**PRACTICAL # 06**

**OBJECT:**

Application UI Design

**THEORY:**

Android applications allow creating beautiful UI design. The two major components in the UI design are containers (eg. layouts) and View elements.

The basic building block for user interface is View class object. It occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The ViewGroup is a subclass of View and provides invisible container that holds other Views or other ViewGroups and define their layout properties. At third level we have different layouts which are subclasses of ViewGroup class.

A typical layout defines the visual structure for an Android user interface and can be created either at run time using View/ViewGroup objects or you can declare your layout file in the res/layout folder of your project.

**Layouts:**

Layouts are containers for other view elements of an application. When working with multiple activities, a layout is created for those activities. If you add a new activity using IDE, a corresponding layout is automatically created and added to your project. Apart from that, an activity can have more than one layouts, nested in one main layout.

Android allows you to create layouts using XML files as well as using Java code (dynamic creation). The default standard is creating XML files for the layouts. The layouts are placed in /res/layout folder.

Out of different layouts, we will be discussing the following layouts:

1. Linear Layout

2. Relative Layout

3. Table Layout

**1. Linear Layout**

In a linear layout, the elements are arranged in linear order, either Horizontally or Vertically. This behavior is set in android:orientation property of LinearLayout.

*<LinearLayout android:orientation="vertical"> .... </LinearLayout>*

*<LinearLayout android:orientation="horizontal"> .... </LinearLayout>*

To test this layout, create a new Android project:

1. File -> New -> Android Project

2. In Package Explorer right click on res/layout folder and create a new Android XML File and name it “linear\_layout.xml” : res/layout -> Right Click -> New -> Android XML File

3. Now open newly created xml file (in my case “linear\_layout.xml”) and type the following code.

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

*<!-- Parent linear layout with vertical orientation -->*

*<LinearLayout*

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

*android:orientation="vertical"*

*android:layout\_width="match\_parent"*

*android:layout\_height="match\_parent">*

*<TextView android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"*

*android:text="Email:" android:padding="5dip"/>*

*<EditText android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"*

*android:layout\_marginBottom="10dip"/>*

*<Button android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"*

*android:text="Login"/>*

*<!-- Child linear layout with horizontal orientation -->*

*<LinearLayout android:layout\_width="fill\_parent"*

*android:layout\_height="wrap\_content"*

*android:orientation="horizontal" android:background="#2a2a2a"*

*android:layout\_marginTop="25dip">*

*<TextView android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"*

*android:text="Home" android:padding="15dip" android:layout\_weight="1"*

*android:gravity="center"/>*

*<TextView android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"*

*android:text="About" android:padding="15dip" android:layout\_weight="1"*

*android:gravity="center"/>*

*</LinearLayout>*

*</LinearLayout>*

4. To set this newly created view as the initial view of your app, Open your MainActivity.java file. You would see the following line inside the onCreate function setContentView(R.layout.main). Change R.layout.main to R.layout.linear\_layout as shown below:

*@Override*

*public void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

***setContentView(R.layout.linear\_layout);***

*}*

5. Run the application and observe the output, which should look as shown in Figure 1.

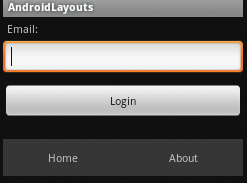


Figure 1: Linear Layout

**2. Relative Layout**

In a relative layout every element arranges itself relative to other elements or a parent element. As an example, consider the layout defined below. The “Cancel” button is placed relatively, to the right of the “Login” button in parallel. Here is the code snippet that achieves the mentioned alignment (Right of Login button in parallel).

Steps to create a relative layout:

1. Create a new project File -> New -> Android Project

2. In Package Explorer right click on res/layout folder and create a new Android XML File and name it “relative\_layout.xml” : **res/layout -> Right Click -> New -> Android XML File**

3. Now open new xml file (“relative\_layout.xml”) and type the following code:

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

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

*android:layout\_width="fill\_parent"*

*android:layout\_height="wrap\_content">*

*<TextView android:id="@+id/label" android:layout\_width="fill\_parent"*

*android:layout\_height="wrap\_content" android:text="Email" />*

*<EditText android:id="@+id/inputEmail" android:layout\_width="fill\_parent"*

*android:layout\_height="wrap\_content"*

*android:layout\_below="@id/label" />*

*<Button android:id="@+id/btnLogin" android:layout\_width="wrap\_content"*

*android:layout\_height="wrap\_content" android:layout\_below="@id/inputEmail"*

*android:layout\_alignParentLeft="true" android:layout\_marginRight="10px"*

*android:text="Login" />*

*<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"*

*android:layout\_toRightOf="@id/btnLogin"*

*android:layout\_alignTop="@id/btnLogin" android:text="Cancel" />*

*<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"*

*android:layout\_alignParentBottom="true" android:text="Register new Account"*

*android:layout\_centerHorizontal="true"/>*

*</RelativeLayout>*

4. Open your MainActivity.java file and set the layout to your newly created relative layout file, that is R.layout.relative\_layout: setContentView(R.layout.relative\_layout);

5. Run the application, and observe the output as shown in Figure 2.

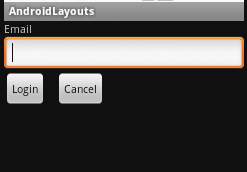


Figure 2: Relative layout output.

