**实 验 报 告**

**课程名称** 移动应用系统

**实验项目** 移动平台下多媒体的运用

**实验仪器** 台式电脑

**系 别** 计算机科学与技术系

**专 业** 计算机科学与技术

**班级/学号** 计科1905 / 2019010132

**学生姓名** 陈浩

**实验日期** 2021/12/17

**成 绩**

**指导教师** 陈雷

实验四 移动平台下多媒体的运用

1. 实验目的：
2. 通过本实验，进一步熟悉Android Studio的用法
3. 通过本实验，掌握Android系统中服务的使用
4. 通过本实验，掌握Android系统中多媒体API的使用
5. 通过本实验，进一步熟悉适配器视图、网络、数据持久化等相关知识。
6. 实验内容及要求：

【注：可以自己设计题目，但需要经过指导教师同意】

设计并实现一个**音乐播放器**，具体要求如下【100分】

1）能够下载并存储mp3音乐文件到Android设备的内部存储空间，下载功能需要以服务的形式实现（可以使用IntentService实现），作为参考，下列网站可以找到部分可供下载的音乐文件：

https://freemusicarchive.org/genre/Classical/

2）以ListView的形式显示播放列表，请设置至少包含5首曲目的播放列表用于测试；点击播放列表中的表项后，若文件不存在则下载文件，下载完成后开始播放对应歌曲；若文件存在则直接开始播放。

参考：http://www.cnblogs.com/wenjiang/p/3140055.html

3）实现一个歌曲播放界面，包括上一首、下一首、播放/暂停、播放进度等按钮并能够显示曲目名称。

【评分标准】：

1）实现了题目所要求的程序功能，将得到对应的分数

2）实现功能的同时，应该尽可能保证程序的稳定性、易用性，出现下列情形，将扣去对应的分数

（1）程序崩溃/闪退，扣10分

（2）程序缺少必要的文字提示，扣5分

3）编写程序的过程中，可以参考网上的资料，但必须能够看懂程序代码并回答老师提出的问题，如果无法解释清楚程序代码，扣15分。

4）雷同的实验报告，提交时间较晚的同学成绩为0

1. 程序代码及运行结果：

注：提供完整的程序代码以及运行结果截图

代码：

1. DownMusicService.Java

*package* com.example.musicplayer;  
  
*import* android.app.IntentService;  
*import* android.content.Intent;  
*import* android.os.Environment;  
*import* android.os.Handler;  
*import* android.os.Looper;  
*import* android.util.Log;  
*import* android.widget.Toast;  
*import* androidx.annotation.*NonNull*;  
  
*import* java.io.File;  
*import* java.io.IOException;  
  
*import* okhttp3.Call;  
*import* okhttp3.Callback;  
*import* okhttp3.OkHttpClient;  
*import* okhttp3.Request;  
*import* okhttp3.Response;  
*import* okio.BufferedSink;  
*import* okio.Okio;  
*import* okio.Sink;  
  
*public class* DownMusicService *extends* IntentService {  
  
 *// 设置音乐存储的相对地址 (位于 SDcard 上)  
 public static final* File PATH = Environment.getExternalStoragePublicDirectory("/Music");  
 *// 下载音乐的名称  
 private* String fileName;  
  
 *protected void* onHandleIntent(Intent intent) {  
  
 *final* String url = intent.getStringExtra("path");  
 System.out.println(url);  
 *// 获取音乐的id* String str = url.substring(0, url.indexOf(".mp3"));  
 fileName = str.substring(str.lastIndexOf("=") + 1);  
  
 *// 设置开始下载的时间  
 final long* startTime = System.currentTimeMillis();  
 Log.i("DOWNLOAD","startTime="+startTime);  
  
 *// 建立http请求  
 // 开启下载  
 final* Request request = *new* Request.Builder()  
 .url(url)  
 .build();  
 *new* OkHttpClient().newCall(request).enqueue(*new* Callback() {  
  
