
				long
clipTimePosition=clip.getMicrosecondPosition();
				clip.stop();
				JOptionPane.showMessageDialog(null,"Press ok
to RESUME");
				clip.setMicrosecondPosition(clipTimePosition);
				clip.start();
				JOptionPane.showMessageDialog(null,"Press ok
to stop playing");
				clip.stop();
			}
			else {
				System.out.println("Can't find file");
			}


		}
		catch(Exception ex){
			ex.printStackTrace();

		}


	}
	}
```

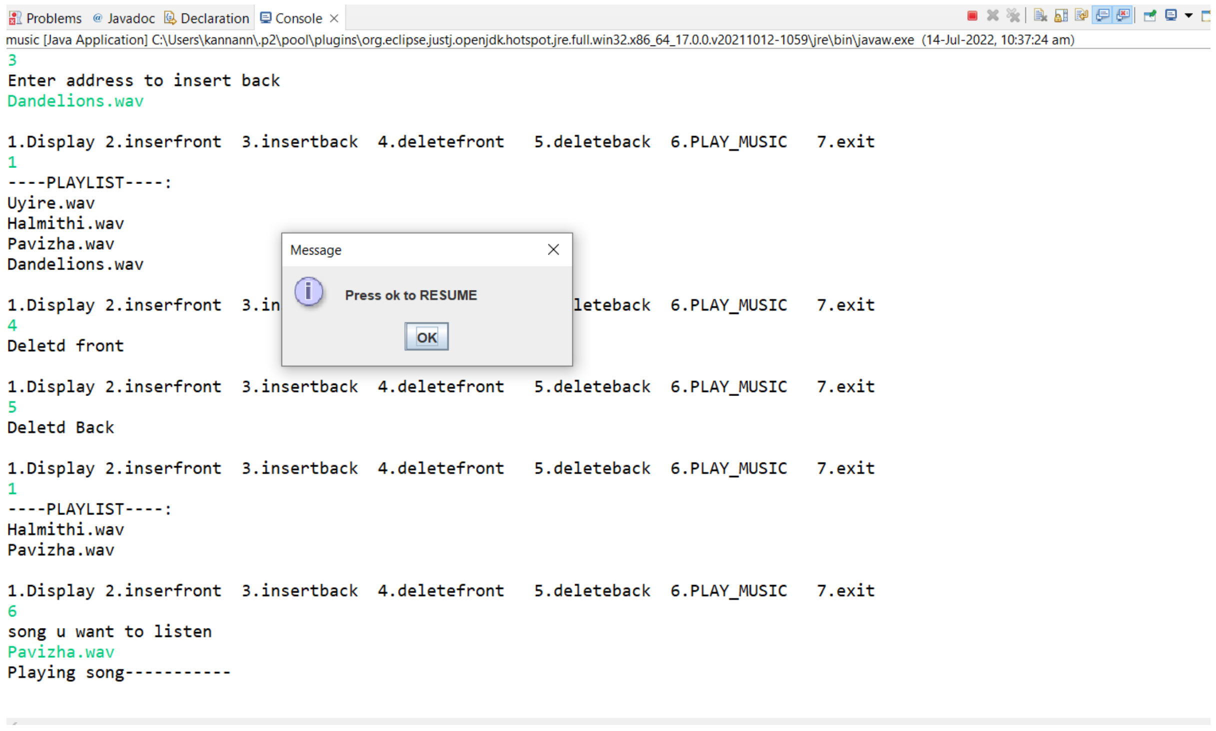
CHAPTER 7

OUTPUT



Problems Javadoc Declaration Console

```
<terminated> music [Java Application] C:\Users\kannann\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe  (14-Jul-2022, 10:37:24 a
******************************
----------SVS PLAYER---------
******************************

