User Interface - Part 3

- Date and Time in Android
- Toast
- AlertDialog
- Images
- Media
- Composite
- Menus

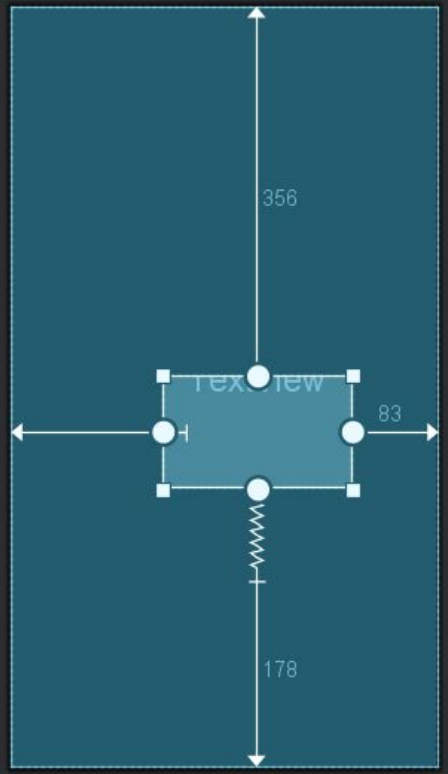
Working with Java File (Display text while running the app)

- Create New Project
- Design activity_main.xml (A single Textview with no text to display)
- Set the id of Textview as tv1
- Write the following code in the MainActivity.java

Working with Java File (Display text while running the app)

```
import android.widget.TextView;
    .
    .
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView tv=(TextView) findViewById(R.id.tv1);
        tv.setText("The Text is here");
      }
```

```java
import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        TextView tv=(TextView) findViewById(R.id.tv1);
        tv.setText(R.string.the_text_is_here);
    }
}
```

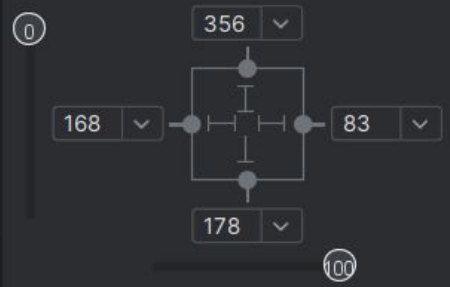


id tv1

Declared Attributes

Layout

Constraint Widget



Constraints

Start → StartOf **parent** (168dp)

End → EndOf **parent** (83dp)

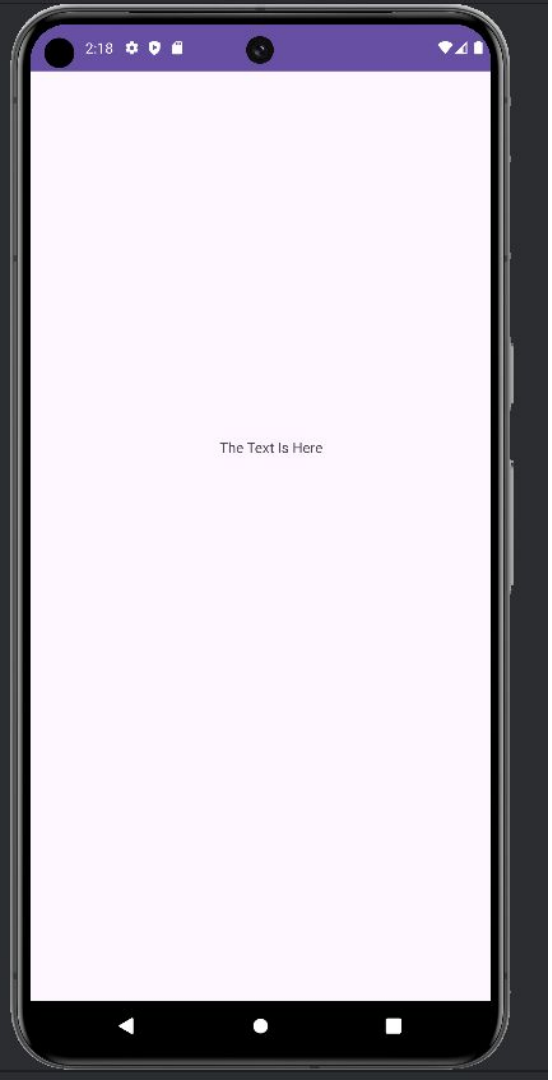
Top → TopOf **parent** (356dp)

Bottom → BottomOf **parent** (178dp)

Vertical Bias (0.0)

Horizontal Bias (1.0)