 *// 若下载失败  
 @Override  
 public void* onFailure( Call call, IOException e) {  
 e.printStackTrace();  
 Log.i("DOWNLOAD","download failed");  
 *// 刷新当前页面UI* Handler handler = *new* Handler(Looper.getMainLooper());  
 handler.post(*new* Runnable(){  
 *public void* run(){  
 Toast.makeText(getApplicationContext(), "下载失败", Toast.LENGTH\_LONG).show();  
 }  
 });  
 }  
  
 *// 成功响应  
 @Override  
 public void* onResponse(Call call, Response response) *throws* IOException {  
  
 *// 设置缓冲区 准备写流数据* BufferedSink bufferedSink = *null*;  
 *try* {  
 *// 设置文件路径 和 存储信息* File dest = *new* File(PATH, fileName + ".mp3");  
 *// sink进行数据流写入* Sink sink = Okio.sink(dest);  
 *// 写缓存* bufferedSink = Okio.buffer(sink);  
 *assert* response.body() != *null*;  
 *// 缓存写入* bufferedSink.writeAll(response.body().source());  
  
 bufferedSink.close();  
 Log.i("DOWNLOAD", "download success");  
 Log.i("DOWNLOAD", "totalTime=" + (System.currentTimeMillis() - startTime));  
  
 *// 刷新当前页面UI* Handler handler = *new* Handler(Looper.getMainLooper());  
 handler.post(*new* Runnable() {  
 *public void* run() { Toast.makeText(getApplicationContext(), "下载成功!", Toast.LENGTH\_LONG).show();  
 }  
 });  
  
 } *catch* (Exception e) {  
 e.printStackTrace();  
 Log.i("DOWNLOAD", "download failed");  
 Handler handler = *new* Handler(Looper.getMainLooper());  
 handler.post(*new* Runnable() {  
 *public void* run() {  
 Toast.makeText(getApplicationContext(), "下载失败!", Toast.LENGTH\_LONG).show();  
 }  
 });  
  
 } *finally* {  
 *if* (bufferedSink != *null*) {  
 *// 关闭缓存* bufferedSink.close();  
 }  
 }  
 }  
 });  
  
 Intent intentNew = *new* Intent(DownMusicService.*this*, MainActivity.*class*);  
 *// 切换回主活动* intentNew.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  
 startActivity(intentNew);  
 }  
  
  
 *public* DownMusicService() {  
 *super*("");  
 }  
}

1. MainActivity.Java

*package* com.example.musicplayer;  
  
*import* android.Manifest;  
*import* android.annotation.SuppressLint;  
*import* android.annotation.TargetApi;  
*import* android.content.Intent;  
*import* android.os.Build;  
*import* android.os.Handler;  
*import* android.os.Message;  
*import* android.os.Bundle;  
*import* android.util.Log;  
*import* android.view.View;  
*import* android.widget.ArrayAdapter;  
*import* android.widget.Button;  
*import* android.widget.EditText;  
*import* android.widget.ListView;  
*import* android.widget.SeekBar;  
*import* android.widget.TextView;  
  
*import* androidx.annotation.RequiresApi;  
*import* androidx.appcompat.app.AppCompatActivity;  
  
  
*import* com.zhy.m.permission.MPermissions;  
  
  
  
*import* java.io.File;  
  
*public class* MainActivity *extends* AppCompatActivity *implements* Runnable {  
  
 *// 开始播放 和 暂停播放的标志位  
 int* flag = 1;  
 *// 播放 和 暂停 的内容显示  
 private* TextView txtInfo;  
 *// 进度条  
 private* SeekBar seekBar;  
 *// 音乐具体服务  
 private* MusicService musicService;  
 *// 处理改变进度条事件  
 private* Handler handler;  
 *// 自动更新时长  
 int* UPDATE = 0x101;  
  
 *// 歌名  
 private* EditText song;  
 *// 开始按钮  
 private* Button btnStart;  
 *// 暂停按钮  
 private* Button btnPause;  
 *// 继续播放按钮  
 private* Button btnContinue;  
 *// 下载按钮  
 private* Button btnDownload;  
 *// 下一首按钮  
 private* Button btnLast;  
 *// 上一首按钮  
 private* Button btnNext;  
 *// 刷新按钮  
 private* Button refresh;  
  