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Empty

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
2
Enter address to insert front
Halmithi.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
3
Enter address to insert back
Pavizha.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
2
Enter address to insert front
Uyire.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
3
Enter address to insert back
Dandelions.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
```



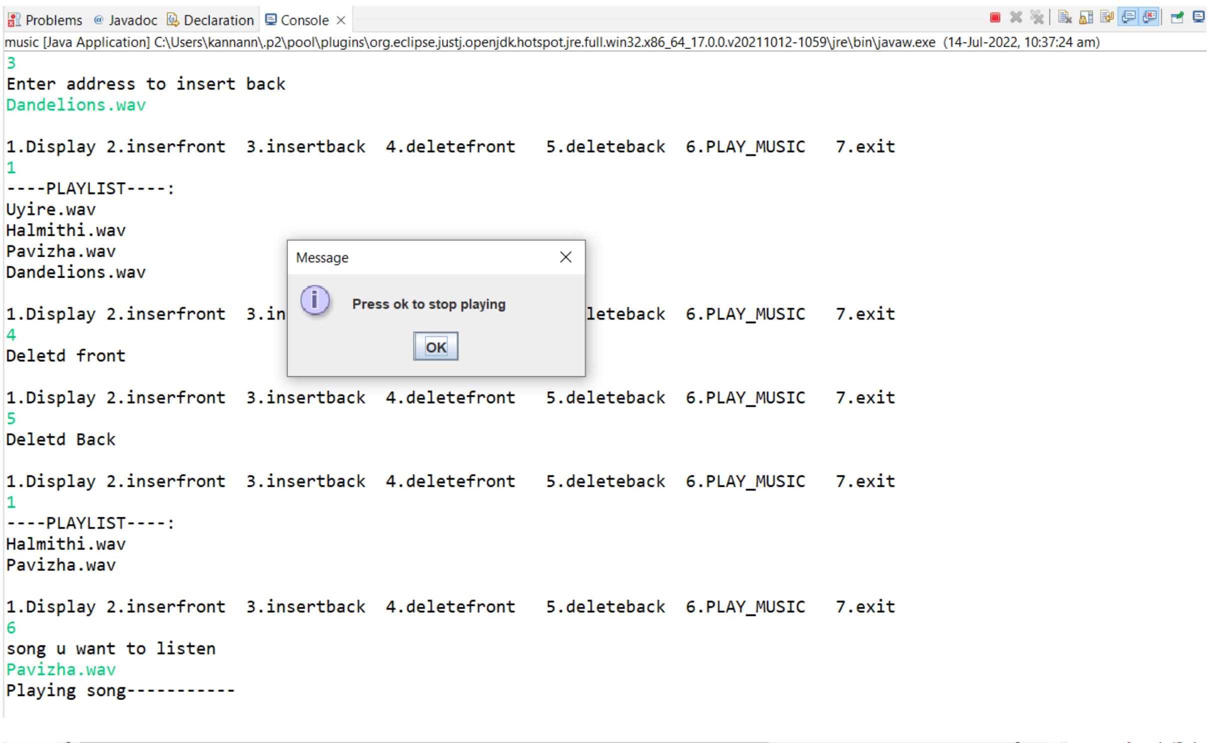
Problems Javadoc Declaration Console

```
music [Java Application] C:\Users\kannann\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe  (14-Jul-2022, 10:37:24 am)
3
Enter address to insert back
Dandelions.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Uyire.wav
Halmithi.wav
Pavizha.wav
Dandelions.wav

1.Display 2.inserfront  3.in                            leteback  6.PLAY_MUSIC   7.exit
4
Deletd front

