

Android Layout

Yuliana Setiowati
Rizky Yuniar Hakkun



Politeknik Elektronika Negeri Surabaya

Outline

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView Layout



The View Class

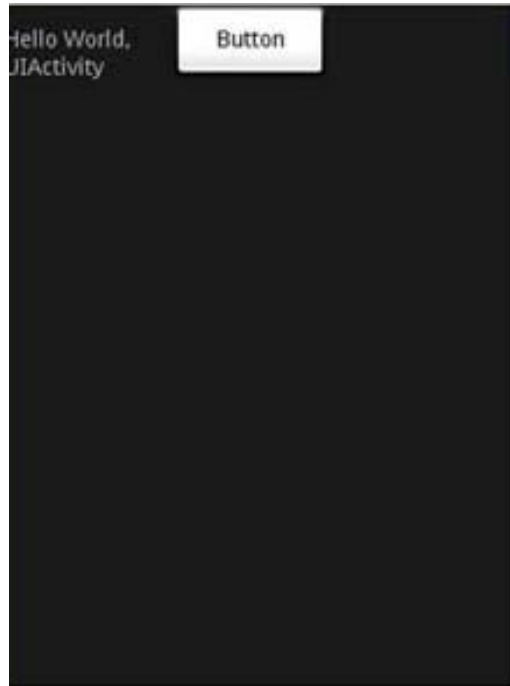
- The **View** class represents the basic for user interface components.
- View is the base class for ***widgets***, which are used to create interactive UI components (*buttons, text fields, etc.*).
- The **ViewGroup** subclass is the base class for ***layouts***, which are invisible containers that hold other Views (or other ViewGroups)



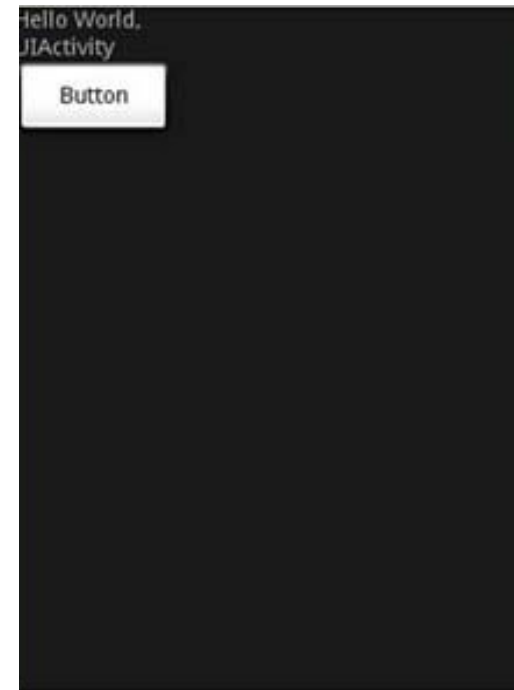
A brief sample of UI components

Linear Layout

A `LinearLayout` is a `ViewGroup` that will lay child `View` elements vertically or horizontally.



Horizontal mode



Vertical mode



A brief sample of UI components

Table Layout

A `TableLayout` is a `ViewGroup` that will lay child `View` elements into rows and columns.



