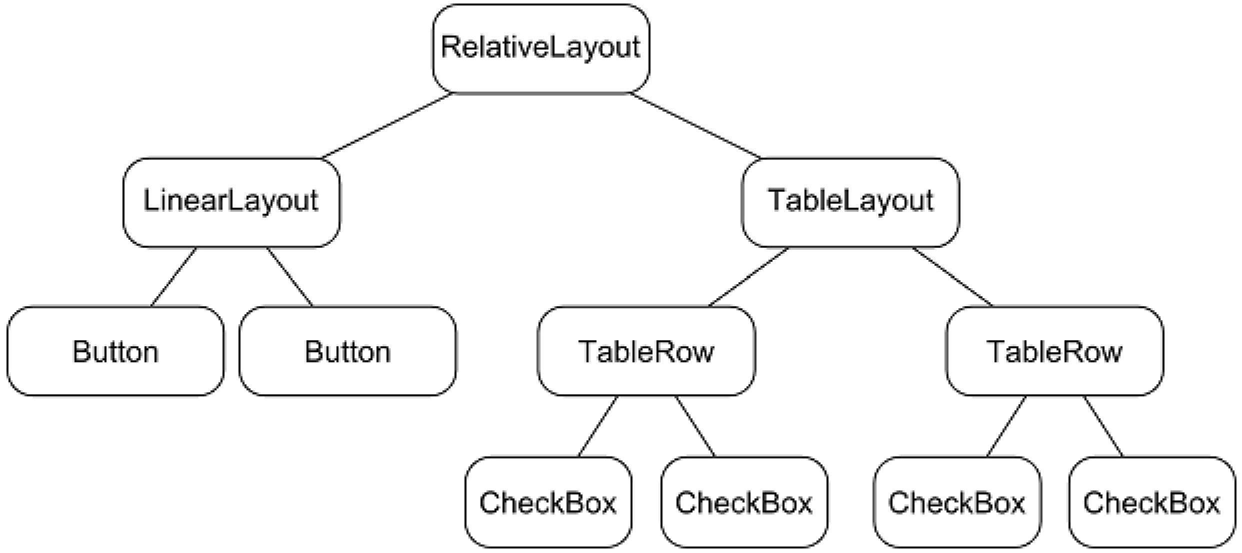
Widgets and Layouts - 1

Android Applications Development

The view Hierarchy

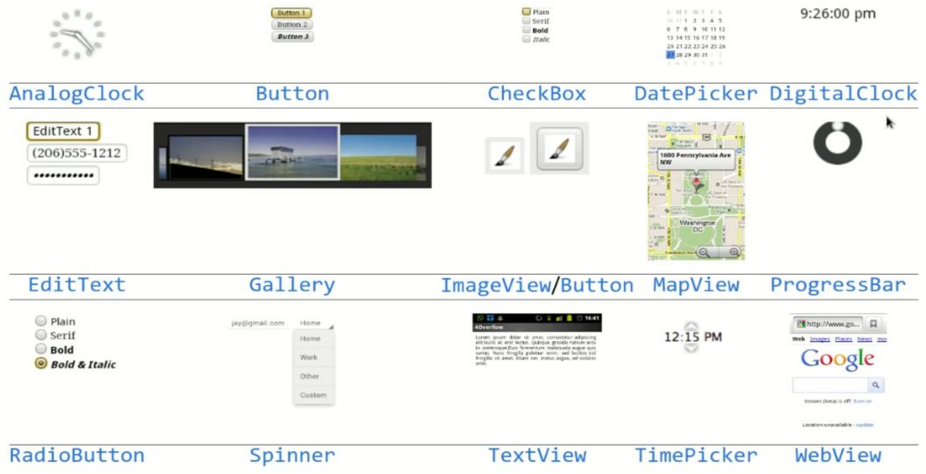
A View inside another View creates an hierarchy, the outer view becomes the parent of the inner view and the inner view is its child. It's just nested views.



Android Widgets

A widget is a small gadget or control of your android application placed on the home screen. You have probably seen some common widgets, such as music widget, weather widget, clock widget e.t.c

Widgets could be of many types such as information widgets, collection widgets, control widgets and hybrid widgets. Android provides us a complete framework to develop our own widgets.

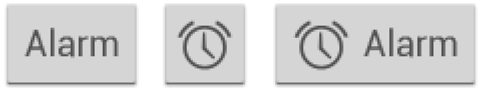


Button

A clickable widget with a text label.

