Practical No. 2

Title: Android program using different layouts and views

Aim: Create an application to demonstrate various layouts and views in android

Introduction

Android Layout Types

There are a number of Layouts provided by Android which you will use in almost all the Android applications to provide a different view, look and feel.

Sr.No	Layout & Description
1	Linear Layout LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	Relative Layout RelativeLayout is a view group that displays child views in relative positions.
3	Table Layout TableLayout is a view that groups views into rows and columns.
4	Absolute Layout AbsoluteLayout enables you to specify the exact location of its children.

5	Frame Layout The FrameLayout is a placeholder on screen that you can use to display a single view.
6	List View ListView is a view group that displays a list of scrollable items.
7	Grid View GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and there are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

Sr.No	Attribute & Description
1	android:id This is the ID which uniquely identifies the view.
2	android:layout_width This is the width of the layout.
3	android:layout_height This is the height of the layout

4	android:layout_marginTop This is the extra space on the top side of the layout.
5	android:layout_marginBottom This is the extra space on the bottom side of the layout.
6	android:layout_marginLeft This is the extra space on the left side of the layout.
7	android:layout_marginRight This is the extra space on the right side of the layout.
8	android:layout_gravity This specifies how child Views are positioned.
9	android:layout_weight This specifies how much of the extra space in the layout should be allocated to the View.
10	android:layout_x This specifies the x-coordinate of the layout.

11	android:layout_y This specifies the y-coordinate of the layout.
12	android:layout_width This is the width of the layout.
13	android:paddingLeft This is the left padding filled for the layout.
14	android:paddingRight This is the right padding filled for the layout.
15	android:paddingTop This is the top padding filled for the layout.
16	android:paddingBottom This is the bottom padding filled for the layout.

Exercise - Create android application to demonstrate List View

Implementation:

Program:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


  <ListView
      android:id="@+id/listView"
      android:layout_width="0dp"
      android:layout_height="700dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      />
</androidx.constraintlayout.widget.ConstraintLayout>
```

mylayout.xml

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

    <LinearLayout
        android:id="@+id/LinearLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="1dp"
        android:orientation="horizontal"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="UseComponentDrawable"
        >

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:background="#9C27B0"
            android:src="@android:drawable/checkbox_on_background"
            tools:ignore="ContentDescription" />
        <TextView
            android:id="@+id/textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginStart="50dp"
            android:layout_weight="1"
            android:gravity="fill_vertical"
            android:text="Demo Text"
            android:textColor="@color/design_default_color_secondary_variant"
            android:textSize="24sp"



            />
    </LinearLayout>


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

myadapter.java

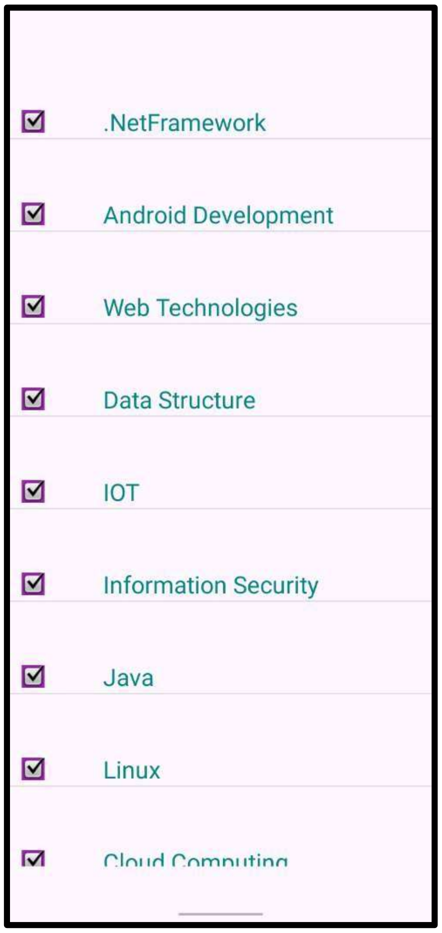
```
package com.example.practicaltwo;

import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import androidx.annotation.NonNull;

public class myadpter extends ArrayAdapter<String> {
    private  final String[] arr;

//    @SuppressLint("ViewHolder")
```

```
    @SuppressLint("ViewHolder")
    public View getView(int position, @NonNull View convertView, @NonNull ViewGroup parent)
    {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.mylayout,parent, false);
        TextView textView = convertView.findViewById(R.id.textView);
        textView.setText(getItem(position));
        return convertView;
    }
    @NonNull
    @Override
    public String getItem(int position){return  arr[position];}

    public myadpter(@NonNull Context context, int resource, @NonNull String[] arr) {
        super(context, resource,arr);
        this.arr = arr;
    }
}
```

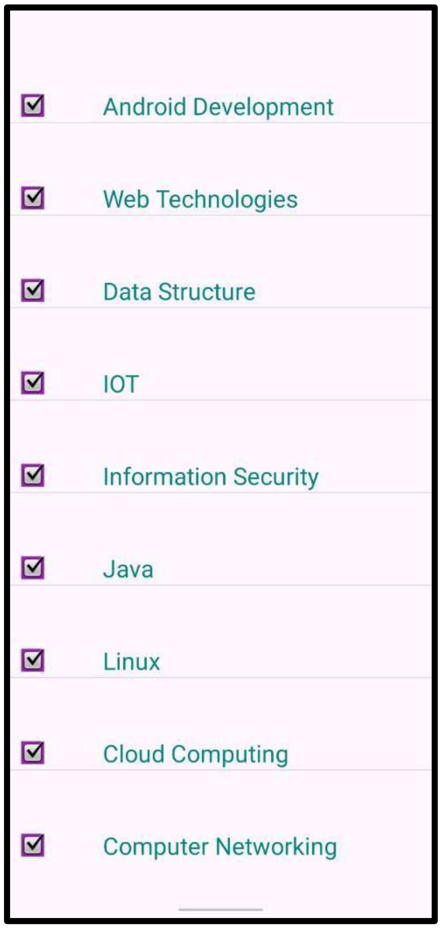
MainActivity.java

```
package com.example.practicaltwo;

import android.os.Bundle;
import android.widget.ListView;

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

public class MainActivity extends AppCompatActivity {

    public ListView l1;
     String[] arr1 ={".NetFramework","Android Development","Web Technologies","Data
Structure","IOT","Information Security","Java","Linux","Cloud Computing","Computer Networking"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        l1= findViewById(R.id.listView);
        myadpter ma = new myadpter(this,R.layout.mylayout,arr1);
        l1.setAdapter(ma);

    }
}
```

Output:



```
☑ .NetFramework
☑ Android Development
☑ Web Technologies
☑ Data Structure
☑ IOT
☑ Information Security
☑ Java
☑ Linux
☑ Cloud Computing
```



```
☑ Android Development
☑ Web Technologies
☑ Data Structure
☑ IOT
☑ Information Security
☑ Java
☑ Linux
☑ Cloud Computing
☑ Computer Networking
```

Conclusion:

This practical demonstrated how to create an Android application using various layouts and views, specifically focusing on implementing a ListView with custom layouts. It provided hands-on experience in managing and customizing Android layouts to build user-friendly interfaces.