Relative Layout

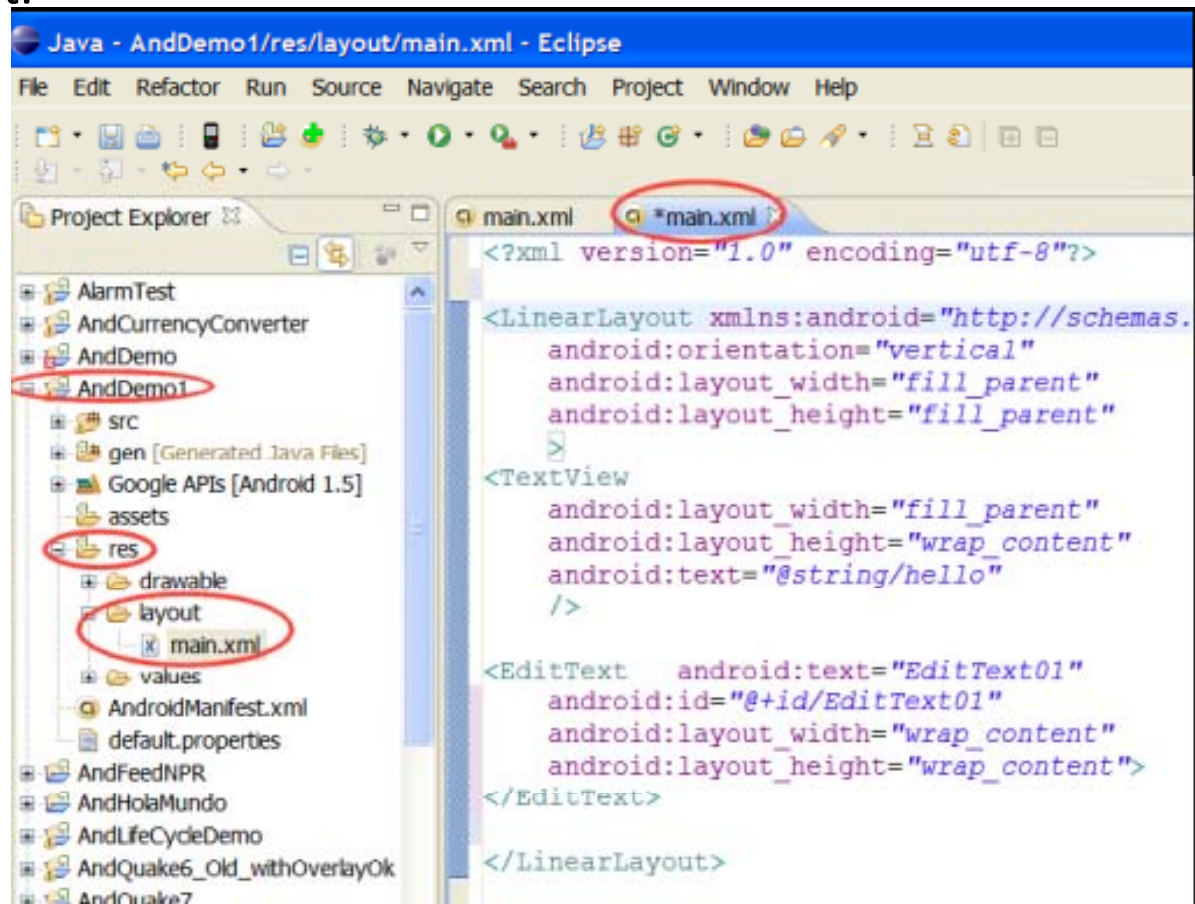
A `RelativeLayout` is a `ViewGroup` that allows you to layout child elements in positions relative to the parent or siblings elements.



What is an XML Layout?

- An XML-based layout is a specification of the various UI components (widgets) and the relationships to each other –and to their containers –all written in XML format.

Android considers XML-based layouts to be **resources**, and as such layout files are stored in the **res/layout** directory inside your Android project.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <EditText android:text="EditText01"
        android:id="@+id/EditText01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```



ANDROID

LinearLayout

- **LinearLayout aligns all children in a single direction —vertically or horizontally depending on the `android:orientation` attribute.**
- **All children are stacked one after the other, so a**
 - *Vertical* list will only have one child per row, no matter how wide they are, and a
 - *Horizontal* list will only be one row high (the height of the tallest child, plus padding).



