```
package com.internshala.echo.fragments
import android.app.Activity
import android.content.Context
import android.media.AudioManager
import android.media.MediaPlayer
import android.net.Uri
import android.os.Bundle
import android.support.v4.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.SeekBar
import android.widget.TextView
import com.internshala.echo.CurrentSongHelper
import com.internshala.echo.R
import com.internshala.echo.Songs
import java.util.*
/**
 * A simple [Fragment] subclass.
 */

class SongPlayingFragment : Fragment() {

    var myActivity: Activity? = null
    var mediaPlayer: MediaPlayer? = null

    /*The different variables defined will be used for their respective purposes*/
    /*Depending on the task they do we name the variables as such so that it gets easier to
identify the task they perform*/
    var startTimeText: TextView? = null
    var endTimeText: TextView? = null
    var playPauseImageButton: ImageButton? = null
    var previousImageButton: ImageButton? = null
    var nextImageButton: ImageButton? = null
    var loopImageButton: ImageButton? = null
    var shuffleImageButton: ImageButton? = null
    var seekBar: SeekBar? = null
    var songArtistView: TextView? = null
    var songTitleView: TextView? = null
    var currentPosition: Int = 0
    var fetchSongs: ArrayList<Songs>? = null

    /*The current song helper is used to store the details of the current song being
played*/
    var currentSongHelper: CurrentSongHelper? = null
    override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        val view = inflater!!.inflate(R.layout.fragment_song_playing, container, false)

        /*Linking views with their ids*/
        seekBar = view?.findViewById(R.id.seekBar)
        startTimeText = view?.findViewById(R.id.startTime)
        endTimeText = view?.findViewById(R.id.endTime)
        playPauseImageButton = view?.findViewById(R.id.playPauseButton)
        nextImageButton = view?.findViewById(R.id.nextButton)
        previousImageButton = view?.findViewById(R.id.previousButton)
        loopImageButton = view?.findViewById(R.id.loopButton)
        shuffleImageButton = view?.findViewById(R.id.shuffleButton)
        songArtistView = view?.findViewById(R.id.songArtist)
        songTitleView = view?.findViewById(R.id.songTitle)
        return view
    }
    override fun onAttach(context: Context?) {
        super.onAttach(context)
        myActivity = context as Activity
    }
    override fun onAttach(activity: Activity?) {
        super.onAttach(activity)
        myActivity = activity
    }
    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)

        /*Initialising the params of the current song helper object*/
        currentSongHelper = CurrentSongHelper()
        currentSongHelper?.isPlaying = true
        currentSongHelper?.isLoop = false
        currentSongHelper?.isShuffle = false
        var path: String? = null
        var _songTitle: String? = null
        var _songArtist: String? = null
        var songId: Long = 0

        try {
            path = arguments.getString("path")
            _songTitle = arguments.getString("songTitle")
            _songArtist = arguments.getString("songArtist")
            songId = arguments.getInt("songId").toLong()

            /*Here we fetch the received bundle data for current position and the list of
all songs*/
            currentPosition = arguments.getInt("position")
            fetchSongs = arguments.getParcelableArrayList("songData")

            /*Now store the song details to the current song helper object so that they
can be used later*/
            currentSongHelper?.songPath = path
            currentSongHelper?.songTitle = _songTitle
            currentSongHelper?.songArtist = _songArtist
            currentSongHelper?.songId = songId
            currentSongHelper?.currentPosition = currentPosition
        } catch (e: Exception) {
            e.printStackTrace()
        }
        mediaPlayer = MediaPlayer()
        mediaPlayer?.setAudioStreamType(AudioManager.STREAM_MUSIC)
        try {
            mediaPlayer?.setDataSource(myActivity, Uri.parse(path))
            mediaPlayer?.prepare()
        } catch (e: Exception) {
            e.printStackTrace()
        }
        mediaPlayer?.start()
        if (currentSongHelper?.isPlaying as Boolean) {
            playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
        } else {
            playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
        }
    }


/*A new click handler function is created to handle all the click functions in the song
playing fragment*/
    fun clickHandler() {

        /*The implementation will be taught in the coming topics*/
        shuffleImageButton?.setOnClickListener({
        })
        /*Here we set the click listener to the next button*/
        nextImageButton?.setOnClickListener({

            /*We set the player to be playing by setting isPlaying to be true*/
            currentSongHelper?.isPlaying = true

            /*First we check if the shuffle button was enabled or not*/
            if (currentSongHelper?.isShuffle as Boolean) {

                /*If yes, then  we play next song randomly
                * The check string is passed as the PlayNextLikeNormalShuffle which plays
the random next song*/
                playNext("PlayNextLikeNormalShuffle")
            } else {

                /*If shuffle was not enabled then we normally play the next song
                * The check string passed is the PlayNextNormal which serves the purpose*/
                playNext("PlayNextNormal")
            }
        })

        /*Here we set the click listener to the next button*/
        previousImageButton?.setOnClickListener({

            /*We set the player to be playing by setting isPlaying to be true*/
            currentSongHelper?.isPlaying = true

            /*First we check if the loop is on or not*/
            if (currentSongHelper?.isLoop as Boolean) {

                /*If the loop was on we turn it off*/
                loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
            }

            /*After all of the above is done we then play the previous song using the