  
 @SuppressLint("HandlerLeak")  
 @RequiresApi(api = Build.VERSION\_CODES.N)  
 @Override  
 *protected void* onCreate(Bundle savedInstanceState) {  
 *super*.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 *// 请求权限 (获取地区信息、读、写 SDcard)* getPermission();  
 musicService = *new* MusicService();  
 *try* {  
 *// 读取MP3列表* setListViewAdapter();  
 } *catch* (Exception e) {  
 Log.i("TAG", "Get ListInfo Error!");  
 }  
  
 *// 开始按钮* btnStart = findViewById(R.id.btn\_star);  
 btnStart.setOnClickListener(*new* View.OnClickListener() {  
 *@Override  
 public void* onClick(View view) {  
 *try* {  
 *// 引入flag作为标志  
 // 当flag为 1 时, 此时 Musicplayer 内没有歌曲, 故执行 musicService.play() 函数 (即自动从第一首歌开始播放)  
 // 第一次播放后 flag自增  
 // 再次点击 “开始/暂停” 时 flag > 1 就执行 继续播放 或 暂停播放  
 if* (flag == 1) {  
 musicService.play();  
 flag++;  
 }  
 *else if* (!musicService.player.isPlaying()) {  
 musicService.goPlay();  
 }  
 *else if* (musicService.player.isPlaying()) {  
 musicService.pause();  
 }  
 } *catch* (Exception e) {  
 Log.i("LAT", "Start Error!");  
 }  
  
 }  
 });  
  
 *// 歌名* song = findViewById(R.id.song);  
  
 *// 下载按钮* btnDownload = findViewById(R.id.btnDownload);  
 btnDownload.setOnClickListener(*new* View.OnClickListener() {  
 @Override  
 *public void* onClick(View v) {  
 *// 获取歌曲id* String pathId = song.getText().toString();  
 System.out.println(pathId);  
 *// 下载地址url* String pathLast = "http://music.163.com/song/media/outer/url?id=" + pathId +".mp3";  
 System.out.println(pathLast);  
 *// 开启下载的活动* Intent intent = *new* Intent(MainActivity.*this*, DownMusicService.*class*);  
 intent.putExtra("path", pathLast);  
 startService(intent);  
 }  
 });  
  
 *// 继续播放按钮* btnContinue = findViewById(R.id.btn\_continue);  
 btnContinue.setOnClickListener(*new* View.OnClickListener() {  
 @SuppressLint("SetTextI18n")  
 @Override  
 *public void* onClick(View view) {  
 *try* {  
 musicService.goPlay();  
 txtInfo.setText("Continue Play!");  
 } *catch* (Exception e) {  
 Log.i("LAT", "Pause Error!");  
 }  
  
 }  
 });  
  
 *// 暂停按钮* btnPause = findViewById(R.id.btn\_stop);  
 btnPause.setOnClickListener(*new* View.OnClickListener() {  
 @SuppressLint("SetTextI18n")  
 @Override  
 *public void* onClick(View view) {  
 *try* {  
 musicService.pause();  
 *// 当点击停止按钮时  
 // flag 默认为 1* flag = 1;  
 txtInfo.setText("Pause Play!");  
 } *catch* (Exception e) {  
 Log.i("LAT", "Pause Error!");  
 }  
  
 }  
 });  
  
 *// 上一首按钮* btnLast = findViewById(R.id.btn\_last);  
 btnLast.setOnClickListener(*new* View.OnClickListener() {  
 @Override  
 *public void* onClick(View view) {  
 *try* {  
 musicService.last();  
 } *catch* (Exception e) {  
 Log.i("LAT", "the Last Music Error!");  
 }  
  
 }  
 });  
  
 *// 下一首按钮* btnNext = findViewById(R.id.btn\_next);  
 btnNext.setOnClickListener(*new* View.OnClickListener() {  
 @Override  
 *public void* onClick(View view) {  
 *try* {  
 musicService.next();  
 } *catch* (Exception e) {  
 Log.i("LAT", "the Next Music Error!");  
 }  
  