LinearLayout

Example 1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

xmlns:android="http://schemas.android.com/a
pk/res/android"
    >
    <TextView
        android:layout_width="105px"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <Button
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Button"
    />
</LinearLayout>
```



LinearLayout

- The default orientation of LinearLayout is set to **horizontal**.
- If you want to change its orientation to vertical, set the orientation attribute to vertical

```
<LinearLayout
```

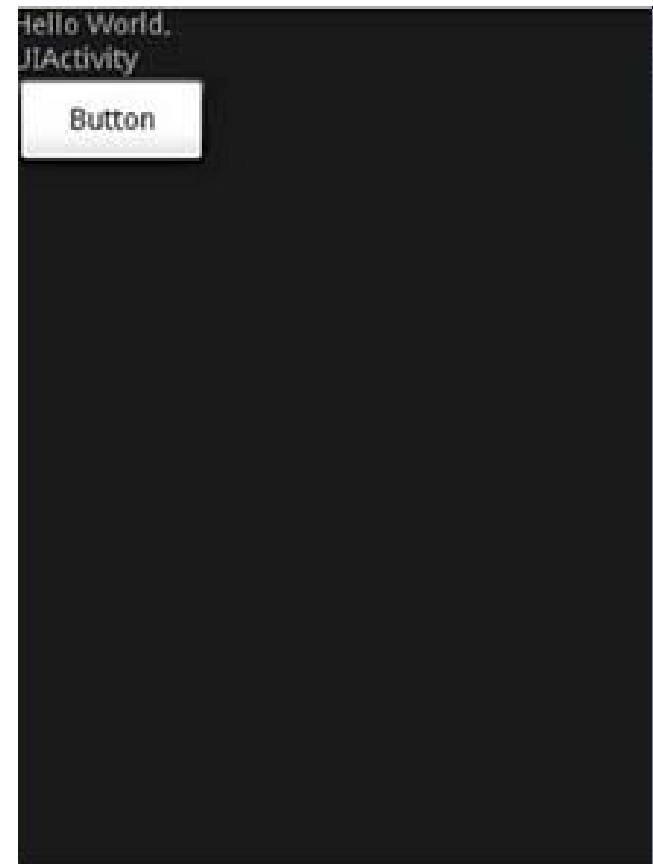
```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical"
```

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

```
>
```



LinearLayout

Example 2

the button is aligned to the right of its parent (which is the LinearLayout) using the `layout_gravity` attribute. At the same time, you use the `layout_weight` attribute to specify the ratio in which the Button and EditText views occupy the remaining space on the screen. The total value for the `layout_weight` attribute must be equal to 1.



LinearLayout

Example 2 - In LinearLayout, you can apply the `layout_weight` and `layout_gravity` attributes to views contained within it

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  >
  <TextView
    android:layout_width="105px"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
  <Button
    android:layout_width="100px"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_gravity="right"
    android:layout_weight="0.2"
    />
  <EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:layout_weight="0.8"
    />
</LinearLayout>
```



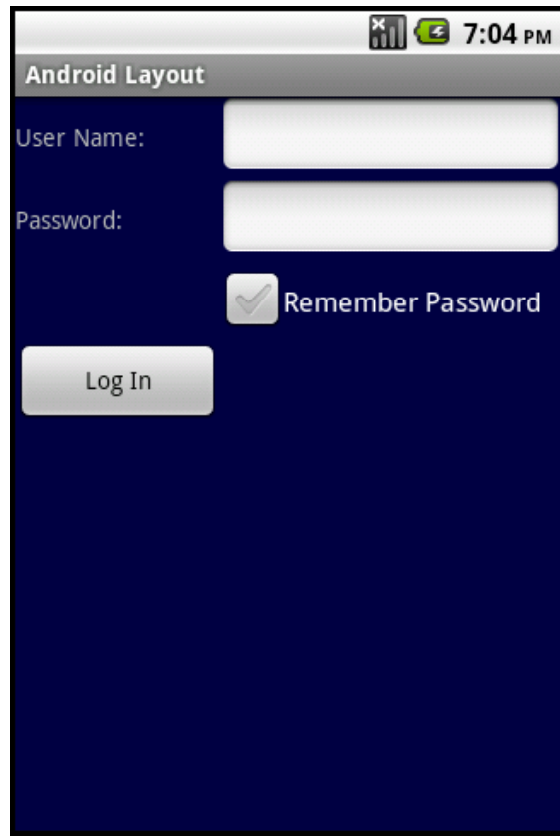
TableLayout

- TableLayout is a ViewGroup that will lay child View elements into rows and columns.



