styles

style resource defines the format and look for a UI. A style can be applied to an individual View (from A within a layout file) or to an entire Activity or application (from within the manifest file).

Defining Styles

- style is defined in an XML resource that is separate from the XML that specifies the layout. This XML file resides under **res/values/** directory of your project and will have **<resources>** as the root node which is mandatory for the style file. The name of the XML file is arbitrary, but it must use the .xml extension.
- You can define multiple styles per file using **<style>** tag but each style will have its name that uniquely identifies the style. Android style attributes are set using **<item>** tag as shown below

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

<resources>

<stylename="CustomFontStyle">

<itemname="android:layout_width">fill_parent</item>

<itemname="android:layout_height">wrap_content</item>

<itemname="android:capitalize">characters</item>

<itemname="android:typeface">monospace</item>

<itemname="android:textSize">12pt</item>

<itemname="android:textColor">#00FF00</item>/>

</style>
```

Using Styles

Once your style is defined, you can use it in your XML Layout file using **style**attribute as follows

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

<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

android:orientation="vertical">



<TextView

android:id="@+id/text_id"

style="@style/CustomFontStyle"

android:text="@string/hello_world"/>



</LinearLayout>
```

Android Themes

- A theme is nothing but an Android style applied to an entire Activity or application, rather than an individual View.
- Thus, when a style is applied as a theme, every **View** in the Activity or application will apply each style property that it supports.
- For example, you can apply the same **CustomFontStyle** style as a theme for an Activity and then all text inside that **Activity** will have green monospace font.

SETTING THEME

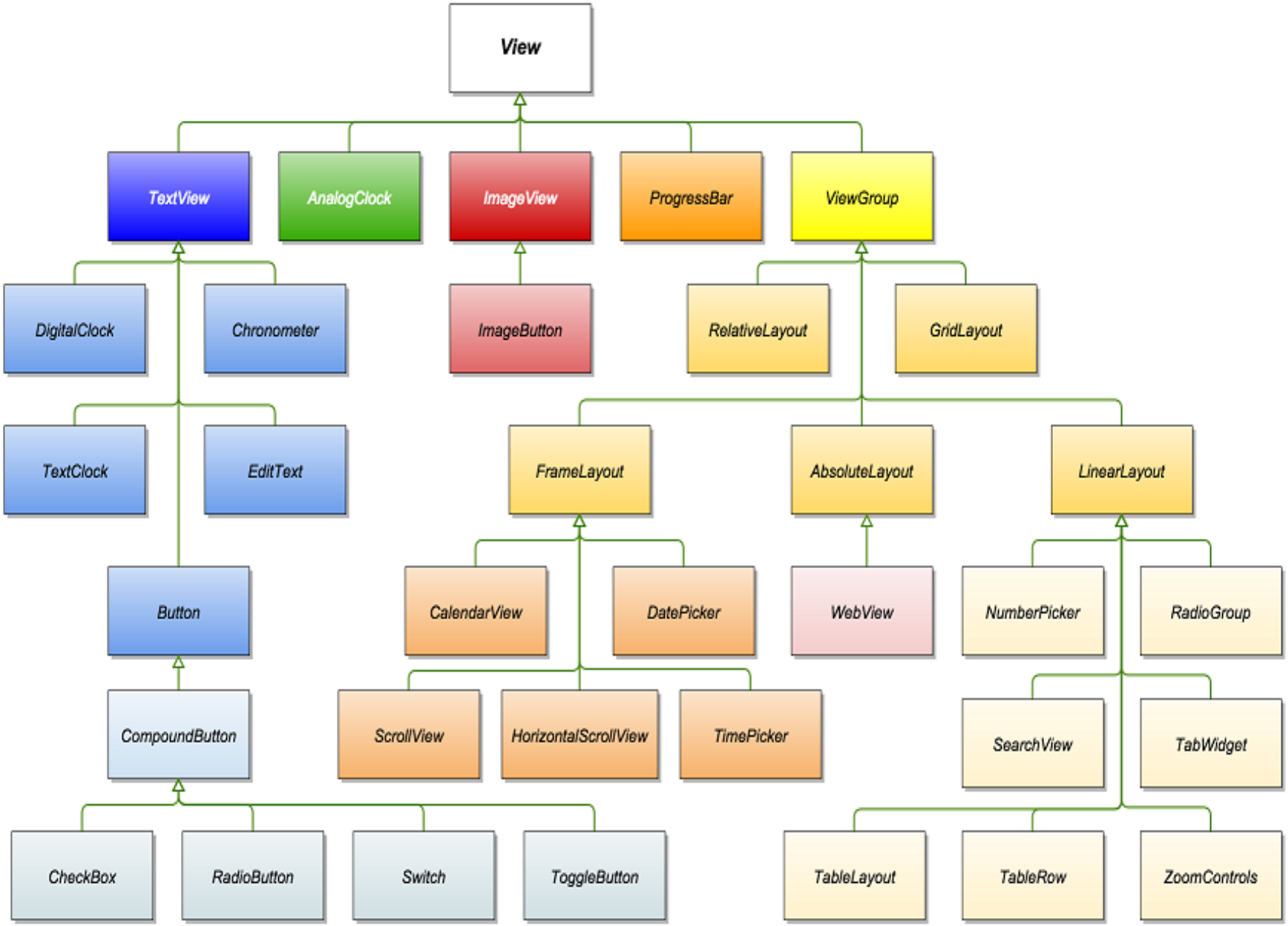
- To set a theme for all the activities of your application, open the **AndroidManifest.xml** file and edit the **<application>** tag to include attribute with the style name. For example −

