PRACTICE EXERCISE 6: LAYOUT

LEARNING OBJECTIVES

After completing this lesson, you will have a broad understanding and flexible practical skills in using and creating layouts in Android applications

Understanding ViewGroups: Learn about different types of ViewGroup containers like LinearLayout, RelativeLayout, ConstraintLayout, etc., and their properties for arranging child views.

Creating Views Programmatically: Understand how to instantiate views such as TextView, Button, EditText, etc., and set their properties programmatically.

LayoutParams: Learn about LayoutParams and how they are used to define the size and positioning of views within a layout.

Managing View Hierarchy: Understand how to add views to a layout and manage the view hierarchy using methods like addView(), removeView(), etc.

Layout Customization: Explore techniques for customizing layouts dynamically based on runtime conditions or user interactions.

Handling Events: Learn how to handle user interactions such as clicks, touches, etc., on views and perform appropriate actions in response to these events.

Layout Editors: While this focuses on Java layout creation, it's also beneficial to be familiar with XML-based layout creation using Android Studio's layout editor.

OVERVIEW

The practical content of learning **Layout** involves understanding different layout types, creating layouts programmatically, setting layout parameters, handling view hierarchies, implementing responsive UI, working with ViewGroup and View, implementing custom layouts, debugging layout issues, adhering to best practices, and integrating layouts with other components of Android applications.

TERMINOLOGY

Layout: The arrangement or organization of elements on the user interface of an application.

View: An object displayed on the screen, including components such as buttons, text fields, images, and layout containers.

ViewGroup: A special type of View in Android that can contain other Views.

LinearLayout: A ViewGroup that arranges Views either horizontally or vertically.

RelativeLayout: A ViewGroup that allows defining the position of Views based on the position of other Views or relative rules.

ConstraintLayout: A flexible ViewGroup that allows defining the relationships between Views based on constraints.

FrameLayout: A simple ViewGroup that displays a single View, often used for switching between fragments.

Margin: The space between Views and the boundaries of the ViewGroup containing them.

Gravity: An attribute that determines how Views are aligned within a ViewGroup.

LayoutParams: An object containing information about how Views are displayed within a ViewGroup, including height, width, margin, etc.

MatchParent: A value of LayoutParams that allows a View to expand to fill the entire size of its containing ViewGroup.

WrapContent: A value of LayoutParams that allows a View to adjust its size to fit its content.

Inflation: The process of converting an XML layout file into corresponding Java objects.

Hierarchy Viewer: A tool to analyze and understand the hierarchical structure of Views in a layout.

ViewStub: A hidden View type that inflates another layout depending on the conditions of the application.

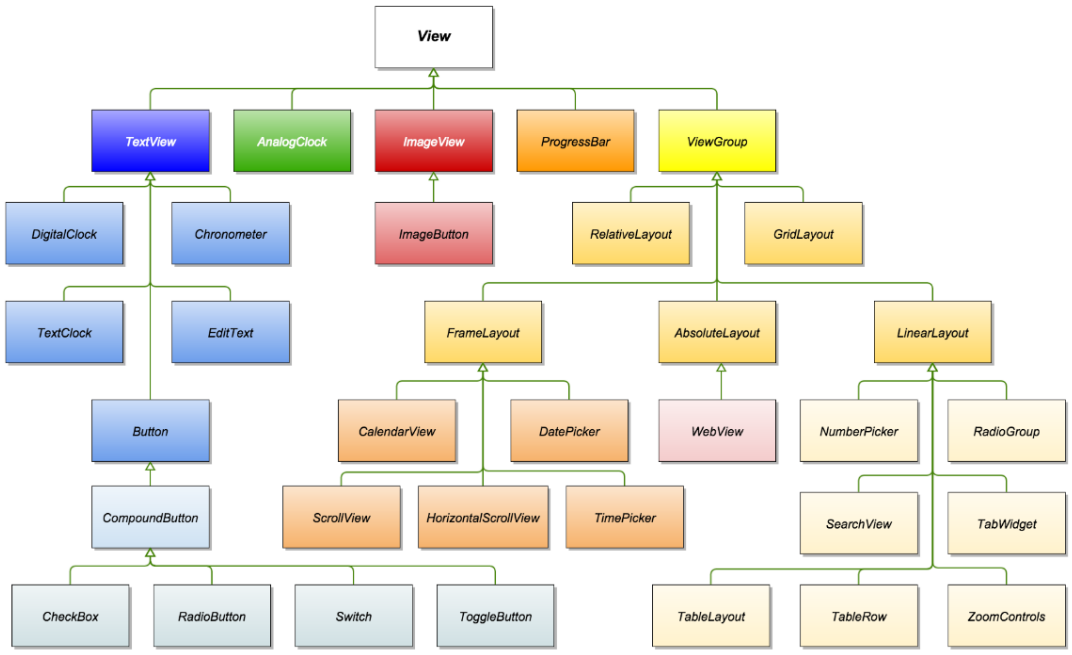
6.1. THEORETICAL SUMMARY

6.1.1. Knowledge

To excel in Layout in Android, it's essential to grasp various layout types such as LinearLayout, RelativeLayout, ConstraintLayout, and their appropriate usage for designing interfaces. Understanding the hierarchical structure of Views within

layouts and manipulating LayoutParams to adjust the size and position of Views are crucial. Implementing techniques for responsive UIs across different screen sizes and orientations is necessary. Additionally, proficiency in programmatically creating layouts, debugging layout issues, and adhering to design best practices is vital. By applying these knowledge and techniques, you'll be capable of crafting efficient and responsive layouts for your Android applications.

The android view class:



Layout:

Android have many kind of Layouts that allows to contain and organize the components of an activity such as button, checkbox and other Views. Layouts allow combining and nesting to create more complex and good interfaces. Most layouts are defined in xml files. Android Studio have default layout: res/layout/activity_main.xml that using for First Activity (MainActivity.java)

We can create more layout for a new Activity, and in Activity use setContentView(R.layout.*activity_new*); function to specify which Activity will use xml file as Layout for it.

Some basic layouts such as: LinearLayout, RelativeLayout, TableLayout, FrameLayout, ScrollView, GridView,…

6.1.2. Skill

To effectively implement Layout in Android app development, you need the following skills

Understanding Layout Types: Proficiency in understanding different layout types such as LinearLayout, RelativeLayout, ConstraintLayout, etc., and knowing when to use each one based on design requirements.

View Hierarchy Management: Ability to manage the hierarchy of views within layouts effectively, including nesting layouts and organizing views to achieve the desired UI structure.

LayoutParams Manipulation: Skill in manipulating LayoutParams to adjust the size, position, and behavior of views within a layout dynamically.

Responsive UI Design: Capability to design responsive user interfaces that adapt seamlessly to various screen sizes, resolutions, and orientations.

Programmatic Layout Creation: Proficiency in creating layouts programmatically using Java code instead of XML, including creating ViewGroup instances and adding views to them dynamically.

Debugging Layout Issues: Competence in identifying and resolving common layout issues such as overlapping views, incorrect sizing, or misaligned elements.

6.2. BASIC PRACTICE

6.2.1. Exercise 1

Develop an android application that have a list of layouts. When user click each item on list, the UI display exactly kind of layout in new Activity. If user want to back to mainActivity, they just click Close button to return main UI (have list).