TableLayout

There are two columns and four rows in the TableLayout. The cell directly under the Password TextView is populated with an empty element. If you don't do this, the Remember Password checkbox will then appear under the Password TextView



TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
  xmlns:android="http://schemas.android.
  com/apk/res/android"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:background="#000044">
  <TableRow>
    <TextView
      android:text="User Name:"
      android:width="120px"
    />
    <EditText
      android:id="@+id/txtUserName"
      android:width="200px" />
  </TableRow>
```



```
<TableRow>
  <TextView
    android:text="Password:"
  />
  <EditText
    android:id="@+id/txtPassword"
    android:password="true"
  />
</TableRow>
<TableRow>
  <TextView />
  <CheckBox android:id="@+id/chkRememberPassword"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Remember Password"
  />
</TableRow>
<TableRow>
  <Button
    android:id="@+id/buttonSignIn"
    android:text="Log In" />
</TableRow>
</TableLayout>
```

AbsoluteLayout

- The AbsoluteLayout lets you specify the exact location of its children. Consider the following UI defined in main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  >
  <Button
    android:layout_width="188px"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_x="126px"
    android:layout_y="361px"
  />
  <Button
    android:layout_width="113px"
    android:layout_height="wrap_content"
    android:text="Button"
    android:layout_x="12px
    android:layout_y="361px"
  />
</AbsoluteLayout>
```



the two Button views located at their specified positions using the android_layout_x and android_layout_y attributes

RelativeLayout

- The RelativeLayout lets you specify how child views are positioned relative to each other
- The following properties manage positioning of a widget respect to other widgets:
 - **android:layout_above** indicates that the widget should be placed above the widget referenced in the property
 - **android:layout_below** indicates that the widget should be placed below the widget referenced in the property
 - **android:layout_toLeftOf** indicates that the widget should be placed to the left of the widget referenced in the property
 - **android:layout_toRightOf** indicates that the widget should be placed to the right of the widget referenced in the property

For example, assigning the parameter

```
android:layout_toLeftOf="@+id/my_button"
```

to a TextView would place the TextView to the left of the View with the ID *my_button*



RelativeLayout

- **android:layout_alignTop** indicates that the widget's top layout_widgets should be aligned with the top of the widget referenced in the property
- **android:layout_alignBottom** indicates that the widget's bottom should be aligned with the bottom of the widget referenced in the property
- **android:layout_alignLeft** indicates that the widget's left should be aligned with the left of the widget referenced in the property
- **android:layout_alignRight** indicates that the widget's right should be aligned with the right of the widget referenced in the property
- **android:layout_alignBaseline** indicates that the baselines of the two widgets should be aligned



RelativeLayout



RelativeLayout

- Consider the following main.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/a
    ndroid" >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
    />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
    />
```

```
<Button
    android:id="@+id/btnSave"
    android:layout_width="125px"
    android:layout_height="wrap_content"
    android:text="Save"
    android:layout_below="@+id/txtComments"
    android:layout_alignRight="@+id/txtComments"
/>
<Button
    android:id="@+id/btnCancel"
    android:layout_width="124px"
    android:layout_height="wrap_content"
    android:text="Cancel"
    android:layout_below="@+id/txtComments"
    android:layout_alignLeft="@+id/txtComments"
/>
</RelativeLayout>
```



FrameLayout

- FrameLayout is the simplest type of layout object. It's basically a *blank space* on your screen that you can later fill with a single object —for example, a picture that you'll swap in and out.
- Views that you add to a FrameLayout is always anchored to the top left of the layout.



FrameLayout

- Here, you have a FrameLayout within an AbsoluteLayout. Within the FrameLayout, you embed an ImageView view.
- Note: This example assumes that the res/drawable folder has an image named androidlogo.png.

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:id="@+id/widget68"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
```