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
5
Deletd Back

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Halmithi.wav
Pavizha.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
6
song u want to listen
Pavizha.wav
Playing song-----------
```

Message

Press ok to RESUME

OK

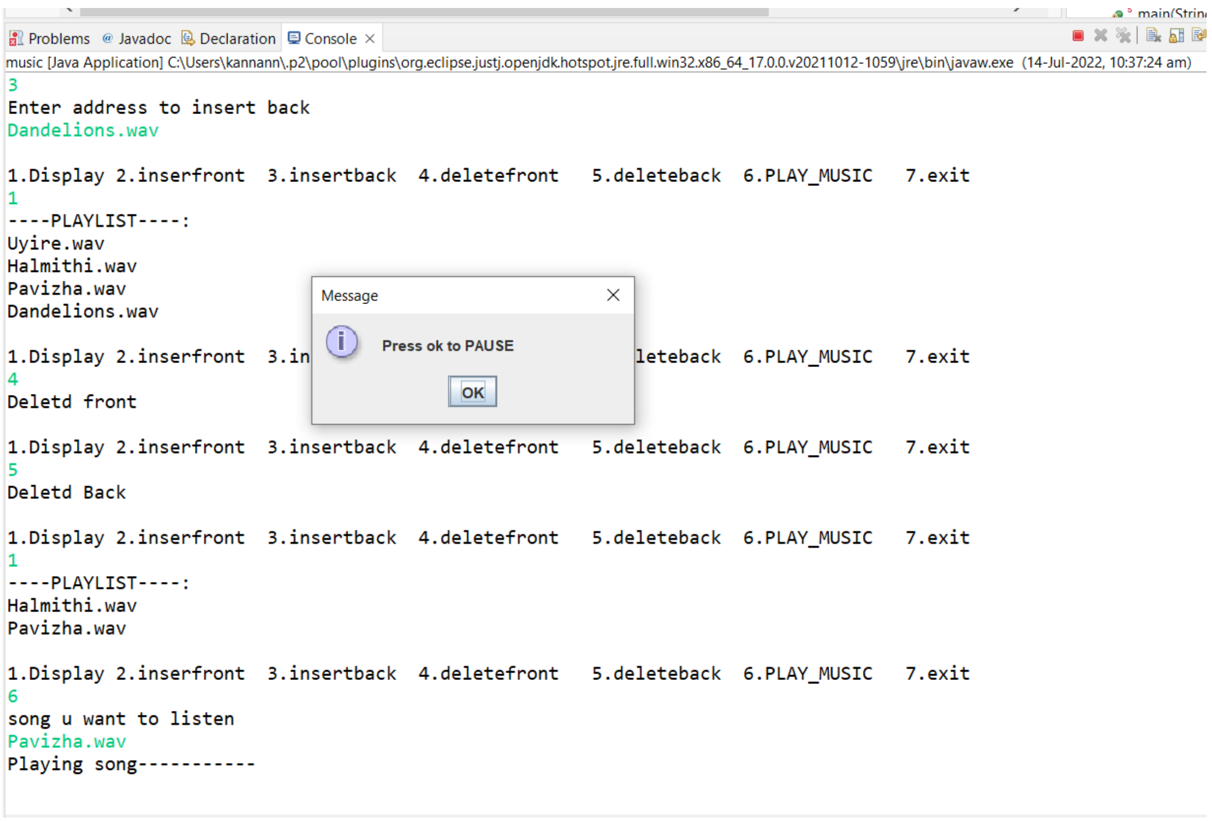


```
music [Java Application] C:\Users\kannann\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe  (14-Jul-2022, 10:37:24 am)
3
Enter address to insert back
Dandelions.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Uyire.wav
Halmithi.wav
Pavizha.wav
Dandelions.wav

1.Display 2.inserfront  3.in                         leteback  6.PLAY_MUSIC   7.exit
4
Deletd front

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
5
Deletd Back

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Halmithi.wav
Pavizha.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
6
song u want to listen
Pavizha.wav
Playing song----------
```

Message

Press ok to stop playing

OK

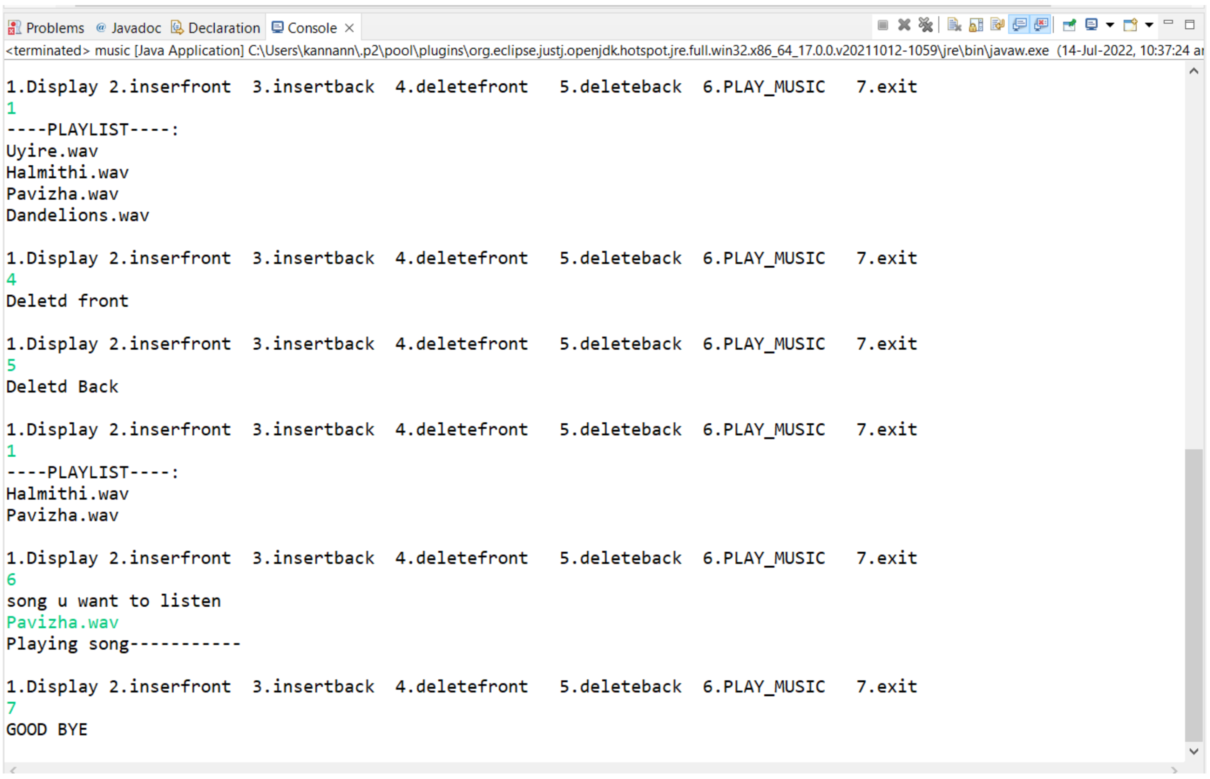