playPrevious() function*/
            playPrevious()
        })

        /*Here we handle the click on the loop button*/
        loopImageButton?.setOnClickListener({

            /*if loop was enabled, we turn it off and vice versa*/
            if (currentSongHelper?.isLoop as Boolean) {

                /*Making the isLoop false*/
                currentSongHelper?.isLoop = false

                /*We change the color of the icon*/
                loopImageButton?.setBackgroundResource(R.drawable.loop_white_icon)
            } else {

                /*If loop was not enabled when tapped, we enable if and make the isLoop to
true*/
                currentSongHelper?.isLoop = true

                /*Loop and shuffle won't work together so we put shuffle false irrespectve
of the whether it was on or not*/
                currentSongHelper?.isShuffle = false

                /*Loop button color changed to mark it ON*/
                loopImageButton?.setBackgroundResource(R.drawable.loop_icon)
```

```
                 /*Changing the shuffle button to white, no matter which color it was
earlier*/
                 shuffleImageButton?.setBackgroundResource(R.drawable.shuffle_white_icon)
             }
         })

         /*Here we handle the click event on the play/pause button*/
         playPauseImageButton?.setOnClickListener({

             /*if the song is already playing and then play/pause button is tapped
             * then we pause the media player and also change the button to play button*/
             if (mediaPlayer?.isPlaying as Boolean) {
                 mediaPlayer?.pause()
                 currentSongHelper?.isPlaying = false
                 playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)

                 /*If the song was not playing the, we start the music player and
                 * change the image to pause icon*/
             } else {
                 mediaPlayer?.start()
                 currentSongHelper?.isPlaying = true
                 playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
             }
         })
     }

     /*The playNext() function is used to play the next song
     * The playNext() function is called when we tap on the next button*/
     fun playNext(check: String) {

         /*Here we check the value of the string parameter passed*/
         if (check.equals("PlayNextNormal", true)) {

             /*If the check string was PlayNextNormal, we normally move on to the next
song*/
             currentPosition = currentPosition + 1
         } else if (check.equals("PlayNextLikeNormalShuffle", true)) {

             /*If the check string was PlayNextLikeNormalShuffle, we then randomly select a
song and play it*/
             /*The next steps are used to choose the select a random number
             * First we declare a variable and then initialize it to a random object*/
             var randomObject = Random()

             /*Now here we calculate the random number
             * The nextInt(val n: Int) is used to get a random number between 0(inclusive)
and the number passed in this argument(n), exclusive.
             * Here we pass the paramter as the length of the list of the songs fetched
             * We add 1 to the size as the length will be one more than the size. For
example if the size of arraylist is 10, then it has items from 0 to 10, which gives the
length as 11*/

             var randomPosition = randomObject.nextInt(fetchSongs?.size?.plus(1) as Int)
             /*Now put the current position i.e the position of the song to be played next
equal to the random position*/
             currentPosition = randomPosition
         }

         /*Now if the current position equals the length of the i.e the current position
points to the end of the list
         * we then make the current position to 0 as no song will be there*/
         if (currentPosition == fetchSongs?.size) {
             currentPosition = 0
         }
         currentSongHelper?.isLoop = false
```

```
        /*Here we get the details of the song which is played as the next song
        * and update the contents of the current song helper*/
        var nextSong = fetchSongs?.get(currentPosition)
        currentSongHelper?.songPath = nextSong?.songData
        currentSongHelper?.songTitle = nextSong?.songTitle
        currentSongHelper?.songArtist = nextSong?.artist
        currentSongHelper?.songId = nextSong?.songID as Long

        /*Before playing the song we reset the media player*/
        mediaPlayer?.reset()
        try {

            /*Similar steps which were done when we started the music*/
            mediaPlayer?.setDataSource(myActivity, Uri.parse(currentSongHelper?.songPath))
            mediaPlayer?.prepare()
            mediaPlayer?.start()
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }

    /*The function playPrevious() is used to play the previous song again*/
    fun playPrevious() {

        /*Decreasing the current position by 1 to get the position of the previous song*/
        currentPosition = currentPosition - 1

        /*If the current position becomes less than 1, we make it 0 as there is no index as
-1*/
        if (currentPosition == -1) {
            currentPosition = 0
        }
        if (currentSongHelper?.isPlaying as Boolean) {
            playPauseImageButton?.setBackgroundResource(R.drawable.pause_icon)
        } else {
            playPauseImageButton?.setBackgroundResource(R.drawable.play_icon)
        }
        currentSongHelper?.isLoop = false

        /*Similar to the playNext() function defined above*/
        var nextSong = fetchSongs?.get(currentPosition)
        currentSongHelper?.songPath = nextSong?.songData
        currentSongHelper?.songTitle = nextSong?.songTitle
        currentSongHelper?.songArtist = nextSong?.artist
        currentSongHelper?.songId = nextSong?.songID as Long
        mediaPlayer?.reset()
        try {
            mediaPlayer?.setDataSource(myActivity, Uri.parse(currentSongHelper?.songPath))
            mediaPlayer?.prepare()
            mediaPlayer?.start()
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }
}
```