```
package com.internshala.echo.fragments
import android.app.Activity
import android.content.Context
import android.os.Bundle
import android.provider.MediaStore
import android.support.v4.app.Fragment
import android.support.v7.widget.DefaultItemAnimator
import android.support.v7.widget.LinearLayoutManager
import android.support.v7.widget.RecyclerView
import android.view.LayoutInflater
import android.view.Menu
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.RelativeLayout
import android.widget.TextView
import com.internshala.echo.R
import com.internshala.echo.Songs
import com.internshala.echo.adapters.FavoriteAdapter
/**
 * A simple [Fragment] subclass.
 */

class FavoriteFragment : Fragment() {

    var myActivity: Activity? = null
    var getSongsList: ArrayList<Songs>? = null
    var noFavorites: TextView? = null
    var nowPlayingBottomBar: RelativeLayout? = null
    var playPauseButton: ImageButton? = null
    var songTitle: TextView? = null
    var recyclerView: RecyclerView? = null

    override fun onCreateView(inflater: LayoutInflater?, container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {

        val view = inflater!!.inflate(R.layout.fragment_favorite, container, false)
        noFavorites = view?.findViewById(R.id.noFavourites)
        nowPlayingBottomBar = view.findViewById(R.id.hiddenBarFavScreen)
        songTitle = view.findViewById(R.id.songTitle)
        playPauseButton = view.findViewById(R.id.playPauseButton)
        recyclerView = view.findViewById(R.id.favoriteRecycler)
        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 onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?) {
        super.onActivityCreated(savedInstanceState)
        getSongsList = getSongsFromPhone()
```

```
        if (getSongsList == null) {
            recyclerView?.visibility = View.INVISIBLE
            noFavorites?.visibility = View.VISIBLE
        } else {
            var favoriteAdapter = FavoriteAdapter(getSongsList as ArrayList<Songs>,
myActivity as Context)
            val mLayoutManager = LinearLayoutManager(activity)
            recyclerView?.layoutManager = mLayoutManager
            recyclerView?.itemAnimator = DefaultItemAnimator()
            recyclerView?.adapter = favoriteAdapter
            recyclerView?.setHasFixedSize(true)
        }
    }

    override fun onResume() {
        super.onResume()
    }

    override fun onPrepareOptionsMenu(menu: Menu?) {
        super.onPrepareOptionsMenu(menu)
    }

    /*As the name suggests, this function is used to fetch the songs present in your phones
and returns the arraylist of the same*/
    fun getSongsFromPhone(): ArrayList<Songs> {
        var arrayList = ArrayList<Songs>()

        /*A content resolver is used to access the data present in your phone
        * In this case it is used for obtaining the songs present your phone*/
        var contentResolver = myActivity?.contentResolver

        /*Here we are accessing the Media class of Audio class which in turn a class of
Media Store, which contains information about all the media files present
        * on our mobile device*/
        var songUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI

        /*Here we make the request of songs to the content resolver to get the music files
from our device*/
        var songCursor = contentResolver?.query(songUri, null, null, null, null)

        /*In the if condition we check whether the number of music files are null or not.
The moveToFirst() function returns the first row of the results*/
        if (songCursor != null && songCursor.moveToFirst()) {
            val songId = songCursor.getColumnIndex(MediaStore.Audio.Media._ID)
            val songTitle = songCursor.getColumnIndex(MediaStore.Audio.Media.TITLE)
            val songArtist = songCursor.getColumnIndex(MediaStore.Audio.Media.ARTIST)
            val songData = songCursor.getColumnIndex(MediaStore.Audio.Media.DATA)
            val dateIndex = songCursor.getColumnIndex(MediaStore.Audio.Media.DATE_ADDED)

            /*moveToNext() returns the next row of the results. It returns null if there
is no row after the current row*/
            while (songCursor.moveToNext()) {
                var currentId = songCursor.getLong(songId)
                var currentTitle = songCursor.getString(songTitle)
                var currentArtist = songCursor.getString(songArtist)
                var currentData = songCursor.getString(songData)
                var currentDate = songCursor.getLong(dateIndex)

                /*Adding the fetched songs to the arraylist*/
```

```
                    arrayList.add(Songs(currentId, currentTitle, currentArtist, currentData,
currentDate))
                }
            }

            /*Returning the arraylist of songs*/
            return arrayList
        }
}
```