layout_width	181dp
layout_height	108dp
visibility	
visibility	

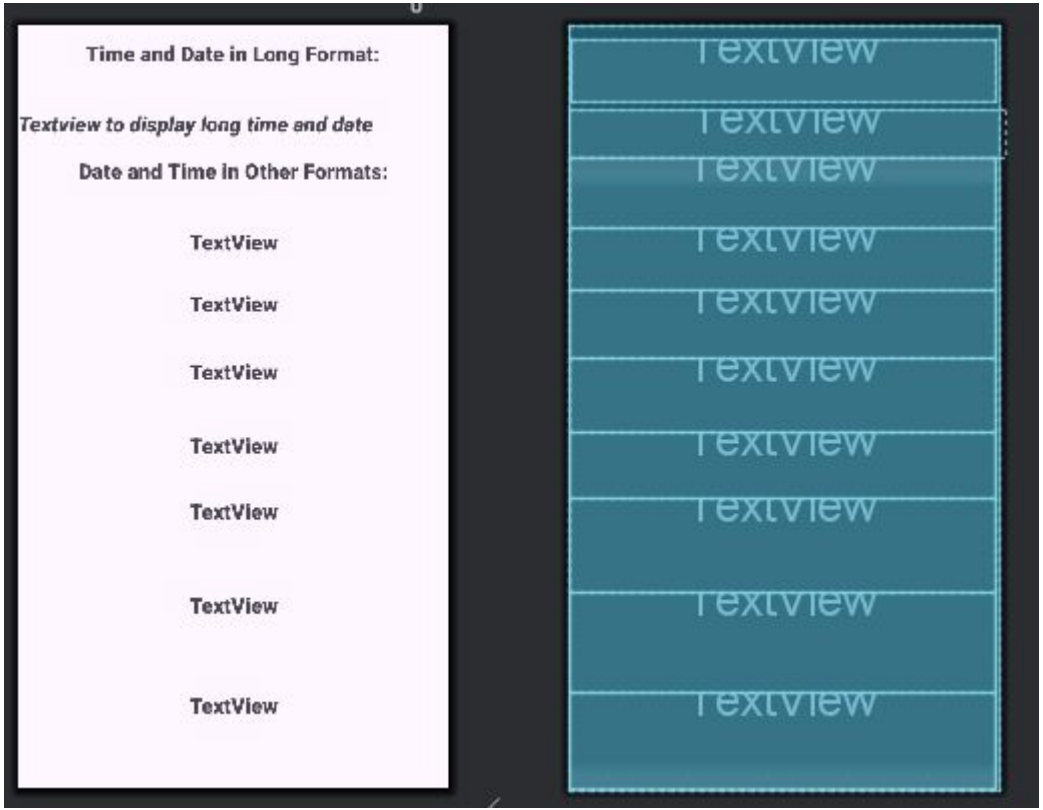


Date and Time in Android

Date and Time in Android

- Date and Time in Android are formatted using the ***SimpleDateFormat*** library from Java, using ***Calendar*** instance which helps to get the current system date and time.
- The current date and time are of the type Long which can be converted to a human-readable date and time.
- Step 1: Create an empty activity project
- Step 2: Working with the activity_main.xml file
    - The main layout of the activity file containing **10 TextViews**.
    - One to show the current system date and time value in Long type.
    - Others to display the same date and time value in a formatted human-readable way.
    - Set the ID for all Textviews

Date and Time in Android



Date and Time in Android

- Step 3: Working with the MainActivity file
  - Create an instance for the Calendar class
  - Desired format of the date and time to be shown is passed to the SimpleDateFormat method.
  - The String should include the following characters and one may include the separators like -, / etc.
  - The below table includes the characters to be used to generate the most used common pattern of date and time.

Character to be used	Output
dd	Date in numeric value
E	Day in String (short form. Ex: Mon)
EEEE	Day in String (full form. Ex: Monday)
MM	Month in numeric value
yyyy	Year in numeric value
LLL	Month in String (short form. Ex: Mar)
LLLL	Month in String (full form. Ex: March)
HH	Hour in numeric value (24hrs timing format)

Date and Time in Android

KK	Hour in numeric value (12hrs timing format)
mm	Minute in numeric value
ss	Seconds in numeric value
aaa	Displays AM or PM (according to 12hrs timing format)
z	Displays the time zone of the region

Date and Time in Android

- Add the code to MainActivity.java

```
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.text.SimpleDateFormat;
import java.util.Calendar;
```

Date and Time in Android

- Declare and initialize TextView instances inside onCreate():
  - **TextView tv1=(TextView) findViewById(R.id.tv1);**
  - Repeat for 10 Textviews with corresponding id.
- Type the remaining code as follows:

```
// get the Long type value of the current system date
Long dtlong = System.currentTimeMillis();
tv2.setText(dtlong.toString());

String dateTime;
Calendar calendar;
SimpleDateFormat simpleDateFormat;
```

Date and Time in Android

```
// format type 1
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "dd.MM.yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv4.setText(dateTime);

// format type 2
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "dd-MM-yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv5.setText(dateTime);
```

Date and Time in Android

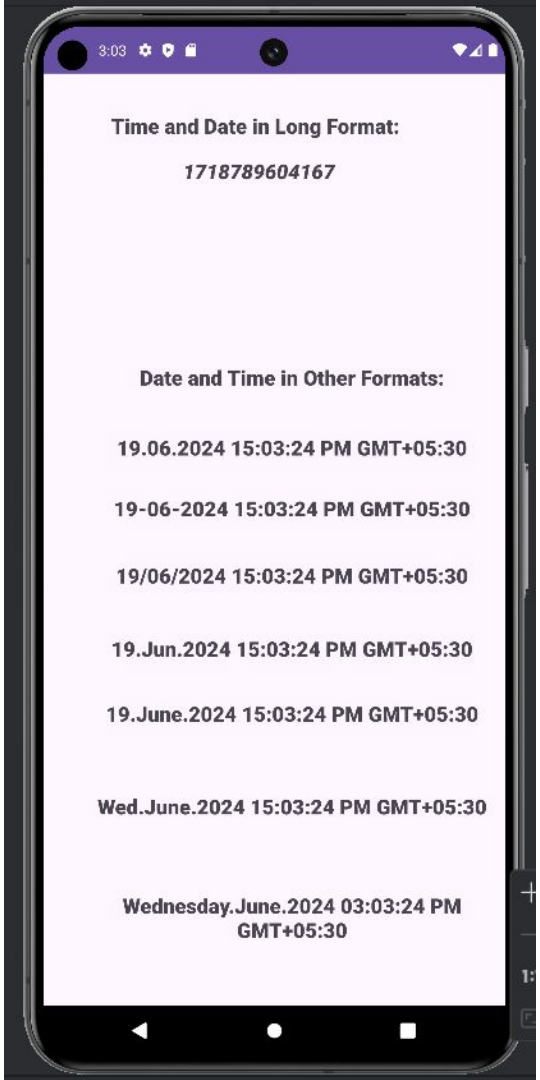
```
// format type 3
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv6.setText(dateTime);

// format type 4
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "dd.LLL.yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv7.setText(dateTime);
```

Date and Time in Android

```
// format type 5
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "dd.LLLL.yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv8.setText(dateTime);

// format type 6
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "E.LLLL.yyyy HH:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv9.setText(dateTime);
```

