TOPICS

1	Graphics & Animation	
2	Bitmap Animation	
3	Frame Animation	
4	Tween Animation	
5	View Animation	
6	Audio & Video Play in Android	
7	Work With Back Ground Services	
8	Broadcast Receiver	
9	Text To Speech	
10	Working with Camera & Taking Picture with Camera	
11	Manage Bluetooth Connection	
12	Manage & Monitor WIFI	
13	Accelerometer Sensor & Gyroscope	

1. Graphics & Animation

Graphics in Android

- Android offers a custom 2D graphics library for drawing and animating shapes and images. The android.graphics.drawable and android.view.animation packages are where you'll find the common classes used for drawing and animating in two-dimensions.
- The android.graphics.Canvas can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.
- The android.graphics.Paint class is used with canvas to draw objects. It holds the information of color and style.

- The very few of the options you have for drawing graphics on Android and which tasks they're best suited for.

- Views to customize their look and feel. When drawing 2D graphics, you'll typically do so in one of two ways:

- Draw your graphics or animations into a View object from your layout. In this manner, the drawing of your graphics is handled by the system's normal View hierarchy drawing process — you simply define the graphics to go inside the View.

- Draw your graphics directly to a Canvas. This way, you personally call the appropriate class's onDraw() method (passing it your Canvas), or one of the Canvas draw...() methods (like drawPicture()).

- In doing so, you are also in control of any animation.

Program to Show Basic Example of Graphics:

File 1: activity_main.xml File

```
<RelativeLayout
xmlns:androclass="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world" />
 </RelativeLayout>
```

File2: MainActivity.java

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;
```

```
public class MainActivity extends Activity {

  DemoView demoview;
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    demoview = new DemoView(this);
    setContentView(demoview);
  }

     private class DemoView extends View{
    public DemoView(Context context){
      super(context);
    }

    @Override protected void onDraw(Canvas canvas) {
      super.onDraw(canvas);

      // custom drawing code here
      Paint paint = new Paint();
      paint.setStyle(Paint.Style.FILL);

      // make the entire canvas white
      paint.setColor(Color.WHITE);
      canvas.drawPaint(paint);

      // draw blue circle with anti aliasing turned off
      paint.setAntiAlias(false);
      paint.setColor(Color.BLUE);
      canvas.drawCircle(20, 20, 15, paint);

      // draw green circle with anti aliasing turned on
      paint.setAntiAlias(true);
      paint.setColor(Color.GREEN);
      canvas.drawCircle(60, 20, 15, paint);
```

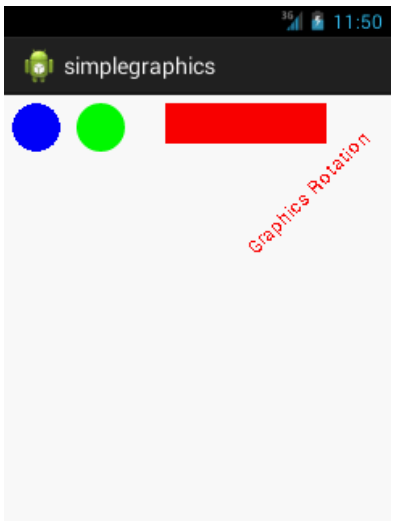
```
        // draw red rectangle with anti aliasing turned off
        paint.setAntiAlias(false);
        paint.setColor(Color.RED);
        canvas.drawRect(100, 5, 200, 30, paint);

        // draw the rotated text
        canvas.rotate(-45);

        paint.setStyle(Paint.Style.FILL);
        canvas.drawText("Graphics Rotation", 40, 180, paint);

        //undo the rotate
        canvas.restore();
      }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
    }
}
```

Output



2. Bitmap Animation

- A bitmap (or raster graphic) is a digital image composed of a matrix of dots.
- When viewed at 100%, each dot corresponds to an individual pixel on a display.
- In a standard bitmap image, each dot can be assigned a different color. In this instance we will simply create a Bitmap directly: Bitmap b = Bitmap.
- Everything that is drawn in android is a Bitmap .
- We can create a Bitmap instance , either by using the Bitmap class which has methods that allow us to manipulate pixels in the 2d coordinate system , or we can can create a Bitmap from an image or a file or a resource by using the BitmapFactory class

What is Canvas?

A canvas is an object which is an instance of the class Canvas . A canvas is used to draw on a bitmap , it has methods to draw shapes on a bitmap . Such as drawing a text , a line , a rectangle or a circle .. We can style the shapes created by the canvas by using a paint object.

What is paint?

A paint object is an instance of the Paint class . it is used to style a shape drawn by a canvas . We can set the stroke , the fill color .. of the drawn shape using the paint object.

Create a bitmap by using Bitmap class

The Bitmap class has many methods that allow us to create an instance of a Bitmap by using java .
These methods will allow us to set the width , height , density , and number of pixels of a Bitmap .

Example for Bitmap Animation

Program:

File 1: MainActivity.java

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

    animate var;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        var = new animate(this);
        setContentView(var);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

File2: Animate.java

```
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.view.View;

public class animate extends View{

    Bitmap bm;
    int x, y;
    public animate(Context context) {
        super(context);
        // TODO Auto-generated constructor stub
        bm=BitmapFactory.decodeResource(getResources(), R.drawable.image4);
        x = 0; y = 0;
    }

    @Override
```

```
    protected void onDraw(Canvas canvas) {
        // TODO Auto-generated method stub
        super.onDraw(canvas);
        Rect myrect = new Rect(0, 0,
canvas.getWidth(),canvas.getHeight()/2);
        Paint pa = new Paint();
        pa.setColor(Color.BLUE);
        pa.setStyle(Paint.Style.FILL);
        canvas.drawRect(myrect, pa);

        if (x < canvas.getWidth()) {
            x += 10;
        }
        else {
            x = 0;
        }
        if (y < canvas.getHeight()) {
            y += 10;
        }
        else {
            y = 0;
        }
        canvas.drawBitmap(bm, x, y, new Paint());
        invalidate();//calls this method again and again
    }

}
```

3. Frame Animation

In Android Frame Animation, you will be swapping frames repeatedly, so that it appears continuous to the human eye and we feel that it is animated.

Frame is referred to an image. So to implement frame by frame animation in android, one needs to have set of images, which describes a motion.

Sequence of Images

The idea behind a frame animation is simple: We'll be cycling through a series of images very quickly, just like an old movie reel. The "frame" refers to a single image. Thus, the first step in creating a custom frame animation is to create a sequence of images.

We have two options here: we can use XML drawables (such as shape drawables) or actual image files. For the sake of simplicity, we're going to use the following series of PNG images:

To create AnimationDrawable Object:

```
AnimationDrawable anim;
anim = (AnimationDrawable) viewComponent.getDrawable();
anim.start(); // To Start Animation
anim.stop(); // To Stop Animation
```

Now What We Need to Make For Frame Animation

- First create animation-list file
- Then Add Images to Drawable Folder with proper name
- Make layout in main acitivity.xml file to start & stop animation
- Write a frame animation code in MainActivity.java file

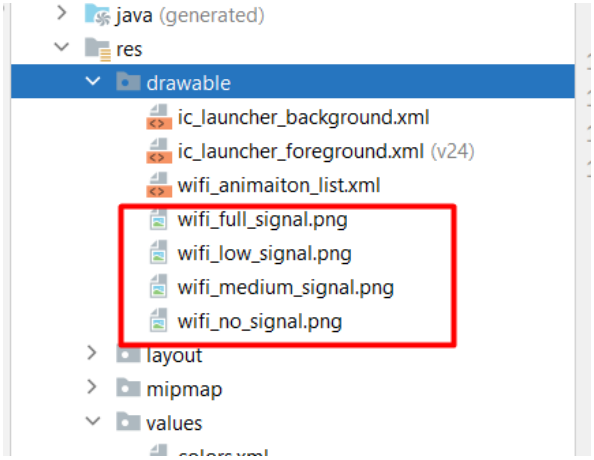
Program For Frame Animation

AIM: Wifi Blinking animation Using Frame Animation, Here we took different images of WIFI signal and then using frame animation Animate it as Wifi Signal Low to High & High to low animation

Steps We Need to Performs

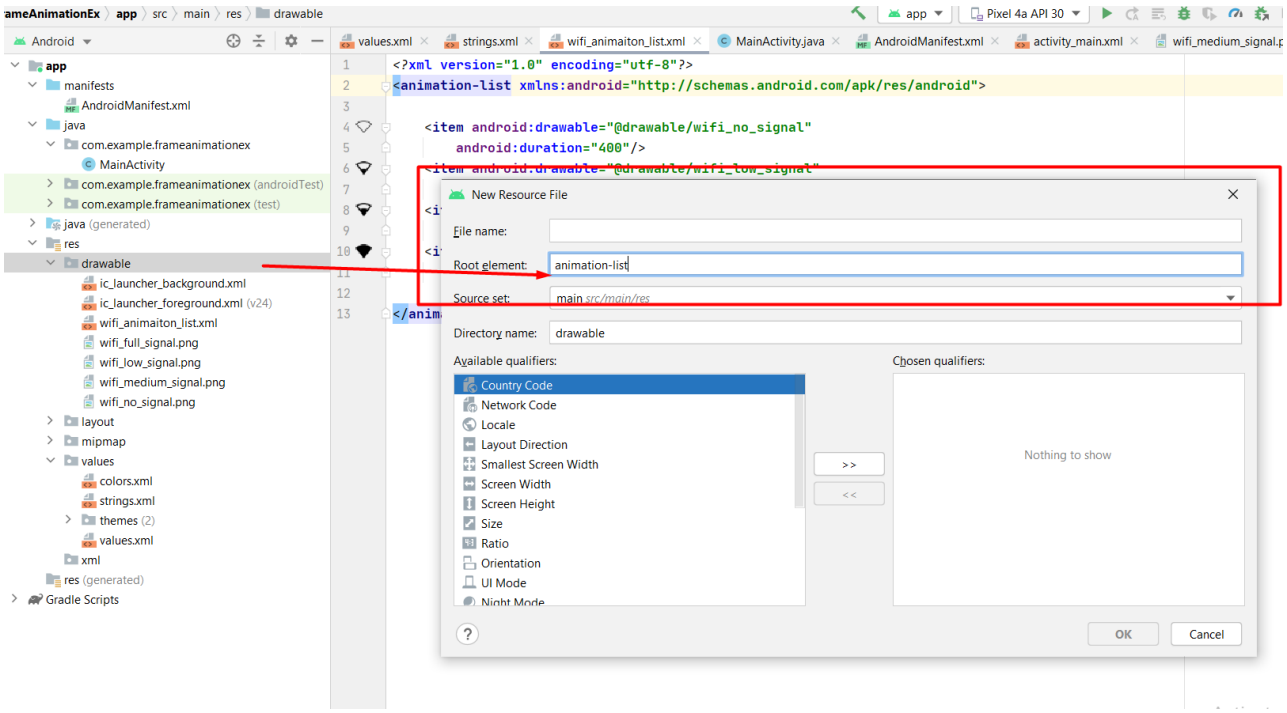
Step 1: Add Images to Drawable Folder

Just do copy from you pc folders to Drawable Folder & Paste them



Step 2: Create Animate-list xml file. For that

Right Click on Drawable Folder → Click on Drawable Resource File → Select animation-list as Root Element give proper name to file & Click on OK.



Step 3: Make activity_main.xml File for Start & Stop Animation

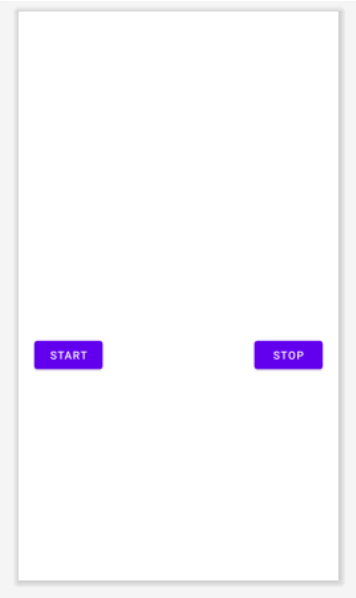
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:layout_width="24dp"
        android:layout_height="24dp"
        android:id="@+id/imageViewID"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"/>

     <Button
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Start"
         android:id="@+id/startBtn"
         android:layout_below="@+id/imageViewID"
         android:layout_marginStart="20dp"
         android:layout_marginTop="40dp"
         />
     <Button
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Stop"
         android:id="@+id/stopBtn"
         android:layout_below="@+id/imageViewID"
```

```
        android:layout_marginEnd="20dp"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="40dp"
        />

</RelativeLayout>
```



