1.	Create the standard calculator application in android 	30
2.	Create a simple "Hello World" application in Android Studio. Explain the purpose of the MainActivity and the activity_main.xml file.	10
3.	Viva	5
4.	Journal	5

✅ Step 1: Create a New Android Project

1. Open Android Studio → Click on **“New Project”**
2. Choose **“Empty Activity”**
3. Name it: CalculatorApp
4. Language: **Java** (or Kotlin if you're asked, let me know)
5. Click **Finish**

✅ Step 2: Design UI in activity_main.xml

Go to res/layout/activity_main.xml and paste this code:

xml

CopyEdit

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
```

```
    android:padding="10dp"
    android:background="#ECECEC">

    <EditText
        android:id="@+id/etInput"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:gravity="right"
        android:layout_marginBottom="10dp"
        android:textSize="32sp"
        android:inputType="none"
        android:background="#FFFFFF" />

    <GridLayout
        android:id="@+id/gridLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:columnCount="4"
        android:rowCount="5"
        android:layout_gravity="center">

        <!-- Buttons -->
        <Button android:text="DEL" android:onClick="onClick"
style="@style/CalcButton"/>
        <Button android:text="C" android:onClick="onClick"
style="@style/CalcButton"/>
        <Button android:text="%" android:onClick="onClick"
style="@style/CalcButton"/>
```

```
    <Button android:text="+" android:onClick="onClick"
style="@style/CalcButton"/>

    <Button android:text="1" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="2" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="3" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="/" android:onClick="onClick" style="@style/CalcButton"/>

    <Button android:text="4" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="5" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="6" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="-" android:onClick="onClick" style="@style/CalcButton"/>

    <Button android:text="7" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="8" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="9" android:onClick="onClick"
style="@style/CalcButton"/>
    <Button android:text="X" android:onClick="onClick"
style="@style/CalcButton"/>

    <Button android:text="." android:onClick="onClick" style="@style/CalcButton"/>
    <Button android:text="0" android:onClick="onClick"
style="@style/CalcButton"/>
```

```
    <Button android:text="=" android:onClick="onClick" style="@style/CalcButton"
android:layout_columnSpan="2"/>

  </GridLayout>

</LinearLayout>
```

✅ **Step 3: Add Button Style in res/values/styles.xml**

xml

CopyEdit

```
<style name="CalcButton">
  <item name="android:layout_width">0dp</item>
  <item name="android:layout_height">wrap_content</item>
  <item name="android:layout_margin">4dp</item>
  <item name="android:layout_columnWeight">1</item>
  <item name="android:textSize">22sp</item>
  <item name="android:background">#D3D3D3</item>
</style>
```

✅ **Step 4: Add Logic in MainActivity.java**

Go to java > your package > MainActivity.java and paste this:

java

```
package com.example.calculatorapp;

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

```
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    EditText etInput;
    String input = "", operator = "", lastNumber = "";

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

        etInput = findViewById(R.id.etInput);
    }

    public void onClick(View v) {
        Button button = (Button) v;
        String value = button.getText().toString();

        switch (value) {
            case "C":
                input = "";
                operator = "";
                lastNumber = "";
                break;

            case "DEL":
                if (!input.isEmpty()) {
```

```
            input = input.substring(0, input.length() - 1);
        }
        break;

    case "=":
        solve();
        break;

    case "X":
    case "/":
    case "+":
    case "-":
    case "%":
        if (!input.isEmpty() && !hasOperator(input)) {
            operator = value;
            input += value;
        }
        break;

    default:
        input += value;
  }

  etInput.setText(input);
}

private boolean hasOperator(String s) {
```

```
        return s.contains("+") || s.contains("-") || s.contains("X") || s.contains("/") ||
s.contains("%");
    }

    private void solve() {
        if (operator.isEmpty()) return;

        String[] tokens = input.split("\\" + operator);
        if (tokens.length < 2) return;

        double result = 0;
        double num1 = Double.parseDouble(tokens[0]);
        double num2 = Double.parseDouble(tokens[1]);

        switch (operator) {
            case "+":
                result = num1 + num2; break;
            case "-":
                result = num1 - num2; break;
            case "X":
                result = num1 * num2; break;
            case "/":
                result = num2 != 0 ? num1 / num2 : 0; break;
            case "%":
                result = num1 % num2; break;
        }

        input = String.valueOf(result);
```

```
        operator = "";
    }
}
```