Please follow these steps:

Step 1: Create a New Project in Android Studio

Create new project with name : **labSix.** (Should choose Empty Activity, don't need to use other layout temmplates). Download layout illustrations: LinearLayout, RelativeLayout, TableLayout, FrameLayout, ScrollView, GridView. Put them into "drawable" folder

Step 2: UX/UI Design:

Update 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:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >
```

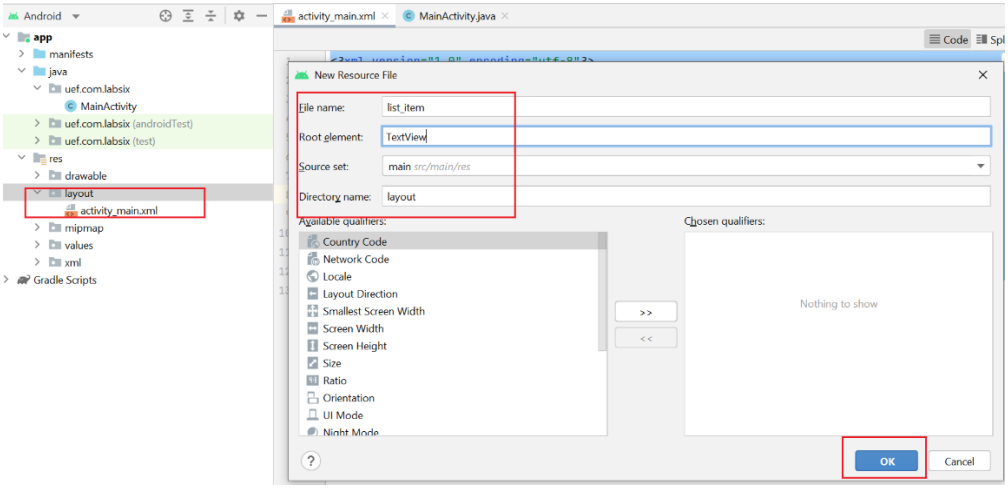
```
    <ListView
        android:id="@+id/listViewMain"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Step 3: Handling events

Update MainActivity.java: mapping ListView with Controller. And handle click event when user choose each item.

```
private ListView lv;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    lv = (ListView) findViewById(R.id.listViewMain);
    lv.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int
position
, long l) {
        TextView tv_selected = (TextView) view;
        Toast.makeText(getApplicationContext(), tv_selected.getText(),
Toast.LENGTH_SHORT).show();
        }
    });
}
```

Next, Create new layout file: list_item.xml. This layout is used for default content of each item in ListView.



Next, Update list_item.xml

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

```
    android:textSize="24dp"
    android:textColor="#000000"
    android:gravity="center"
    android:padding="10dp"
    android:background="#ffffff">
</TextView>
```

Next, create new array for Layout name in MainActivity.java

```
String[]layoutName={"LinearLayoutV","LinearLayoutH","RelativeLayout","Tab
leLayout","FrameLayout","ScrollLayout","GridLayout",
"LinearLayoutV","LinearLayoutH","RelativeLayout","TableLayout","FrameLayo
ut","ScrollLayout","GridLayout",
"LinearLayoutV","LinearLayoutH","RelativeLayout","TableLayout","FrameLayo
ut","ScrollLayout","GridLayout"};
```

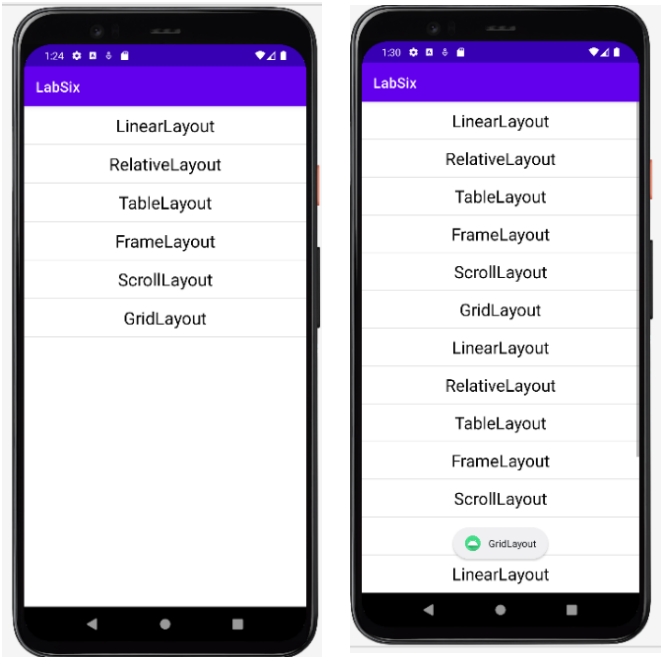
Next, create an ArrayAdapter (binding data with the view) connect to data source. Assign data for Listview

```
ArrayAdapter<String> mAdapter = new
ArrayAdapter<String>(this,R.layout.list_item,layoutName);
```

Next, update MainActivity.java

```
public class MainActivity extends AppCompatActivity {
public ListView lv;
String[]layoutName={"LinearLayoutV","LinearLayoutH","RelativeLayout","TableLayo
ut","FrameLayout","ScrollLayout","GridLayout",
"LinearLayoutV","LinearLayoutH","RelativeLayout","TableLayout","FrameLayout","S
crollLayout","GridLayout",
"LinearLayoutV","LinearLayoutH","RelativeLayout","TableLayout","FrameLayout","S
crollLayout","GridLayout"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        lv = (ListView) findViewById(R.id.listViewMain);
        ArrayAdapter<String> mAdapter = new ArrayAdapter<>(this,
R.layout.list_item,      layoutName);

        lv.setAdapter(mAdapter);
        lv.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int
position
, long              l) {
                TextView tv_selected = (TextView) view;
                Toast.makeText(getApplicationContext(), tv_selected.getText(),
                    Toast.LENGTH_SHORT).show();
            }
        });


    }
}
```

Run app on Virtual device: Click on each item, your app display Toast notification.

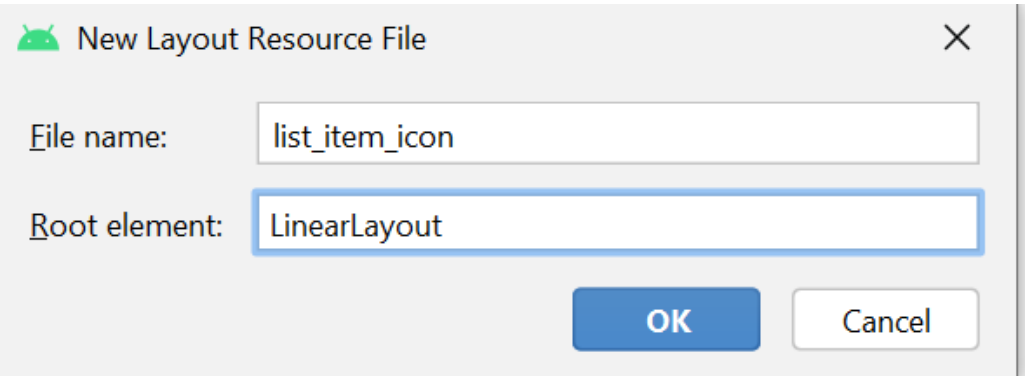


Optional: Another way you can define data resource with strings.xml file and load them into array

```
<resources>
    <string name="app_name">LabSix</string>
    <string-array name="layout_name_string">
        <item>LinearLayoutV</item>
        <item>LinearLayoutH</item>
        <item>RelativeLayout</item>
        <item>…</item> <!--Create more items-->
    </string-array>
</resources>
     String[]layoutName=
getResources().getStringArray(R.array.layout_name_string);
```

Update UI for item in List: We want to display icon and text for each item in ListView

Create new layout file:



Next, design your layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_marginLeft="5dp"
        android:layout_marginTop="5dp"
        android:layout_marginRight="20dp"
        android:src="@drawable/ic_launcher_foreground" />

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Large Text"
        android:gravity="center"
        android:textColor="#000000"
        android:textAppearance="?android:attr/textAppearanceLarge" />
</LinearLayout>
```

Next, update MainActivity.java: Declare icon array

```
int[]layoutIcon={R.drawable.linearlayout_v,R.drawable.linearlayout_h,R.drawable.relativelayout,R.drawable.tablelayout,R.drawable.framelayout,R.drawable.scrollview,R.drawable.gridlayout,R.drawable.linearlayout_v,R.drawable.linearlayout_h,R.drawable.relativelayout,R.drawable.tablelayout,R.drawable.framelayout,R.drawable.scrollview,R.drawable.gridlayout,R.drawable.linearlayout_v,R.drawable.linearlayout_h,R.drawable.relativelayout,R.drawable.tablelayout,R.drawable.framelayout,R.drawable.scrollview,R.drawable.gridlayout};
```

Next, create a new class contains element definitions in an item:

