UWS UNIVERSITY OF THE WEST of SCOTLAND

13954 - COMP 11062: Mobile Networks And Smartphone Applications

Integration of Multimedia in Android App

Section Agenda

- What is multimedia and how can be used in Android App?
- MediaPlayer Class
- Design and develop App for Playing your favorite song
    - Draw wireframe
    - Design & technical constraints
- Design UI in IDE
- Edit properties of inputs
- Build up Java file

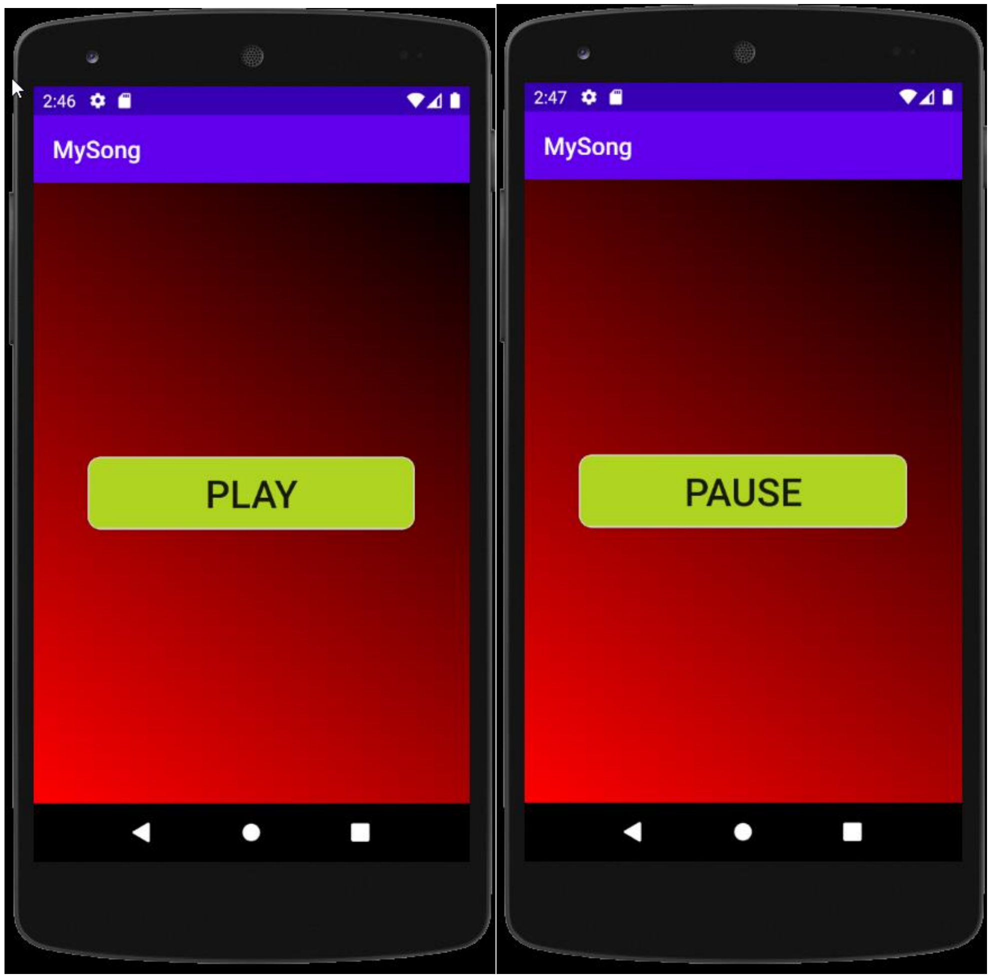
What is Multimedia ?