✅ **Step 5: Run the App**

- Click the **green “Run” button**
- Choose your emulator or real device
- ✅ You’ll see a working **calculator app**

Q.2

✅ **1. Steps to Create "Hello World" App in Android Studio**

🧩 **Step-by-Step:**

📦 **Step 1: Open Android Studio**

- Click **“Start a new Android Studio project”**
- Choose **“Empty Activity”**
- Name your app (e.g., HelloWorldApp)
- Click **Finish**

📄 **MainActivity.java (Java)**

java

CopyEdit

```
package com.example.helloworldapp;

import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); // links to the XML layout
    }
}
```

🖼️ activity_main.xml

```
xml
CopyEdit
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="24sp" />
</LinearLayout>
```

📘 Explanation

🔹 What is MainActivity.java?

- It's the **starting point** of your app.
- Contains the code that runs when the app launches.
- Uses setContentView() to show the UI from activity_main.xml.

🔹 What is activity_main.xml?

- It defines the **UI layout** of the app.
- In this case, it displays a TextView with "Hello World!".
- XML is used for designing the interface visually or with code.

1.	Create an android application to pass the data from one activity to another activity in the same application using intent.	10
2.	Create a user interface with a TextView, Button, and ImageView. When the button is clicked, the image should be changed dynamically.	30
3.	Viva	5
4.	Journal	5

✅ Question 1: Pass data from one activity to another using Intent

🔧 Step-by-Step:

🧩 Step 1: Create two activities

- MainActivity.java
- SecondActivity.java

🖼️ activity_main.xml

xml

CopyEdit

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name" />

    <Button
        android:id="@+id/buttonSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Data" />

</LinearLayout>
```

🧠 **MainActivity.java**

java

CopyEdit

```
package com.example.intentexample;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    EditText editTextName;
    Button buttonSend;

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

        editTextName = findViewById(R.id.editTextName);
        buttonSend = findViewById(R.id.buttonSend);

        buttonSend.setOnClickListener(v -> {
            String name = editTextName.getText().toString();
            Intent intent = new Intent(MainActivity.this, SecondActivity.class);
            intent.putExtra("user_name", name);
            startActivity(intent);
        });
    }
}
```

🖼️ **activity_second.xml**

xml

CopyEdit

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textViewResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp" />
</LinearLayout>
```

🎯 **SecondActivity.java**

java

CopyEdit

```
package com.example.intentexample;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

    TextView textViewResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);

    textViewResult = findViewById(R.id.textViewResult);
    String name = getIntent().getStringExtra("user_name");

    textViewResult.setText("Hello, " + name + "!");
  }
}
```

✅ **Update AndroidManifest.xml**

```
xml
CopyEdit
<activity android:name=".SecondActivity"></activity>
```

✅ **Question 2: TextView, Button, and ImageView. Change image dynamically.**

🎨 **activity_main.xml**

```
xml
CopyEdit
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="16dp">

  <TextView
    android:id="@+id/textView"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click button to change image"
    android:textSize="20sp"
    android:layout_gravity="center"/>

  <ImageView
    android:id="@+id/imageView"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_gravity="center"
    android:src="@drawable/image1" />

  <Button
    android:id="@+id/buttonChange"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Change Image"
    android:layout_gravity="center" />

</LinearLayout>
```

📦 **Place 2 images in res/drawable**

Use names like image1.png, image2.png

💻 **MainActivity.java**

```
java
CopyEdit
package com.example.changeimage;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    ImageView imageView;
    Button buttonChange;
    boolean imageChanged = false;

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

        imageView = findViewById(R.id.imageView);
        buttonChange = findViewById(R.id.buttonChange);

        buttonChange.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (imageChanged) {
                    imageView.setImageResource(R.drawable.image1);
                } else {
                    imageView.setImageResource(R.drawable.image2);
```

```
        }
            imageChanged = !imageChanged;
        }
    });
  }
}
```

1.	Explain the Android Activity life cycle in detail. Implement a simple app that logs the lifecycle methods (onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()).	40
2.	Viva	5
3.	Journal	5

📄 **Question:**

Explain the Android Activity life cycle in detail. Implement a simple app that logs the lifecycle methods (onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()).

✅ **Part 1: Explanation of Android Activity Lifecycle**

Android activities have **7 main lifecycle methods** that manage the state of an app:

Method	Purpose
onCreate()	Called when the activity is first created (initial setup).
onStart()	Activity is becoming visible to the user.
onResume()	Activity is now interacting with the user.