```
//In public class MainActivity extends AppCompatActivity
public static class View_An_Item
{
    public ImageView imageview;
    public TextView textview;
}
//Create your own adapter that extends from BaseAdapter
public class myadapter extends BaseAdapter
{
    Context context;
    public myadapter(Context c)
    {
        context=c;
    }
    public int getCount() {
        // TODO Auto-generated method stub
        return layoutIcon.length;
    }
    public Object getItem(int position) {
        // TODO Auto-generated method stub
```

```
        return layoutIcon[position];
    }
    public long getItemId(int arg0) {
        // TODO Auto-generated method stub
        return 0;
    }
    public View getView(int arg0, View arg1, ViewGroup arg2) {
        // TODO Auto-generated method stub
        View_An_Item an_item;
        LayoutInflater layoutinflater= ((Activity)context).getLayoutInflater();

        if(arg1==null)
        {
            an_item = new View_An_Item();
            arg1 = layoutinflater.inflate(R.layout.list_item_icon, null);
            an_item.textview = (TextView) arg1.findViewById(R.id.textView1);
            an_item.imageview = (ImageView)arg1.findViewById(R.id.imageView1);
            arg1.setTag(an_item);
        }
        else
            an_item =(View_An_Item)arg1.getTag();
        an_item.imageview.setImageResource(layoutIcon[arg0]);
        an_item.textview.setText(layoutName[arg0]);
        return arg1;
    }
}
```

Next, change default Adapter by your Adapter:

```
//ArrayAdapter<String> mAdapter = new ArrayAdapter<>(this,
R.layout.list_item, layoutName);
//lv.setAdapter(mAdapter);
lv.setAdapter(new myadapter(this));
```

Next, update Listview “onItemClick”:

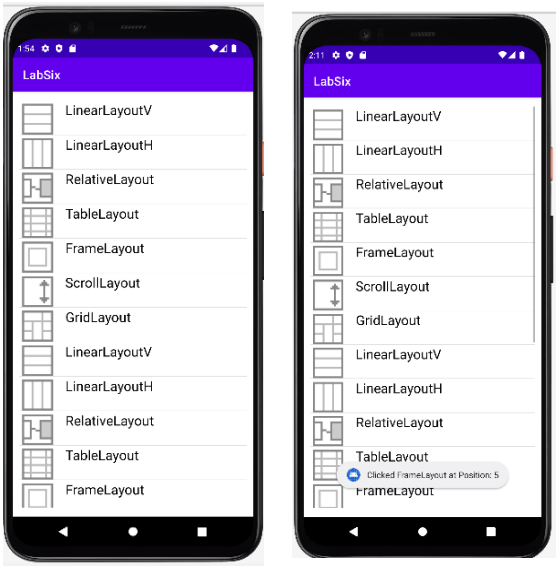
```
//TextView tv_selected = (TextView) view;
//Toast.makeText(getApplicationContext(),
tv_selected.getText(),Toast.LENGTH_SHORT).show();
String itemList = layoutName[position];
Toast.makeText(MainActivity.this, "Clicked "+ itemList+" at Position:
"+(position+1), Toast.LENGTH_SHORT).show();
```

Next, update activity_main.xml to make UI clean

```
<?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" >
    <ListView
        android:id="@+id/listViewMain"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingTop="15dp"
        android:paddingBottom="15dp"
```

```
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:cacheColorHint="#00000000"
        />
</LinearLayout>
```

Next, Run your app:



Now, your user click on each item in List, android app display another Activity (Screen) with Layout style, Button back to Main screen. Take a look Intent in this Lab. There are two intents available in android as Implicit Intents and Explicit Intents

Explicit Intent: It going to connect the internal world of an application such as start activity or send data between two activities. To start new activity we have to create Intent object and pass source activity and destination activity as shown below

```
Intent send = new Intent(MainActivity.this, SecondActivity.class);
startActivity(send);
```

And we should declare about second activity in AndroidManifest.xml file or else it going to show run time exception. sample declaration is as shown below.

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

In SecondActivity, If we want to finish and back to MainActivity, must using **finish()** method. If not, SecondActivity is still running in the background even though it's switched back to Main screen.

Implicit Intents: It going to connect with out side application such as call, mail, phone, see any website ..etc. In implicit intent we have to pass an action using setAction() as shown below example.

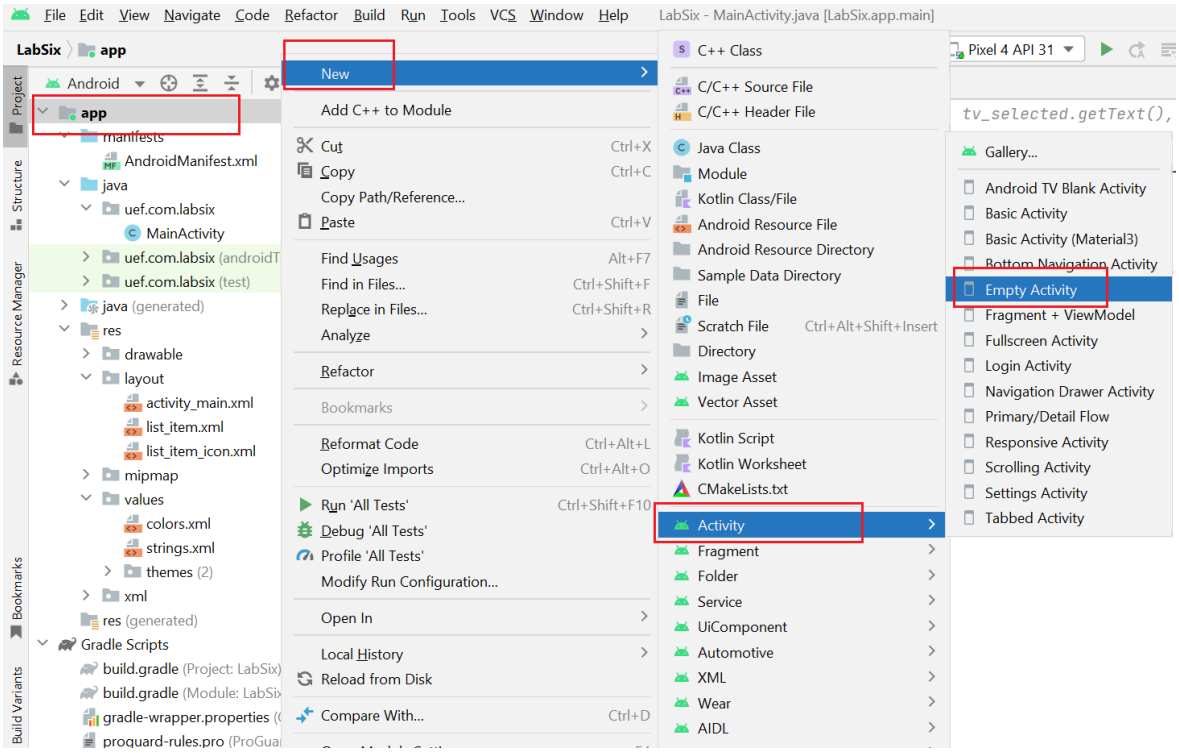
```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
i.setData(Uri.parse("https://www.uef.edu.vn/"));
startActivity(i);
```