Date and Time in Android

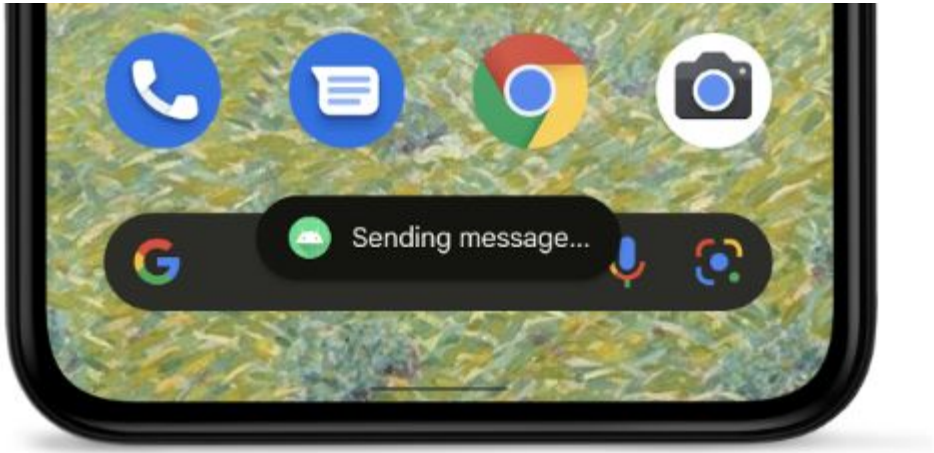
```
// format type 7
calendar = Calendar.getInstance();
simpleDateFormat = new SimpleDateFormat( pattern: "EEEE.LLLL.yyyy KK:mm:ss aaa z");
dateTime = simpleDateFormat.format(calendar.getTime()).toString();
tv10.setText(dateTime);
```


3:03
Time and Date in Long Format:
1718789604167
Date and Time in Other Formats:
19.06.2024 15:03:24 PM GMT+05:30
19-06-2024 15:03:24 PM GMT+05:30
19/06/2024 15:03:24 PM GMT+05:30
19.Jun.2024 15:03:24 PM GMT+05:30
19.June.2024 15:03:24 PM GMT+05:30
Wed.June.2024 15:03:24 PM GMT+05:30
Wednesday.June.2024 03:03:24 PM GMT+05:30

Toast

Toast in Android

- A toast provides simple feedback about an operation in a small popup.
- It only fills the amount of space required for the message and the current activity remains visible and interactive.
- Toasts automatically disappear after a timeout.



Toast in Android

- **import android.widget.Toast;**
- Add the following code to onCreate() of MainActivity.java:

  Toast.makeText(MainActivity.this, "Activity Launched successfully", Toast.LENGTH_SHORT).show();

  Syntax: *Toast.makeText(context, message,duration).show();*

Alert Dialogs

Alert Dialog

- Alert Dialog shows the Alert message and gives the answer in the form of yes or no.
- Alert Dialog displays the message to warn you and then according to your response, the next step is processed.
- Alert Dialog code has three methods:
  - setTitle() method for displaying the Alert Dialog box Title
  - setMessage() method for displaying the message
  - setIcon() method is used to set the icon on the Alert dialog box.
- Then we add the two Buttons, setPositiveButton and setNegativeButton to our Alert Dialog Box.

MainActivity.java

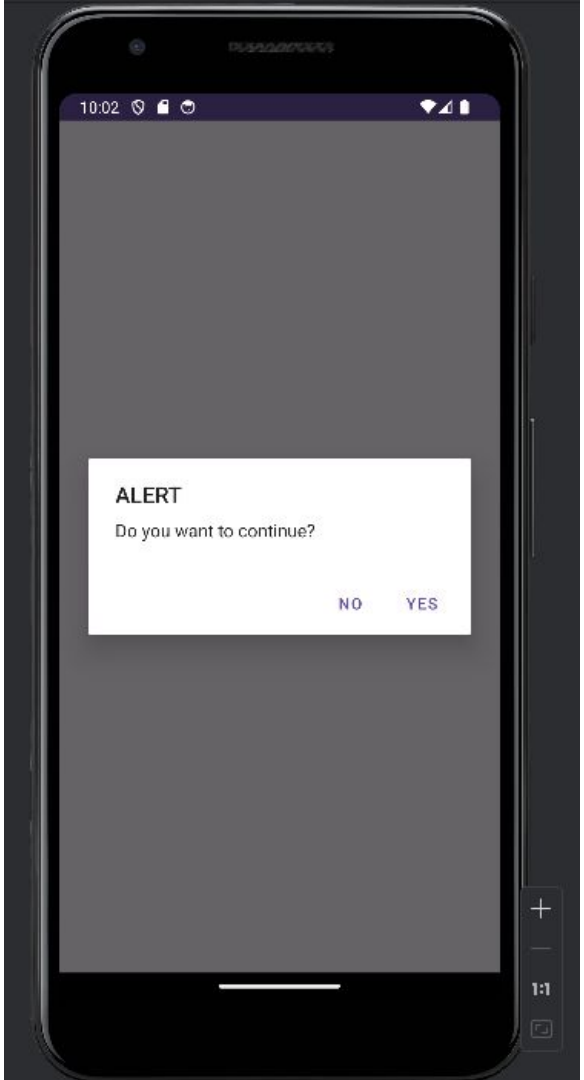
```java
package com.example.alertdialogdemo;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;

//import the following packages
import android.app.AlertDialog;
import android.content.DialogInterface;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

MainActivity.java

```
//Shows a dialogue box when the app is launched everytime
AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);

//set title and message to display when the dialogue box appears
builder.setTitle("ALERT").setMessage("Do you want to continue?");

// Set Cancelable true: when the user clicks on outside the Dialog Box then it will disappear
//Set Cancelable false: when the user clicks on outside the Dialog Box then it will remain visible
builder.setCancelable(true);
```

MainActivity.java

