



SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution



Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

COURSE NAME: 19CS603-Mobile Application Development

III YEAR /VI SEMESTER

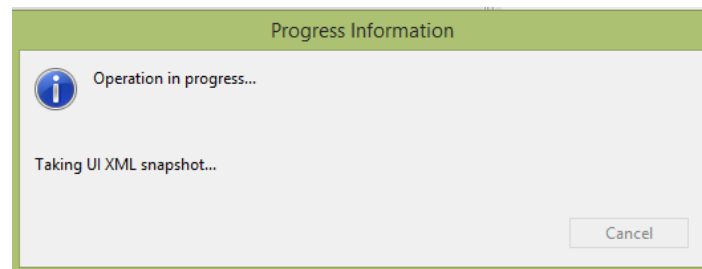
UNIT – II Basic Design

UI screen components

A typical user interface of an android application consists of action bar and the application content area.

- Main Action Bar
- View Control
- Content Area
- Split Action Bar

These components have also been shown in the image below –



Understanding Screen Components

The basic unit of android application is the activity. A UI is defined in an xml file. During compilation, each element in the XML is compiled into equivalent Android GUI class with attributes represented by methods.

View and ViewGroups

An activity is consisting of views. A view is just a widget that appears on the screen. It could be button e.t.c. One or more views can be grouped together into one ViewGroup. Example of ViewGroup includes layouts.

Types of layouts

There are many types of layouts. Some of which are listed below –

- Linear Layout
- Absolute Layout
- Table Layout
- Frame Layout
- Relative Layout

Linear Layout

Linear layout is further divided into horizontal and vertical layout. It means it can arrange views in a single column or in a single row. Here is the code of linear layout (vertical) that includes a text view.

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

Absolute Layout

The Absolute Layout enables you to specify the exact location of its children. It can be declared like this.

```
<AbsoluteLayout
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android" >
<Button
android:layout_width="188dp"
android:layout_height="wrap_content"
android:text="Button"
android:layout_x="126px"
android:layout_y="361px" />
</AbsoluteLayout>
```

Table Layout

The Table Layout groups views into rows and columns. It can be declared like this.

```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="fill_parent"
android:layout_width="fill_parent" >
<TableRow>
<TextView
```

```

android:text="User
Name:"android:width
="120dp"
/>
<EditText
android:id="@+id/txtUserName"
android:width="200dp" />
</TableRow>
</TableLayout>

```

Relative Layout

The Relative Layout enables you to specify how child views are positioned relative to each other. It can be declared like this.

```

<RelativeLayout
android:id="@+id/RLayout"
android:layout_width="fill_pare
nt"
android:layout_height="fill_pare
nt"
xmlns:android="http://schemas.android.com/apk/res/android" >
</RelativeLayout>

```

Frame Layout

The Frame Layout is a placeholder on screen that you can use to display a single view. It can be declared like this.

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
android:layout_width="wrap_content"
android:layout
android:layout_alignLeft="@+id/lblComments"
android:layout_below="@+id/lblComments"
android:layout_centerHorizontal="true" >
<ImageView
android:src = "@drawable/droid"
android:layout_width="wrap_conte
nt"
android:layout_height="wrap_conte
nt" />
</FrameLayout>

```

Apart from these attributes, there are other attributes that are common in all views and ViewGroups. They are listed below –

Sr.No	View & description
1	layout_width Specifies the width of the View or ViewGroup
2	layout_height Specifies the height of the View or ViewGroup
3	layout_marginTop Specifies extra space on the top side of the View or ViewGroup
4	layout_marginBottom Specifies extra space on the bottom side of the View or ViewGroup
5	layout_marginLeft Specifies extra space on the left side of the View or ViewGroup
6	layout_marginRight Specifies extra space on the right side of the View or ViewGroup
7	layout_gravity Specifies how child Views are positioned
8	layout_weight Specifies how much of the extra space in the layout should be allocated to the View

Units of Measurement

When you are specifying the size of an element on an Android UI, you should remember the following units of measurement.

Sr.No	Unit & description
1	dp Density-independent pixel. 1 dp is equivalent to one pixel on a 160 dpi screen.
2	sp Scale-independent pixel. This is similar to dp and is recommended for specifying font sizes
3	pt Point. A point is defined to be 1/72 of an inch, based on the physical screen size.
4	px Pixel. Corresponds to actual pixels on the screen

Screen Densities

Sr.No	Density & DPI
1	Low density (ldpi) 120 dpi
2	Medium density (mdpi) 160 dpi
3	High density (hdpi) 240 dpi
4	Extra High density (xhdpi) 320 dpi

Optimizing layouts

Here are some of the guidelines for creating efficient layouts.

- Avoid unnecessary nesting
- Avoid using too many Views
- Avoid deep nesting