Step 5: Write a Java Code for Animation the Images & also map code for Start & Stop Button

```
package com.example.frameanimationex;

import androidx.appcompat.app.AppCompatActivity;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    ImageView wifi;
    Button start, stop;
    AnimationDrawable anim;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        wifi = findViewById(R.id.imageViewID);
        start = findViewById(R.id.startBtn);
        stop = findViewById(R.id.stopBtn);

        wifi.setImageResource(R.drawable.wifi_animaiton_list);
```

```
        // To Start Animation Click on Start Button Event
        start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                anim = (AnimationDrawable) wifi.getDrawable();
                anim.start();
            }
        });


        // To Stop Animation Click on Stop Button Event
        stop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                anim.stop();
            }
        });



    }
}
```

Animation Methods & Attributes

Property Animations :

Property animations are highly customizable, you can specify the duration, the number of repeats, the type of interpolation, and the frame rate of the animation. The Property Animation system is always preferred for more complex animations.

Animation Syntax:

```
Animation animation =AnimationUtils.loadAnimation(getApplicationContext(), R.anim.
myanimation);
```

Second parameter refers to our animation.xml file.It is created under res directory(res->anim->myanimation.xml)

The Animation class has many methods given below:

- start(): This method will start the animation.

- setDuration(long duration): This method sets the duration of an animation.
- getDuration(): This method gets the duration.
- end(): This method ends the animation.
- cancel(): This method cancels the animation.

Setting The Animation Listeners (Optional)

Animation listeners can be set by implementing AnimationListener in our Activity. If you will use AnimationListener, then you will have to override following methods.

onAnimationStart –

```
@Override
public void onAnimationStart(Animation animation) {
}
```

onAnimationEnd –

```
@Override
public void onAnimationEnd(Animation animation) {
}
```

onAnimationRepeat –

```
@Override
public void onAnimationRepeat(Animation animation) {
}
```

Scale Animation

Scale Animation is used to make a smaller or larger view either on x axis or y axis. Pivot point can also be specified around which we want the animation to take place.

Rotate Animation

Rotate Animation is used to rotate a view around a pivot point by a certain number of degrees.

Translate Animation

Translate Animation is used to move a view along the x or y axis.

Alpha Animation

Transparency of a view can be changed by Alpha Animation

Interpolator:

The dictionary meaning of Interpolate is to alter, intercept or insert in between two things or events.

It is used to insert or apply an additional animation effects between the start and the end value of the property of an object.It is also used to define the rate of change of an animation.

Types of Interpolator:

1.**Linear Interpolator:** It is the default interpolator of all animations. It defines that the rate of the change of the animation is constant.

2.**Accelerate Interpolator:** Accelerates the moving of the view, it starts out slowly and then accelerates until it reaches the final position.

3.**Decelerate Interpolator:** Does the opposite of the accelerate interpolator. It starts out fast and the slows down.

4.**Accelerate Decelerate Interpolator:** Makes the moving go slow at the beginning and the end, but fast in the middle.

5.**Anticipate Interpolator:** Moves the animation backwards before it starts

6.**Overshoot Interpolator:** Does the opposite of the anticipate interpolator. It make the animation to go further than the defined destintion, and then get back to the defined destination.

7.**Anticipate Overshoot Interpolator:** Combines the anticipate and the overshoot interpoplator. The change starts backward then flings forward and overshoots the target value and finally goes back to the final value.

8.**Bounce Interpolator:** Makes an bounce animation before the desired animation ends.

9.**Cycle Interpolator:** Repeats the animation for a specified number of cycles.

Important XML Animation Attributes In Android:

1. android:duration: The duration in which animation is completed is referred to as duration attribute. It refers to the ideal duration to show the transition on the screen.

```
android:duration="1500"
```

2. android:fillAfter: This attribute defines whether the view should be visible or not at the end of the animation.We have set its value true in our animation. If it would set to false then element changes comes to its previous state after the animation.

```
android:fillAfter="true"
```

3. android:interpolator: This property refers to the rate of change in animation. You can define your own interpolators using the time as the constraint. In this animation we have used an inbuilt interpolator i.e. linear interpolator.

```
android:interpolator="@android:anim/linear_interpolator"
```

4. android:startOffset: It refers to the waiting time before an animation starts.

```
android:startOffset="2000"
```

5. android:repeatMode: When user wants the animation to be repeated then this attribute is used

```
android:repeatMode="restart"
```

6. android:repeatCount: This attribute defines the number of repetitions on animation.

```
android:repeatCount="infinite"
```

4. Tween Animation

- Tween Animation takes some parameters such as start value , end value, size , time duration , rotation angle e.t.c and perform the required animation on that object. It can be applied to any type of object. So in order to use this , android has provided us a class called Animation.

- In order to perform animation in android , we are going to call a static function loadAnimation() of the class AnimationUtils. We are going to receive the result in an instance of Animation Object. Its syntax is as follows –

```
Animation animation =
AnimationUtils.loadAnimation(getApplicationContext(),
   R.anim.myanimation);
```

Program For Tween Animations

AIM:

- **What we do in file?**
- Blink animation
- Fade animation
- Move animation
- Rotate animation
- Slide animation
- Zoom in – out Animation

Steps

- Create **anim** Folder, Under Resource directory as per Previous steps shown in View Animation
- Write all files for different animations
- Write xml file for Layout xml
- Write java code for MainActivity.java File

File 1: blink_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <alpha android:fromAlpha="0.5"
        android:toAlpha="1.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:duration="500"
        android:repeatMode="reverse"
        android:repeatCount="3" />

</set>
```

File 2: fade_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator">

    <alpha
        android:toAlpha="1"
        android:fromAlpha="0"
        android:duration="1000"
    />

    <alpha
        android:toAlpha="0"
        android:fromAlpha="1"
        android:startOffset="1000"
        android:duration="3000"
        />
</set>
```

File 3: move_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:interpolator="@android:anim/linear_interpolator"
    android:fillAfter="true"
    >

    <translate
        android:fromXDelta="0%p"
        android:toXDelta="75%p"
        android:fromYDelta="10%p"
        android:toYDelta="45%p"
        android:duration = "2000"/>
</set>
```

File 4: rotate_animation.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:andorid="http://schemas.android.com/apk/res-auto"
    android:fillAfter="true">

    <rotate xmlns:android="http://schemas.android.com/apk/res/android"
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2500" >
    </rotate>

    <rotate xmlns:android="http://schemas.android.com/apk/res/android"
        android:startOffset="5000"
        android:fromDegrees="360"
        android:toDegrees="0"
        android:pivotX="70%"
        android:pivotY="50%"
        android:duration="2500" >
    </rotate>
```

```
</set>
```