```
/*Intent i = new
Intent(Intent.ACTION_VIEW,Uri.parse("https://www.uef.edu.vn/"));
startActivity(i);*/
```

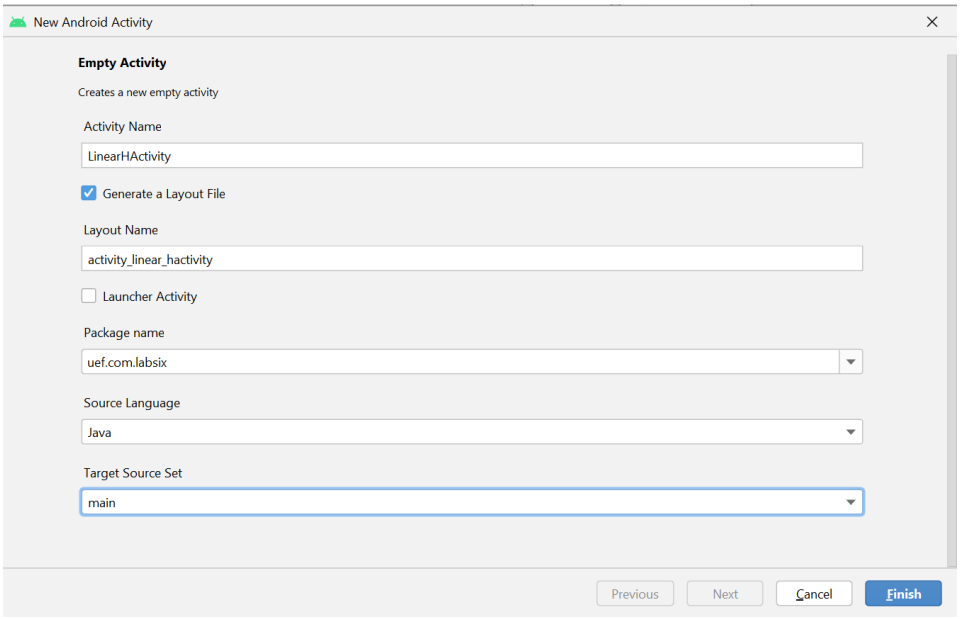
Note: To start website, It required internet permission. Add an internet permission in AndroidManifest.xml as shown below

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name = "android.permission.INTERNET"/>
    <application
        ...
```

Next, **back to current project**, create new Activity with LinearLayout (Horizontal):



Next, fill in the information and press the **Finish** button



Next, should to Check AndroidManifest.xml have registered LineaHActivity

```
<activity
    android:name=".LinearHActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, update ***activity_linear_hactivity.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="horizontal"
    tools:context=".LinearHActivity">

    <Button
        android:id="@+id/buttonLinearHActivity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Back to MainActivity" />
</LinearLayout>
```

Next, update ***LinearHActivity.java***

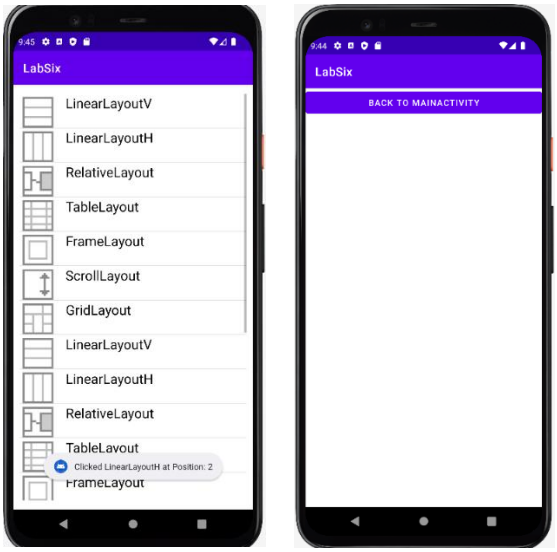
```
    package uef.com.labsix;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
public class LinearHActivity extends AppCompatActivity {
    Button mybt;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_linear_hactivity);
        mybt = (Button) findViewById(R.id.buttonLinearHActivity);
        mybt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
    }
}
```

Next, in ***MainActivity.java*** update Listview ***"onItemClick":***

```
lv.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {

        String itemList =  layoutName[position];
        Toast.makeText(MainActivity.this, "Clicked "+ itemList+" at Position:
"+(position+1), Toast.LENGTH_SHORT).show();

        switch (itemList){
            case "LinearLayoutV":
                break;
            case "LinearLayoutH":
                Intent send = new Intent(MainActivity.this,
LinearHActivity.class);
                startActivity(send);
                break;
            case "RelativeLayout":
                break;
            case "TableLayout":
                break;
            case "FrameLayout":
                break;
            case "ScrollLayout":
                break;
            case "GridLayout":
                break;
        }
    }
});
```

Run app and click item "LinearLayoutH", click "Back to MainActivity" button. Intent works:



6.2.2. Exercise 2

You continue with exercise number 1. Get data from one screen and transfer it to another screen.

- When Activity1 calls Activity2, you can pass data from Activity1 to Activity2.
- When Activity is finished, you can return data to Activity 1.
- Explicit Intent and Implicit Intent can both transmit data.
- Implicit Intent can rely on passed data to decide which component to use to open.
- When more than one program can serve the request, Android displays a dialog allowing the user to select the desired program.
- You can declare in AndroidManifest.xml to register with the system which request your application can serve.

Example: Your application can view images, you register with the system through AndroidManifest.xml. When there is a certain program that needs to call the image viewing application, then the system will put your application on the service list for users to choose from.

Passing data from Activity (Activity 1) to another Activity (Activity 2):

After initializing the Intent, we can give data to the Intent to send to another Activity using the putExtra function.

```
Intent send = new Intent(MainActivity.this, LinearHActivity.class);
send.putExtra("firstValue", new String("Helllo world!"));
send.putExtra("secondValue",new Integer(20));
```

In ***Activity* 2** (receiving side) using getIntent().getExtras() with getString() or getInt(),…

```
String value1 = getIntent().getExtras().getString("firstValue");
int value2 = getIntent().getExtras().getInt("secondValue");
```

Return data from the called Activity **(*Activity 2*)** to the calling activity (***Activity 1***).

When ***Activity1*** calls ***Activity2***, after the user interacts with ***Activity2***, can return data back to Activity before finish.

Step 1: In ***Activity1***, We use Activity Result API. The Activity Result APIs provide components for registering for a result, launching the result, and handling the result once it is dispatched by the system.

```
Intent intent = new Intent(this,optionActivity.class);
mGetContent.launch(intent);
```

Implement mGetContent method with ***ActivityResultLauncher<T>***

```
ActivityResultLauncher<Intent> mGetContent = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == Activity.RESULT_OK) {
                    // Here, no request code
                    Intent data = result.getData();
                    // get Bundle data
                    Bundle bundle = data.getExtras();
                    int index1 = bundle.getInt("ForeColor");
                    int index2 = bundle.getInt("BackColor");
                    //using Toast to print index1 and index2 value
                }
               if (result.getResultCode() == 999) {…}

            }
        });
```

Step 2: In Activity2, create an Intent, push data into Intent and using setResult() function to set result code and your Intent.

```
public void onSend(View view) {
    // give data for previous activity
    Intent intent = new Intent();
    Bundle bundle = new Bundle();
    bundle.putInt("ForeColor", index1); // get textColor value
    bundle.putInt("BackColor", index2); // get background color value
    intent.putExtras(bundle); //send with data
    setResult(RESULT_OK, intent); // return result
    //setResult(999, intent); // using this line to return result with
your own result code
    finish(); // close activity2 and back to Activity1
}
```

Now apply ***Intent*** for current project: **Create an Activity with LinearLayout (Vertical) (named:** LinearVActivity**) and** Should to Check AndroidManifest.xml have registered LineaVActivity

```
<activity
    android:name=".LinearVActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, Update ***activity_linear_vactivity.xml*** file (1 TextView, 1 EditText, 1 Button). We can put LinearLayout in ConstraintLayout (recommend)

```
<?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=".LinearVActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Your Name" />

        <EditText
            android:id="@+id/editTextName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:hint="Type your Name"
            android:inputType="textPersonName"
            android:minHeight="48dp"
            android:textColorHint="#757575" />

