BICT- Level III – Semester II
Topic – Mobile Application Development (ICT3233)
Lab Sheet 05

Layouts

A layout defines the structure for a user interface in your app, such as in an activity.

Layout Classes:

1. Frame Layout
2. Linear Layout
3. Relative Layout
4. Constraint Layout

Frame Layout

- Frame Layout is one of the simplest layout to organize view controls. They are designed to block an area on the screen to display a single item.

Frame Layout - Attributes

Example 01 – foreground attribute

- Create a new project.
- Change the layout to **FrameLayout**.
- By default your app is created with a **ConstraintLayout** you can change the layout to **FrameLayout**.
- Set an **id** for your layout.

activity_main.xml MainActivity.java

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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/framelayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

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

</FrameLayout>
```

android:foreground Attribute

- Foreground defines the drawable to draw over the content and this may be a color value.
- Insert the following code.

```
<FrameLayout 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/framelayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:foreground="#00FF00"> <!--foreground color for a FrameLayout-->

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

        <!--Textview will not be shown because of foreground color is drawn over it-->

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:text="abhiAndroid"/>

    </LinearLayout>

</FrameLayout>
```

- In here we have set the **foreground** attribute of the frame layout as **"#00ff00" - "Green".**
- And we have inserted a LinearLayout inside the FrameLayout and we have inserted a TextView inside the LinearLayout.
- But, here the LinearLayout will not be visible because for the FrameLayout we have set the foreground to Green color.
- The FrameLayout comes in the foreground.

Example 02: android:layout_gravity

- It is used to set the gravity for the views of the layout. That means where the views should be placed within the layout.
- Create a new project and add the following code.
- You can get an idea about how you can place the views inside a FrameLayout by using **layout_gravity** attribute.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    >
    <TextView android:text="LeftTop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="RightTop"
        android:layout_gravity="top|right" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="CentreTop"
        android:layout_gravity="top|center_horizontal" />
    <TextView android:text="Left"
        android:layout_gravity="left|center_vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Right"
        android:layout_gravity="right|center_vertical" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Centre"
        android:layout_gravity="center" />
    <TextView android:text="LeftBottom"
        android:layout_gravity="left|bottom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="RightBottom"
        android:layout_gravity="right|bottom" />
    <TextView android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="CenterBottom"
        android:layout_gravity="center|bottom" />
</FrameLayout>
```

Linear Layout

- In the linear Layout all the elements are displayed in linear fashion means all the child elements of a Linear layout are displayed according to its orientation.

Example 03

android:orientation

- Create a new project.
- Change the layout to a **LinearLayout**.
- Add two buttons and set the orientation to horizontal as given in the below code.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="horizontal"
    >

    <!-- Child Views(In this case 2 Button) are here -->


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6" />
</LinearLayout>
```

- You can change the **orientation to "vertical"** and see the difference.

android:gravity

- You can change the gravity attribute of the layout to **left, right, center, top, bottom** and check how they behave.

```
activity_main.xml   MainActivity.java
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="horizontal"
    android:gravity="right"
    >

    <!-- Child Views(In this case 2 Button) are here -->


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6" />
</LinearLayout>
```

android:layout_weight

- You can add **layout_weight** attribute for both buttons and check how they behave.

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

    <!-- Child Views(In this case 2 Button) are here -->


    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32"
        android:layout_margin="5dp"
        android:layout_weight="2"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6"
        android:layout_margin="5dp"
        android:layout_weight="1"
        />
</LinearLayout>
```

android:weightSum

- Change the code of the example as the below one and check how the layout change with "**weightsum**" attribute.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:weightSum="5">
    <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Linear Layout (With Weight and weight sum)"
    android:id="@+id/textView"
    android:layout_gravity="center_horizontal"
    android:layout_weight="2"
    />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button1"
        android:id="@+id/button"
        android:background="#358a32"
        android:layout_margin="5dp"
        android:layout_weight="2"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button2"
        android:id="@+id/button2"
        android:background="#0058b6"
        android:layout_margin="5dp"
        android:layout_weight="1"
        />
</LinearLayout>
```

- You can **increase** the value of **weightsum** attribute **more than the addition of the layout_weights of the views** and check how the views behave on the screen.

  Eg: In the above example the sum=5, you can just assign weightSum=6 and check how it differs.

Relative Layout

- The Relative Layout is very flexible layout used in android for custom layout designing.
- It gives us the flexibility to position our component/view based on the relative or sibling component's position.
- There are a lot of attributes which you can use in Relative Layouts such as:

android:layout_above

Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name"

android:layout_alignBottom

Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignLeft

Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignParentBottom

If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentEnd

If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentLeft

If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentRight

If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentStart

If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignParentTop

If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".

android:layout_alignRight

Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignStart

Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_alignTop

Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_below

Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

android:layout_centerHorizontal

If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".

android:layout_centerInParent

If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".

android:layout_centerVertical

If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".

android:layout_toEndOf Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toLeftOf Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toRightOf Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".
android:layout_toStartOf Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@[+][package:]type:name".

Example 04

- Create a new project
- Change the Layout to RelativeLayout.
- Change the code as follows and see how the layout behaves.

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

    <!--Text View for Displaying SIGN IN Text At Top of UI-->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="SIGN IN"
        android:id="@+id/textView3"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <!--Text View for Displaying Username-->

    <TextView
        android:id="@+id/userName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="110dp"
        android:text="UserName:"
        android:textColor="#000000"
        android:textSize="20sp" />
```