Method	Purpose
onPause()	Called when activity is partially hidden (like on screen rotation or popup).
onStop()	Called when activity is no longer visible.
onRestart()	Called after onStop(), before starting again.
onDestroy()	Final call before the activity is destroyed.

✅ Part 2: Steps to Create the App in Android Studio

🛠️ 1. Create a New Project

- Open Android Studio
- Click: **“New Project” → Empty Activity**
- Name it: ActivityLifeCycleApp
- Click **Finish**

📓 2. Java Code for Logging Lifecycle

📄 MainActivity.java

java

CopyEdit

```
package com.example.activitylifecycleapp;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "ActivityLifecycle";
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate() called");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "onStart() called");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d(TAG, "onResume() called");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(TAG, "onPause() called");
    }

    @Override
    protected void onStop() {
```

```
    super.onStop();
    Log.d(TAG, "onStop() called");
  }

  @Override
  protected void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "onDestroy() called");
  }

  @Override
  protected void onRestart() {
    super.onRestart();
    Log.d(TAG, "onRestart() called");
  }
}
```

📄 3. Layout XML (Optional)

📄 activity_main.xml

```
xml
CopyEdit
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:gravity="center"
  android:padding="16dp">
```

```
    <TextView
        android:id="@+id/lifecycleText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Check Logcat for lifecycle logs"
        android:textSize="20sp"
        android:textColor="#000000" />
</LinearLayout>
```

✅ 4. Run the App and Check Lifecycle

- Run the app on emulator or real device.
- Open **Logcat** (bottom of Android Studio).
- Filter by ActivityLifecycle.
- You will see logs like:

```
scss
CopyEdit
onCreate() called
onStart() called
onResume() called
```

Rotate the device or press Home/Back buttons to see other methods being called.

✅ Conclusion:

You have now:

- Learned the full Android Activity Lifecycle.
- Implemented a working app that logs each method in Logcat.

Seat No: Max. Marks: 50

1.	Insert the new contents in the following resources and demonstrate their uses in the android application Android Resources: (Color, Theme, String, Drawable, Dimension, Image)	20
2.	Create an android application which automatically notify the user when Aeroplane mode is turned on or off using broadcast receiver.	20
3.	Viva	5
4.	Journal	5

✅ Q1: Insert contents in these resources and demonstrate in the app:

Resources: Color, Theme, String, Drawable, Dimension, Image
Marks: 20

🎯 Goal:

Create an app that uses all these resources in the UI.

🧱 Step-by-Step Implementation:

➤ 1. Create a New Project

- File → New → New Project → Empty Activity → Name it ResourceDemoApp

---

➤ 2. Add Resource Files

🔸 res/values/colors.xml:

xml

CopyEdit

```
<resources>
    <color name="my_background">#E0F7FA</color>
    <color name="my_text_color">#00796B</color>
</resources>
```

🔸 res/values/strings.xml:

xml

CopyEdit

<resources>

  <string name="app_name">ResourceDemoApp</string>

  <string name="greeting">Hello, Resources!</string>

</resources>

🔹 **res/values/dimens.xml:**

xml

CopyEdit

<resources>

  <dimen name="text_size">24sp</dimen>

  <dimen name="padding">16dp</dimen>

</resources>

🔹 **Add Image in res/drawable/:**

- Right click res/drawable → New → Image Asset OR paste an image like logo.png in drawable/.

🔹 **res/values/themes.xml:**

Change primary color:

xml

CopyEdit

<item name="colorPrimary">#4CAF50</item>

➤ **3. Update activity_main.xml**

xml

CopyEdit

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

  xmlns:android="http://schemas.android.com/apk/res/android"

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:background="@color/my_background"
    android:padding="@dimen/padding">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/greeting"
        android:textSize="@dimen/text_size"
        android:textColor="@color/my_text_color"/>

    <ImageView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/logo"/>
</LinearLayout>
```

✅ Q2: Create an app to notify when Aeroplane mode is ON/OFF using BroadcastReceiver

Marks: 20

🧱 Step-by-Step Implementation:

➤ 1. Create a New Project

- File → New → New Project → Empty Activity → Name it AirplaneModeApp

---

➤ 2. Create Broadcast Receiver

📄 AirplaneReceiver.java

java

CopyEdit