 }  
 });  
  
 *// 刷新按钮* refresh = findViewById(R.id.refresh);  
 refresh.setOnClickListener(*new* View.OnClickListener() {  
 @Override  
 *public void* onClick(View v) {  
 *try* {  
 *// 获取最新的当前页面* Intent newIntent = getIntent();  
 finish();  
 startActivity(newIntent);  
 } *catch* (Exception e){  
 Log.i("TAG","Refresh Error!");  
 }  
 }  
 });  
  
 *// 进度条* seekBar = findViewById(R.id.sb);  
 *// 设置监听 (长度 停止状态 停止时间)* seekBar.setOnSeekBarChangeListener(*new* SeekBar.OnSeekBarChangeListener() {  
 *// 用于监听SeekBar进度值改变时* @Override  
 *public void* onProgressChanged(SeekBar seekBar, *int* i, *boolean* b) {  
 }  
  
 *// 用于监听SeekBar开始拖动时* @Override  
 *public void* onStartTrackingTouch(SeekBar seekBar) {  
 }  
  
 *// 用于监听SeekBar停止拖动时* @Override  
 *public void* onStopTrackingTouch(SeekBar seekBar) {  
 *// 显示当前进度  
 int* progress = seekBar.getProgress();  
 Log.i("TAG:", "" + progress + "");  
  
 *// 得到该首歌曲最长秒数  
 int* musicMax = musicService.player.getDuration();  
  
 *// 跳到该曲该秒  
 int* seekBarMax = seekBar.getMax();  
 musicService.player.seekTo(musicMax \* progress / seekBarMax);  
 }  
 });  
  
 *// 歌曲信息* txtInfo = findViewById(R.id.tv1);  
 *// 自动改变进度条的线程* Thread t = *new* Thread(*this*);  
 *// 实例化一个handler对象 用于处理当前进程* handler = *new* Handler() {  
 @SuppressLint("SetTextI18n")  
 @Override  
 *public void* handleMessage(Message msg) {  
 *super*.handleMessage(msg);  
 *// 更新UI  
 // 获得最大秒数  
 int* mMax = musicService.player.getDuration();  
  
 *// 歌曲名称变更  
 if* (msg.what == UPDATE) {  
 *try* {  
 *// 获取新进程的进度条 和 歌曲信息 (时长)* seekBar.setProgress(msg.arg1);  
 *// (ms为单位计算)* txtInfo.setText(setPlayInfo(msg.arg2 / 1000, mMax / 1000));  
 } *catch* (Exception e) {  
 e.printStackTrace();  
 }  
 } *else* {  
 seekBar.setProgress(0);  
 txtInfo.setText("Pause Play!");  
 }  
 }  
 };  
  
 t.start();  
  
 }  
  
 *// 定义需要启用的权限数量  
 private static final int* ACCESS\_FINE\_LOCATION = 3;  
  
 *// 开启权限* @TargetApi(Build.VERSION\_CODES.M)  
 *private void* getPermission() {  
 MPermissions.requestPermissions(MainActivity.*this*, ACCESS\_FINE\_LOCATION,  
 Manifest.permission.READ\_EXTERNAL\_STORAGE,  
 Manifest.permission.WRITE\_EXTERNAL\_STORAGE,  
 Manifest.permission.ACCESS\_FINE\_LOCATION  
 );  
 }  
  
 *// 向当前ListView 添加 音乐的MP3名称  
 private void* setListViewAdapter() {  
 String[] str = *new* String[musicService.musicList.size()];  
 *int* i = 0;  
 *// 将文件路径中的MP3名称读出 并加载到 ListView 中  
 for* (String path : musicService.musicList) {  
 File file = *new* File(path);  
 str[i++] = file.getName();  
 }  
 ArrayAdapter adapter = *new* ArrayAdapter(*this*, android.R.layout.simple\_list\_item\_1, str);  
 ListView listView = findViewById(R.id.lv1);  
 listView.setAdapter(adapter);  
 }  
  