- Multimedia involves
  - Songs mp3, mp4 etc.
  - Photos
  - Videos

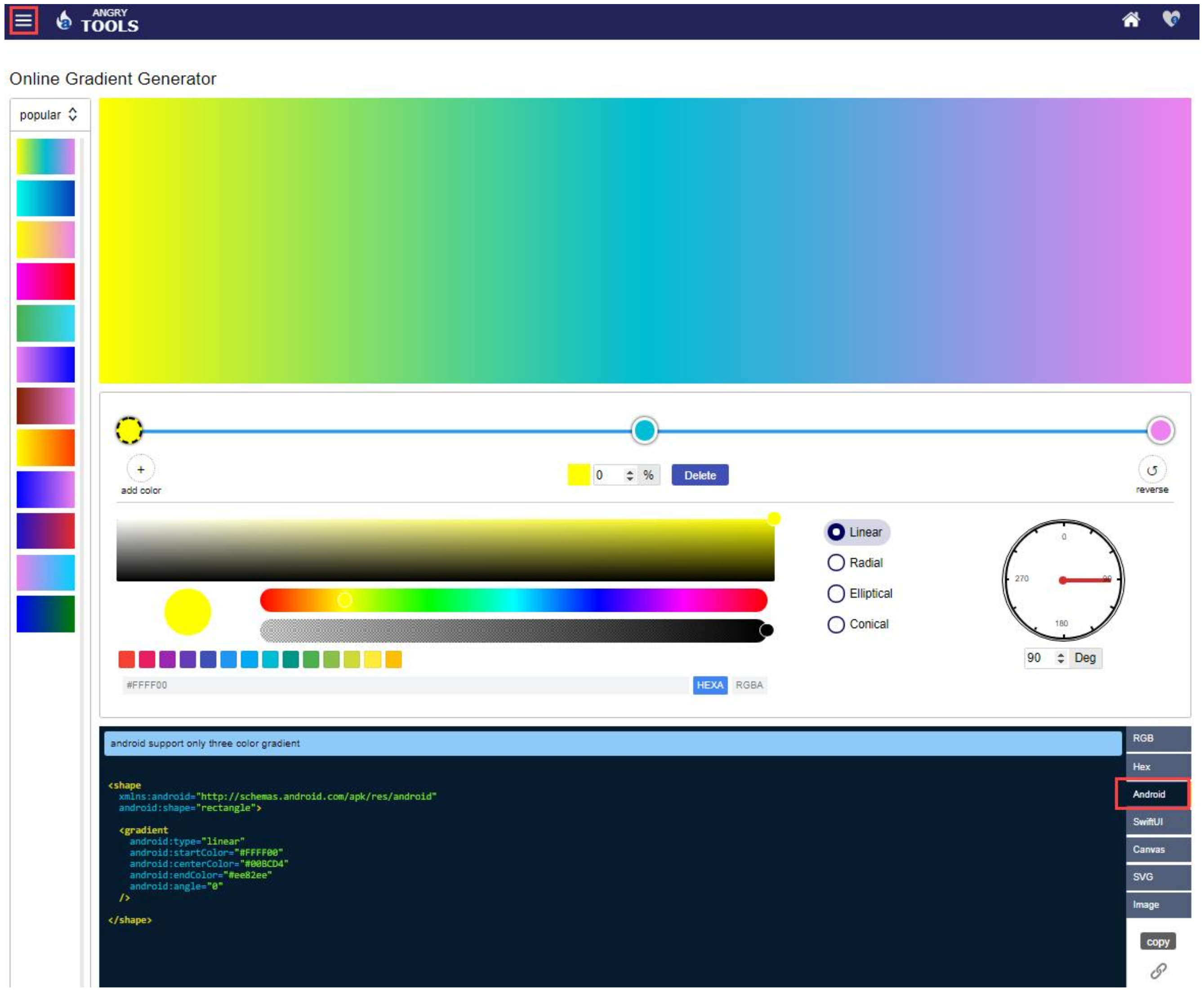
Wireframe

- UI layout
  - Widgets requirements: Button or Buttons
  - Some color may be matched with the song genre
- Technical requirements
  - Toggling between Play and Pause functions
  - Display song duration
  - Any other requirements