File 5 : Slide_animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:andorid="http://schemas.android.com/apk/res-auto">

    <scale
        android:duration="2000"
        android:fromXScale="1.0"
        android:fromYScale="1.0"
        android:interpolator="@android:anim/linear_interpolator"
        android:toXScale="1.0"
        android:toYScale="0.0" />
</set>
```

Flie 6: zoom_animation.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <scale xmlns:android="http://schemas.android.com/apk/res/android"
        android:fromXScale="0.5"
        android:toXScale="3.0"
        android:fromYScale="0.5"
        android:toYScale="3.0"
        android:duration="4000"
        android:pivotX="50%"
        android:pivotY="50%" >
    </scale>


    <scale xmlns:android="http://schemas.android.com/apk/res/android"
        android:startOffset="5000"
        android:fromXScale="3.0"
        android:toXScale="0.5"
        android:fromYScale="3.0"
        android:toYScale="0.5"
        android:duration="4000"
        android:pivotX="50%"
        android:pivotY="50%" >
    </scale>

</set>
```

File 7: 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"
    tools:context=".MainActivity">

    <ImageView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginTop="20dp"
        android:id="@+id/logo"
        android:layout_centerHorizontal="true"
        android:contentDescription="APP NAME"
        android:src="@drawable/ic_launcher_foreground"
        />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/logo"
        android:layout_marginTop="30dp"
        android:orientation="horizontal"
        android:id="@+id/linear1">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Blink Button"
            android:layout_margin="10dp"
            android:id="@+id/blinkBtn"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Fade Button"
            android:layout_margin="10dp"
            android:id="@+id/fadeBtn"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Clock Wise"
            android:layout_margin="10dp"
            android:id="@+id/rotateBtn"/>

    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/linear1"
        android:layout_marginTop="30dp"
        android:orientation="horizontal"
        android:id="@+id/linear2">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Zoom Button"
            android:layout_margin="10dp"
```

```
            android:id="@+id/zoomBtn"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Slide Button"
            android:layout_margin="10dp"
            android:id="@+id/slideBtn"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5dp"
            android:text="Move"
            android:layout_margin="10dp"
            android:id="@+id/moveBtn"/>

    </LinearLayout>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="STOP"
        android:id="@+id/stopBtn"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/linear2"
        />

</RelativeLayout>
```

File 8: MainActivity.java File

```
package com.example.androidanimationex_1;

import androidx.appcompat.app.AppCompatActivity;

import android.animation.ObjectAnimator;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    Button blinkBtn, fadeBtn, moveBtn, rotateBtn, zoomBtn, slideBtn,
stopBtn;

    private Animation objectAnimator;
    ImageView logo;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        blinkBtn = findViewById(R.id.blinkBtn);
```

```
        fadeBtn = findViewById(R.id.fadeBtn);
        moveBtn = findViewById(R.id.moveBtn);
        rotateBtn = findViewById(R.id.rotateBtn);
        zoomBtn = findViewById(R.id.zoomBtn);
        slideBtn = findViewById(R.id.slideBtn);
        logo = findViewById(R.id.logo);
        stopBtn = findViewById(R.id.stopBtn);


        blinkBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animation =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.blink_animation
);
                logo.startAnimation(animation);
                }
        });

        rotateBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animation =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.rotate_animatio
n);
                logo.startAnimation(animation);
            }
        });
        zoomBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Animation animation =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.zoom_animation)
;
                objectAnimator =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.zoom_animation)
;
                logo.startAnimation(objectAnimator);
            }
        });

        stopBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                v.clearAnimation();
                v.animate().cancel();

                objectAnimator.cancel();

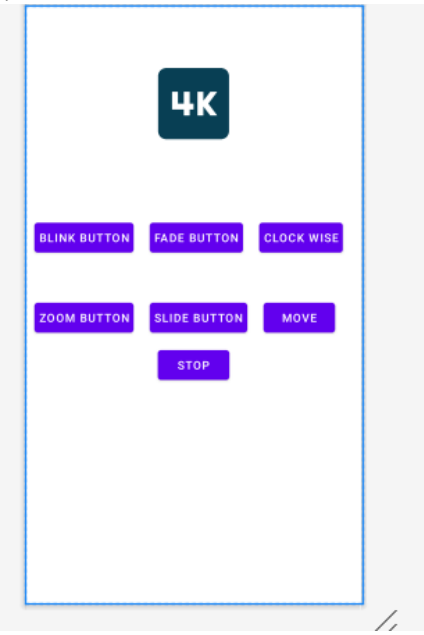
            }
        });

        fadeBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                objectAnimator =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.fade_animation)
;
                logo.startAnimation(objectAnimator);
            }
```

```
        });

        moveBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                objectAnimator =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.move_animation)
;
                logo.startAnimation(objectAnimator);
            }
        });

        slideBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                objectAnimator =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.slide_animation
);
                logo.startAnimation(objectAnimator);
            }
        });
    }
}
```



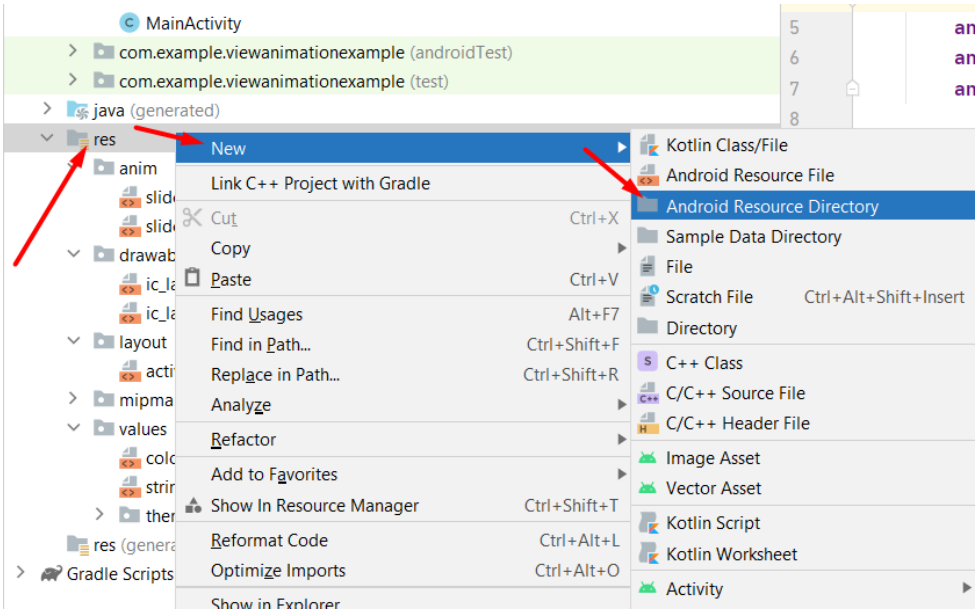
5. View Animation

- This is the simplest animation used in Android. It define the properties of our Views that should be animated using a technique called Tween Animation.
- It take the following parameters i.e. size, time duration , rotation angle, start value , end value, and perform the required animation on that object.

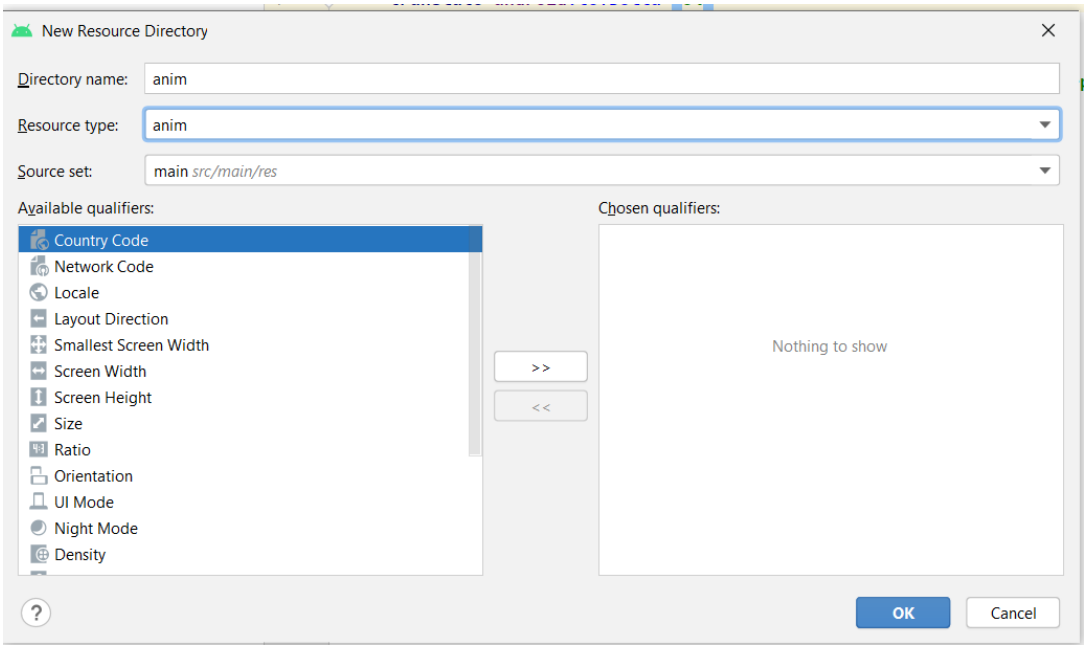
- You can execute the animation by specifying transformations on your View. This can be done in XML resource files or programmatically.
- Android View animation can make animation on any View objects, such as ImageView, TextView or Button objects.
- View animation can only animate simple properties like position, size, rotation, and the alpha property that allows you animate the transparency of a View.

Steps for Animation

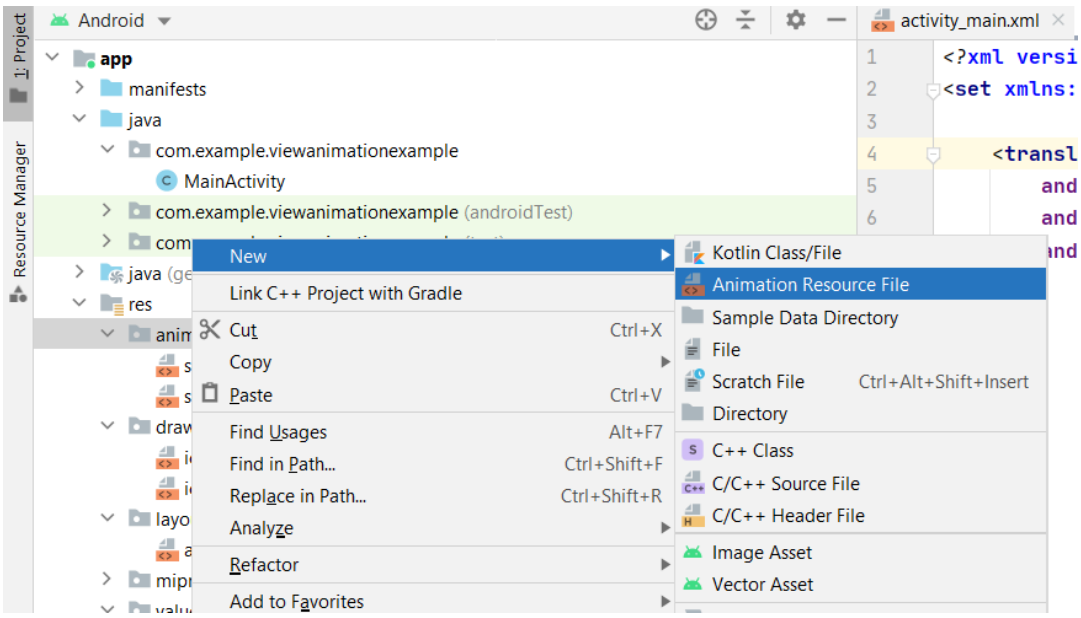
- Right click on Resource folder → Click on Resource Directory
- → Select Resource Type as anim



- Select Resource Type anim



- Right Click on **anim** Folder, Select android Animation Resource File



- Create a File slide_in_top.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:toYDelta="0%"
        android:fromYDelta="-100%"
        android:duration="600"

android:interpolator="@android:anim/accelerate_decelerate_interpolator"/
```

```
>
</set>
```