```
        // Add the buttons, display "YES" and "NO"
        builder.setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // User taps OK button.
                //continue in the screen, dialogue box cancels
                dialog.cancel();
            }
        });
        builder.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                // User cancels the dialog.
                // GO BACK
                finish();
            }
        });

        // Create the AlertDialog.
        AlertDialog dialog = builder.create();
        builder.show();
    }
}
```


10:02
ALERT
Do you want to continue?
NO
YES
1:1

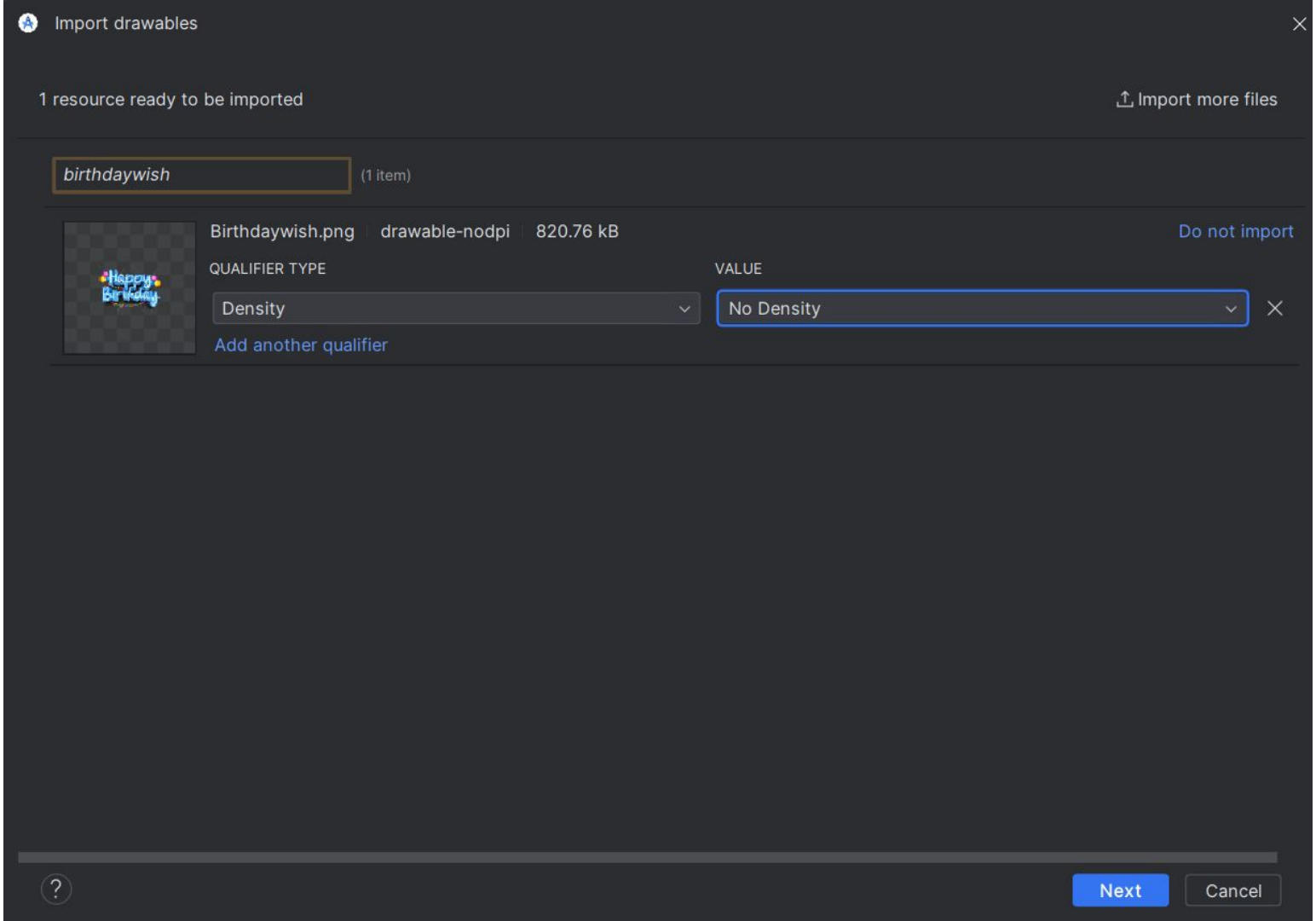
Images

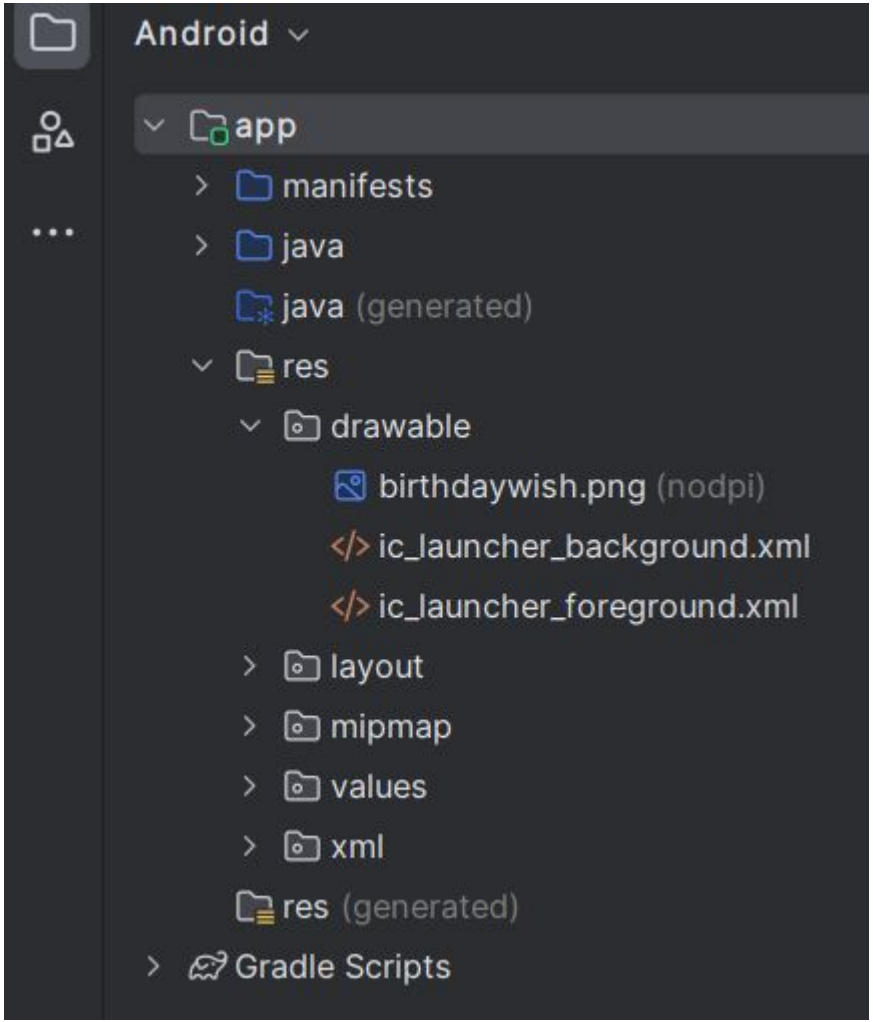
How to add an image to your android app (Sample Greeting Card with an image)

- Create a new project in android studio
- Design the UI as per your requirements
- Download the image you want to add to the app and save it in a location.
- Add the image to the Android studio
  - Tools -> Resource Manager
  - Click on the “+” sign (Add resources to the module)
  - Click on “Import Drawables”
  - Select your image from the location
  - Click on Ok

How to add an image to your android app (Sample Greeting Card with an image)

- Select “Density” from the QUALIFIER TYPE and “No Density” in VALUE
- Click on Next -> Import
- Find the imported image from res-> drawable