```
music [Java Application] C:\Users\kannann\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe  (14-Jul-2022, 10:37:24 am)
3
Enter address to insert back
Dandelions.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Uyire.wav
Halmithi.wav
Pavizha.wav
Dandelions.wav

1.Display 2.inserfront  3.in                         leteback  6.PLAY_MUSIC   7.exit
4
Deletd front

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
5
Deletd Back

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Halmithi.wav
Pavizha.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
6
song u want to listen
Pavizha.wav
Playing song-----------
```

Message

Press ok to PAUSE

OK



```
<terminated> music [Java Application] C:\Users\kannann\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\jre\bin\javaw.exe  (14-Jul-2022, 10:37:24 a
1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Uyire.wav
Halmithi.wav
Pavizha.wav
Dandelions.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
4
Deletd front

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
5
Deletd Back

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
1
----PLAYLIST----:
Halmithi.wav
Pavizha.wav

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
6
song u want to listen
Pavizha.wav
Playing song-----------

1.Display 2.inserfront  3.insertback  4.deletefront   5.deleteback  6.PLAY_MUSIC   7.exit
7
GOOD BYE
```

CHAPTER 8
CONCLUSION

This project thus allows the user to listen to the song of their choice. The doubly linked list makes the Music player, even more, better as deletion of the node is easier and the insertion can be performed efficiently at any node. When you add a new track to a playlist, you tack it on to the end. In a linked list, adding a new element is a constant time — O(1) operation. Note that as the songs are read in from a data source and added to the playlist, this will be done as a sequence of calls to append. A linked list has head and tail properties, this provides for an easy way to delineate the beginning and end of a playlist.

Future Scope

- The project can be made as an APP making it available for all. It will integrate the advantages of existing music players on the market. We can have numerous options like a change of screen modes (drive mode and night mode), a trim option to trim the favorite part of the music, and many more making it user-friendly.
- We can also implement a music player based on the mode of the users.

REFERENCES

- https://docs.oracle.com/javase/8/docs/api/javax/sound/sampled/package-summary.html
- https://www.youtube.com/watch?v=TErboGLHZGA
- https://mobcup.net
- Online Audio Converter - Convert audio files to MP3, WAV, MP4, M4A, OGG or iPhone Ringtones (online-audio-converter.com)
- Pan Yong-Cai, Liu Wen-chao, & Li Xiao. (2010). *Development and Research of Music Player Application Based on Android. 2010 International Conference on Communications and Intelligence Information Security*. doi:10.1109/icciis.2010.28
- Qing, Z., Ying, L., Yuan, P. G., & Sheng, L. Z. (2015). Music Player Based on the Cordova Cross-Platform. 2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence. doi:10.1109/acit-csi.2015.85

.