- Create a File slide_out_bottom.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:fromYDelta="0%"
        android:toYDelta="100%"
        android:duration="600"
        android:fromXDelta="0%"
        android:toXDelta="100%"

android:interpolator="@android:anim/accelerate_decelerate_interpolator"
        />
</set>
```

- Create activity_main.xml File with Button with text: Slide in Topthatis used for Slide in Top Animation & Text view for Slide Out Bottom Animation

```
<?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"
    tools:context=".MainActivity"
    android:background="#AAFDC6">


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Slide IN TOP"
        android:layout_margin="10dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:id="@+id/slideInBtn"

        />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#F44336"
        android:gravity="center"
        android:id="@+id/animateView"
        android:textColor="@color/white"
        android:textSize="20dp"
```

```
        android:visibility="gone"
        android:text="Click To Slide Out"/>


</RelativeLayout>
```

- Create a Java File for **MainActivity.java File**

```
- package com.example.viewanimationexample;

  import androidx.appcompat.app.AppCompatActivity;

  import android.graphics.drawable.AnimationDrawable;
  import android.os.Bundle;
  import android.view.View;
  import android.view.animation.Animation;
  import android.view.animation.AnimationUtils;
  import android.widget.Button;
  import android.widget.TextView;

  public class MainActivity extends AppCompatActivity {

      Button slideInBtn, slideUpBtn;
      TextView animateView;
      private Animation objectAnimator;
      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);

          slideInBtn = findViewById(R.id.slideInBtn);
          animateView = findViewById(R.id.animateView);

          slideInBtn.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {

                  objectAnimator =
  AnimationUtils.loadAnimation(getApplicationContext(),R.anim.slide_in)
  ;
                  objectAnimator.setAnimationListener(new
  Animation.AnimationListener() {
                      @Override
                      public void onAnimationStart(Animation animation)
  {
                          slideInBtn.setVisibility(View.GONE);
                          animateView.setVisibility(View.VISIBLE);
                      }

                      @Override
                      public void onAnimationEnd(Animation animation) {

                      }

                      @Override
                      public void onAnimationRepeat(Animation
  animation) {
```

```
                    }
                });
                animateView.startAnimation(objectAnimator);
            }
        });


        animateView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                objectAnimator =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.slide_out
_bottom);
                objectAnimator.setAnimationListener(new
Animation.AnimationListener() {
                    @Override
                    public void onAnimationStart(Animation animation)
{

                    }

                    @Override
                    public void onAnimationEnd(Animation animation) {


                        slideInBtn.setVisibility(View.VISIBLE);
                        animateView.setVisibility(View.GONE);
                    }


                    @Override
                    public void onAnimationRepeat(Animation
animation) {

                    }
                });
                animateView.startAnimation(objectAnimator);
            }
        });

    }


}
```

6. Audio & Video Play in Android

Android provides many ways to control playback of audio/video files and streams. One of this way is through a class called MediaPlayer.

Android is providing MediaPlayer class to access built-in mediaplayer services like playing audio,video e.t.c. In order to use MediaPlayer, we have to call a static Method create() of this class. This method returns an instance of MediaPlayer class. Its syntax is as follows –

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.song);
```

The second parameter is the name of the song that you want to play. You have to make a new folder under your project with name raw and place the music file into it.

Once you have created the Mediaplayer object you can call some methods to start or stop the music. These methods are listed below.

```
mediaPlayer.start();
mediaPlayer.pause();
```

On call to **start**() method, the music will start playing from the beginning. If this method is called again after the **pause**() method, the music would start playing from where it is left and not from the beginning.

In order to start music from the beginning, you have to call **reset**() method. Its syntax is given below.

```
mediaPlayer.reset();
```

Methods For media Player

1
isPlaying()

This method just returns true/false indicating the song is playing or not

2
seekTo(position)

This method takes an integer, and move song to that particular position millisecond

3
getCurrentPosition()

This method returns the current position of song in milliseconds

4
getDuration()

This method returns the total time duration of song in milliseconds

5
reset()

This method resets the media player

6
release()

This method releases any resource attached with MediaPlayer object

7
setVolume(float leftVolume, float rightVolume)

This method sets the up down volume for this player

8
setDataSource(FileDescriptor fd)

This method sets the data source of audio/video file

9
selectTrack(int index)

This method takes an integer, and select the track from the list on that particular index

10
getTrackInfo()

This method returns an array of track information

Program That Shows Both Audio & Video File Integration

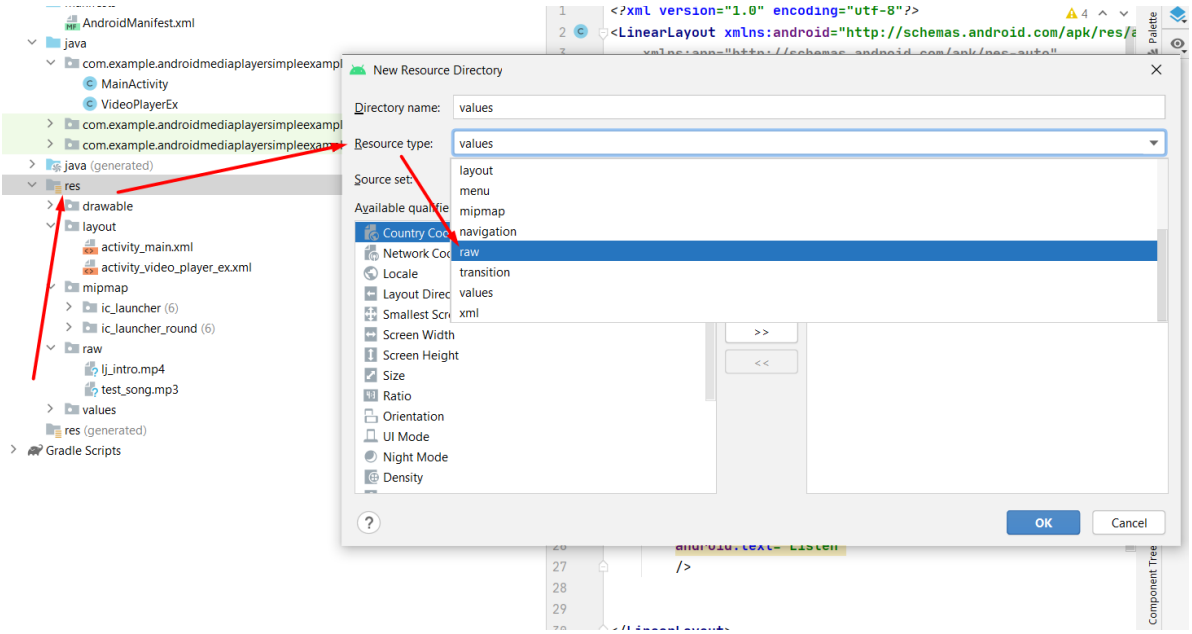
For Audio

Create a folder with name raw under the Resource Folder.