```
package com.example.airplanemodeapp;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.provider.Settings;
import android.widget.Toast;

public class AirplaneReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        boolean isAirplaneModeOn = Settings.Global.getInt(
                context.getContentResolver(),
                Settings.Global.AIRPLANE_MODE_ON, 0) != 0;

        if (isAirplaneModeOn) {
            Toast.makeText(context, "Airplane Mode is ON",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(context, "Airplane Mode is OFF",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

➤ 3. Register Receiver in AndroidManifest.xml

xml

CopyEdit

```
<receiver android:name=".AirplaneReceiver">
    <intent-filter>
        <action android:name="android.intent.action.AIRPLANE_MODE"/>
    </intent-filter>
</receiver>
```

➤ 4. Layout: activity_main.xml

xml

CopyEdit

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="20dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Toggle Airplane Mode to test"
        android:textSize="20sp"/>
</LinearLayout>
```

✅ How to Test It

- Run the app.
- Pull down system settings and toggle Airplane Mode.
- You'll see a toast like: **"Airplane Mode is ON/OFF"**

✅ Conclusion:

You now have:

1. **Used Android resources**: color, theme, string, dimension, drawable.
2. **Built a BroadcastReceiver app** to detect Airplane Mode status change.

1.	Create a **RelativeLayout** for a profile screen with: • A **Profile Image** centered at the top of the screen. • A **TextView** displaying the name below the profile image, aligned to the center. • A **Button** below the name, aligned to the center.	20
2.	Write a program that logs the different lifecycle methods of an **Activity** (e.g., onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()) to the logcat.	20
3.	Viva	5
4.	Journal	5

✅ Q1. Create a RelativeLayout for a profile screen

Requirements:

- A **Profile Image** centered at the top
- A **TextView** below the image, centered
- A **Button** below the TextView, also centered

Marks: 20

🎯 Goal: Build a simple profile layout using RelativeLayout

🧱 Step-by-Step Implementation

➤ 1. Open or Create New Android Project

- File → New → New Project → Empty Activity → Name it ProfileScreenApp

---

➤ 2. Replace activity_main.xml content

xml

CopyEdit

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

```
    android:padding="20dp"
    android:background="#F9F9F9">

    <ImageView
        android:id="@+id/profileImage"
        android:layout_width="120dp"
        android:layout_height="120dp"
        android:src="@drawable/ic_profile"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:contentDescription="Profile Image" />

    <TextView
        android:id="@+id/profileName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="John Doe"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_below="@id/profileImage"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp" />

    <Button
        android:id="@+id/btnEditProfile"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Edit Profile"
```

android:layout_below="@id/profileName"

android:layout_centerHorizontal="true"

android:layout_marginTop="20dp" />

</RelativeLayout>

🔹 Add any dummy image in res/drawable/ folder and name it ic_profile.png or use Android Studio's default vector icon.

✅ Q2. Write a program that logs all activity lifecycle methods

Methods:

- onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()
  Marks: 20

🎯 Goal: Show log messages for each activity lifecycle method

🧱 Step-by-Step Implementation

➤ 1. Open your existing MainActivity.java

Update the file as below:

📄 MainActivity.java

java

CopyEdit

```
package com.example.lifecyclelogger;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
    private static final String TAG = "ActivityLifecycle";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d(TAG, "onCreate called");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "onStart called");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.d(TAG, "onResume called");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.d(TAG, "onPause called");
    }
```

```
    @Override
    protected void onStop() {
        super.onStop();
        Log.d(TAG, "onStop called");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d(TAG, "onDestroy called");
    }
}
```

✅ How to View Logs

- Run the app → Go to **Logcat** tab in Android Studio → Filter using ActivityLifecycle

You will see logs like:

```
makefile
CopyEdit
D/ActivityLifecycle: onCreate called
D/ActivityLifecycle: onStart called
...
```

✅ Conclusion:

1. First app uses RelativeLayout with a centered image, name, and button.
2. Second app shows logs for all Android lifecycle methods using Log.d.

1.	Design a ListView that displays a list of items. Each item should be a simple TextView with a string from an array of names. The list should be populated dynamically in your Activity.	40
2.	Viva	5
3.	Journal	5

✅ Q1. Design a ListView that displays a list of items.

🔸 **Each item = simple TextView**

🔸 **Data = array of names**

🔸 **List must be populated dynamically in your Activity**

Marks: 40

🧱 Step-by-Step Guide

🎯 Goal: Display a list of names using ListView and ArrayAdapter.

➤ Step 1: Create a New Android Project

- Open Android Studio → File → New → New Project
- Select **Empty Activity** → Name it ListViewDemo

➤ Step 2: Modify activity_main.xml

Replace the content with:

xml

CopyEdit

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <ListView
        android:id="@+id/nameListView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

➤ Step 3: Update MainActivity.java

java

CopyEdit