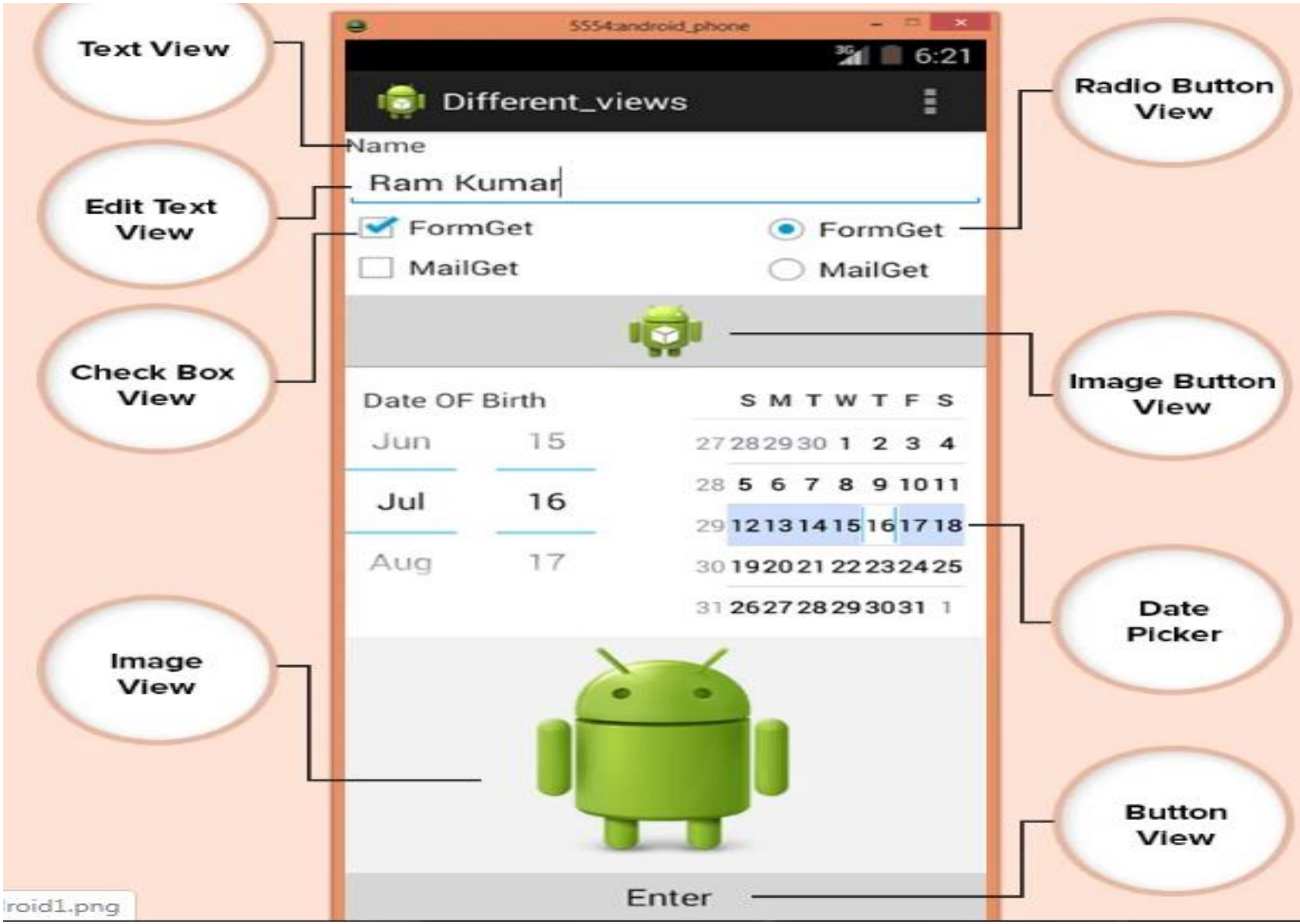
<application android:theme="@style/CustomFontStyle">