For that:

Right click on res →

Select Resource Directory → Select raw as Resource Type



- **Copy Audio file and paste it in raw Folder.**

File 1: activity_main.xml File

```
<?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">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Simple Media - Audio Player"
        android:layout_gravity="center"
        android:textSize="20dp"
        android:gravity="center"
        android:textStyle="bold"
```

```
        android:layout_margin="20dp"
         />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listenBtn"
        android:layout_gravity="center"
        android:text="Listen"
        />
</LinearLayout>
```

File 2: MainActivity.java File

```
package com.example.androidmediaplayersimpleexamples;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button listenBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listenBtn = findViewById(R.id.listenBtn);
        listenBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MediaPlayer mediaPlayer =
MediaPlayer.create(getApplicationContext(),R.raw.test_song);
                mediaPlayer.start();
                mediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
                    @Override
                    public void onCompletion(MediaPlayer mp) {
                        Toast.makeText(getApplicationContext(),"Completed
Song",Toast.LENGTH_LONG).show();
                    }
                });
            }
        });
    }
}
```

For Video

- Copy & Paste video file under the raw Folder

Activity_video_player.xml File

```
<?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"
    tools:context=".VideoPlayerEx">

    <VideoView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/videoViewID"
        android:layout_margin="20dp"
        />


</LinearLayout>
```

VideoPlayerEx.java File

```
package com.example.androidmediaplayersimpleexamples;

import androidx.appcompat.app.AppCompatActivity;

import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class VideoPlayerEx extends AppCompatActivity {

    VideoView videoView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video_player_ex);
        videoView = findViewById(R.id.videoViewID);
        String videoPath  = "android.resource://" +getPackageName()+ "/"
+R.raw.lj_intro;

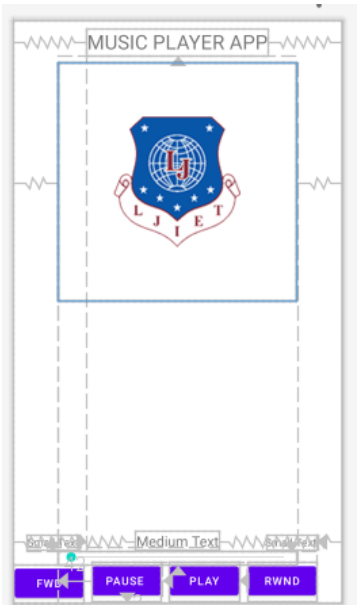
        Uri uri = Uri.parse(videoPath);
        videoView.setVideoURI(uri);
        MediaController mediaController = new MediaController(this);
        videoView.setMediaController(mediaController);

        mediaController.setAnchorView(videoView);


    }
}
```

Program 2: Example That Shows Usage of Media Player with Seek Bar Like Play, Stop, Forward & Backward Functionality (For Audio)

We are going to develop an application which shows below layout.



Activity_main.xml File

```
<?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"
    tools:context=".MainActivity">

   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="MUSIC PLAYER APP"
       android:textSize="25dp"
       android:layout_margin="10dp"
       android:layout_centerHorizontal="true"
       android:id="@+id/nameAppTextView"/>

    <ImageView
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:id="@+id/logoImageView"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/nameAppTextView"
        android:src="@drawable/ljiet_logo"/>

    <Button
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="FWD"
        android:id="@+id/forwardBtn"
        android:layout_alignParentBottom="true"

        android:layout_marginLeft="15dp"
        android:layout_marginEnd="10dp"
        />

    <Button
        android:id="@+id/pauseBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignStart="@id/logoImageView"
        android:layout_alignLeft="@+id/forwardBtn"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="42dp"
        android:layout_marginLeft="98dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="3dp"
        android:text="PAUSE " />

    <Button
        android:id="@+id/playBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/pauseBtn"
        android:layout_toRightOf="@id/pauseBtn"
        android:layout_toEndOf="@+id/pauseBtn"
        android:layout_marginRight="10dp"
        android:text="Play" />

    <Button
        android:id="@+id/rewindBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/playBtn"
        android:layout_toRightOf="@id/playBtn"
        android:layout_toEndOf="@+id/playBtn"
        android:text="RWND" />

    <SeekBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/seekbar"
        android:layout_alignLeft="@+id/logoImageView"
        android:layout_alignStart="@+id/logoImageView"
        android:layout_alignRight="@+id/logoImageView"
        android:layout_above="@+id/forwardBtn"
        android:layout_alignEnd="@+id/logoImageView"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Small Text"
        android:id="@+id/textView2"
        android:layout_above="@id/seekbar"
        android:layout_toLeftOf="@id/nameAppTextView"
        android:layout_toStartOf="@id/nameAppTextView"
        android:textAppearance="@style/TextAppearance.AppCompat.Small"/>
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Small Text"
        android:id="@+id/textView3"
        android:layout_above="@id/seekbar"
        android:layout_alignRight="@id/rewindBtn"
        android:layout_alignEnd="@id/rewindBtn"
        android:textAppearance="@style/TextAppearance.AppCompat.Small"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="@style/TextAppearance.AppCompat.Medium"
        android:id="@+id/textView4"
        android:layout_alignBaseline="@+id/textView2"
        android:text="Medium Text"
        android:layout_alignBottom="@+id/textView2"
        android:layout_centerHorizontal="true"/>


</RelativeLayout>
```

MainActivity.java File

```
package com.example.androidmediaplayer;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import java.util.concurrent.TimeUnit;

public class MainActivity extends AppCompatActivity {

    TextView txt1, txt2, txt3, txt4;
    ImageView logo;
    Button forwardBtn, pauseBtn, playBtn, rewindBtn;
    private MediaPlayer mediaPlayer;
    private double startTime = 0;
    private double finalTime = 0;
    private Handler myHandler = new Handler();
    private int forwardTime = 5000;
    private int backwardTime = 5000;
    private SeekBar seekBar;
    private static int oneTimeOnly = 0 ;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        forwardBtn = findViewById(R.id.forwardBtn);
        playBtn = findViewById(R.id.playBtn);
        pauseBtn = findViewById(R.id.pauseBtn);
        rewindBtn = findViewById(R.id.rewindBtn);
        logo = findViewById(R.id.logoImageView);
        txt2 = findViewById(R.id.textView2);
        txt3 = findViewById(R.id.textView3);
        txt4 = findViewById(R.id.textView4);
        txt3.setText("Garba.mp3");



        mediaPlayer = MediaPlayer.create(this,R.raw.music_garba);
        seekBar = findViewById(R.id.seekbar);
        seekBar.setClickable(false);

        pauseBtn.setEnabled(false);

        playBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),"Play
Sound",Toast.LENGTH_LONG).show();
                mediaPlayer.start();
                finalTime = mediaPlayer.getDuration();
                startTime = mediaPlayer.getCurrentPosition();

                if(oneTimeOnly == 0)
                {
                    seekBar.setMax((int)finalTime);
                    oneTimeOnly = 1;
                }
                 txt3.setText(String.format("%d min, %d sec",
TimeUnit.MICROSECONDS.

toMinutes((long)finalTime),TimeUnit.MICROSECONDS.toSeconds((long)finalTime)
-
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)finalTime)
)));



                txt2.setText(String.format("%d min, %d sec",
TimeUnit.MICROSECONDS.

toMinutes((long)startTime),TimeUnit.MICROSECONDS.toSeconds((long)startTime)
-
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes((long)startTime)
)));

                seekBar.setProgress((int)startTime);
                myHandler.postDelayed(UpdateSongTime,100);
                pauseBtn.setEnabled(true);
                playBtn.setEnabled(false);
            }
        });
```

```
        pauseBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(),"Pausing
Sound",Toast.LENGTH_LONG).show();
                mediaPlayer.pause();
                pauseBtn.setEnabled(false);
                playBtn.setEnabled(true);
            }
        });


        forwardBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int temp = (int)startTime;
                if((temp+forwardTime)<=finalTime){
                    startTime = startTime + forwardTime;
                    mediaPlayer.seekTo((int)startTime);
                    Toast.makeText(getApplicationContext(),"5 Seconds
Forwarded",Toast.LENGTH_LONG).show();
                }
                else{
                    Toast.makeText(getApplicationContext(),"Can Not
Forward",Toast.LENGTH_LONG).show();
                }
            }
        });

        rewindBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int temp = (int)startTime;
                if((temp-backwardTime)>0)
                {
                    startTime = startTime - backwardTime;
                    mediaPlayer.seekTo((int)startTime);
                    Toast.makeText(getApplicationContext(),"5 Seconds
Backward ",Toast.LENGTH_LONG).show();

                }
                else{
                    Toast.makeText(getApplicationContext(),"Can Not
Backward",Toast.LENGTH_LONG).show();
                }
            }
        });



    }
    private Runnable UpdateSongTime = new Runnable() {
        @Override
        public void run() {
            startTime = mediaPlayer.getCurrentPosition();
            txt2.setText(String.format("%d min, %d sec",
                    TimeUnit.MILLISECONDS.toMinutes((long) startTime),
                    TimeUnit.MILLISECONDS.toSeconds((long) startTime) -
```

```
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.
                                    toMinutes((long) startTime))));
            seekBar.setProgress((int)startTime);
            myHandler.postDelayed(this,100);
        }


    };


}
```

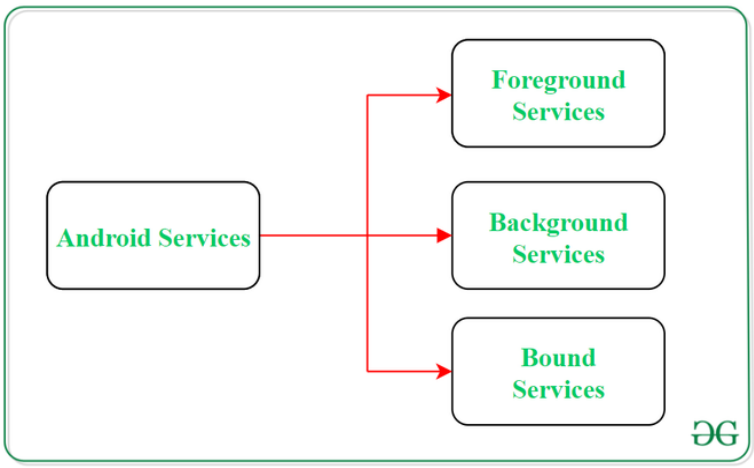
7. Work With Back Ground Services

- Android service is a component that is used to perform operations on the background such as playing music, handle network transactions, interacting content providers etc. It doesn't has any UI (user interface).