App Output



Gradient Background/Button Shape/Image

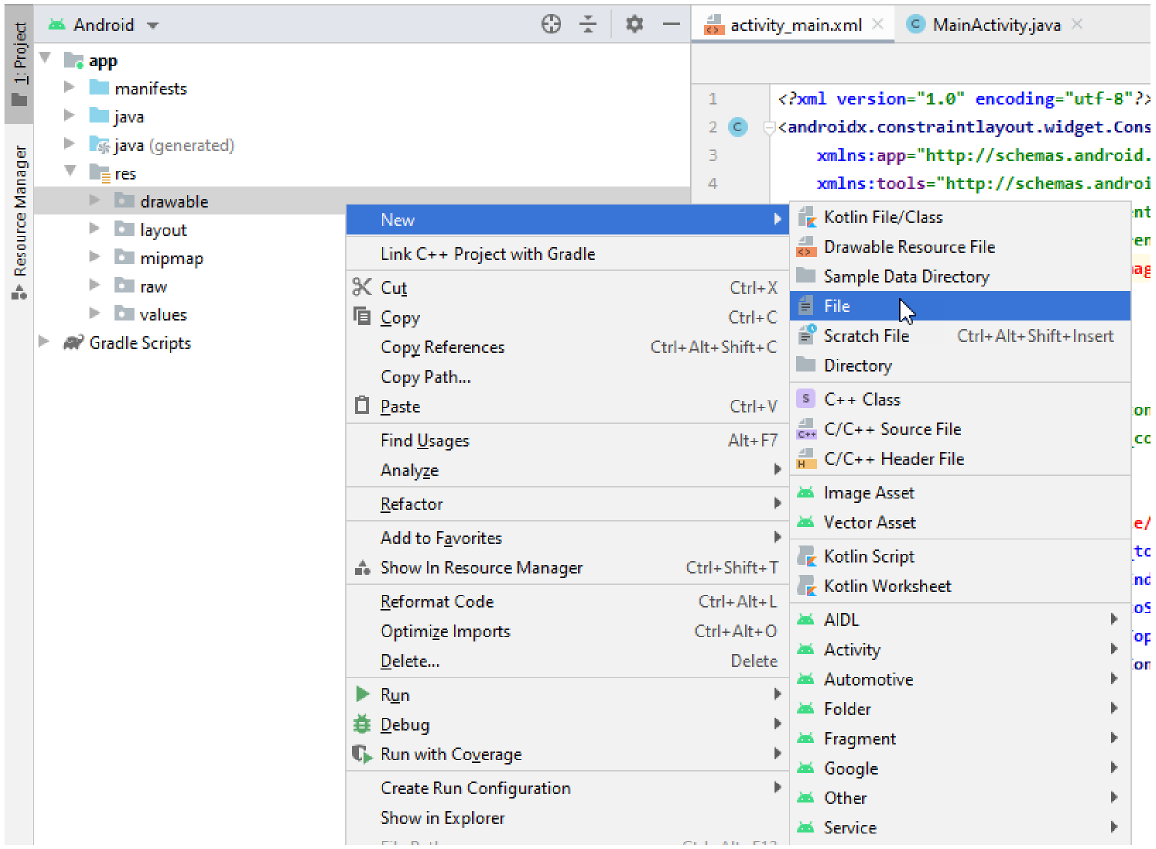


android support only three color gradient

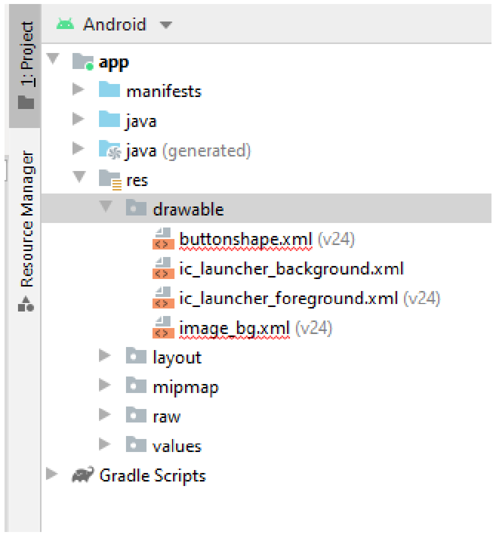
```
<shape
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:shape="rectangle">

  <gradient
    android:type="linear"
    android:startColor="#FFFF00"
    android:centerColor="#00BCD4"
    android:endColor="#ee82ee"
    android:angle="0"
  />

</shape>
```