Import drawables
1 resource ready to be imported
Import more files
birthdaywish
(1 item)
Birthdaywish.png
drawable-nodpi
820.76 kB
Do not import
QUALIFIER TYPE
VALUE
Density
No Density
Add another qualifier
Next
Cancel


Android
app
manifests
java
java (generated)
res
drawable
birthdaywish.png (nodpi)
ic_launcher_background.xml
ic_launcher_foreground.xml
layout
mipmap
values
xml
res (generated)
Gradle Scripts

How to add an image to your android app (Sample Greeting Card with an image)

- Add an ImageView from the Palette
- Select the image to add to the ImageView
- Arrange the image according to your requirements
- Run the application

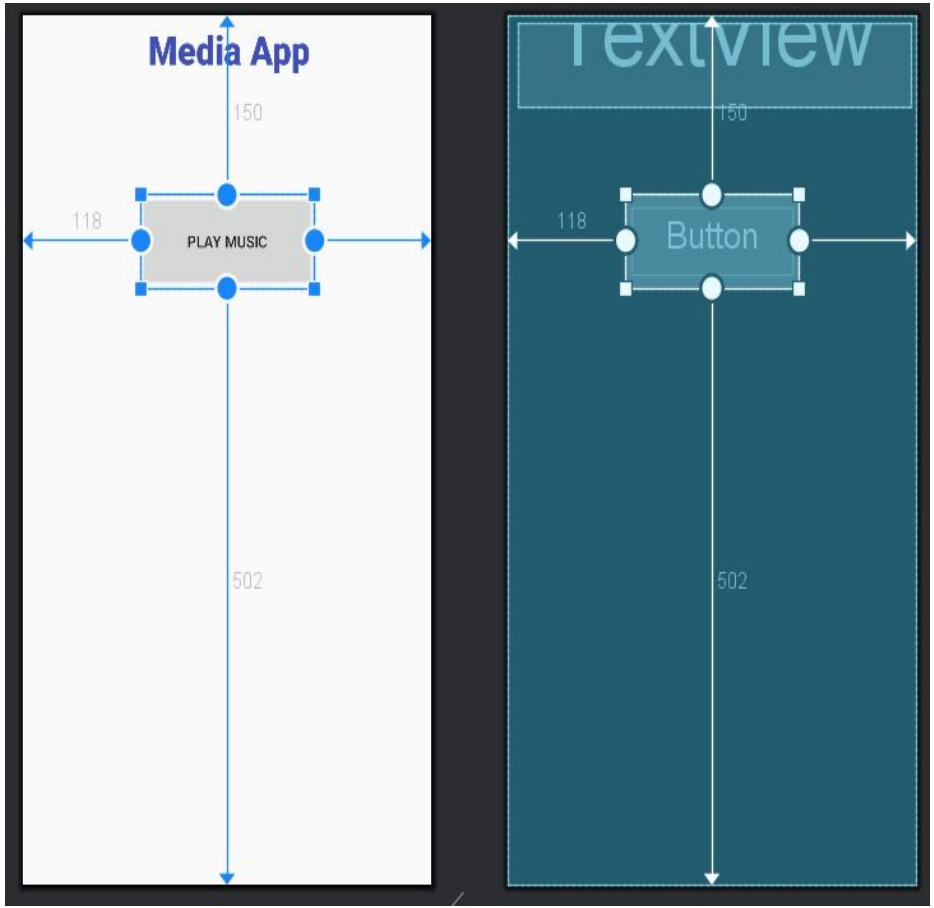

Happy Birthday
Happy Birthday Sunu Fathima
210
ImageView
TextView
210


3:06
Happy Birthday
Happy Birthday Sunu Fathima

Media (Audio & Video)

Add audio to the app

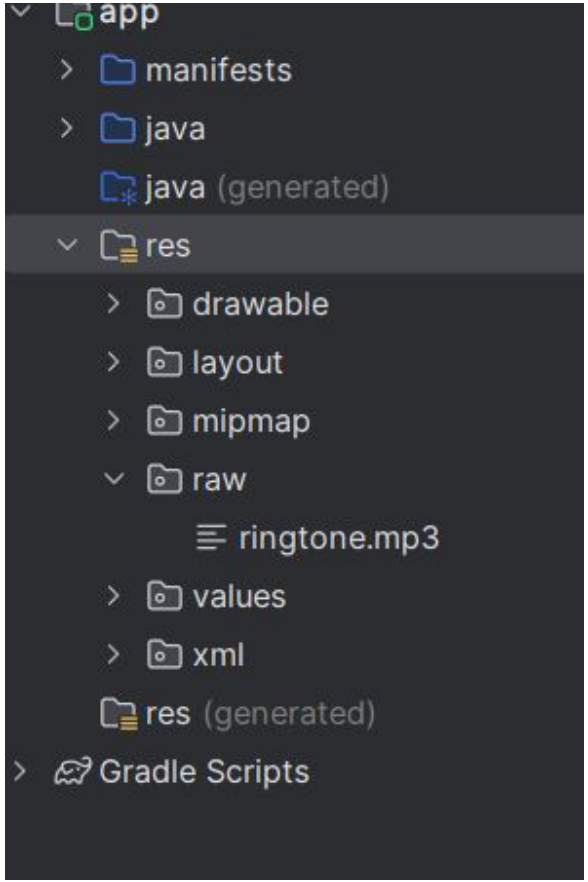
- Create a new project and design the interface.
- Add a Button and an optional TextView.
- Display “PLAY MUSIC” as the text on the Button
- Download audio file and save it in any location on your device (with lowercase letters)



Add audio to the app

- Right click on the res folder
- Click on New-> Directory
- Give the name of the folder as “raw”
- Click on the audio file, copy - paste to the raw folder
- Edit the MainActivity.java file as follows:

```
//Import the following class

import android.view.View;

import android.media.MediaPlayer;
```



Add audio to the app

- Add the new function “PlayMusic” after onCreate() method:

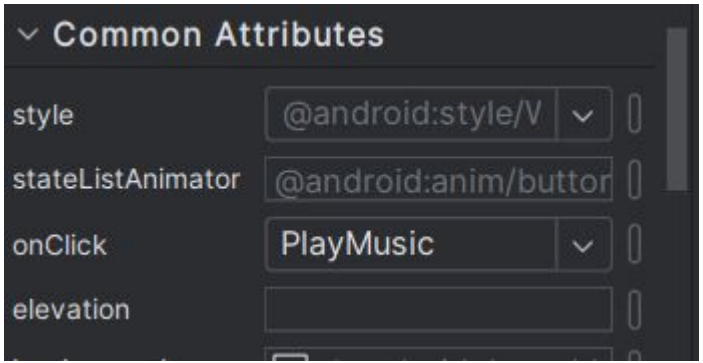
```
public void PlayMusic(View view)
   {
MediaPlayer ring= MediaPlayer.create(MainActivity.this,R.raw.ringtone);
ring.start();
   }
```

```java
package com.example.mediaapp;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
//Import the following class
import android.view.View;
import android.media.MediaPlayer;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    2 usages
    public void PlayMusic(View view)
    {
        MediaPlayer ring= MediaPlayer.create( context: MainActivity.this,R.raw.ringtone);
        ring.start();
    }
}
```

Add audio to the app

- Edit the “onclick” property on the Button and add the function “PlayMusic” (through the activity_main.xml file or Attributes panel for the Button)

```
<Button
    android:id="@+id/button"
    android:layout_width="175dp"
    android:layout_height="79dp"
    android:layout_marginStart="118dp"
    android:layout_marginTop="150dp"
    android:layout_marginEnd="118dp"
    android:layout_marginBottom="502dp"
    android:onClick="PlayMusic"
    android:text="Play Music"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```



Add video to the app

- Add a new Button to the app and display the text “Play Video”
- Add the function “PlayVideo” to the onclick property of the new button.
- Drag and drop a VideoView into the UI from the Palette.
- Adjust the position of the VideoView and create an id.
- Create a reference object for the VideoView in the java file

VideoView v= findViewById(R.id.videoView);

Add video to the app

- To play a downloaded video,
  - Download the video and copy paste to the raw folder on res directory.
  - Add the following code to the MainActivity.java

```
import android.view.View;

import android.widget.VideoView;

import android.widget.MediaController;

import android.media.MediaPlayer;
```

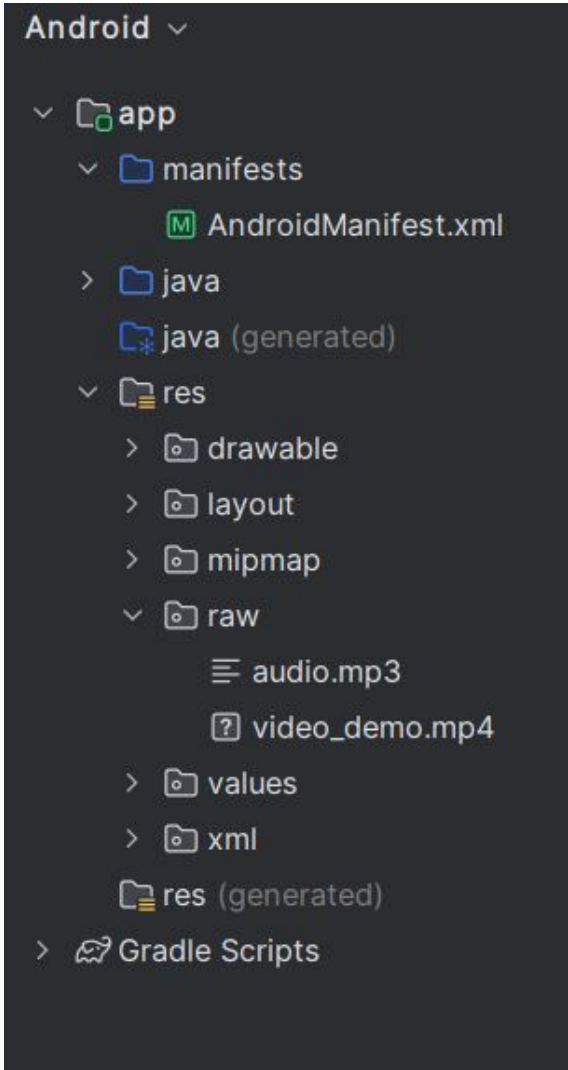
```
public void PlayVideo(View view) {
    VideoView v= findViewById(R.id.videoView);
    //set the path of the video that we need to use in our VideoView
    v.setVideoPath("android.resource://" + getPackageName() + "/" + R.raw.video_demo);
    // creating object of media controller class
    MediaController mediaController = new MediaController( context: this);
    // sets the anchor view for the videoView
    mediaController.setAnchorView(v);
    // sets the media player to the videoView
    mediaController.setMediaPlayer(v);
    // sets the media controller to the videoView
    v.setMediaController(mediaController);
    // starts the video
    v.start();
}
```