- The service runs in the background indefinitely even if application is destroyed.

- Moreover, service can be bounded by a component to perform interactivity and inter process communication (IPC).

Android Services Types



A. Foreground Services
- Services that notify the user about its ongoing operations are termed as Foreground Services.
- Users can interact with the service by the notifications provided about the ongoing task.
- Such as in downloading a file, the user can keep track of the progress in downloading and can also pause and resume the process.

B. Background Services:
- Background services do not require any user intervention. These services do not notify the user about ongoing background tasks and users also cannot access them.
- The process like schedule syncing of data or storing of data fall under this service.

C. Bound Services:
- This type of android service allows the components of the application like activity to bound themselves with it.
- Bound services perform their task as long as any application component is bound to it.
- More than one component is allowed to bind themselves with a service at a time.
- In order to bind an application component with a service bindService() method is used.

The Life Cycle of Android Services

In android, services have 2 possible paths to complete its life cycle namely Started and Bounded.

1. Started Service (Unbounded Service):

By following this path, a service will initiate when an application component calls the startService() method. Once initiated, the service can run continuously in the background even if the

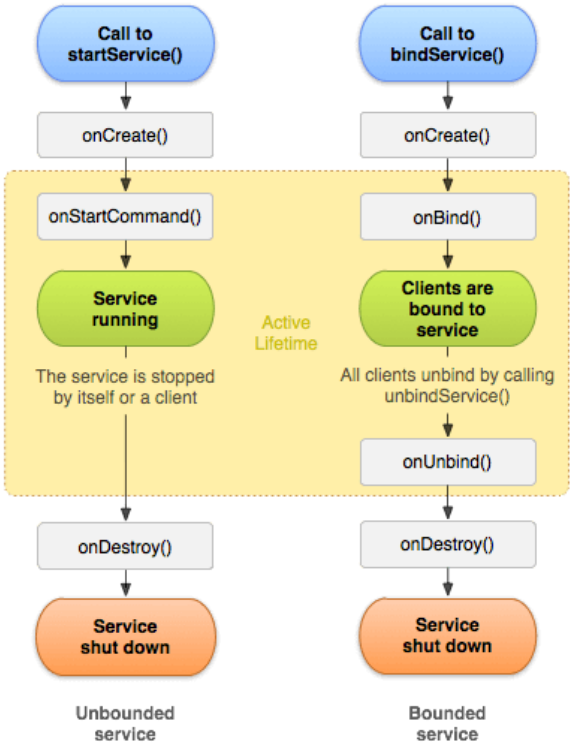
component is destroyed which was responsible for the start of the service. Two option are available to stop the execution of service:

By calling stopService() method,

The service can stop itself by using stopSelf() method.

2. Bounded Service:

It can be treated as a server in a client-server interface. By following this path, android application components can send requests to the service and can fetch results. A service is termed as bounded when an application component binds itself with a service by calling bindService() method. To stop the execution of this service, all the components must unbind themselves from the service by using unbindService() method.



Methods of Service Lifecycle

1- onStartCommand()
The system calls this method when another component, such as an activity, requests that the service be started, by calling startService(). If you implement this method, it is your responsibility to stop the service when its work is done, by calling stopSelf() or stopService() methods.

2- onBind()
The system calls this method when another component wants to bind with the service by calling bindService(). If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an IBinder object. You must always implement this method, but if you don't want to allow binding, then you should return null.

3- onUnbind()
The system calls this method when all clients have disconnected from a particular interface published by the service.

4- onRebind()
The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its onUnbind(Intent).

5- onCreate()

The system calls this method when the service is first created using onStartCommand() or onBind(). This call is required to perform one-time set-up.

6- onDestroy()

The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc.

Example of Android Services

Playing music in the background is a very common example of services in android. From the time when a user starts the service, music play continuously in the background even if the user switches to another application. The user has to stop the service explicitly in order to pause the music.

Program : Here, We are going to see Service Lifecycle with Background Music Player.

Steps

- Create MyService.java file

File : MyService.java

```
package com.example.serviceexample1;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.media.audiofx.Equalizer;
import android.os.IBinder;
import android.provider.Settings;
import android.widget.MediaController;
import android.widget.Toast;

import androidx.annotation.Nullable;

public class MyService extends Service {


    private MediaPlayer player;
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {


        player = MediaPlayer.create(this,
Settings.System.DEFAULT_ALARM_ALERT_URI);

        player.setLooping(true);

        player.start();
```

```
        Toast.makeText(this,"Service Started",Toast.LENGTH_LONG).show();
        return START_STICKY;
    }


    @Override
    public void onDestroy() {
        super.onDestroy();
        player.stop();
        Toast.makeText(this,"Service Destroyed",Toast.LENGTH_LONG).show();

    }

}
```

- Add Service.java file to AndroidManifest.xml File

- ```
  <?xml version="1.0" encoding="utf-8"?>
  <manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.serviceexample1">

      <application
          android:allowBackup="true"
          android:icon="@mipmap/ic_launcher"
          android:label="@string/app_name"
          android:roundIcon="@mipmap/ic_launcher_round"
          android:supportsRtl="true"
          android:theme="@style/Theme.ServiceExample1">
          <activity android:name=".MainActivity">
              <intent-filter>
                  <action
  android:name="android.intent.action.MAIN" />

                  <category
  android:name="android.intent.category.LAUNCHER" />
              </intent-filter>
          </activity>
          <service android:name=".MyService"/>
      </application>

  </manifest>
  ```

- Activity_main.xml File with start & stop service button

```
<?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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```
```

```
        android:id="@+id/nameTV"
        android:text="SERVICE EXAMPLE"
        android:textSize="30dp"
        android:layout_margin="50dp"
        android:layout_centerHorizontal="true"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="START"
        android:id="@+id/startBtn"
        android:textSize="10dp"
        android:layout_below="@+id/nameTV"
        android:layout_margin="20dp"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="STOP"
        android:id="@+id/stopBtn"
        android:textSize="10dp"
        android:layout_below="@+id/nameTV"
        android:layout_margin="20dp"
        android:layout_alignParentEnd="true"/>
</RelativeLayout>
```

- **MainActivity.java File**

```
- package com.example.serviceexample1;

  import androidx.appcompat.app.AppCompatActivity;

  import android.content.Intent;
  import android.os.Bundle;
  import android.util.Log;
  import android.view.View;
  import android.widget.Button;

  public class MainActivity extends AppCompatActivity {

      String msg = "Android";
      Button startBtn, stopBtn;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
          Log.d(msg,"OnCreate Method Called");

          startBtn = findViewById(R.id.startBtn);
          stopBtn = findViewById(R.id.stopBtn);
          startBtn.setOnClickListener(new View.OnClickListener()
  {
              @Override
              public void onClick(View v) {

                  startService(new
  Intent(getBaseContext(),MyService.class));
              }
```

```
        });

        stopBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                stopService(new
Intent(getBaseContext(),MyService.class));
            }
        });
    }
}
```

8. Broadcast Receiver

- Broadcast in android is the system-wide events that can occur when the device starts, when a message is received on the device or when incoming calls are received, or when a device goes to airplane mode, etc.
- Broadcast Receivers are used to respond to these system-wide events.
- Broadcast Receivers allow us to register for the system and application events, and when that event happens, then the register receivers get notified.
- **There are mainly two types of Broadcast Receivers:**
- **Static Broadcast Receivers:** These types of Receivers are declared in the manifest file and works even if the app is closed.
- **Dynamic Broadcast Receivers:** These types of receivers work only if the app is active or minimized.

Following are some of the important system wide generated intents.

1. **android.intent.action.BATTERY_LOW** : Indicates low battery condition on the device.
2. **android.intent.action.BOOT_COMPLETED** : This is broadcast once, after the system has finished booting
3. **android.intent.action.CALL** : To perform a call to someone specified by the data
4. **android.intent.action.DATE_CHANGED** : The date has changed
5. **android.intent.action.REBOOT** : Have the device reboot
6. **android.net.conn.CONNECTIVITY_CHANGE** : The mobile network or wifi connection is changed(or reset)

Program For STATIC BROADCAST RECEIVER

AIM: Our aim is to write a program for Static broadcast receiver, where Static broadcast receiver receives event when : Boot completed & Connectivity change

For this First Add

```
<receiver android:name=".ExampleBroadcastReceiver">

    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
    </intent-filter>
</receiver>
```

Into manifest file.

File1: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.broadcastreceiverstatic">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BroadCastReceiverStatic">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <receiver android:name=".ExampleBroadcastReceiver">

            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED"/>
                <action
```

```
android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
            </intent-filter>
        </receiver>
    </application>

</manifest>
```

Create Broadcast Receiver for Static

File 2: ExampleBroadcastReceiver.java

```
package com.example.broadcastreceiverstatic;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.widget.Toast;

public class ExampleBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {

        if(Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction()))
        {
            Toast.makeText(context,"Boot Completed",
Toast.LENGTH_LONG).show();
        }

if(ConnectivityManager.CONNECTIVITY_ACTION.equals(intent.getAction()))
        {
            Toast.makeText(context,"Connectivity Changed",
Toast.LENGTH_LONG).show();
        }
    }
}
```

File 3: Keep MainActivity.java File As it is.