Gradient Background/Button Shape/Image

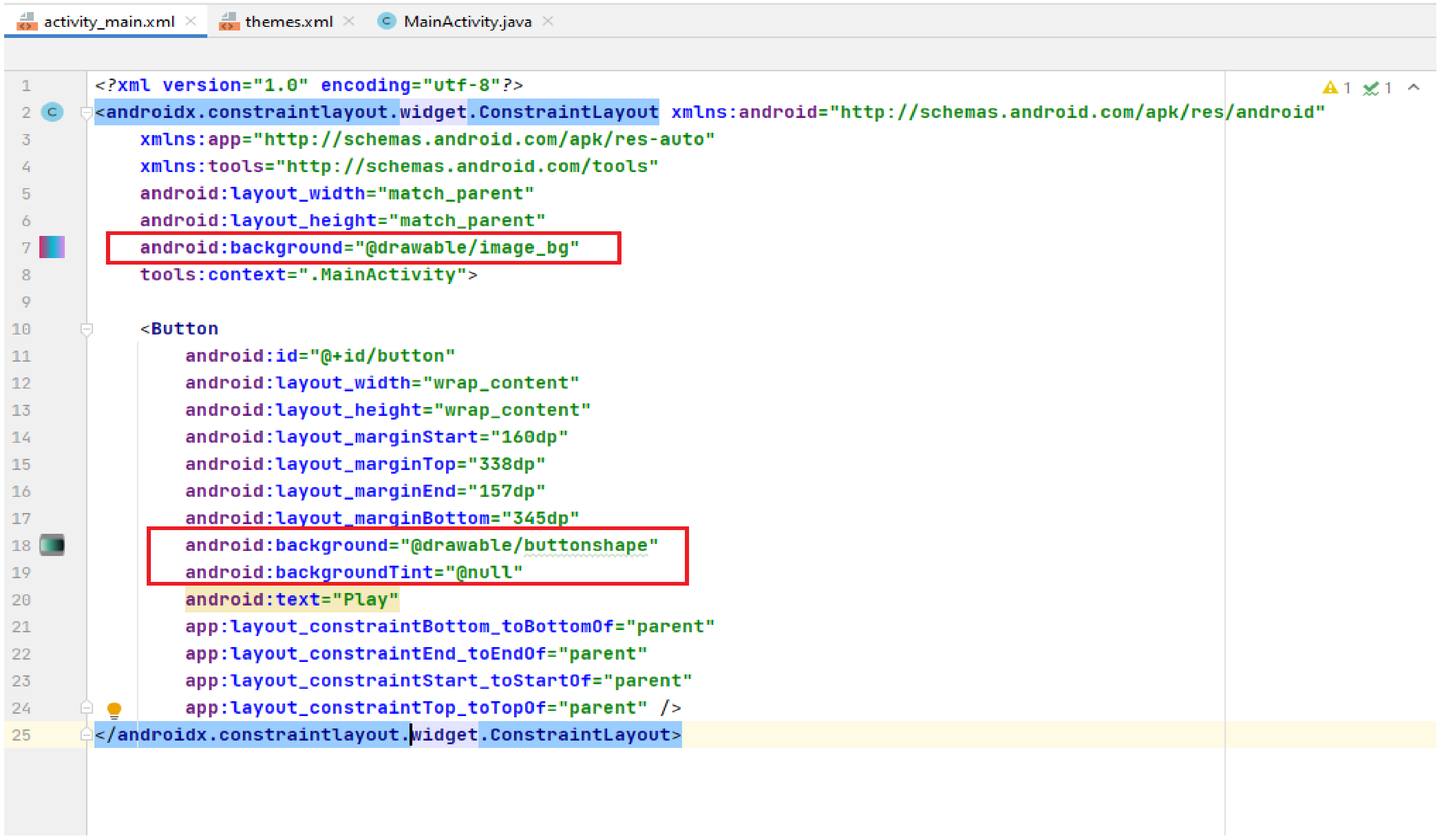
- To Create Gradient Background, we will use the site called http://angrytools.com/gradient/.
- This site will help us to create our background and buttons easily and generate the android code.
- After designing our background image, we will copy the android code. Now create a new file under drawable folder and name it image_bg.xml or image_bg with type xml, then paste the appropriate code.
- We will follow the similar approach for the button with name buttonshape.xml. It is to be noted that the button can be designed using the same tool. Click the three straight line top left of the website then select Android button maker. You will see available tools to design your button.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.Cons
    xmlns:app="http://schemas.android.
    xmlns:tools="http://schemas.androi
```

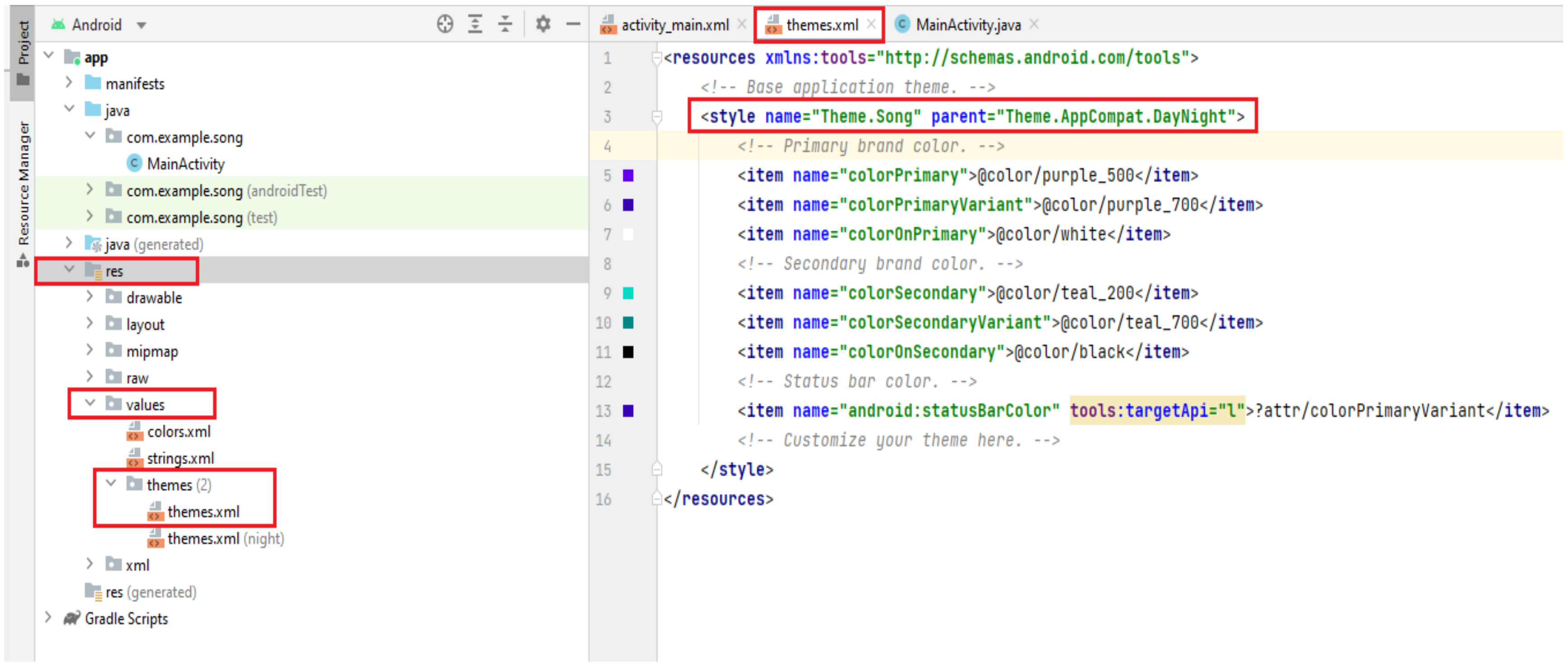