Add video to the app

- Set the Video Path to the VideoView

  v.setVideoPath("android.resource://" + getPackageName() + "/" + R.raw.video_demo);

- **MediaController:** Allows an app to interact with an ongoing media session. Media buttons and other commands can be sent to the session.
- **MediaController.setAnchorView(videoView):** specifies the view to which the controller will be anchored. This determines where the controls appear on the screen.



Add video to the app

```
MediaController mediaController = new MediaController(this);
// sets the anchor view for the videoView
mediaController.setAnchorView(v);
// sets the media player to the videoView
mediaController.setMediaPlayer(v);
// sets the media controller to the videoView
v.setMediaController(mediaController);
// starts the video
v.start();
```

Add video to the app

- To play a video from internet,
  - Add the following import statement:

    import android.net.Uri;

    - Add the permission to access internet into AndroidManifest.xml

      <uses-permission android:name="android.permission.INTERNET"/>

```
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>

</manifest>
```

Add video to the app

- Copy the video address of the video to be played.
- Store the address into a string.

String videoUrl =
"https://cdn.pixabay.com/video/2016/04/02/2637-161442811_medium.mp4";

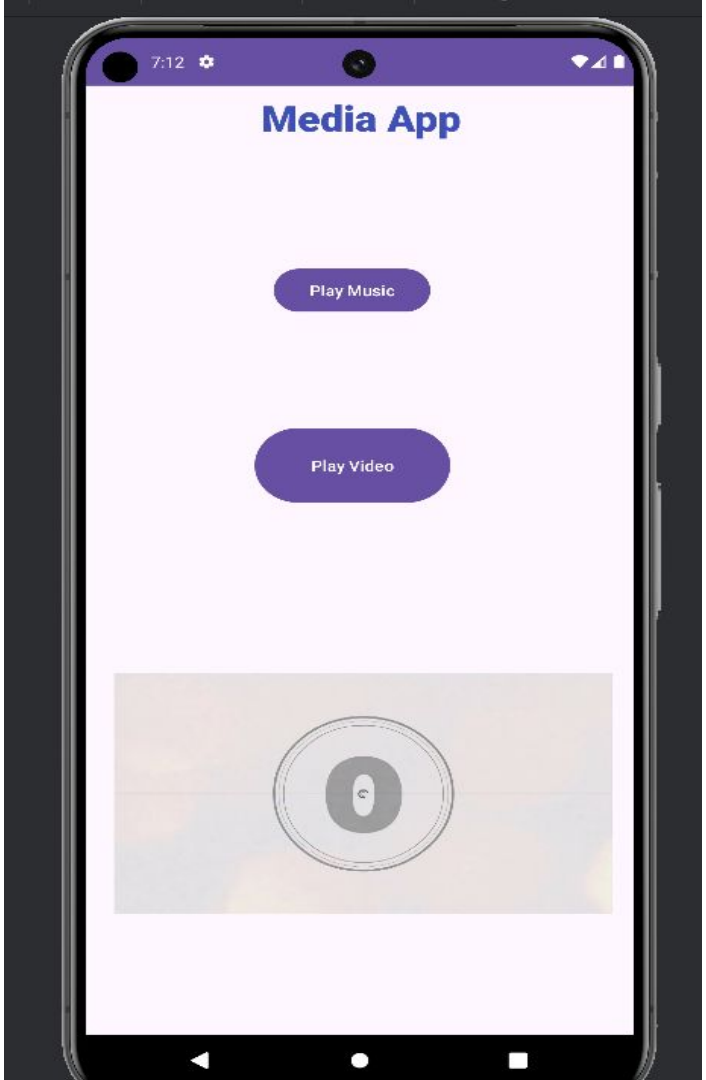
- Uri.parse(): Creates a Uri which parses the given encoded URI string.

Uri uri = Uri.parse(videoUrl);

- Set the video to the VideoView

v.setVideoURI(uri);