        <Button
            android:id="@+id/buttonSend"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Send Name to MainActivity" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Next, update ***LinearVActivity.java***

```
package uef.com.labsix;

import androidx.appcompat.app.AppCompatActivity;
```

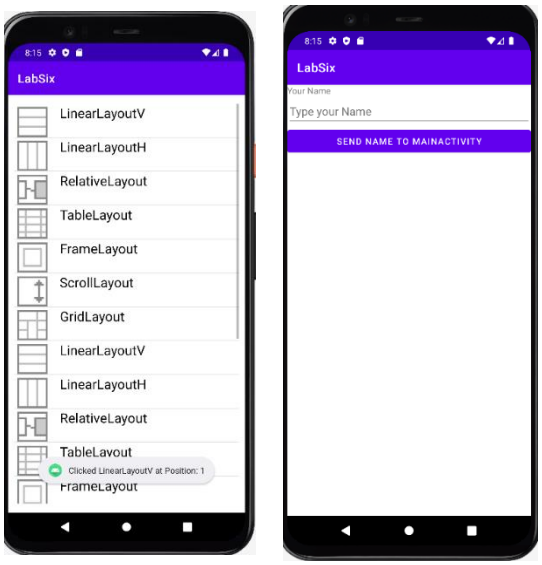
```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class LinearVActivity extends AppCompatActivity {

    EditText myEditText;
    Button myButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_linear_vactivity);
        myEditText = (EditText) findViewById(R.id.editTextName);
        myButton = (Button) findViewById(R.id.buttonSend);
        myButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // give data for previous activity
                Intent intent = new Intent();
                Bundle bundle = new Bundle();

bundle.putString("YourName",myEditText.getText().toString() ); // get
Name value
                intent.putExtras(bundle); //send with data
                setResult(RESULT_OK, intent); // return result
                finish(); // close activity2 and back to Activity1

            }
        });

    }
}
```

Next, in ***MainActivity.java,*** update Listview ***"onItemClick":***

```
lv.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {
        /*TextView tv_selected = (TextView) view;
        Toast.makeText(getApplicationContext(),
tv_selected.getText(),Toast.LENGTH_SHORT).show();*/
        String itemList =  layoutName[position];
        Toast.makeText(MainActivity.this, "Clicked "+ itemList+" at Position:
"+(position+1), Toast.LENGTH_SHORT).show();
        switch (itemList){
            case "LinearLayoutV":
                Intent intent = new Intent(MainActivity.this,
LinearVActivity.class);
                mGetContentFromIntent.launch(intent);
                break;
                 . . .
        }
    }
});
Implement mGetContentFromIntent
ActivityResultLauncher<Intent> mGetContentFromIntent =
registerForActivityResult(
```

```
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == Activity.RESULT_OK) {
                    // Here, no request code
                    Intent data = result.getData();
                    // get Bundle data
                    Bundle bundle = data.getExtras();
                    String index1 = bundle.getString("YourName");
                    //using Toast to print index1 value
                    Toast.makeText(MainActivity.this, "Your Name is "+ index1,
Toast.LENGTH_SHORT).show();
                }
            }
        });
```

Run app and click item "LinearLayoutV", type your Name in EditText and click "Send Name to MainActivity" button. Intent works:

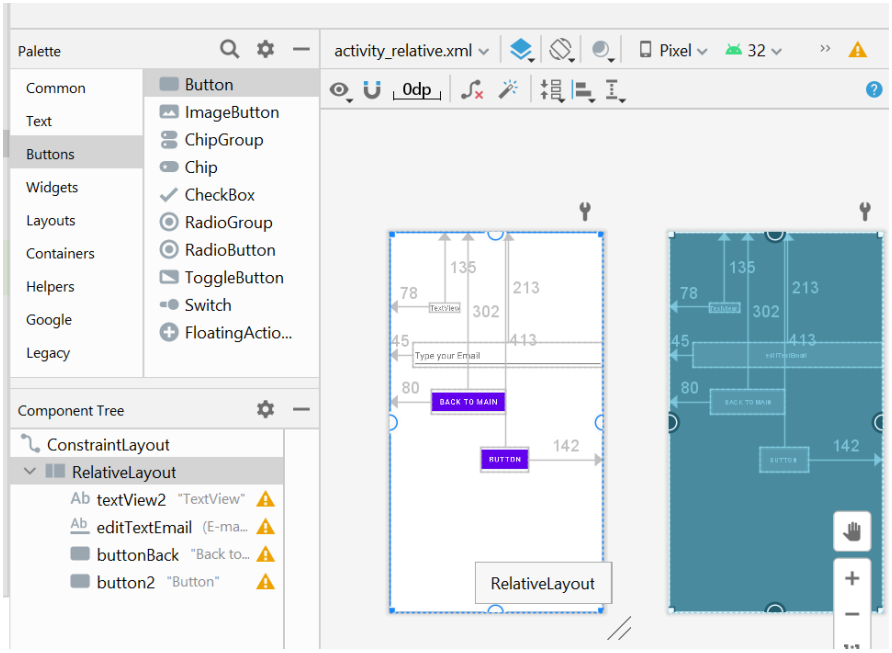


RelativeLayout

Create new Activity with name: RelativeActivity with Empty Activity. Check AndroidManifest.xml with Activity registered

```
<activity
    android:name=".RelativeActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Design your UI:




```
<?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=".RelativeActivity">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="78dp"
            android:layout_marginTop="135dp"
            android:text="TextView" />

        <EditText
            android:id="@+id/editTextEmail"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="45dp"
            android:layout_marginTop="213dp"
            android:ems="10"
            android:hint="Type your Email"
            android:inputType="textEmailAddress"
            android:minHeight="48dp"
            android:textColorHint="#757575" />
```

```
        <Button
            android:id="@+id/buttonBack"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginStart="80dp"
            android:layout_marginTop="302dp"
            android:text="Back to Main" />

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_alignParentEnd="true"
            android:layout_marginTop="413dp"
            android:layout_marginEnd="142dp"
            android:text="Button" />
    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

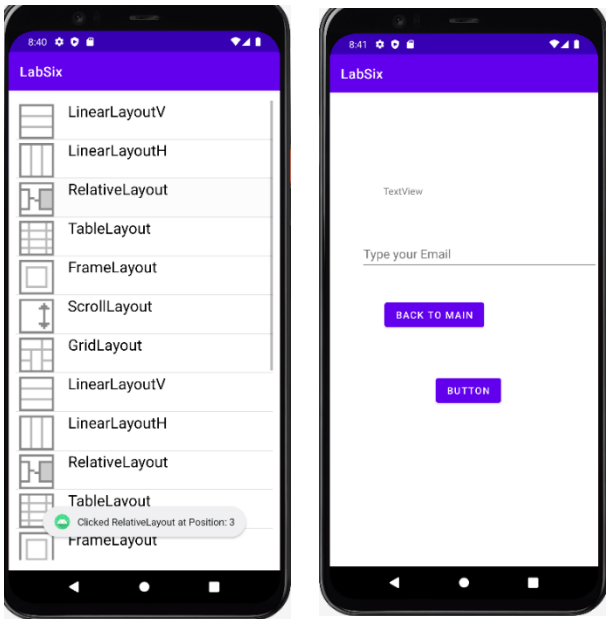
Next, update ***RelativeActivity***

```
package uef.com.labsix;

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

public class RelativeActivity extends AppCompatActivity {
    Button mybt;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_relative);
        mybt = (Button) findViewById(R.id.buttonBack);
        mybt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
    }
}
```

Next, update ***MainActivity.java***

```
…
case "RelativeLayout":
    Intent mRelativeLayout = new Intent(MainActivity.this,
RelativeActivity.class);
    startActivity(mRelativeLayout);
    break;
…
```

Next, run app:

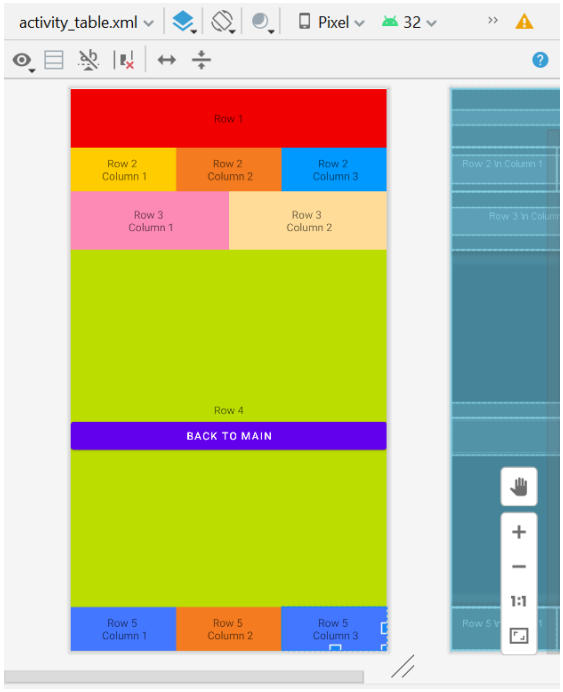


TableLayout

Create new Activity with name: TableActivity with Empty Activity. Check AndroidManifest.xml with Activity registered

```
<activity
    android:name=".TableActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, design your UI:



```
<?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=".TableActivity">

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:stretchColumns="*">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:background="#beeeef"
            android:minHeight="200px"
            android:gravity="center">

            <TableRow
                android:id="@+id/tableRow1"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:gravity="center"
                android:background="#f00000">

                <TextView
                    android:id="@+id/textView3"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:text="Row 1"
                    android:textAlignment="center" />

            </TableRow>

        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:minHeight="150px"
            android:background="#f47b20" >
            <TableRow
                android:id="@+id/tableRow2"
                android:layout_height="match_parent"
                android:layout_width="0dip"
                android:layout_weight="1"
                android:gravity="center"
                android:background="#ffcc00">

                <TextView
                    android:id="@+id/textView4"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
```

```
                android:textAlignment="center"
                android:text="Row 2 \n Column 1" />

        </TableRow>
        <TableRow
            android:id="@+id/tableRow3"
            android:layout_height="match_parent"
            android:layout_width="0dip"
            android:layout_weight="1"
            android:gravity="center" >

            <TextView
                android:id="@+id/textView5"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAlignment="center"
                android:text="Row 2 \n Column 2" />

        </TableRow>
        <TableRow
            android:id="@+id/tableRow4"
            android:layout_height="match_parent"
            android:layout_width="0dip"
            android:layout_weight="1"
            android:gravity="center"
            android:background="#0099ff">

            <TextView
                android:id="@+id/textView6"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAlignment="center"
                android:text="Row 2 \n Column 3" />

        </TableRow>

    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:minHeight="200px">

        <TableRow
            android:id="@+id/tableRow5"
            android:layout_height="match_parent"
            android:layout_width="0dip"
            android:layout_weight="1"
            android:gravity="center"
            android:background="#fd8ab5">

            <TextView
                android:id="@+id/textView7"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Row 3 \n Column 1"
                android:textAlignment="center" />
```

```
        </TableRow>

        <TableRow
            android:id="@+id/tableRow6"
            android:layout_height="match_parent"
            android:layout_width="0dip"
            android:layout_weight="1"
            android:gravity="center"
            android:background="#ffdd99">

            <TextView
                android:id="@+id/textView8"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Row 3 \n Column 2"
                android:textAlignment="center" />

        </TableRow>
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:background="#bbdd00" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:gravity="center"
            android:orientation="vertical">
            <TextView
                android:id="@+id/textView9"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textAlignment="center"
                android:text="Row 4" />

            <Button
                android:id="@+id/buttonBack"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Back to Main" />
        </LinearLayout>


    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:minHeight="150px"
        android:background="#f47b20" >
        <!-- Column 1 -->
        <TextView
            android:id="@+id/textView1"
```

```
                android:layout_height="match_parent"
                android:layout_width="0dip"
                android:layout_weight="1"
                android:gravity="center"
                android:textAlignment="center"
                android:text="Row 5 \n Column 1"
                android:background="#f47f"/>

            <!-- Column 2 -->
            <TextView
                android:id="@+id/textView2"
                android:layout_height="match_parent"
                android:layout_width="0dip"
                android:layout_weight="1"
                android:gravity="center"
                android:textAlignment="center"
                android:text="Row 5 \n Column 2" />

            <!-- Column 3 -->
            <TextView
                android:id="@+id/textView"
                android:layout_height="match_parent"
                android:layout_width="0dip"
                android:layout_weight="1"
                android:gravity="center"
                android:textAlignment="center"
                android:text="Row 5 \n Column 3"
                android:background="#f47f"/>
        </TableRow>
    </TableLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

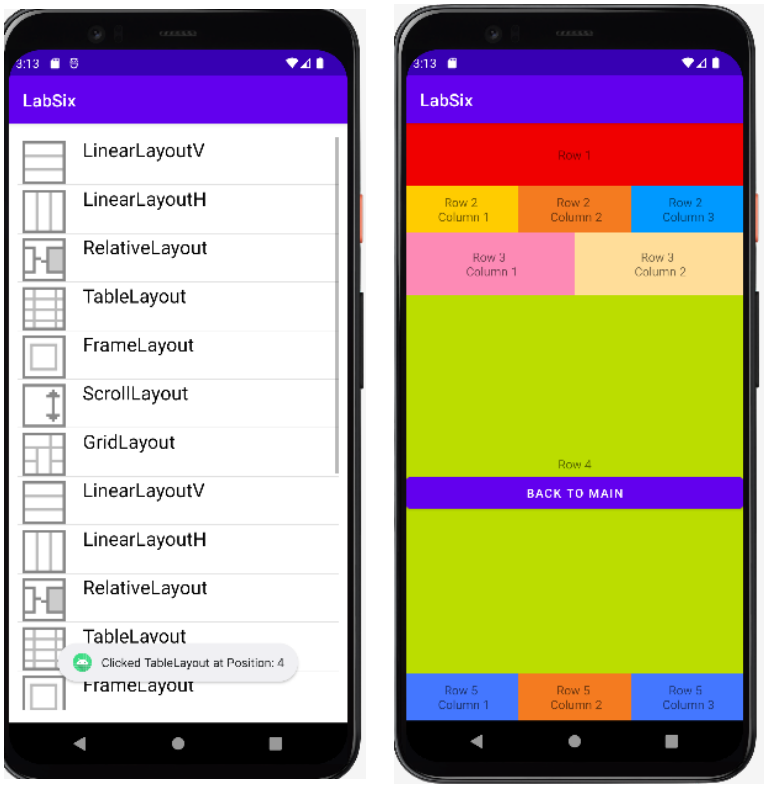
Next, update ***TableActivity***

```
package uef.com.labsix;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class TableActivity extends AppCompatActivity {
    Button mybt;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_table);
        mybt = (Button) findViewById(R.id.buttonBack);
        mybt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
    }
}
```

Next, update ***MainActivity.java***

```
…
case "TableLayout":
    Intent mTableLayout = new Intent(MainActivity.this, TableActivity.class);
    startActivity(mTableLayout);
    break;
…
```

Run app:



FrameLayout

Create new Activity with name: FrameActivity with Empty Activity. Check AndroidManifest.xml with Activity registered

```
<activity
    android:name=".FrameActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, design your UI:

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

```
    <FrameLayout
        android:id="@+id/FrameLayoutDemo"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </FrameLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:layout_marginBottom="20dp">

        <Button
            android:id="@+id/buttonBackFrame"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Back to MainActivity" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Next, update ***FrameActivity***