UI Properties



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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/image_bg"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="160dp"
        android:layout_marginTop="338dp"
        android:layout_marginEnd="157dp"
        android:layout_marginBottom="345dp"
        android:background="@drawable/buttonshape"
        android:backgroundTint="@null"
        android:text="Play"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

To see the button shape correctly, we need to change the theme of the app. On the project view of the directories go to res-> values->themes and open themes.xml. Replace the theme name as shown in the picture.



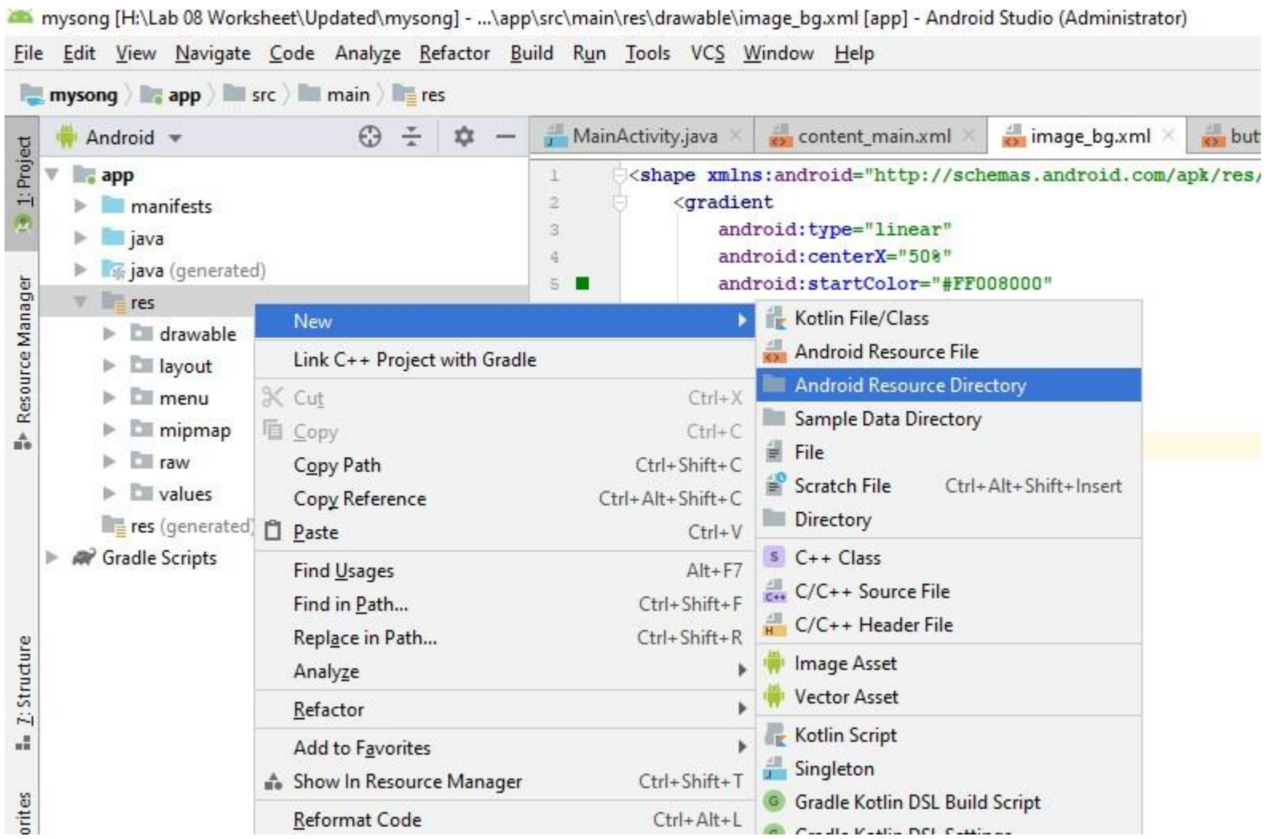
```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.Song" parent="Theme.AppCompat.DayNight">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Media Player