```
package com.example.listviewdemo;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    ListView nameListView;

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

```
        nameListView = findViewById(R.id.nameListView);

        // Array of names
        String[] names = {"Alice", "Bob", "Charlie", "David", "Eve", "Fiona", "George"};

        // Adapter to populate the list
        ArrayAdapter<String> adapter = new ArrayAdapter<>(
                this,
                android.R.layout.simple_list_item_1,
                names
        );

        // Set adapter to ListView
        nameListView.setAdapter(adapter);
    }
}
```

✅ **Output:**

A simple screen displaying a list of names like:

```
python-repl
CopyEdit
Alice
Bob
Charlie
David
...
```

Each item is a TextView created dynamically using ArrayAdapter.

🛠 Notes:

- ArrayAdapter binds the array data to the ListView.
- simple_list_item_1 is a built-in layout that uses just a single TextView.
- You don’t need to create any custom layout for this basic example.

✅ Conclusion:

- You created a **dynamic ListView** with names.
- Used **ArrayAdapter** and **array of strings** in Java code.
- No hardcoding in XML for list items.

1.	Create a **Registration Form** with the following fields: • **Username** (Text Input) • **Email Address** (Text Input) • **Gender** (Radio Buttons with options "Male" and "Female") • **Password** (Password Input) • **Submit Button**	40
2.	Viva	5
3.	Journal	5

✅ Q1: Create a Registration Form

The form must include:

- Username (Text Input)
- Email Address (Text Input)
- Gender (Radio Buttons: Male & Female)
- Password (Password Input)
- Submit Button

Marks: 40

🧱 Step-by-Step Android App

🎯 Goal: Create a simple registration form UI with basic validation (optional).

➤ Step 1: Create a New Android Project

- Open Android Studio → File → New → New Project
- Select **Empty Activity** → Name it RegistrationFormApp

➤ Step 2: Update activity_main.xml

(This defines the UI layout of the form)

xml

CopyEdit

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <EditText
            android:id="@+id/editUsername"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Username"
            android:inputType="textPersonName" />

        <EditText
            android:id="@+id/editEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email Address"
            android:inputType="textEmailAddress" />

        <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gender"
        android:textStyle="bold"
        android:layout_marginTop="10dp" />

    <RadioGroup
        android:id="@+id/radioGender"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <RadioButton
            android:id="@+id/radioMale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male" />

        <RadioButton
            android:id="@+id/radioFemale"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female" />
    </RadioGroup>

    <EditText
        android:id="@+id/editPassword"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/btnSubmit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:layout_marginTop="20dp" />

  </LinearLayout>
</ScrollView>
```

➤ Step 3: Write Java Code in MainActivity.java

(Handles form submission)

```
java
CopyEdit
package com.example.registrationformapp;

import android.os.Bundle;
import android.widget.*;
import android.view.View;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
    EditText editUsername, editEmail, editPassword;
    RadioGroup radioGender;
    Button btnSubmit;

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

        // Initialize components
        editUsername = findViewById(R.id.editUsername);
        editEmail = findViewById(R.id.editEmail);
        editPassword = findViewById(R.id.editPassword);
        radioGender = findViewById(R.id.radioGender);
        btnSubmit = findViewById(R.id.btnSubmit);

        // Set click listener
        btnSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String username = editUsername.getText().toString();
                String email = editEmail.getText().toString();
                String password = editPassword.getText().toString();

                int selectedGenderId = radioGender.getCheckedRadioButtonId();
                RadioButton selectedGender = findViewById(selectedGenderId);
                String gender = (selectedGender != null) ?
selectedGender.getText().toString() : "Not selected";
```