 *// 线程启动 (同步更新歌曲进度)* @Override  
 *public void* run() {  
 *int* position, mMax, sMax;  
 *while* (!Thread.currentThread().isInterrupted()) {  
 *// 若当前线程正在播放歌曲  
 if* (musicService.player != *null* && musicService.player.isPlaying()) {  
 *// 当前歌曲播放进度 (秒)* position = musicService.getCurrentProgress();  
 *// 当前歌曲最大秒数* mMax = musicService.player.getDuration();  
 *// seekBar的最大值 (需要计算百分比)* sMax = seekBar.getMax();  
  
 *//获取一个Message* Message m = handler.obtainMessage();  
 *// seekBar进度条的百分比* m.arg1 = position \* sMax / mMax;  
 *// 当前位置* m.arg2 = position;  
 *// 设置为更新状态* m.what = UPDATE;  
 *// 对线程进行处理 (即发送信息告诉活动进行更新)* handler.sendMessage(m);  
  
 *// 每间隔1秒发送一次更新消息  
 try* {  
 Thread.sleep(1000);  
 } *catch* (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 }  
  
 }  
  
 *// 设置当前播放的信息  
 private* String setPlayInfo(*int* position, *int* max) {  
 String info = "正在播放:\t\t" + musicService.songName + "\t\t\t\t";  
  
 *// 设置播放时间的跳转逻辑  
  
 // 当前进度：  
 int* pMinutes = 0;  
 *// 当前秒数 >= 60  
 while* (position >= 60) {  
 pMinutes++;  
 position -= 60;  
 }  
  
 *// 格式 00:00* String now = (pMinutes < 10 ? "0" + pMinutes : pMinutes) + ":"  
 + (position < 10 ? "0" + position : position);  
  
 *// 总进度：  
 int* mMinutes = 0;  
 *while* (max >= 60) {  
 mMinutes++;  
 max -= 60;  
 }  
 String all = (mMinutes < 10 ? "0" + mMinutes : mMinutes) + ":"  
 + (max < 10 ? "0" + max : max);  
  
 *// 返回格式： 歌名 + 当前播放时间 / 总歌曲时间  
 return* info + now + " / " + all;  
 }  
  
}

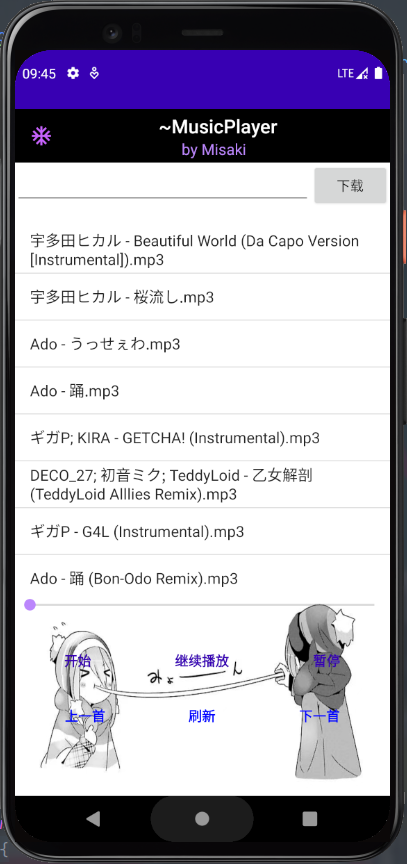
1. MusicService.Java

*package* com.example.musicplayer;  
  
*import* android.media.AudioManager;  
*import* android.media.MediaPlayer;  
*import* android.os.Environment;  
*import* android.util.Log;  
*import* android.widget.ArrayAdapter;  
*import* android.widget.ListView;  
  
  
*import* java.io.File;  
*import* java.io.FilenameFilter;  
*import* java.util.ArrayList;  
*import* java.util.List;  
*import* java.util.Objects;  
*import* java.util.Random;  
  