- Android multimedia framework includes support for playing variety of media content/files such as audio, video and images.
- MediaPalyer Class is the most important component of the media framework in Android App development to fetch, decode and play both audio and video content/files.
- Files or content can be:
    - Stored in App's resources (raw resources)
    - Arrived from data stream over network connection

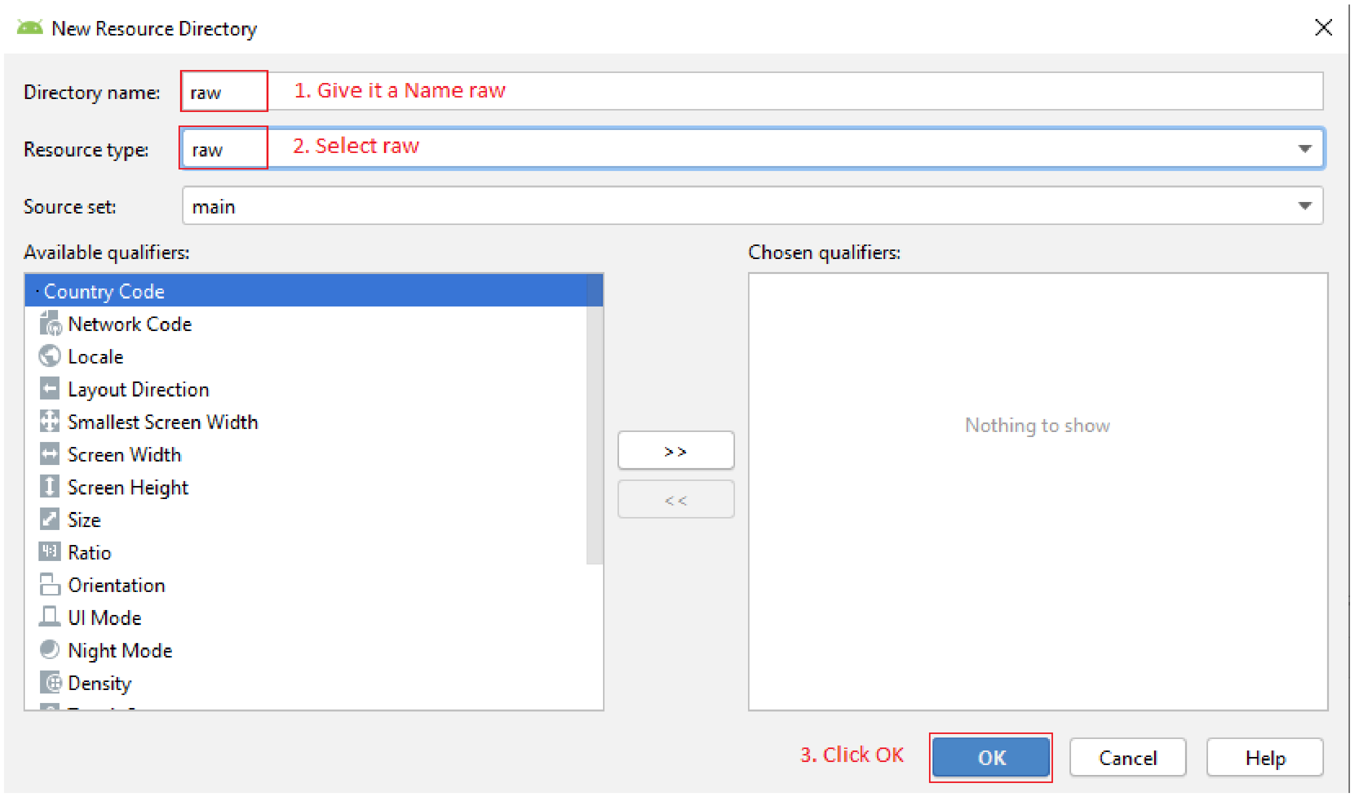
Music File

- Add the sound file into the App. Add the files to the raw folder. If there is no raw folder, then create a raw folder under res directory.



```
<shape xmlns:android="http://schemas.android.com/apk/res/
    <gradient
        android:type="linear"
        android:centerX="50%"
        android:startColor="#FF008000"
```

