COMP3040 Coursework 1-2 MP3Player

Summary

In this exercise you are required to build an Android MP3 player application. This is an assessed exercise and will account for **10% of your final module mark**. This is an individual coursework, and your submission must be entirely your own work – please pay particular attention to the section of this document regarding plagiarism. This document sets out general requirements and broad instructions for developing the application.

Your application should be submitted no later than:

- **5pm on Monday the 12th of November 2018**

Submissions should be made electronically via Moodle. Standard penalties of 5% per working day will be applied to late submissions.

Your application should be submitted as a .zip or .tar.gz file containing all relevant source code, configuration and related files, and a compiled .apk file – i.e. the contents of the directory containing your Android Studio project. Do not submit RAR files.

Specification

You should create an application with the functionality of a simple music player for Android, which allows users to select from a number of music files stored on the SD card storage of the device to be played, and allows the music to continue to play in the background while the user performs other tasks.

At minimum your application should support:

- An *Activity* presenting an interface for the user that:
  - Allows the user to select a music file to play from the /sdcard/Music folder
  - Allows the user to stop or pause playback
  - Displays the current progress of the playback
- A *Service* that provides continued playback in the background
- A *Notification* that allows the user to return to the *Activity*

You must implement a **Service** to handle the music-playing element of the application, as this is a long-running task and the user can be expected to leave the initial activity to perform other tasks. You should think carefully about the relationship between the Activity and Service in your application, and how these should be used appropriately to perform the task. There is **no requirement** that your service will be used remotely, i.e. you do not need to use *AIDL*.

A simple MP3Player class is provided that wraps a basic MediaPlayer object for loading and playing an MP3 file. It is left up to you to decide how best to design and implement Activities for selecting and controlling the music playback.

Your application must be written in Java and make use of the Android SDK. There are no requirements to target a specific Android API version, however you should assume that your application would be tested on an emulated Nexus 6 device (1440x2560 560dpi) running Android API version 25 (Android 7.1.1 Nougat).

You should consider the following when implementing your application:

- Decomposition of the task into logical, discrete Activity components.
- Appropriate use of Activities, Intents and appreciation of the Activity life- cycle
- Appropriate use of Widgets and ViewGroups for layouts that support devices of differing screen sizes and resolutions
- Appropriate use of Services, Notifications and appreciation of the Service life-cycle
- Your application should have appropriate comments and variable / class names, so that a reader can easily understand how it works at the code level

Adding further additional functionality to the application is encouraged, but please note that this coursework will be primarily assessed on the above specification, and further additions must not detract from the required functionality given above. You can extend MP3Player.java as you see fit.

Plagiarism

N.B. Use of third party assets (tutorials, images, example code, libraries etc.) MUST be credited and referenced, and you MUST be able to demonstrate that they are available under a license that allows their reuse.

Making significant use of tutorial code while referencing it is poor academic practice, and will result in a lower mark that reflects the significance of your own original contribution.

Copying code from other students, from previous students, from any other source, or soliciting code from online sources and submitting it as your own is plagiarism and will be penalized as such. FAILING TO ATTRIBUTE a source will result in a mark of zero – and can potentially result in failure of coursework, module or degree.

All submissions are checked using both plagiarism detection software and manually for signs of cheating. If you have any doubts, then please ask.

Assessment Criteria

	Marks Available
Basic Application Functionality	
The application allows the user to play music files	20
The application runs appropriately in the background, and stops appropriately	20
Application Structure and Implementation	
Activities appropriately support the task	5
Appropriate use of Views and Layouts	5
Appropriate use of a Service	15
Handling of the Service lifecycle	10
Appropriate use of Notifications	5
Programming style	
The application is easy to understand, with comments explaining each part of the code, correct formatting, and meaningful variable names	10
Extension	
Appropriate additional functionality not in the specification	10
Total	100

Each element of your coursework will be assessed against the standard criteria (please see the Marking Criteria document in Moodle).

The following areas will be taken into account for each part of the assessment:

- Demonstrating knowledge of the area
- Quality of the concept, including appropriateness and novelty
- Quality of the technological design, including appropriate use of software design concepts, and appropriate good coding practice (abstraction, commenting, naming)
- Quality of the realization, including how well it works and elaborations over and above the basic requirements
- Including all of the above aspects, clarity of structure, quality of argument / evidence, and insight / novelty

Instructions

Note that this coursework aims to serve as an assessment of the Services material covered in lectures and used in Lab Exercise 4 – as such try to think about how to make use of the concepts covered in those. It is an incorrect approach to start by googling “how to build an mp3 player app”

MP3Player

Begin by creating a new application in Android Studio as usual.