Observe how the relative attributes *android:layout\_below, android:layout\_toRightOf* etc are used to relatively align the elements in the layout.

**3. Table Layout:**

Table layout in Android is similar to HTML table. You can divide the container into rows and columns. Figure 3 shows how a table layout can be partitioned:

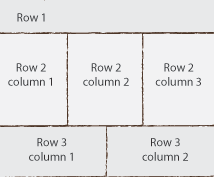


Figure 3: Example table layout

Steps to create a table layout:

1. In the previous project’s Package Explorer right click on res/layout folder and create a new Android XML File and name it “table\_layout.xml” : res/layout -> Right Click -> New -> Android XML File

2. Open “table\_layout.xml” file and enter following XML code.

*<TableLayout*

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

*android:layout\_width="match\_parent"*

*android:layout\_height="match\_parent"*

*android:shrinkColumns="\*" android:stretchColumns="\*" android:background="#ffffff">*

*<!-- Row 1 with single column -->*

*<TableRow*

*android:layout\_height="wrap\_content"*

*android:layout\_width="fill\_parent"*

*android:gravity="center\_horizontal">*

*<TextView*

*android:layout\_width="match\_parent" android:layout\_height="wrap\_content"*

*android:textSize="18dp" android:text="Row 1" android:layout\_span="3"*

*android:padding="18dip" />*

*</TableRow>*

*<!-- Row 2 with 3 columns -->*

*<TableRow*

*android:id="@+id/tableRow1"*

*android:layout\_height="wrap\_content"*

*android:layout\_width="match\_parent">*

*<TextView*

*android:id="@+id/TextView04" android:text="Row 2 column 1"*

*android:layout\_weight="1"*

*android:padding="20dip" android:gravity="center"/>*

*<TextView*

*android:id="@+id/TextView04" android:text="Row 2 column 2"*

*android:layout\_weight="1"*

*android:padding="20dip" android:gravity="center"/>*

*<TextView*

*android:id="@+id/TextView04" android:text="Row 2 column 3"*

*android:layout\_weight="1"*

*android:padding="20dip" android:gravity="center"/>*

*</TableRow>*

*<!-- Row 3 with 2 columns -->*

*<TableRow*

*android:layout\_height="wrap\_content"*

*android:layout\_width="fill\_parent"*

*android:gravity="center\_horizontal">*

*<TextView*

*android:id="@+id/TextView04" android:text="Row 3 column 1"*

*android:layout\_weight="1"*

*android:padding="20dip" android:gravity="center"/>*

*<TextView*

*android:id="@+id/TextView04" android:text="Row 3 column 2"*

*android:layout\_weight="1"*

*android:padding="20dip" android:gravity="center"/>*

*</TableRow>*

*</TableLayout>*

3. Open MainActivity.java file and set the layout to the new table layout file R.layout.table\_layout:

*setContentView(R.layout.table\_layout);*

4. Run the application to observe the output.

**View Elements:**

The visible elements that make up the UI of an Android application are the view elements. In the layouts section, we have used some of the most common view elements. These include:

- Button

- TextView

- EditText

Figure 4 shows some of the view elements that you can access from the Palette panel when you open design Tab in for the XML layout file.

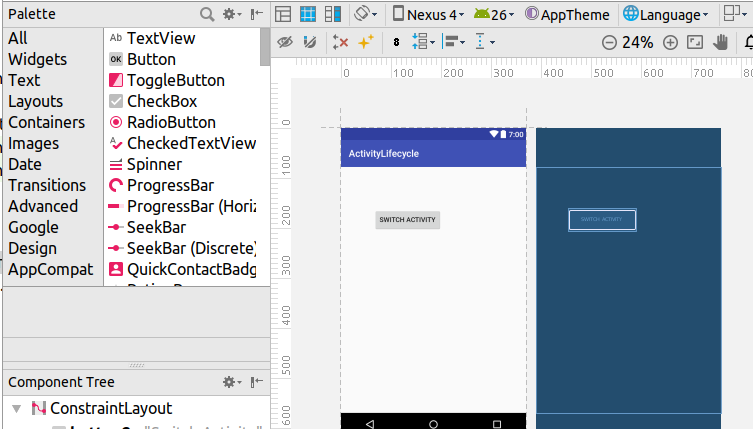


Figure 4: Some of the view elements in Palette pane in Android Studio

**ACTIVITIES**

**Activity 1**

Create a login activity UI containing username, password and login buttons. The UI should be created using: 1) linear layout, 2) Relative Layout, 3) Nested Linear Layouts

**REVIEW QUESTIONS**

1. What are layouts?
2. What are different properties of linear layout?
3. What are the advantages of relative layout?
4. How does table layout works?