- But if you want a theme applied to just one Activity in your application, then add the android:theme attribute to the <activity> tag only. For example −

<activity android:theme="@style/CustomFontStyle"

The Android View Class




Text View
Edit Text View
Check Box View
Image View
Radio Button View
Image Button View
Date Picker
Button View
5554:android_phone
6:21
Different_views
Name
Ram Kumar
FormGet
MailGet
FormGet
MailGet
Date OF Birth
Jun 15
Jul 16
Aug 17
S M T W T F S
Enter

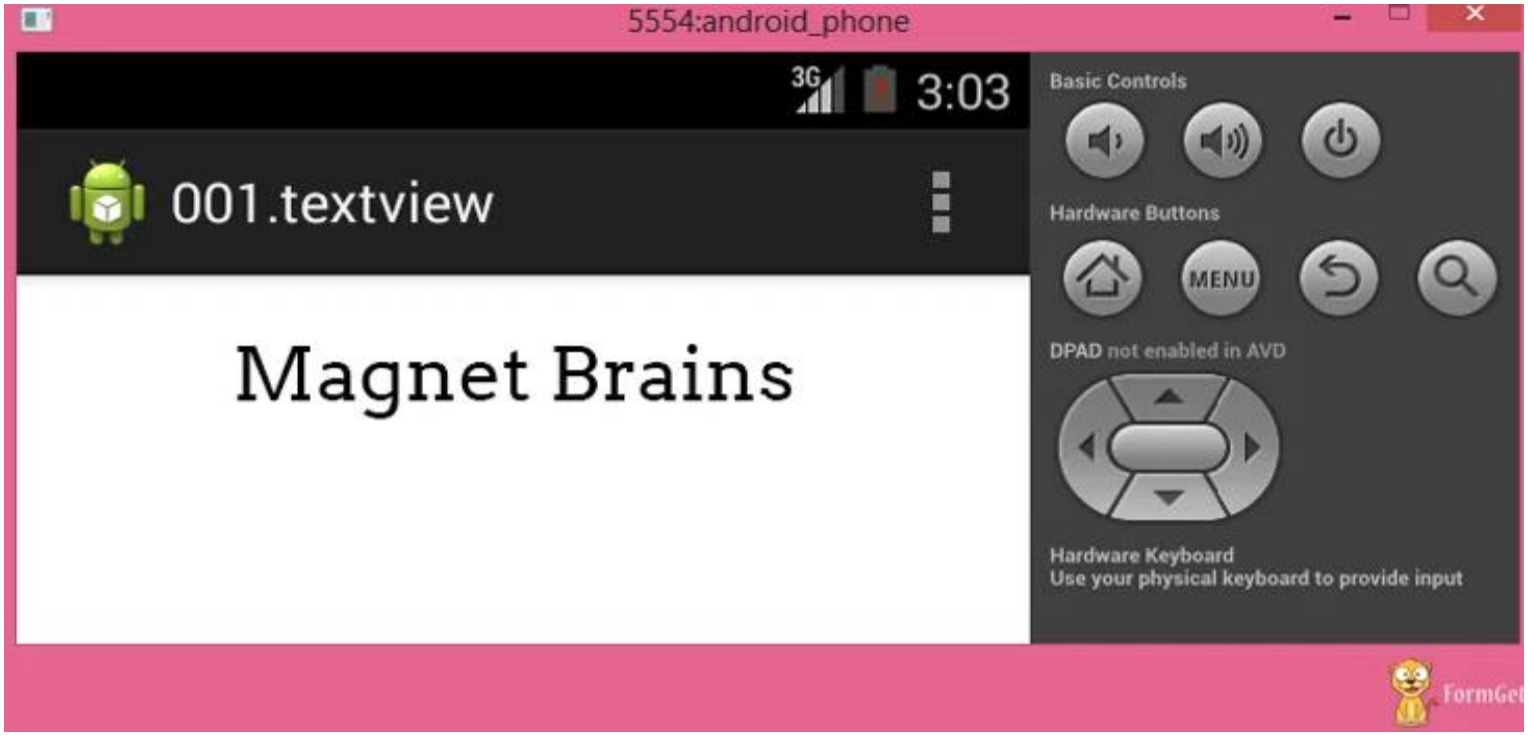
Most Commonly Used Android View classes:

These views can be used to create a useful input and output fields.