Add the class MP3Player.java to your app project. This class is available on Moodle.

You can either copy the file directly into your project's source directory (*projectname/app/src/main/java/…*) or create a new MP3Player Java class in your project and copy / paste the code into it, updating the package qualifier accordingly.

MP3Player is a simplistic wrapper for an Android MediaPlayer object. MediaPlayer has its own internal thread for actually doing the work of playing an MP3, so there is no need to spawn a new thread to contain it. There is also no need to asynchronously load an MP3 in a separate thread. It is worth adding the MP3Player class directly to an Activity and controlling it directly with buttons to make sure you understand how it works before moving it into a Service.

The MP3Player has a *state* variable that reflects the stateful nature of the underlying MediaPlayer, and getState() will return one of the following:

```
public enum MP3PlayerState {
ERROR,
PLAYING,
PAUSED,
STOPPED
}
```

The MP3Player begins in the *STOPPED* state on instantiation.

The class has a few simple methods for loading and playing an MP3:

```
public void load(String filePath)
```

…attempts to load and play the file specified by *filePath*. If all goes well the music will start playing and the MP3Player will now be in the *PLAYING* state. If something went wrong – usually if the file is not found, or is not a playable type, the MP3Player will be in the *ERROR* state.

The music can be paused or unpaused when playing:

```
public void play()
public void pause()
```

Or finally stopped:

```
public void stop()
```

Note that stopping releases and cleans up the MediaPlayer, so the MP3 must be loaded again from the beginning if it needs to start playing again.
The duration and current position of the MP3 can be queried, in milliseconds, and so can the current filename:

```
        public int getDuration()
        public int getProgress()
        public String getFilePath()
```

Files

To transfer mp3 files onto the sdcard of the emulator, you can either use the Android Device Monitor from within Android Studio.

If you don't have access to any music in mp3 format, a variety of royalty free / creative commons licensed files are available here http://freemusicarchive.org/

The Android permissions system by default prevents the application from reading from the sdcard, so you will need to add the READ_EXTERNAL_STORAGE permission to the manifest as shown below. Depending on the emulator API version you are using you will also need to enable this permission from within the phone settings via
Settings->Apps->**My MP3Player**->Permissions->Storage

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
```

The code below handles much of the work of listing files in the /Music/ directory and populating a ListView lv with the resultant list. onItemClick is called when one of the entries in the list is selected by the user. You are free to extend this code as you wish. Note that there is no requirement to recurse into directories or handle directories in any way.

```
import android.os.Environment;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import java.io.File;

final ListView lv = (ListView) findViewById(R.id.listView);

File musicDir = new File(
        Environment.getExternalStorageDirectory().getPath()+ "/Music/");

File list[] = musicDir.listFiles();

lv.setAdapter(new ArrayAdapter<File>(this,android.R.layout.simple_list_item_1, list));

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> myAdapter, View myView, int myItemInt,
        long mylng) {
                File selectedFromList =(File) (lv.getItemAtPosition(myItemInt));
                Log.d("g53mdp", selectedFromList.getAbsolutePath());
                // do something with selectedFromList...
        }
});
```

Service and Notification

The main element of this coursework is to implement a Service component that contains an instance of the MP3Player class, and to control this Service from your main Activity component.

With the previous exercises in mind, you should think about:

- How should the Service be *started*
- How should the Activity tell the Service what to do (play, pause etc)
- How can the Activity know what the Service is doing (display progress etc)
- When should the Servce be *stopped*

The MP3Player class and the view code above should give you a starting point for developing a Service and an Activity respectively.

Note that you should explicitly design your Service so that it continues running when the Activity component has been *destroyed*, and when the Activity is recreated it should consistently regain control of the Service.

Your Service should maintain a *Notification* while it is running, to allow the user to return to the Activity even after the Activity has been destroyed via a Pending Intent.

References

http://developer.android.com/guide/topics/ui/declaring-layout.html

http://developer.android.com/guide/topics/ui/controls.html

http://developer.android.com/guide/components/activities.html

http://developer.android.com/guide/components/services.html

http://developer.android.com/guide/topics/ui/notifiers/notifications.html