01

```xml
    <!--Text View for Displaying Password-->

    <TextView
        android:id="@+id/password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/userName"
        android:text="Password:"
        android:textColor="#000000"
        android:textSize="20sp" />

    <!--Edit Text for Filling Username-->

    <EditText
        android:id="@+id/edt_userName"
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:layout_marginTop="100dp"
        android:layout_toRightOf="@+id/userName"
        android:hint="User Name" />

    <!--Edit Text for Filling Password-->

    <EditText
        android:layout_width="fill_parent"
        android:layout_height="40dp"
        android:layout_below="@+id/edt_userName"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@+id/password"
        android:hint="Password" />
```

02

03

04

```
    <!--Button for Clicking after filling details-->

    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/password"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:background="#03B424"
        android:text="Login"
        android:textColor="#ffffff"
        android:textStyle="bold" />
</RelativeLayout>
```

05

01 – You are placing SIGN IN TextView at the top and horizontal center of the layout.

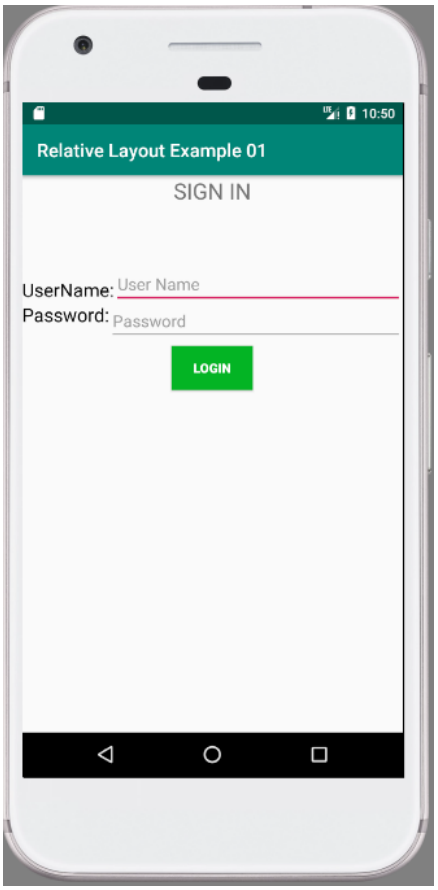
02 – You are placing the password TextView at the bottom of user name Text View.

03 – You are placing user name EditText to the right of user name Text View.

04 – You are placing password EditText below user name EditText, centering it vertically and place it to the right of password TextView.

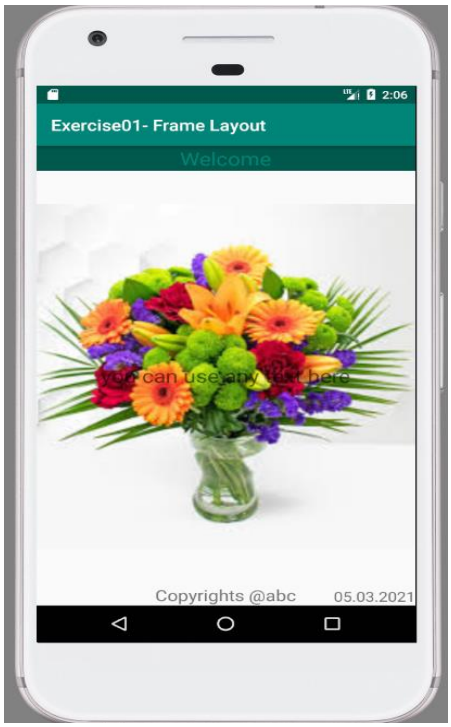
05 – You are placing the LOGIN button below password TextView and centering it horizontally.

Output:



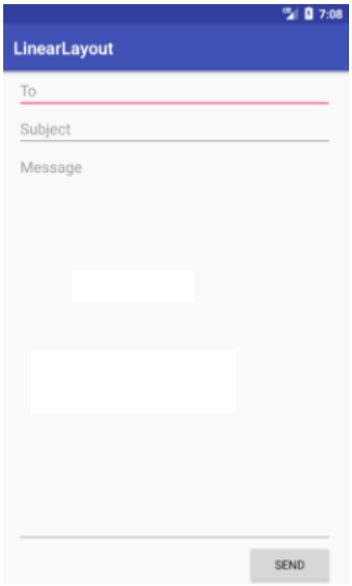
Exercise 01 – Frame Layouts

- Create the following user interface using frame layouts.
- You can change text, colors and the image as you wish.



Exercise 02 – Linear Layouts

- Create the following user interface using Linear layouts.
- There are 3 EditText and 1 Button.



Exercise 03 – Linear Layouts

- Create the following user interface using Linear layouts.
- **Hint: You can place one Linear Layout inside another Linear Layout as parents and children.**
- **You can change the texts and images as you wish. Consider the arrangement of the views only.**

Exercise 04 – Relative Layouts

- Create the following user interface using Relative layouts.