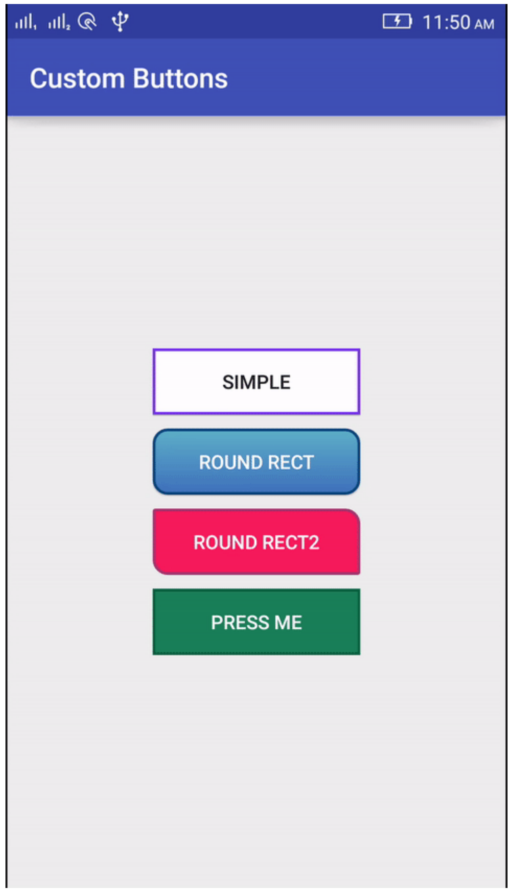
`android:clickable="`*`bool`*`"`	set to `false` to disable the button
`android:id="@+id/`*`theID`*`"`	unique ID for use in Java code
`android:onClick="`*`function`*`"`	function to call in activity when clicked (must be `public`, `void`, and take a `View` arg)
`android:text="`*`text`*`"`	text to put in the button

Key attributes in XML



Java Code

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```



TextView

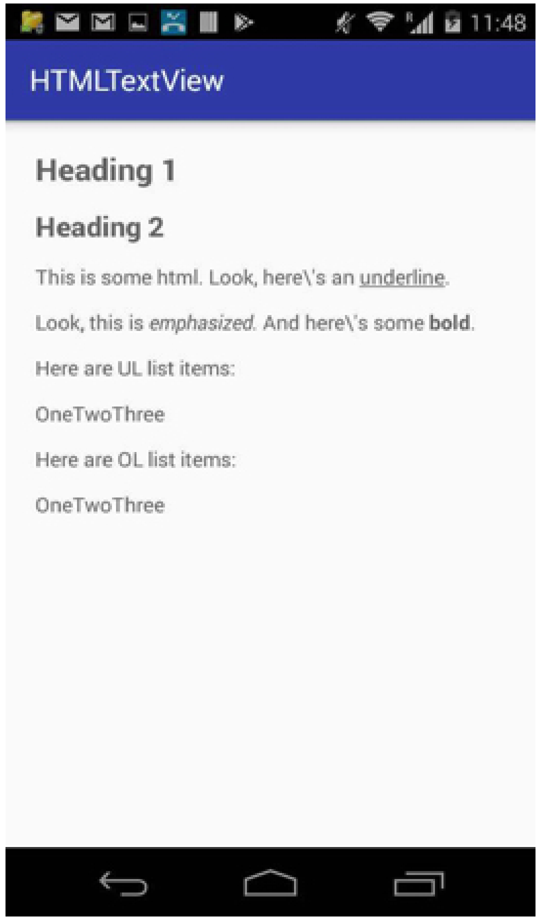
- A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing.

`android:id="@+id/theID"`	unique ID for use in Java code
`android:text="string"`	text to display

Key attributes in XML

Java Code

```
TextView textView = (TextView) findViewById(R.id.textView);
```



Heading 1

Heading 2

This is some html. Look, here\'s an underline.

Look, this is *emphasized*. And here\'s some **bold**.

Here are UL list items:

OneTwoThree

Here are OL list items:

OneTwoThree

Sizing and positioning

How does the programmer specify where each component appear, how big each component should be , etc. ?

- **Absolute positioning** (C++ , C# , others) :

  - Programmer specify where the component will appear.

  - "Put this button at (x=15,y=20) and make it 70 X 30 px in size".

- **Layout managers** (Java, Android) :

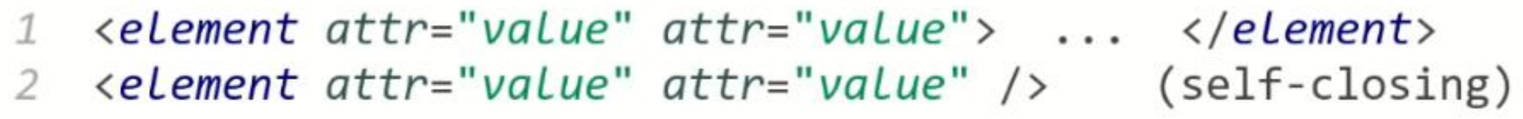
  - Objects that decide where to position each component based on some general rules or criteria.