- Text View
- EditText
- Button
- ImageView
- ImageButton
- CheckBox
- Radio button
- RadioGroup
- ListView
- Spinner
- AutoCompleteTextView

Text View

- This class is used to display text on the android application screen. It also allows user to optionally edit it. Although it contains text editing operations, the basic class does not allow editing, So Edit Text class is included to do so.

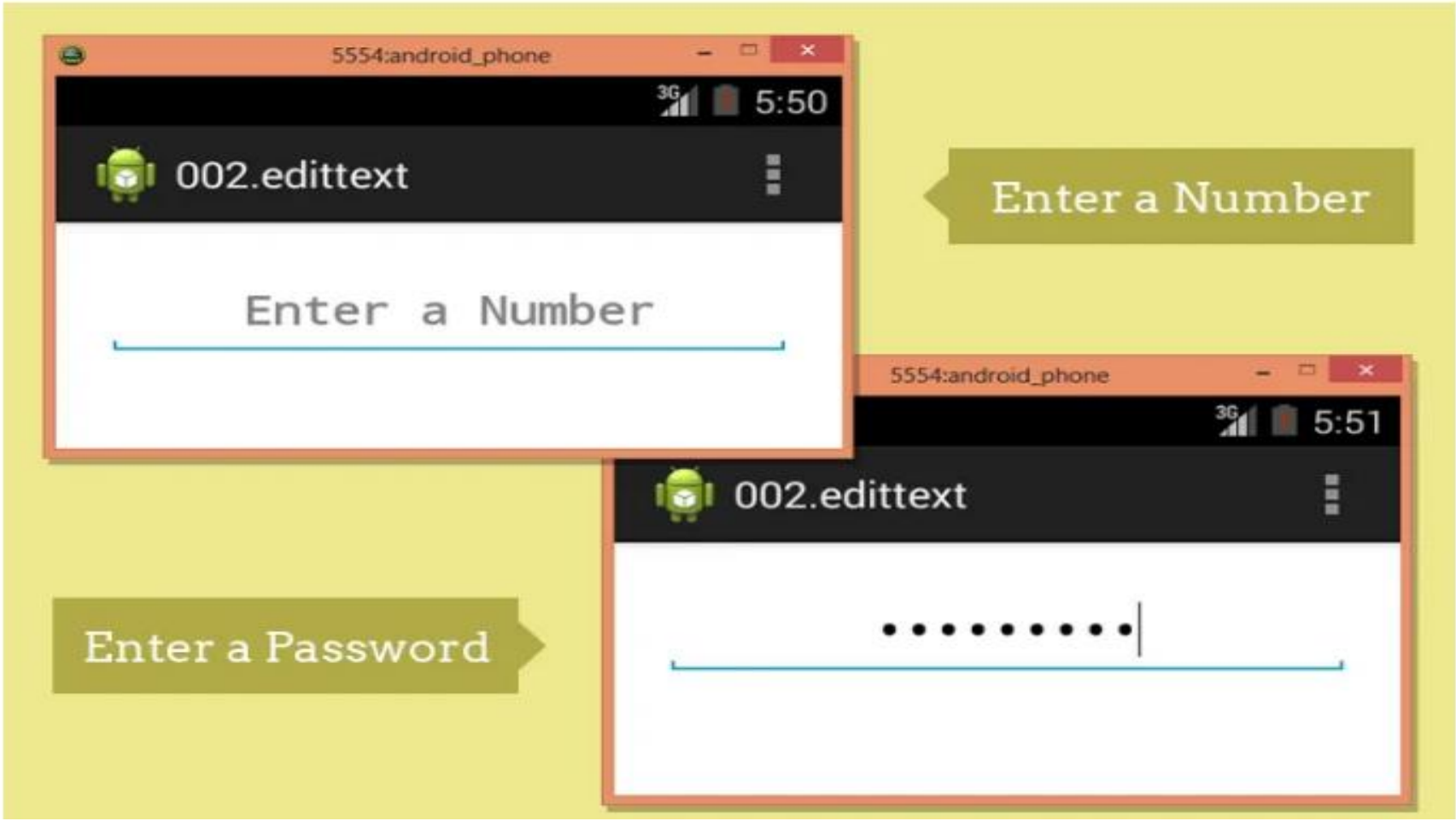


XML coding of text view:

```
<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:id="@+id/myTextview"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Mangnet Brains"
android:textSize="25dp"
android:textColor="@android:color/black"
android:typeface="serif"
android:gravity="center"
android:padding="10dp"
android:layout_margin="20dp" />
</LinearLayout>
```