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

```
>
```

<FrameLayout

```
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_x="40px"
  android:layout_y="35px"
```

```
>
```

<ImageView

```
  android:src = "@drawable/androidlogo"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
```

```
/>
```

```
</FrameLayout>
```

```
</AbsoluteLayout>
```



ANDROID

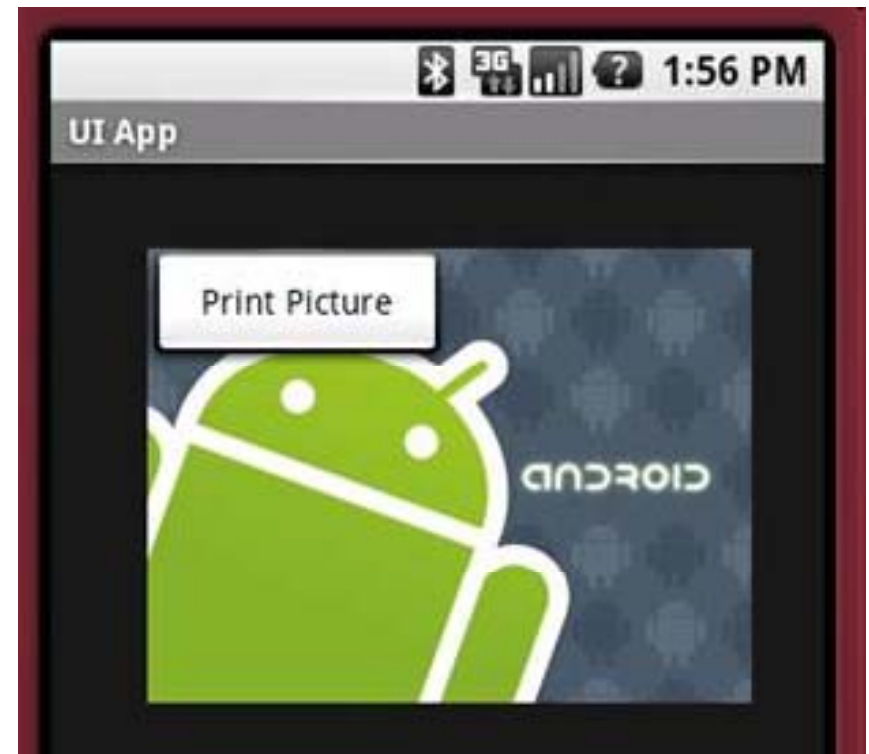
Politeknik Elektronika Negeri Surabaya



FrameLayout

- If you add another view (such as a Button view) within the FrameLayout, the view will overlap the previous view

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:id="@+id/widget68"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  >
  <FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_x="40px"
    android:layout_y="35px"
    >
    <ImageView
      android:src="@drawable/androidlogo"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      />
    <Button
      android:layout_width="124px"
      android:layout_height="wrap_content"
      android:text="Print Picture"
      />
  </FrameLayout>
</AbsoluteLayout>
```



ScrollView

- A ScrollView is a special type of FrameLayout in that it allows users to scroll through a list of views that occupy more space than the physical display.
- The ScrollView can contain only one child view or ViewGroup, which normally is a LinearLayout.
- Note: Do not use a ListView together with the ScrollView. The ListView is designed for showing a list of related information and is optimized for dealing with large lists.



ScrollView

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    android:id="@+id/widget54"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <LinearLayout
        android:layout_width="310px"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1"
            />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2"
            />
```



```
<Button
    android:id="@+id/button3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 3"
    />
<EditText
    android:id="@+id/txt"
    android:layout_width="fill_parent"
    android:layout_height="300px"
    />
<Button
    android:id="@+id/button4"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 4"
    />
<Button
    android:id="@+id/button5"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Button 5"
    />
</LinearLayout>
</ScrollView>
```


ScrollView