```
package com.example.broadcastreceiverstatic;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

```
    }
}
```

File 4 : Keep activity_main.xml File as it is

```
package com.example.broadcastreceiverstatic;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

Program 2: For Dynamic Broadcast Receiver

Aim : This will work when internet connection get changed.

There is no need to add anything in Manifest file.

We will register broadcast receiver & unregister them as well as in MainActivity.java File

File 1: No Change in Android Manifest.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.broadcastreceiverdynamic">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BroadcastReceiverDynamic">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>
```

```
</manifest>
```

File 2: BroadCastReceiver file : ExampleBroadcastReceiver

```
package com.example.broadcastreceiverdynamic;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.renderscript.ScriptIntrinsicBlend;
import android.widget.Toast;


public class ExampleBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {


if(ConnectivityManager.CONNECTIVITY_ACTION.equals(intent.getAction())){


            boolean noConnectivity =
intent.getBooleanExtra(ConnectivityManager.EXTRA_NO_CONNECTIVITY,false);

            if(noConnectivity)
            {
                Toast.makeText(context,"No Connectivity",
Toast.LENGTH_LONG).show();
            }
            else{
                Toast.makeText(context,"Connected",
Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

File 3: MainActivity.java File

```
package com.example.broadcastreceiverdynamic;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.IntentFilter;
import android.net.ConnectivityManager;
import android.net.Uri;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    ExampleBroadcastReceiver exampleBroadcastReceiver = new
ExampleBroadcastReceiver();
```

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

    @Override
    protected void onStart() {
        super.onStart();
        /*Rhe fololowing code is used to work broadcast reciever duting
forground activity
        * if you want to run it in background then :
        * Register Braodcast Reciever during onCreate & unregister in
Destroy.
        *
        * In this We register Broadcast Reciever during onStart &
unregister in Destroy. */


        IntentFilter filter = new
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);

        // We can also do this :-->
filter.addAction(ConnectivityManager.CONNECTIVITY_ACTION);

        registerReceiver(exampleBroadcastReceiver,filter);

    }


    @Override
    protected void onStop() {
        super.onStop();
        unregisterReceiver(exampleBroadcastReceiver);
    }
}
```

File 4: activity_main.xml File

```
<?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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Program 3: Custom Broadcast Receiver [Going to make 2 projects one for sender one for reciever]

Here we are going to make custom broadcast receiver, through which we Send a Message From One APP ---- The Second app will receive it & we can see it.

For this You need to run both projects in Split Screen or Run the broadcast receiver un register in activity On Create & on Destroy.

Here we are also going to send Broad cast message in one app as well as From one app to another app.

Project 1: for BroadcastSender (This will do Inter app message send and receive as well as send to other app)

File 1: AndroidManifest.xml [Keep this file as it is]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.broadcastsender">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BroadcastSender">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File 2: 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:orientation="vertical"
    android:gravity="center"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:id="@+id/textView"
        android:layout_height="wrap_content"
        android:text="Message!"
        android:textSize="20dp"
        android:layout_margin="20dp"

        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/sendBroadcastBtn"
        android:textSize="15dp"
        android:text="Send Broadcast"
        />
</LinearLayout>
```

File 3: MainActivity.java File

```
package com.example.broadcastsender;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.view.ViewCompat;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView tv;
    Button sendBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        tv = findViewById(R.id.textView);
        sendBtn = findViewById(R.id.sendBroadcastBtn);



        sendBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent("com.example.TEST_ACTION");
                intent.putExtra("com.example.TEST_ACTION","Broadcast
received");
                sendBroadcast(intent);
            }
        });
    }

// Here I have created Anonymous Inner Class for Broadcast Receiver as
given below .
// You can  create that class in the same way as previous programs.

    private BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String receivedText =
intent.getStringExtra("com.example.TEST_ACTION");
            tv.setText(receivedText);
        }
    };

    @Override
    protected void onStart() {

        super.onStart();
        IntentFilter filter = new IntentFilter("com.example.TEST_ACTION");
        registerReceiver(broadcastReceiver,filter);

    }

    @Override
    protected void onStop() {
        super.onStop();
        unregisterReceiver(broadcastReceiver) ;
    }
}
```

Project 2: for BroadcastReciever (This will receive message from First app project that we have ran previous)

File 1: Androidmanifest.xml File as it is.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.broadcastreceivercustom">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BroadcastReceiverCustom">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File2: Create a class that extends BroadcastReceiver, ExampleCustomBroadCastReceiver.java

```
package com.example.broadcastreceivercustom;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.widget.Toast;

public class ExampleCustomBroadCastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if("com.example.TEST_ACTION".equals(intent.getAction())){
            String receivedText =
intent.getStringExtra("com.example.TEST_ACTION");
            Toast.makeText(context,receivedText,Toast.LENGTH_LONG).show();

          }

    }
}
```

File3: MainActivity.java File for Register & Unregister Broadcast Receiver

```
package com.example.broadcastreceivercustom;

import androidx.appcompat.app.AppCompatActivity;

import android.content.IntentFilter;
import android.net.ConnectivityManager;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    ExampleCustomBroadCastReceiver exampleCustomBroadCastReceiver = new
ExampleCustomBroadCastReceiver();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        IntentFilter filter = new IntentFilter("com.example.TEST_ACTION");
        registerReceiver(exampleCustomBroadCastReceiver,filter);
    }


    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(exampleCustomBroadCastReceiver);
    }
}
```

9. Text To Speech

Android allows you convert your text into voice. Not only you can convert it but it also allows you to speak text in variety of different languages.

Android provides TextToSpeech class for this purpose. In order to use this class, you need to instantiate an object of this class and also specify the initListener. Its syntax is given below

```
tts = new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if(status!= TextToSpeech.ERROR)
        {
            tts.setLanguage(Locale.UK);
        }
    }
});
```

To set the language : **tts.setLanguage(Locale.US);**

Methods in TEXT TO SPEECH

1
addSpeech(String text, String filename)

This method adds a mapping between a string of text and a sound file.

2
getLanguage()

This method returns a Locale instance describing the language.

3
isSpeaking()

This method checks whether the TextToSpeech engine is busy speaking.

4
setPitch(float pitch)

This method sets the speech pitch for the TextToSpeech engine.

5
setSpeechRate(float speechRate)

This method sets the speech rate.

6
shutdown()

This method releases the resources used by the TextToSpeech engine.

7
stop()

This method stop the speak.

Example Program:

AIM : We need to write a code for an app that has one Edit text, we will write content in it and by clicking on Text To Speech, It will speak out the content that your have written in edit text

File1: 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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Text To Speech Ex"
        android:layout_centerHorizontal="true"
        android:textSize="20dp"
        android:layout_margin="20dp"
        android:id="@+id/titleTextView"
         />
```

```
    <EditText
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:ems="15"
        android:id="@+id/textEdt"
        android:layout_below="@id/titleTextView"
        android:layout_margin="20dp"
        android:textColor="#AAFFDD"
        android:textColorHint="#AADDAA"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textToSpeechBtn"
        android:text="Text To Speech"
        android:layout_below="@+id/textEdt"
        android:layout_centerHorizontal="true"
        android:layout_margin="20dp"
        />

</RelativeLayout>
```

File2: MainActivity.xml

```
package com.example.texttospeechex1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    EditText textEdt;
    Button ttsBtn;
    TextToSpeech tts;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        textEdt = findViewById(R.id.textEdt);
        ttsBtn = findViewById(R.id.textToSpeechBtn);

        tts = new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if(status!= TextToSpeech.ERROR)
                {
                    tts.setLanguage(Locale.UK);
                }
            }
        });

        ttsBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String toSpeak = textEdt.getText().toString();
                Toast.makeText(MainActivity.this, "Data = "+toSpeak,
Toast.LENGTH_SHORT).show();
                tts.speak(toSpeak,TextToSpeech.QUEUE_FLUSH,null);
            }
        });
    }

    @Override
    protected void onPause() {
        super.onPause();
        if(tts!=null)
        {
            tts.stop();
            tts.shutdown();
        }
    }
}
```

10. Working with Camera & Taking Picture with Camera

#1 Using Camera By Using Camera Application

We can capture pictures without using the instance of Camera class. Here you will use an intent action type of MediaStore.ACTION_IMAGE_CAPTURE to launch an existing Camera application on your phone. In Android MediaStore is a type of DataBase which stores pictures and videos in android.

```
Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE)
;
```

#2 Using Camera By using Camera Api
This class is used for controlling device cameras. It can be used to take pictures when you are building a camera application.
Camera API works in following ways:

1. Camera Manager: This is used to get all the cameras available in the device like front camera back camera each having the camera id.

2. CameraDevice: You can get it from Camera Manager class by its id.

3. CaptureRequest: You can create a capture request from camera device to capture images.

4. CameraCaptureSession: To get capture request's from Camera Device create a CameraCaptureSession.

5. CameraCaptureSession.CaptureCallback: This is going to provide the Capture session results.

    - Camera Permission Declarations In Manifest
      First, you should declare the Camera requirement in your Manifest file if Camera is compulsory for your application and you don't want your application to be installed on a device that does not support Camera.