Edit Text

This class makes text to be editable in Android application. It helps in building the data interface taken from any user, also contains certain features through which we can hide the data which are confidential.



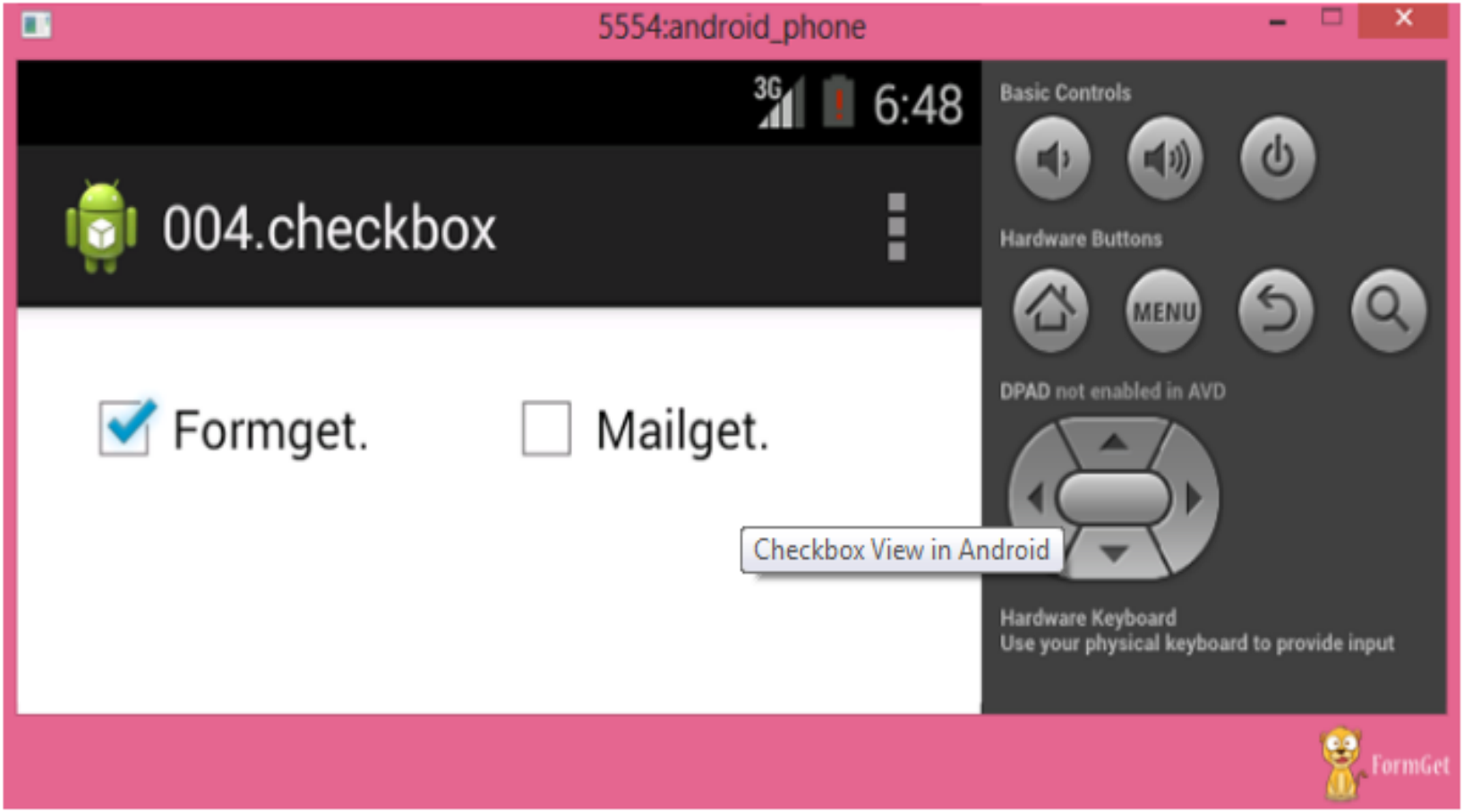
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >

<EditText
android:id="@+id/myEdittext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="20dp"
android:textStyle="bold"
android:typeface="serif"
android:gravity="center"
android:padding="10dp"
android:layout_margin="20dp"
android:hint="Enter a Number"
android:singleLine="true"
android:inputType="textPassword" />

</LinearLayout>
```

Image view:

Image view helps to display images in an android application. Any image can be selected, we just have to paste our image in a drawable folder from where we can access it. For example: In below Code **"@drawable/ic_laucher"** has been taken.

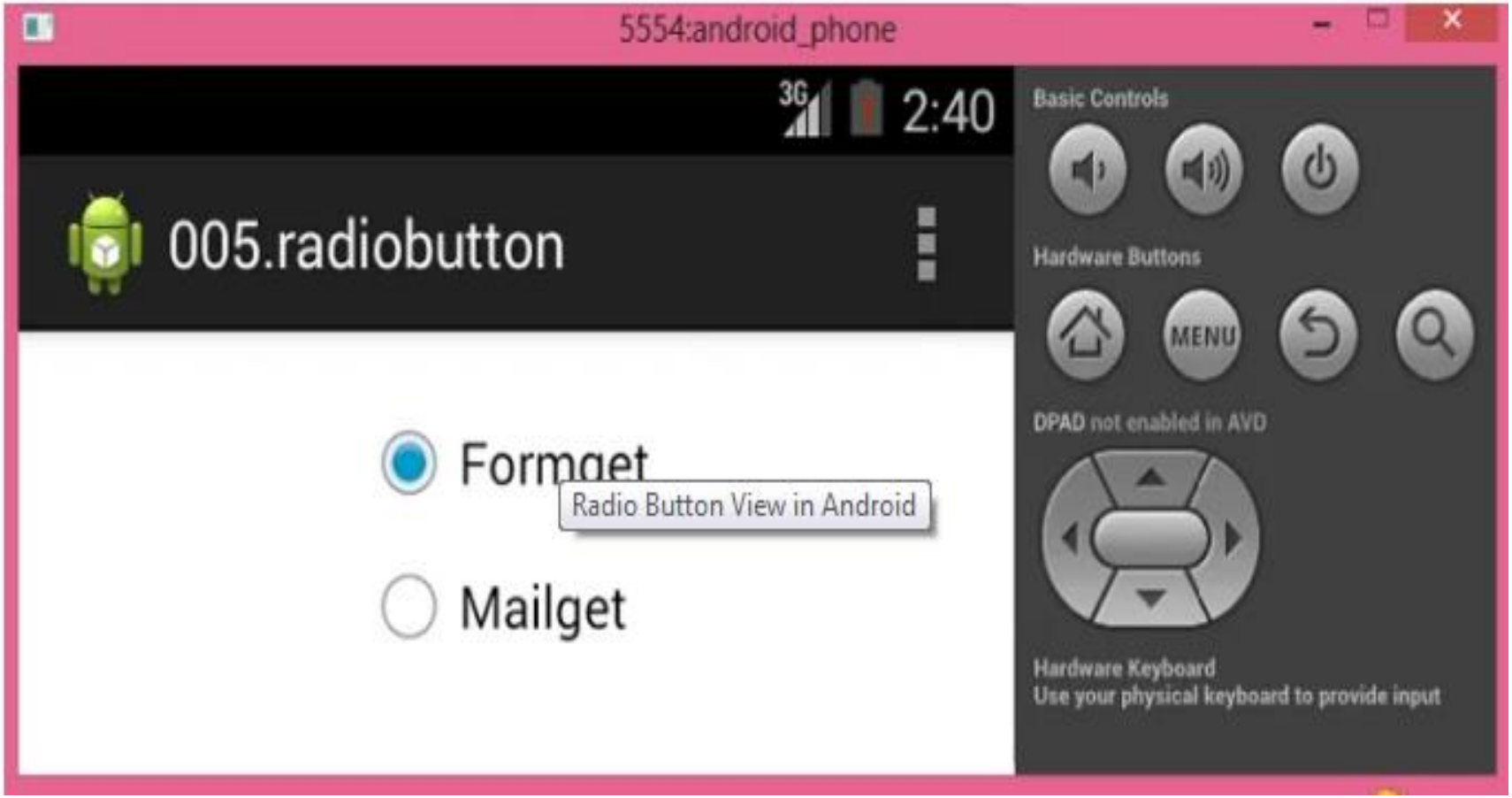
Check Box:

Checkbox is used in that applications where we have to select one option from multiple provided. Checkbox is mainly used when 2 or more options are present.



Radio Button:

Radio button is like checkbox, but there is slight difference between them. Radio button is a two-states button that can be either checked or unchecked.



Button View:

This class is used to create a button on an application screen. Buttons are very helpful in getting into a content. Android button represents a clickable push-button widget.

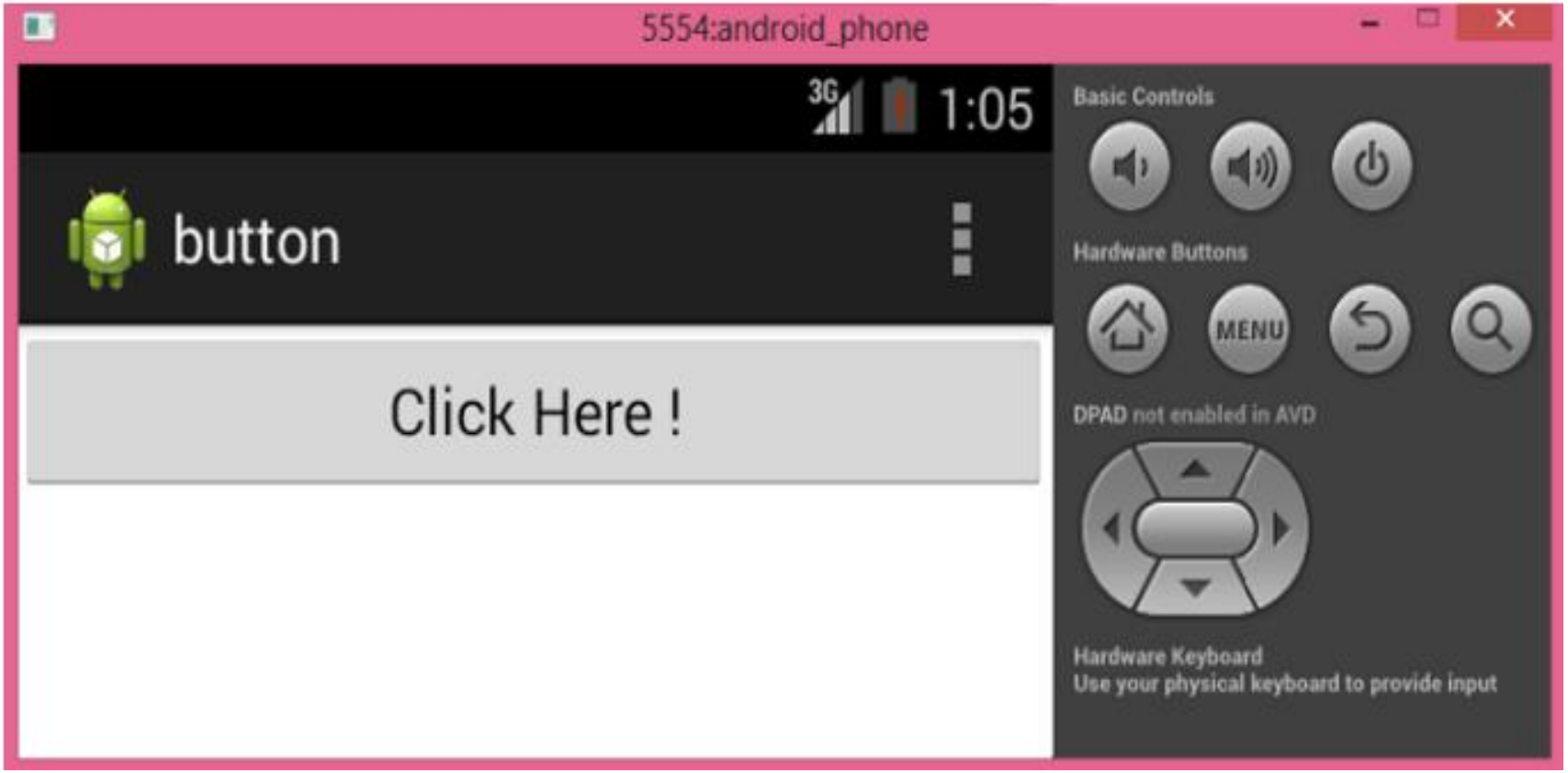
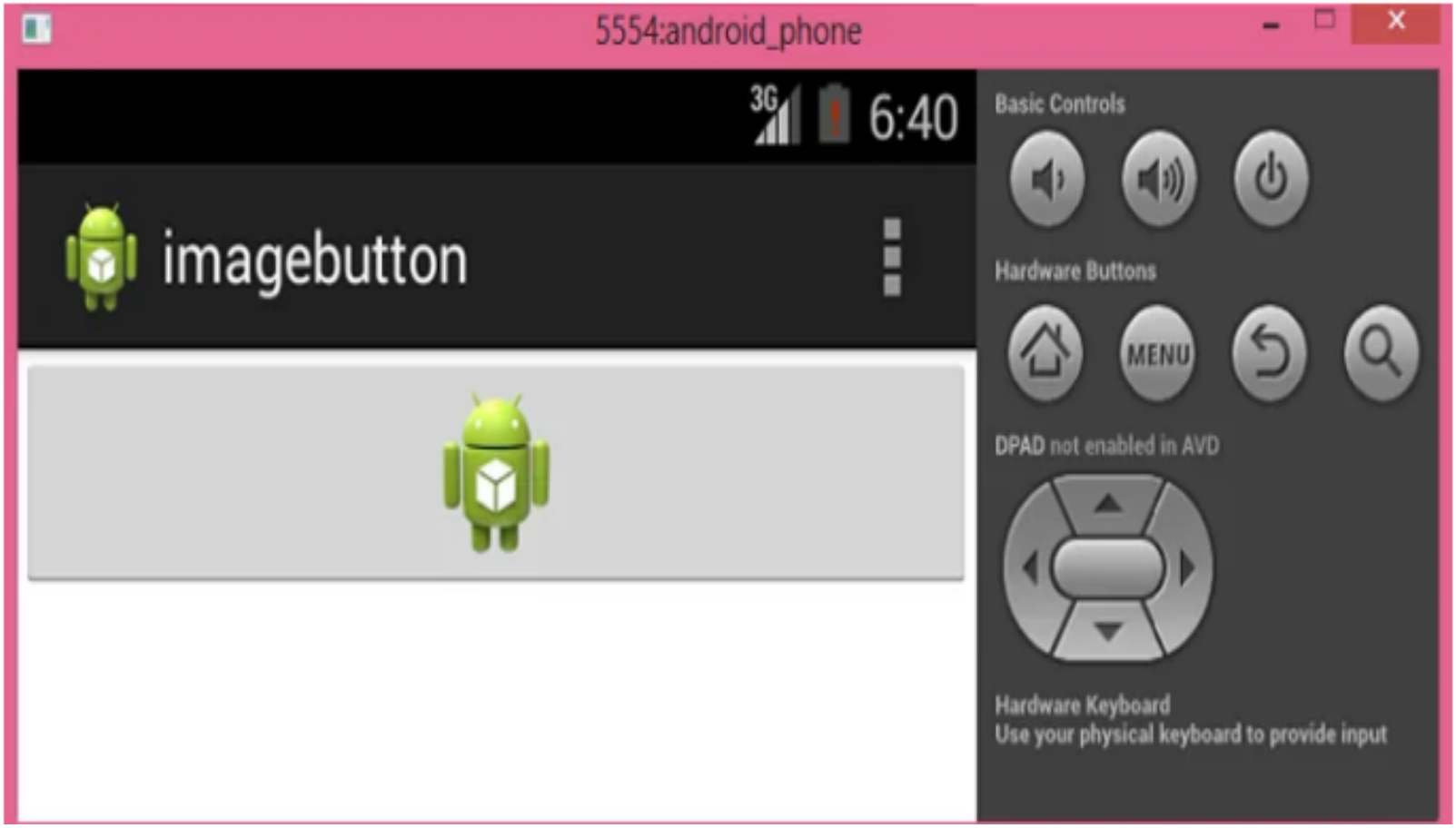


Image Button View:

Image button is a button but it carries an image on it. We can put an image or a certain text on it and when we click it provides the operations assigned to it.



LAYOUT

1	Linear Layout LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	Relative Layout RelativeLayout is a view group that displays child views in relative positions.
3	Table Layout TableLayout is a view that groups views into rows and columns.
4	Absolute Layout AbsoluteLayout enables you to specify the exact location of its children.
5	Frame Layout The FrameLayout is a placeholder on screen that you can use to display a single view.
6	List View ListView is a view group that displays a list of scrollable items.
7	Grid View GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.