Music File


New Resource Directory

Directory name: raw

Resource type: raw

Source set: main

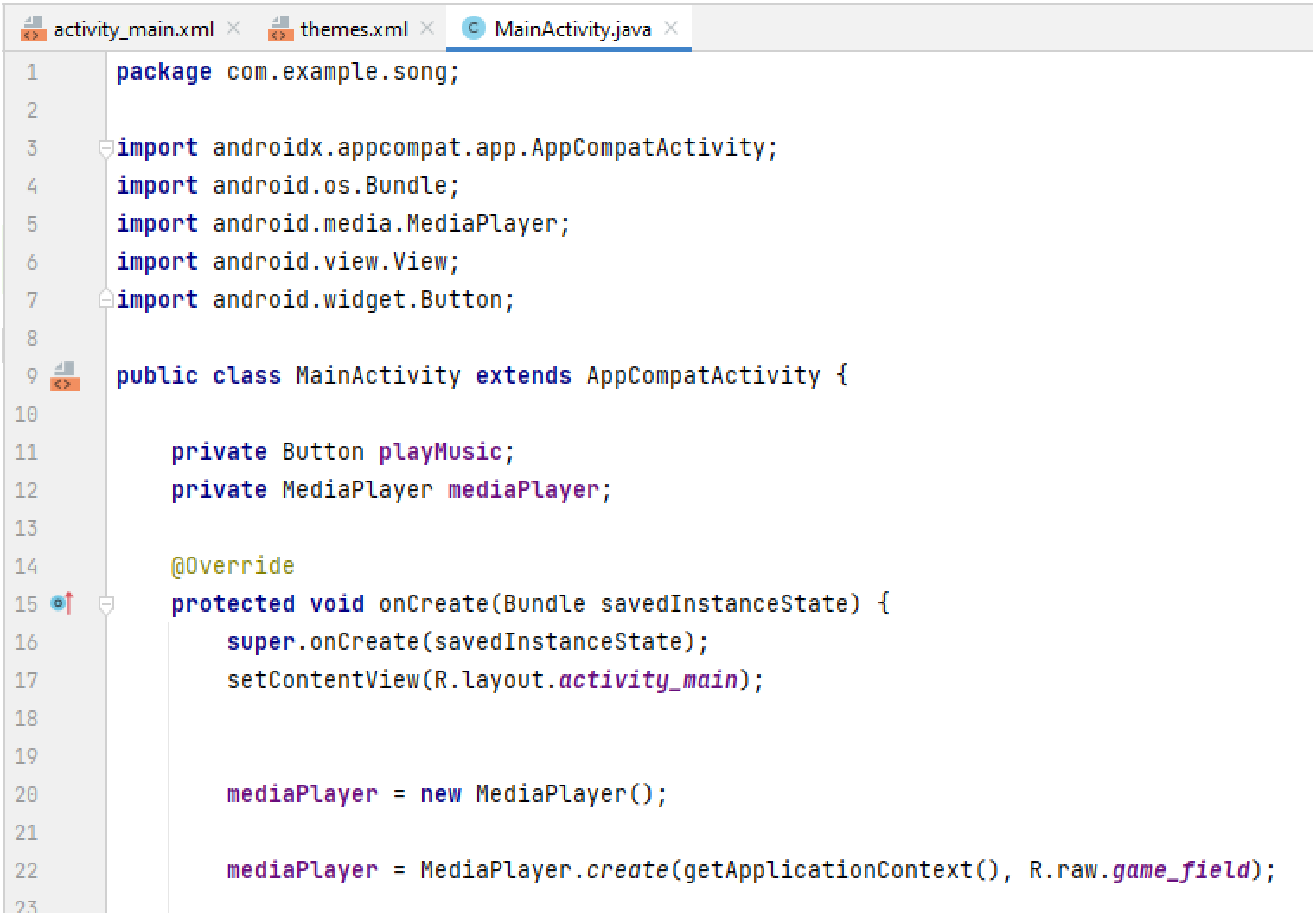
Available qualifiers: Country Code, Network Code, Locale, Layout Direction, Smallest Screen Width, Screen Width, Screen Height, Size, Ratio, Orientation, UI Mode, Night Mode, Density

Chosen qualifiers: Nothing to show

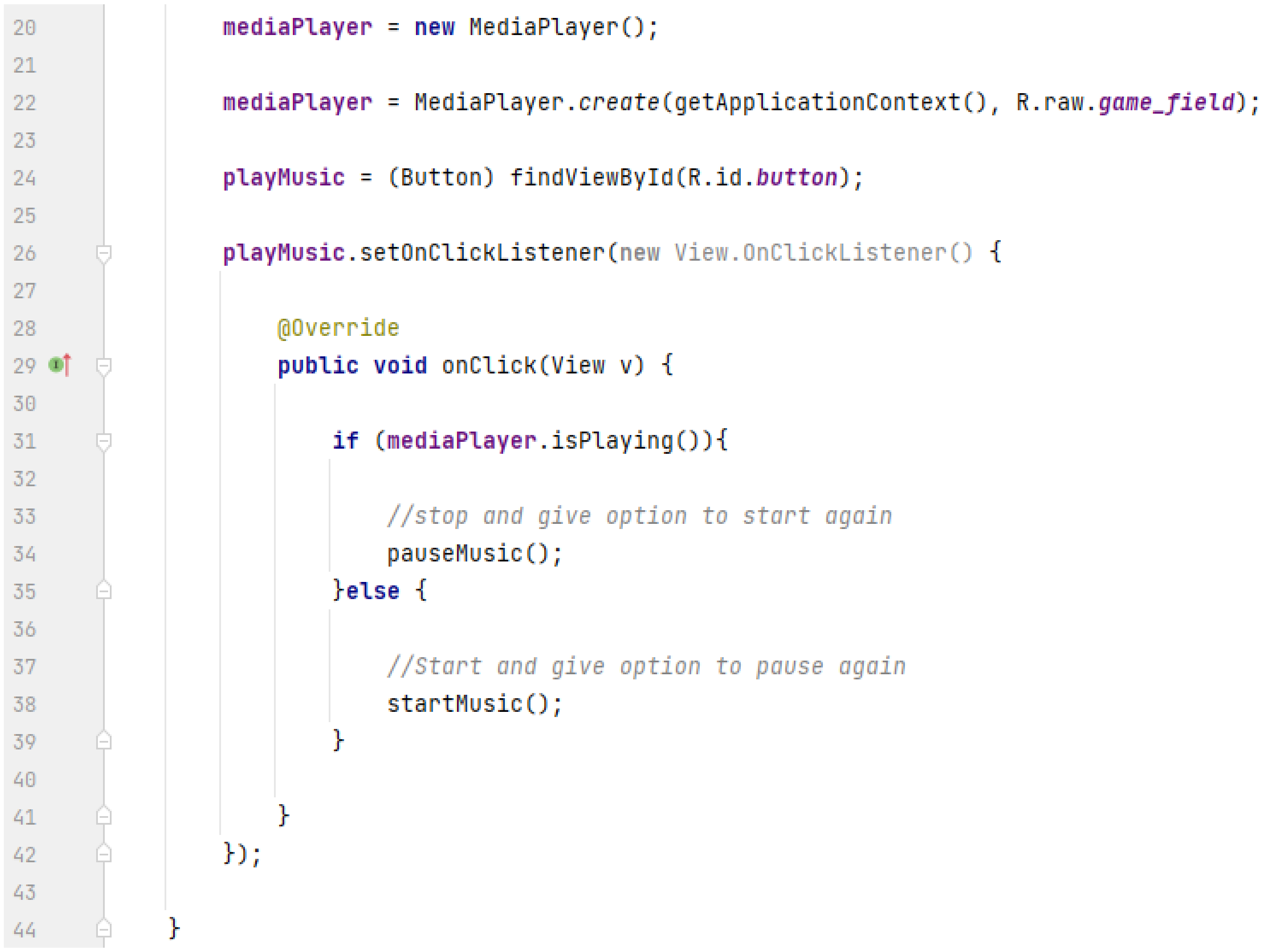
- Copy and paste the available music file (game_field.mp3) to raw the directory.