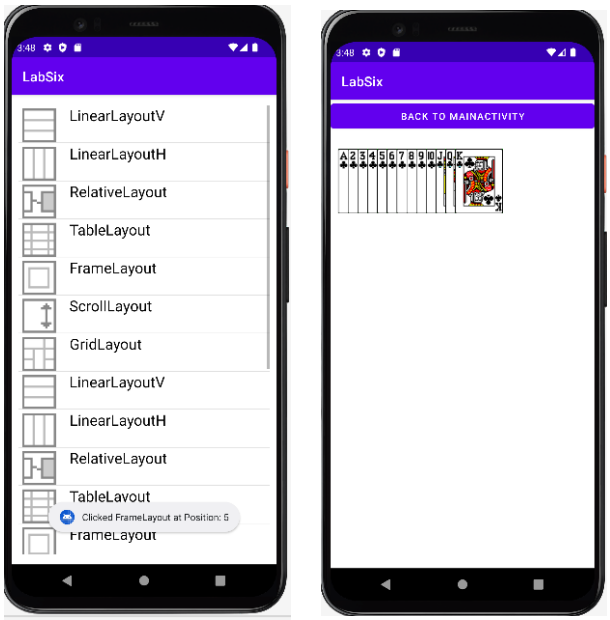
```
package uef.com.labsix;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.ImageView;

public class FrameActivity extends AppCompatActivity implements
View.OnClickListener {
    Button mybt;
    FrameLayout fl;
    int card_array[]={


R.drawable.c1,R.drawable.c2,R.drawable.c3,R.drawable.c4,R.drawable.c5,

R.drawable.c6,R.drawable.c7,R.drawable.c8,R.drawable.c9,R.drawable.c10,
            R.drawable.cj,R.drawable.cq,R.drawable.ck,



R.drawable.d1,R.drawable.d2,R.drawable.d3,R.drawable.d4,R.drawable.d5,

R.drawable.d6,R.drawable.d7,R.drawable.d8,R.drawable.d9,R.drawable.d10,
            R.drawable.dj,R.drawable.dq,R.drawable.dk,



R.drawable.h1,R.drawable.h2,R.drawable.h3,R.drawable.h4,R.drawable.h5,

R.drawable.h6,R.drawable.h7,R.drawable.h8,R.drawable.h9,R.drawable.h10,
            R.drawable.hj,R.drawable.hq,R.drawable.hk,
```

```
R.drawable.s1,R.drawable.s2,R.drawable.s3,R.drawable.s4,R.drawable.s5,

R.drawable.s6,R.drawable.s7,R.drawable.s8,R.drawable.s9,R.drawable.s10,
            R.drawable.sj,R.drawable.sq,R.drawable.sk
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_frame);
        mybt = (Button) findViewById(R.id.buttonBackFrame);
        mybt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
        fl=(FrameLayout)findViewById(R.id.FrameLayoutDemo);
        for(int i=0;i<13;i++)
        {
            ImageView iv1=new ImageView(this);
            iv1.setImageResource(card_array[i]);
            FrameLayout.LayoutParams lay1=new
FrameLayout.LayoutParams(FrameLayout.LayoutParams.WRAP_CONTENT,
FrameLayout.LayoutParams.WRAP_CONTENT);
            lay1.setMargins(30+(i*40), 200, 100, 200);
            iv1.setLayoutParams(lay1);
            iv1.setOnClickListener(this);
            fl.addView(iv1);
        }
    }

    @Override
    public void onClick(View view) {
        // TODO Auto-generated method stub
        ImageView iv=(ImageView)view;
        FrameLayout.LayoutParams l=(FrameLayout.LayoutParams)
iv.getLayoutParams();

        l.setMargins(l.leftMargin, l.topMargin-10, l.rightMargin,
l.bottomMargin);
        iv.setLayoutParams(l);
    }
}
```

Next, update ***MainActivity.java***

```
…
case "FrameLayout":
    Intent mFrameLayout = new Intent(MainActivity.this, FrameActivity.class);
    startActivity(mFrameLayout);
    break;
…
```

Run app:

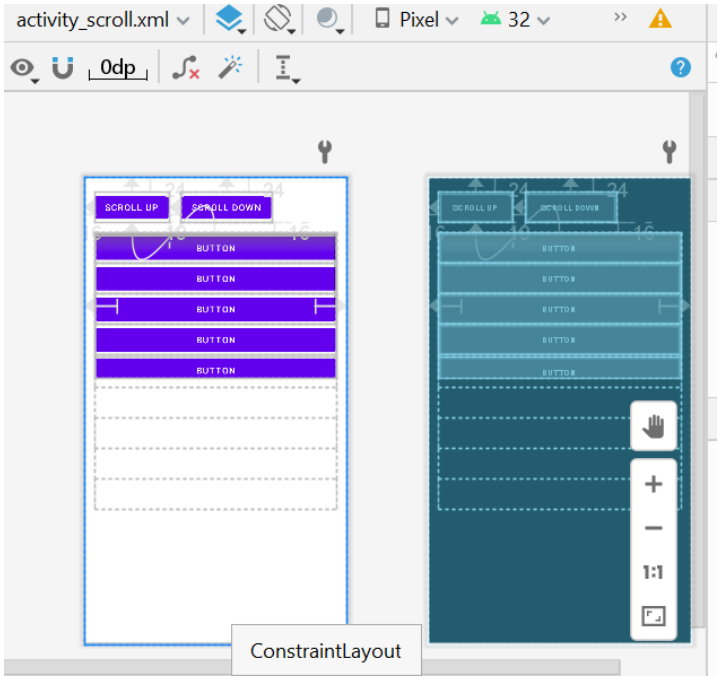


ScrollView

Create new Activity with name: ScrollActivity with Empty Activity. Check AndroidManifest.xml with Activity registered

```
<activity
    android:name=".ScrollActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, design your UI:




```
<?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=".ScrollActivity">

    <Button
        android:id="@+id/button_scrollUp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="24dp"
        android:text="Scroll Up"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button_scrollDown"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="19dp"
        android:layout_marginLeft="19dp"
        android:layout_marginTop="24dp"
        android:text="Scroll Down"
        app:layout_constraintStart_toEndOf="@+id/button_scrollUp"
        app:layout_constraintTop_toTopOf="parent" />

    <ScrollView
        android:id="@+id/scrollView"
        android:layout_width="0dp"
        android:layout_height="229dp"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button_scrollUp">

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

            <Button
                android:id="@+id/button11"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button10"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />
```

```
            <Button
                android:id="@+id/button9"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button8"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button7"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button6"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button5"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button4"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />

            <Button
                android:id="@+id/button3"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Button" />
        </LinearLayout>
    </ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Update **ScrollActivity**

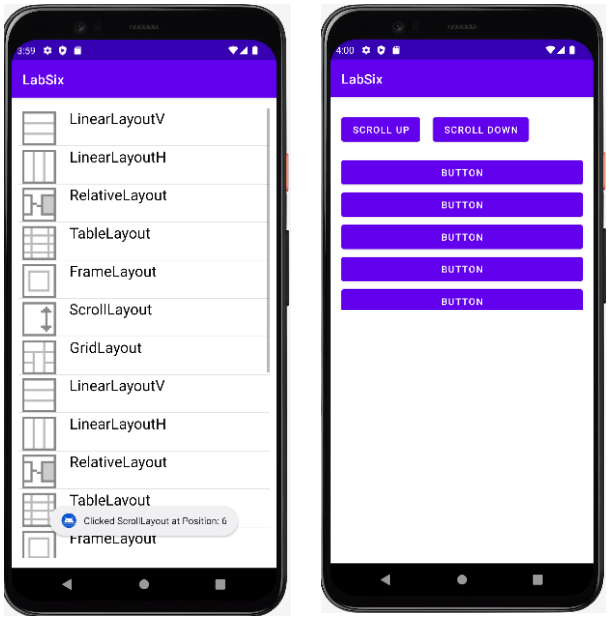
```
package uef.com.labsix;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ScrollView;

public class ScrollActivity extends AppCompatActivity {
    private ScrollView scrollView;

    private Button buttonScrollUp;
    private Button buttonScrollDown;
```

```
    public static final int SCROLL_DELTA = 15; // Pixel.
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scroll);
        this.scrollView = (ScrollView) this.findViewById(R.id.scrollView);
        this.buttonScrollUp = (Button) this.findViewById(R.id.button_scrollUp);
        this.buttonScrollDown = (Button)
this.findViewById(R.id.button_scrollDown);
        this.buttonScrollUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                doScrollUp();
            }
        });

        this.buttonScrollDown.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                doScrollDown();
            }
        });
    }
    private void doScrollUp() {

        int x = this.scrollView.getScrollX();
        int y = this.scrollView.getScrollY();
        if(y - SCROLL_DELTA >= 0) {
            this.scrollView.scrollTo(x, y-SCROLL_DELTA);
        }
    }
    private void doScrollDown() {
        int maxAmount = scrollView.getMaxScrollAmount();
        int x = this.scrollView.getScrollX();
        int y = this.scrollView.getScrollY();
        if(y + SCROLL_DELTA <= maxAmount) {
            this.scrollView.scrollTo(x, y + SCROLL_DELTA);
        }
    }
}
```