*public class* MusicService {  
  
 *// 获取SD卡下的音乐目录  
 public static final* File PATH = Environment.getExternalStoragePublicDirectory("/Music");  
 *// 存放所有MP3的绝对路径  
 public* List<String> musicList;  
 *// 定义多媒体对象 (音乐媒体)  
 public* MediaPlayer player;  
 *// 当前播放的歌曲标号  
 public int* songNum;  
 *// 当前播放的歌曲名  
 public* String songName;  
  
 *// 引入过滤器 (方便读取文件)  
 static class* MusicFilter *implements* FilenameFilter {  
 *public boolean* accept(File dir, String name) {  
 *// 返回当前目录所有以.mp3结尾的文件  
 return* (name.endsWith(".mp3"));  
 }  
 }  
  
 *// 扫描获取文件  
 public* MusicService() {  
 *super*();  
 player = *new* MediaPlayer();  
 *// 构造动态存储数组* musicList = *new* ArrayList<String>();  
 *try* {  
 *// 获取Music文件的二级目录* File MUSIC\_PATH = *new* File(String.valueOf(PATH));  
 *if* (MUSIC\_PATH.exists()){  
 *// 将获取到的所有文件存入数组中* File[] files = MUSIC\_PATH.listFiles(*new* MusicFilter());  
 *if* (files == *null* || files.length == 0) {  
 Log.e("TAG", String.format("Data Empty!"));  
 *return*;  
 }  
 *int* length = files.length;  
 *// 当数组不空时，遍历数组文件  
 if* (length > 0) {  
 *for* (File file : Objects.requireNonNull(MUSIC\_PATH.listFiles(*new* MusicFilter()))) {  
 *// 将各个文件的绝对路径写入数组中* musicList.add(file.getAbsolutePath());  
 }  
 }  
 }  
 } *catch* (Exception e) {  
 Log.i("TAG", String.format("Load File Error!%s", e.getMessage()));  
 }  
 }  
  
 *// 截取音乐文件名  
 // 格式：/A/B/C/xxx.mp3  
 public void* setPlayName(String dataSource) {  
 File file = *new* File(dataSource);  
 String name = file.getName();  
 *int* index = name.lastIndexOf(".");  
 songName = name.substring(0, index);  
 }  
  
 *// 准备播放音乐  
 public void* play() {  
 *try* {  
 *// 重置多媒体 (初始化)* player.reset();  
 *// 得到当前播放音乐的路径* String dataSource = musicList.get(songNum);  
 *// 截取歌名* setPlayName(dataSource);  
 *// 指定参数为音频文件* player.setAudioStreamType(AudioManager.STREAM\_MUSIC);  
 *// 为多媒体对象设置播放路径* player.setDataSource(dataSource);  
 *// 准备播放* player.prepare();  
 *// 开始播放* player.start();  
 *// 当前多媒体对象播放完成时的监听时间* player.setOnCompletionListener(*new* MediaPlayer.OnCompletionListener() {  
 *public void* onCompletion(MediaPlayer arg0) {  
 *// 自动下一首歌* next();  
 }  
 });  
  
 } *catch* (Exception e) {  
 Log.v("MusicService", e.getMessage());  
 }  
 }  
  
 *// 继续播放  
 public void* goPlay() {  
 *// 获取当前播放进度  
 int* position = getCurrentProgress();  
  
 *// 设置当前MediaPlayer的播放位置 (ms为单位)* player.seekTo(position);  
 *try* {  
 *// 装载流媒体文件 准备播放* player.prepare();  
 } *catch* (Exception e) {  
 e.printStackTrace();  
 }  
 player.start();  
 }  
  
 *// 获取当前进度  
 public int* getCurrentProgress() {  
 *if*(player == *null*) {  
 *return* 0;  
 }  
 *else* {  
 *return* player.getCurrentPosition();  
 }  
 }  
  
 *// 下一首  
 public void* next() {  
 *// 若当前为最后一首 则自动跳转至开头歌曲  
 // 否则自增为下一首* songNum = (songNum == musicList.size() - 1 ? 0 : songNum + 1);  
 play();  
 }  
  