Build up Java file

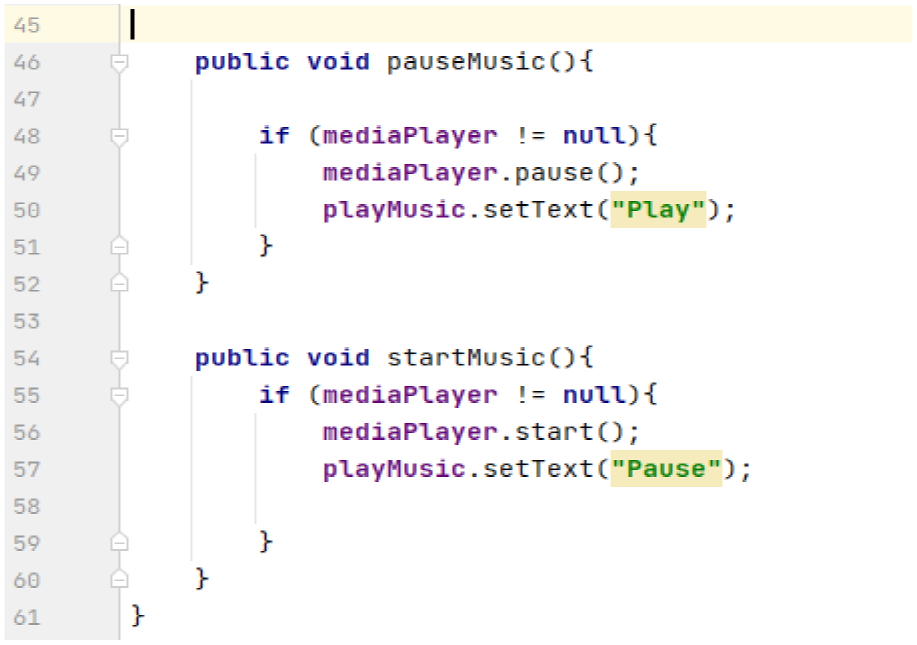
- Initialise the variables
  - Based on inputs we have in UI
  - Import any missing packages
  - Check: for any possible spelling error
- Link UI with Java
  - Map the widgets on UI plane using correct IDs
  - Create media content resource folder and media player
- Create events
  - Two functions with the click on button: Either play a song or Pause a song
  - Two methods using if-then-else



```java
package com.example.song;

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

public class MainActivity extends AppCompatActivity {

    private Button playMusic;
    private MediaPlayer mediaPlayer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        mediaPlayer = new MediaPlayer();

        mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.game_field);
```



```
        mediaPlayer = new MediaPlayer();

        mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.game_field);

        playMusic = (Button) findViewById(R.id.button);

        playMusic.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {

                if (mediaPlayer.isPlaying()){

                    //stop and give option to start again
                    pauseMusic();
                }else {

                    //Start and give option to pause again
                    startMusic();
                }

            }
        });

    }
```



```

    public void pauseMusic(){

        if (mediaPlayer != null){
            mediaPlayer.pause();
            playMusic.setText("Play");
        }
    }

    public void startMusic(){
        if (mediaPlayer != null){
            mediaPlayer.start();
            playMusic.setText("Pause");

        }
    }
}
```

Exercise

If you let the song play until the end, you will notice the text of the button stays set to “Pause”. Modify MainActivity.java to automatically set the button text to “Play” when the song ends.
Hint: Inside the startMusic() method, call setOnCompletionListener() on the mediaPlayer object!

Discussions!

- Can we play more songs ?
- Can we have duration of songs?
- Issues to take care when stream the video in App?
- Any other features a multimedia player should have?