Update ***MainActivity.java***

```
…
case "ScrollLayout":
    Intent mScrollLayout = new Intent(MainActivity.this, ScrollActivity.class);
    startActivity(mScrollLayout);
    break;
…
```

Run app:

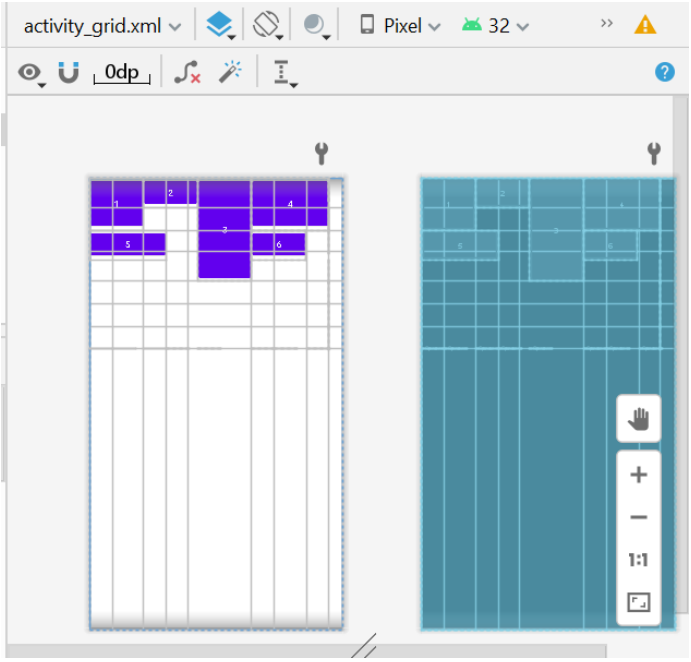


GridView

Create new Activity with name: GridActivity with Empty Activity. Check AndroidManifest.xml with Activity registered

```
<activity
    android:name=".GridActivity"
    android:exported="false">
    <meta-data
        android:name="android.app.lib_name"
        android:value="" />
</activity>
```

Next, design your UI:




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".GridActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <GridLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:columnCount="9"
            android:orientation="horizontal"
            android:rowCount="8" >
            <Button
                android:layout_columnSpan="2"
                android:layout_gravity="fill"
                android:layout_rowSpan="2"
                android:text="1" />
            <Button
                android:layout_columnSpan="2"
                android:layout_gravity="fill_horizontal"
                android:text="2" />
            <Button
                android:layout_gravity="fill_vertical"
                android:layout_rowSpan="4"
                android:text="3" />

            <Button
                android:layout_columnSpan="3"
                android:layout_gravity="fill"
                android:layout_rowSpan="2"
                android:text="4" />
            <Button
                android:layout_columnSpan="3"
                android:layout_gravity="fill_horizontal"
                android:text="5" />
            <Button
                android:layout_columnSpan="2"
                android:layout_gravity="fill_horizontal"
                android:text="6" />
            <Space
                android:layout_width="36dp"
                android:layout_column="0"
                android:layout_row="7" />
            <Space
                android:layout_width="36dp"
                android:layout_column="1"
                android:layout_row="7" />
            <Space
                android:layout_width="36dp"
                android:layout_column="2"
                android:layout_row="7" />
            <Space
                android:layout_width="36dp"
                android:layout_column="3"
```

```
            android:layout_row="7" />
        <Space
            android:layout_width="36dp"
            android:layout_column="4"
            android:layout_row="7" />
        <Space
            android:layout_width="36dp"
            android:layout_column="5"
            android:layout_row="7" />
        <Space
            android:layout_width="36dp"
            android:layout_column="6"
            android:layout_row="7" />
        <Space
            android:layout_width="36dp"
            android:layout_column="7"
            android:layout_row="7" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="0" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="1" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="2" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="3" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="4" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="5" />
        <Space
            android:layout_height="36dp"
            android:layout_column="8"
            android:layout_row="6" />
    </GridLayout>
</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Next, update ***GridActivity***

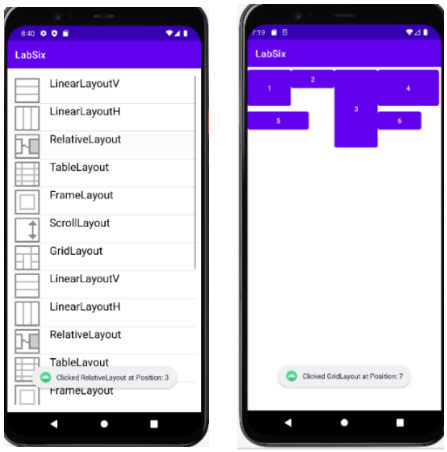
```
Package uef.com.labsix;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class RelativeActivity extends AppCompatActivity {
    Button mybt;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_relative);
        mybt = (Button) findViewById(R.id.buttonBack);
        mybt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });
    }
}
```

Next, update ***MainActivity.java***

```
…
case "GridLayout":
    Intent mGridLayout = new Intent(MainActivity.this, GridActivity.class);
    startActivity(mGridLayout);
    break;
…
```

Run app:



Store your project on Git

- From Android studio menu bar. Choose VCS => Create Git Repo…
- Switch Porject view to “Android”, Right click on app folder → Git → + Add
- Choose Git => Manage remotes… => Paste your Git

Ex: https://examplegit.com/exampleproject/mylab.git

- Authorize by your account
- Click Git → Commit …→Write commit mess
- Click Git → Push

6.3. APPLICATION EXERCISES

You practice additional topics to gain a better understanding of the Layout in Android

1. **Profile Manager App**
   a. Use LinearLayout or RecyclerView to display a list of profiles.
   b. Each profile should include name, age, address, and a profile picture.
   c. Allow users to add, edit, and delete profiles.
2. **Weather App**
   a. Use RelativeLayout or ConstraintLayout to display weather information.
   b. Include a chart or a series of buttons or weather items.
   c. Connect to a weather data API to display real-time weather information.
3. **Book Lookup App**
   a. Use GridLayout or RecyclerView to display a list of books.
   b. Each book entry should include title, author, and a book cover image.
   c. Allow users to search for books by title or author.

REFERENCES

[1]. Android Developers. Android Developers Documentation. Retrieved from https://developer.android.com/docs

[2]. Jumpstart Academy. (n.d.). Android Layout Cookbook. Big Nerd Ranch.

[3]. Phillips, B., Stewart, C., & Marsicano, K. (2017). Android Programming: The Big Nerd Ranch Guide. Big Nerd Ranch.

[4]. Griffiths, D., & Griffiths, D. (2017). Head First Android Development. O'Reilly Media.

[5]. Clifton, I. G. (2013). Android User Interface Design: Turning Ideas and Sketches into Beautifully Designed Apps. Addison-Wesley Professional.

[6]. Mew, K. (2016). Mastering Android Layouts: Best Practices for Mobile App Design. Packt Publishing.