 *// 上一首  
 public void* last() {  
 *// 若当前为第一首 则自动跳转至结尾歌曲  
 // 否则自减为上一首* songNum = (songNum == 0 ? musicList.size() - 1 : songNum - 1);  
 play();  
 }  
  
 *// 暂停播放  
 public void* pause() {  
 *if* (player != *null* && player.isPlaying()) {  
 *// 获取当前播放进度  
 int* position = player.getCurrentPosition();  
 *// 设置存储当前播放进度 (ms为单位)* player.seekTo(position);  
 player.pause();  
 }  
 }  
}

1. activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity"  
 android:background="@drawable/zimalin">  
  
 <androidx.appcompat.widget.Toolbar  
 android:background="@color/black"  
 app:title="~MusicPlayer"  
 app:titleTextColor="@color/white"  
 android:id="@+id/toolbar"  
 app:subtitle=" by Misaki"  
 app:subtitleTextColor="@color/purple\_200"  
 app:titleMarginStart="95dp"  
 app:navigationIcon="@drawable/ic\_baseline\_ac\_yuki\_24"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"/>  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
 <EditText  
 android:id="@+id/song"  
 android:layout\_width="310dp"  
 android:layout\_height="wrap\_content"  
 android:inputType="text"/>  
 <Button  
 android:id="@+id/btnDownload"  
 android:text="下载"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
 </LinearLayout>  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="400dp"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <ListView  
 android:id="@+id/lv1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content" />  
 </LinearLayout>  
  
  
 <SeekBar  
 android:id="@+id/sb"  
 android:layout\_width="match\_parent"  
 android:layout\_height="30dp"  
 android:maxHeight="2dp"  
 android:minHeight="2dp"  
 android:paddingBottom="3dp"  
 android:paddingLeft="12dp"  
 android:max="200"  
 android:progressTint="@color/black"  
 android:thumbTint="@color/purple\_200"  
 android:paddingRight="12dp"  
 android:paddingTop="3dp" />  
 <TextView  
 android:id="@+id/tv1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColor="@color/black"  
 android:textStyle="bold"/>  
  
 <LinearLayout  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content">  
  
 <Button  
 android:id="@+id/btn\_star"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.35"  
 android:text="开始"  
 android:textStyle="bold"  
 android:textColor="@color/colorPrimaryDark"  
 style="?attr/buttonBarButtonStyle"  
 />  
 <Button  
 android:id="@+id/btn\_continue"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.25"  
 android:text="继续播放"  
 android:textStyle="bold"  
 android:textColor="@color/colorPrimaryDark"  
 style="?attr/buttonBarButtonStyle"  
 />  
 <Button  
 android:id="@+id/btn\_stop"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.35"  
 android:text="暂停"  
 android:textStyle="bold"  
 android:textColor="@color/colorPrimaryDark"  
 style="?attr/buttonBarButtonStyle"  
 />  
 </LinearLayout>  
  
 <LinearLayout  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 style="?attr/buttonBarButtonStyle"  
 >  
 <Button  
 android:id="@+id/btn\_last"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.33"  
 android:text="上一首"  
 android:textStyle="bold"  
 android:textColor="@color/DarkBlue"  
 style="?attr/buttonBarButtonStyle"  
 />  
 <Button  
 android:id="@+id/refresh"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.34"  
 android:text="刷新"  
 android:textStyle="bold"  
 android:textColor="@color/DarkBlue"  
 style="?attr/buttonBarButtonStyle"  
 />  
 <Button  
 android:id="@+id/btn\_next"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.33"  
 android:text="下一首"  
 android:textStyle="bold"  
 android:textColor="@color/DarkBlue"  
 style="?attr/buttonBarButtonStyle"  
 />  
  
 </LinearLayout>  
  
</LinearLayout>

实验截图：



仓库地址：

<https://github.com/Minakami-Yuuki/Assignments.git>

1. 实验总结：

注：对实验中遇到的问题和所采取的解决方案做简要总结

# 说明：若发现雷同的实验报告，均以0分论处。