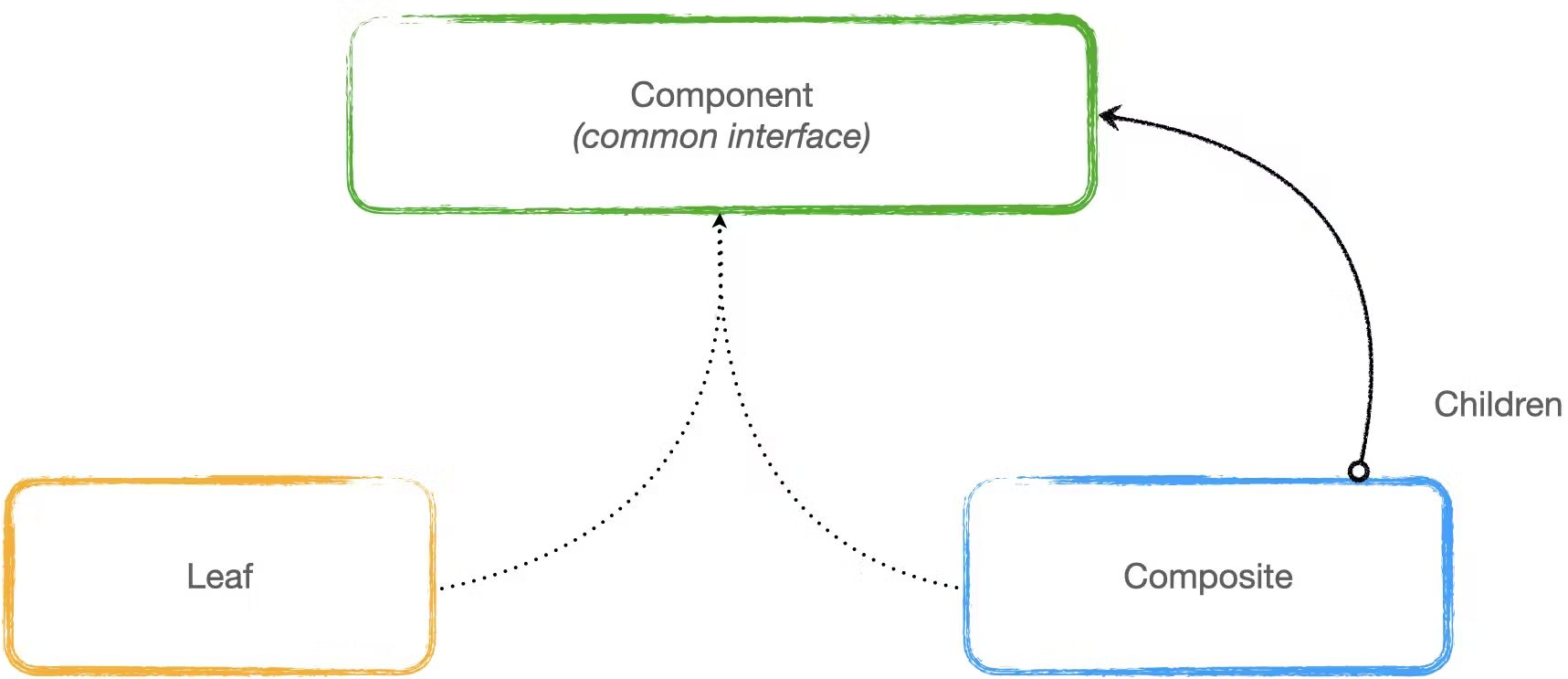
```java
public void PlayVideo(View view) {
    VideoView v= findViewById(R.id.videoView);
    // Your Video URL or Path to the location
    String videoUrl = "https://cdn.pixabay.com/video/2016/04/02/2637-161442811_medium.mp4";
    // Uri object to refer the resource from the videoUrl
    Uri uri = Uri.parse(videoUrl);
    // sets the resource from the videoUrl to the videoView
    v.setVideoURI(uri);
    // creating object of media controller class
    MediaController mediaController = new MediaController( context: this);
    // sets the anchor view for the videoView
    mediaController.setAnchorView(v);
    // sets the media player to the videoView
    mediaController.setMediaPlayer(v);
    // sets the media controller to the videoView
    v.setMediaController(mediaController);
    // starts the video
    v.start();
}
```


7:12
Media App
Play Music
Play Video

Composite Views in Android

Composite Views in Android

- The composite design pattern allows us to create a tree structure of objects.
- Given a root component, we can access all nodes in the structure.
- Such access is granted using a common interface that refers to either of two object types:
  - Leaf - a primitive object
  - Composite - a group of component objects

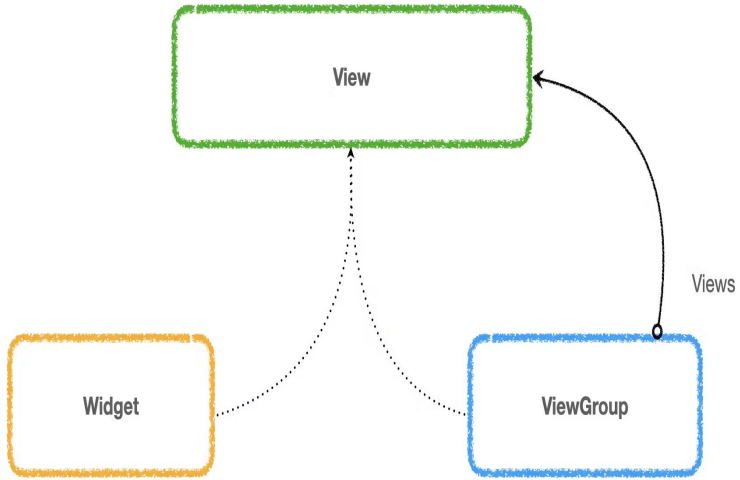

Component
(common interface)
Children
Leaf
Composite

Composite Views in Android

- The component is the common interface for accessing the structure.
- A composite stores instances of child components -- either leaves or other composites.
- With the composite pattern, you only need a reference to the root component and with it, you have access to the rest of the tree.

Composite Views in Android

- **The Android View(Group):**
  - View is the base class for building UI elements (the component)
  - A widget is a view representing an interactive UI component (a leaf)
  - A Viewgroup is a view representing a container component to hold other Views (the composite)



Composite Views in Android

- Each layout is referenced through a single View instance i.e. the root node of the view tree structure.
- The Activity class provides a setContentView api that accepts a root View instance for representing its UI.

LAB ASSIGNMENT

- Popup Menus:
  - Option Menu
  - Context Menu
  - Sub menu
  - Menu from xml
  - Menu via code
  - Application Menu
  - ActionBar
  - ActionBar & Tabs
  - View Pager
  - ActionBar & View Pager

Thank You