```
            // Show data using Toast (Optional)
            Toast.makeText(MainActivity.this,
                    "Username: " + username + "\nEmail: " + email +
                    "\nGender: " + gender + "\nPassword: " + password,
                    Toast.LENGTH_LONG).show();
        }
    });
  }
}
```

✅ **Output Preview:**

- A form with:
  - Username input
  - Email input
  - Gender selection (Male / Female)
  - Password field
  - Submit button
- On click, it shows all values in a **Toast**.

✅ **Conclusion:**

You have successfully built a **Registration Form** in Android with:

- Text input for Username and Email
- Password input
- Radio buttons for Gender
- Submit button with basic data handling

1.	Create a simple **Registration Form** with the following fields: • **Name** (Text Input) • **Email** (Text Input) • **Phone Number** (Text Input) • **Password** (Password Input) • **Confirm Password** (Password Input) • **Submit Button**	**40**
2.	Viva	**5**
3.	Journal	**5**

Step-by-Step Guide with Java

1. **Set Up Android Studio (Already Done)**
    - Ensure you have created a new project with **Empty Activity** and selected **Java** as the language during setup.
2. **Use the Same Layout**
    - Open res/layout/activity_main.xml and ensure it contains the XML code provided earlier:

xml

Copy

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Registration Form"
        android:textSize="20sp"
```

```
    android:textStyle="bold"/>

  <EditText
    android:id="@+id/etName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Name"
    android:inputType="text"/>

  <EditText
    android:id="@+id/etEmail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:inputType="textEmailAddress"/>

  <EditText
    android:id="@+id/etPhone"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Phone Number"
    android:inputType="phone"/>

  <EditText
    android:id="@+id/etPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Password"
    android:inputType="textPassword"/>

  <EditText
```

```
    android:id="@+id/etConfirmPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Confirm Password"
    android:inputType="textPassword"/>

  <Button
    android:id="@+id/btnSubmit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Submit"
    android:layout_gravity="center"/>

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Viva"
    android:textSize="18sp"
    android:textStyle="bold"
    android:layout_marginTop="16dp"/>

  <EditText
    android:id="@+id/etJournal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Journal"
    android:inputType="textMultiLine"
    android:lines="4"
    android:gravity="top"/>

</LinearLayout>
```

3. **Add Logic in Java**
    - Open java/com.example.registrationform/MainActivity.java.
    - Replace the code with the following Java code to handle the form:

java

Copy

```
package com.example.registrationform;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        // Initialize UI elements
        EditText etName = findViewById(R.id.etName);
        EditText etEmail = findViewById(R.id.etEmail);
        EditText etPhone = findViewById(R.id.etPhone);
        EditText etPassword = findViewById(R.id.etPassword);
        EditText etConfirmPassword = findViewById(R.id.etConfirmPassword);
        EditText etJournal = findViewById(R.id.etJournal);
        Button btnSubmit = findViewById(R.id.btnSubmit);

        // Set click listener for Submit button
```

```
    btnSubmit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String name = etName.getText().toString().trim();
            String email = etEmail.getText().toString().trim();
            String phone = etPhone.getText().toString().trim();
            String password = etPassword.getText().toString().trim();
            String confirmPassword = etConfirmPassword.getText().toString().trim();
            String journal = etJournal.getText().toString().trim();

            // Check if all fields are filled
            if (!name.isEmpty() && !email.isEmpty() && !phone.isEmpty() &&
                !password.isEmpty() && !confirmPassword.isEmpty() && !journal.isEmpty()) {
                // Check if passwords match
                if (password.equals(confirmPassword)) {
                    Toast.makeText(MainActivity.this, "Registration Successful!",
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(MainActivity.this, "Passwords do not match!",
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(MainActivity.this, "Please fill all fields!", Toast.LENGTH_SHORT).show();
            }
        }
    });
  }
}
```

4. **Run the Project**
    - Connect an Android device via USB or set up an emulator:
        - Go to **Tools > Device Manager** and create a virtual device if needed.
    - Click the **Run** button (green play triangle) in Android Studio.

- Select your device or emulator and wait for the app to install and launch.
- The registration form should appear on the screen.

5. **Test the App**
    - Enter data in the fields (Name, Email, Phone Number, Password, Confirm Password, and Journal).
    - Click the Submit button to see a toast message. The app checks if all fields are filled and if the passwords match.

Notes

- This Java implementation provides basic functionality with toast messages for feedback.
- For persistent storage or advanced validation (e.g., email format), you can extend the code using SQLite or add regex checks.
- The "Viva" label is included as a TextView above the journal EditText as per the requirements.