Module 4

Graphics and animation, Multimedia

Playing audio

MediaPlayer class

•One of the most important components of the media framework is the MediaPlayer class.

• An object of this class can fetch, decode, and play both audio and video with minimal setup.

• It supports several different media sources such as:

- Local resources
- Internal URIs, such as one you might obtain from a Content Resolver
- External URLs (streaming)

Methods of MediaPlayer Class

- public void setDataSource(String path) : sets the data source (file path or http url) to use.
- public void prepare() : prepares the player for playback synchronously.
- public void start() : it starts or resumes the playback.
- public void stop() : it stops the playback.
- public void pause() : it pauses the playback.
- public boolean isPlaying() : checks if media player is playing.
- public void seekTo(int millis) : seeks to specified time in milliseconds.

Methods of MediaPlayer Class

- public void setLooping(boolean looping) : sets the player for looping or non-looping.
- public boolean isLooping() : checks if the player is looping or non-looping.
- public void selectTrack(int index) : it selects a track for the specified index.
- public int getCurrentPosition() : returns the current playback position.
- public int getDuration() : returns duration of the file.
- public void setVolume(float leftVolume,float rightVolume) :sets the volume on this player

•How to play audio that's available as a local raw resource (saved in your application's res/raw/ directory):

•MediaPlayer mediaPlayer =

MediaPlayer.create(.MainActivity.this,R.raw.sound_file_1);

mediaPlayer.start();

• no need to call prepare(); create() does that for you

•If you want from Uri available locally in the system (that you obtained through a Content Resolver, for instance):

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioAttributes(
                                    new AudioAttributes.Builder()
                                   .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
                                   .setUsage(AudioAttributes.USAGE_MEDIA)
                                   .build()
                            );
                            mediaPlayer.setDataSource(getApplicationContext(), myUri);
                            mediaPlayer.prepare();
                            mediaPlayer.start();
```

- **Playing from a remote URL via HTTP streaming looks like this:**

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioAttributes(
    new AudioAttributes.Builder()
        .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
        .setUsage(AudioAttributes.USAGE_MEDIA)
        .build()
);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Releasing the MediaPlayer

- A MediaPlayer can consume valuable system resources. Therefore, you should always take extra precautions to make sure you are not hanging on to a MediaPlayer instance longer than necessary. When you are done with it, you should always call release() to make sure any system resources allocated to it are properly released.
- For example, if you are using a MediaPlayer and your activity receives a call to onStop(), you must release the MediaPlayer, because it makes little sense to hold on to it while your activity is not interacting with the user (unless you are playing media in the background).
- When your activity is resumed or restarted, of course, you need to create a new MediaPlayer and prepare it again before resuming playback.

```
mediaPlayer.release()

mediaPlayer = null;
```

- **Write an android application to play,pause and stop an audio file.**

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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:background="@drawable/music"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="bottom|center_horizontal"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnPlay"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:backgroundTint="#C6A8A8"
            android:text="Play"
            android:textColor="#01579B" />

        <Button
        android:id="@+id/btnPause"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
        android:layout_margin="10dp"

        android:backgroundTint="#C6A8A8“
        android:text="Pause"
        android:textColor="#01579B" />
         <Button
        android:id="@+id/btnStop"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
        android:layout_margin="10dp"

        android:backgroundTint="#C6A8A8"
        android:text="Stop"
        android:textColor="#01579B" />
        </LinearLayout>

        </RelativeLayout>
```

MainActivity.java

```java
package com.example.audio;

import androidx.appcompat.app.AppCompatActivity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {
    Button play,pause,stop;
    MediaPlayer mp;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        play=findViewById(R.id.btnPlay);
        pause=findViewById(R.id.btnPause);
        stop=findViewById(R.id.btnStop);
        play.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(mp==null)
                {
                    mp=MediaPlayer.create(getApplicationContext(),R.raw.song);
                } mp.start();
            }
        });
```

MainActivity.java contd..

```
pause.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(mp!=null)
                {
                    mp.pause();
                }
            }
        });
        stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(mp!=null)
                {
                    mp.release();
                    mp=null;
                }
            }
        });
    }
}
```