```
uses-permission android:name="android.permission.CAMERA"/>
```

Program For Capture Image with Camera

File 1: Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cameraex1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
```

```
        android:supportsRtl="true"
        android:theme="@style/Theme.CameraEx1">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File 2: MainActivity.java

```
package com.example.cameraex1;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

    private static final int CAMERA_REQUEST = 1008;
    ImageView imgview;
    Button cameraBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        cameraBtn = findViewById(R.id.cameraBtn);
        imgview = findViewById(R.id.imageView1);

        cameraBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent cameraIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(cameraIntent,CAMERA_REQUEST);
            }
        });


    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {

        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CAMERA_REQUEST) {
```

```
            Bitmap photo = (Bitmap) data.getExtras().get("data");
            imgview.setImageBitmap(photo);

        }
    }
}
```

11. Manage Bluetooth Connection

Bluetooth is a way to exchange data with other devices wirelessly. Android provides Bluetooth API to perform several tasks such as:

- scan bluetooth devices
- connect and transfer data from and to other devices
- manage multiple connections etc.

Android Bluetooth API

The android.bluetooth package provides a lot of interfaces classes to work with bluetooth such as:

- BluetoothAdapter
- BluetoothDevice
- BluetoothSocket
- BluetoothServerSocket
- BluetoothClass
- BluetoothProfile
- BluetoothProfile.ServiceListener
- BluetoothHeadset
- BluetoothA2dp
- BluetoothHealth
- BluetoothHealthCallback
- BluetoothHealthAppConfiguration

Methods of BluetoothAdapter class

Commonly used methods of BluetoothAdapter class are as follows:

- **static synchronized BluetoothAdapter getDefaultAdapter()** returns the instance of BluetoothAdapter.
- **boolean enable()** enables the bluetooth adapter if it is disabled.
- **boolean isEnabled()** returns true if the bluetooth adapter is enabled.
- **boolean disable()** disables the bluetooth adapter if it is enabled.
- **String getName()** returns the name of the bluetooth adapter.
- **boolean setName(String name)** changes the bluetooth name.
- **int getState()** returns the current state of the local bluetooth adapter.
- **Set<BluetoothDevice> getBondedDevices()** returns a set of paired (bonded) BluetoothDevice objects.
- **boolean startDiscovery()** starts the discovery process.

Program : Bluetooth Functionality & Management Program – First need to write Permissions in AndroidManifest.xml File

File1: AndroidManifest.xml File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bluetoothex1">

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BluetoothEx1">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
```

```
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File 2: activity_main.xml File

```
<?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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bluetooth Example"
        android:id="@+id/titleTextView"
        android:layout_centerHorizontal="true"
        android:layout_margin="30dp"
        android:textSize="20dp"

       />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Turn ON"
        android:id="@+id/onBtn"
        android:layout_margin="20dp"
       android:layout_alignParentLeft="true"
        android:layout_below="@+id/titleTextView"
        />


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Get Visible"
        android:id="@+id/visibleBtn"
        android:layout_margin="10dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/onBtn"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="List Devices"
        android:id="@+id/listDeviceBtn"
        android:layout_margin="10dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/visibleBtn"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Turn OFF"
```

```
        android:id="@+id/offBtn"
        android:layout_margin="20dp"

       android:layout_alignParentRight="true"
        android:layout_below="@id/titleTextView"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/listDeviceBtn"
        android:layout_centerHorizontal="true"
        android:text="Paired Devices List"
        android:id="@+id/pairedTextView"
        android:textSize="30dp"
        android:textStyle="bold|italic"
        />
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listOfDevices"
        android:layout_below="@+id/pairedTextView"
        android:divider="@color/black"
        android:dividerHeight="2dp"
        android:layout_margin="30dp"
        />

        />


</RelativeLayout>
```

File 3: MainActivity.java

```
package com.example.bluetoothex1;

import androidx.appcompat.app.AppCompatActivity;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.Set;

public class MainActivity extends AppCompatActivity {

    Button onBtn, offBtn, visibleBtn, pairedDevicesBtn;
```

```
    ListView listOfPairedDevices;
    private BluetoothAdapter BA;
    private Set<BluetoothDevice> pairedDevices;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        onBtn = findViewById(R.id.onBtn);
        offBtn = findViewById(R.id.offBtn);
        visibleBtn = findViewById(R.id.visibleBtn);
        pairedDevicesBtn = findViewById(R.id.listDeviceBtn);
        listOfPairedDevices = findViewById(R.id.listOfDevices);
        BA = BluetoothAdapter.getDefaultAdapter();
        onBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!BA.isEnabled())
                {
                    Intent turnOn = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                    startActivity(turnOn);
                    Toast.makeText(MainActivity.this, "Turned ON",
Toast.LENGTH_SHORT).show();
                }
                else {
                    Toast.makeText(MainActivity.this, "Already ON",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        offBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                BA.disable();
                Toast.makeText(MainActivity.this, "Turned off",
Toast.LENGTH_SHORT).show();
            }
        });

        visibleBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent getVisible = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);

                startActivityForResult(getVisible,0);
            }
        });

        pairedDevicesBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pairedDevices = BA.getBondedDevices();
                ArrayList list = new ArrayList();
                for(BluetoothDevice bt : pairedDevices)
                {
                    list.add(bt.getName());
                }
```

```
                Toast.makeText(MainActivity.this, "Showing Paired Devices",
Toast.LENGTH_SHORT).show();

                final ArrayAdapter adapter = new
ArrayAdapter(getApplicationContext(),
android.R.layout.simple_list_item_1,list);

                listOfPairedDevices.setAdapter(adapter);
            }
        });


    }
}
```

12. Manage & Monitor WIFI

- Android allows applications to access to view the access the state of the wireless connections at very low level.
- Application can access almost all the information of a wifi connection.
- The information that an application can access includes connected network's link speed,IP address, negotiation state, other networks information.
- Applications can also scan, add, save, terminate and initiate Wi-Fi connections.

WIFI – Methods & Description

1

addNetwork(WifiConfiguration config)

This method add a new network description to the set of configured networks.

2

createWifiLock(String tag)

This method creates a new WifiLock.

3

disconnect()

This method disassociate from the currently active access point.

4

enableNetwork(int netId, boolean disableOthers)

This method allow a previously configured network to be associated with.

5

getWifiState()

This method gets the Wi-Fi enabled state

6

isWifiEnabled()

This method return whether Wi-Fi is enabled or disabled.

7

setWifiEnabled(boolean enabled)

This method enable or disable Wi-Fi.

8

updateNetwork(WifiConfiguration config)

This method update the network description of an existing configured network.

Program For WIFI

File 1 : AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.wifiexample">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WifiExample">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

File 2: 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"
```

```
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Wifi Manager Example"
        android:layout_centerHorizontal="true"
        android:textSize="20dp"
        android:layout_margin="30dp"
        android:id="@+id/titleTextView"
         />

    <Button
        android:id="@+id/onWifiBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Enable Wifi"
        android:layout_margin="20dp"
        android:layout_below="@+id/titleTextView" />

    <Button
        android:id="@+id/offWifiBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Enable Wifi"
        android:layout_margin="20dp"
        android:layout_below="@+id/onWifiBtn" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/showWifiListBtn"
        android:layout_below="@+id/offWifiBtn"
        android:layout_centerHorizontal="true"
        android:text="Show WIFI"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/showWifiTextView"
        android:layout_below="@+id/showWifiListBtn"/>
</RelativeLayout>
```

File 3: MainActivity.java

```
package com.example.wifiexample;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
import android.widget.Toast;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    Button onBtn, offBtn, showBtn;
    WifiManager wifiManager;
    TextView showWifiListTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        onBtn = findViewById(R.id.onWifiBtn);
        offBtn = findViewById(R.id.offWifiBtn);
        showBtn = findViewById(R.id.showWifiListBtn);

        showWifiListTextView = findViewById(R.id.showWifiTextView);
        wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);


        onBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                wifiManager.setWifiEnabled(true);
                Toast.makeText(MainActivity.this, "WIFI ENABLED",
Toast.LENGTH_SHORT).show();

            }
        });


        offBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                wifiManager.setWifiEnabled(false);
               Toast.makeText(MainActivity.this, "WIFI DISABLED",
Toast.LENGTH_SHORT).show();
            }
        });

/*
        showBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                WifiManager wmgr =
(WifiManager)getApplicationContext().getSystemService(Context.WIFI_SERVICE)
;
// Get List of Available Wifi Networks
                List<ScanResult> availNetworks = wmgr.getScanResults();
                if (availNetworks.size() > 0) {
                    String wifis[] = new String[availNetworks.size()];
                    // Get Each network detail
                    for (int i=0; i<availNetworks.size();i++) {

                        wifis[i] = availNetworks.get(i).toString();
                        showWifiListTextView.setText(wifis[i]+"\n");
```

```
                    }
                }
            }
        });*/
    }
}
```

13. Accelerometer Sensor & Gyroscope

File1: 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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="List of Sensors"
        android:id="@+id/titleTextView"
        android:layout_centerHorizontal="true"
        android:textSize="30dp"
        android:layout_margin="20dp"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Data"
        android:textSize="20dp"
        android:layout_centerHorizontal="true"
        android:id="@+id/sensorListTextView"
        android:layout_below="@+id/titleTextView"
        />

</RelativeLayout>
```

File 2: MainActivity.xml File

```
package com.example.sensorex1;

import androidx.appcompat.app.AppCompatActivity;

import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.TextView;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    TextView sensorListTV;
    private SensorManager sensorManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorListTV = findViewById(R.id.sensorListTextView);
        sensorListTV.setVisibility(View.GONE);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        List<Sensor> list = sensorManager.getSensorList(Sensor.TYPE_ALL);


        for(int i = 1; i < list.size(); i++)
        {
            sensorListTV.setVisibility(View.VISIBLE);
            sensorListTV.append("\n-------------
"+list.get(i).getName()+"\n==> " +

""+list.get(i).getVendor()+"\n==>"+list.get(i).getVersion());


        }
    }
}
```