  - " Put these four buttons into 2 X 2 grid and put these text boxes in a horizontal flow in south part of the app"

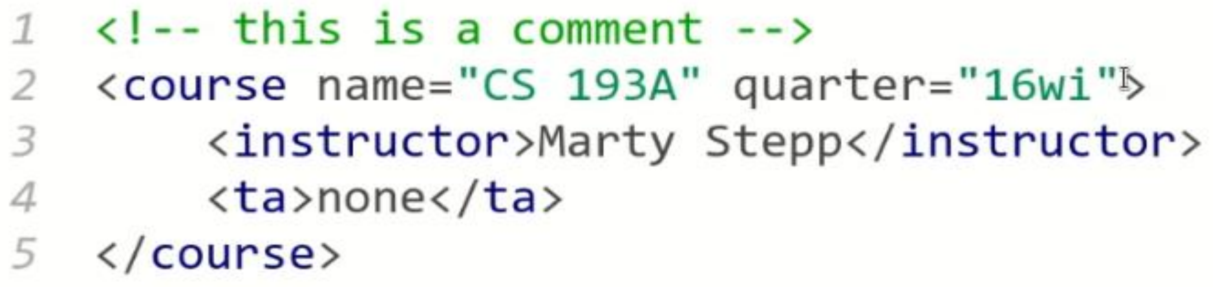
  - More flexible and general; works better with a variety of device.

XML (eXtensible Markup Language)

- **XML** : a language for describing hierarchical text data.
- Uses tags that consist of element and attributes. Tags can be nested.
- Some tag are opened and closed; others self closed.



```
<element attr="value" attr="value">  ...  </element>
<element attr="value" attr="value" />    (self-closing)
```



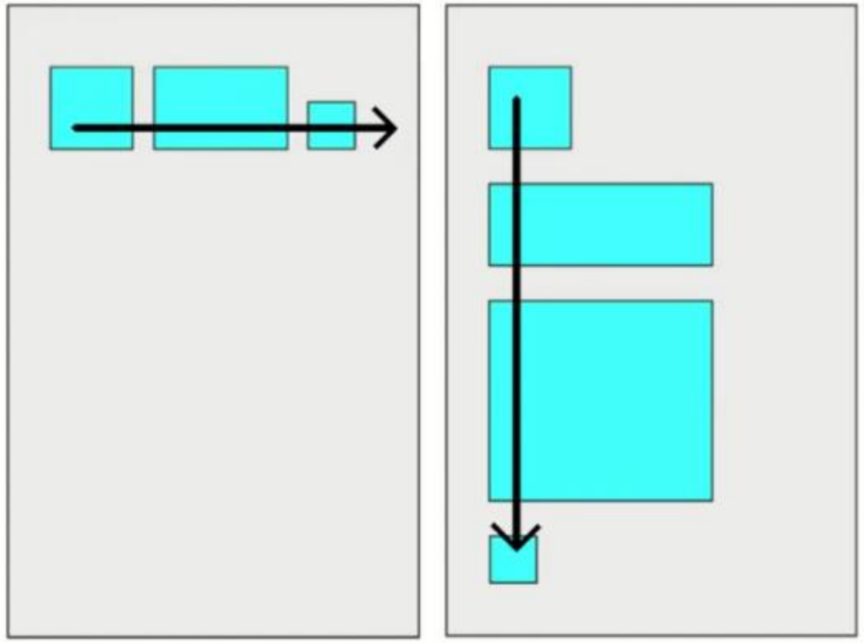
```
<!-- this is a comment -->
<course name="CS 193A" quarter="16wi">
    <instructor>Marty Stepp</instructor>
    <ta>none</ta>
</course>
```

Types of layout

1. Linear Layout
2. Relative Layout
3. Constraint Layout
4. Table Layout
5. Absolute Layout
6. Frame Layout
7. Grid Layout

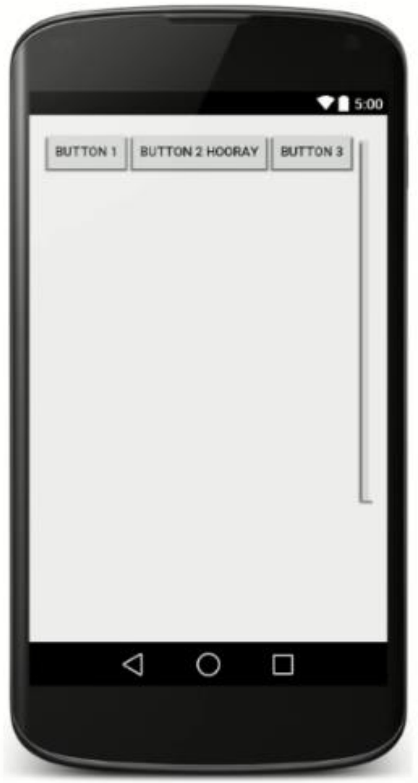
LinearLayout

- Lays out views / widgets in a single line.
- Orientation of horizontal or vertical.
- Items do not wrap if they reach edge of screen.



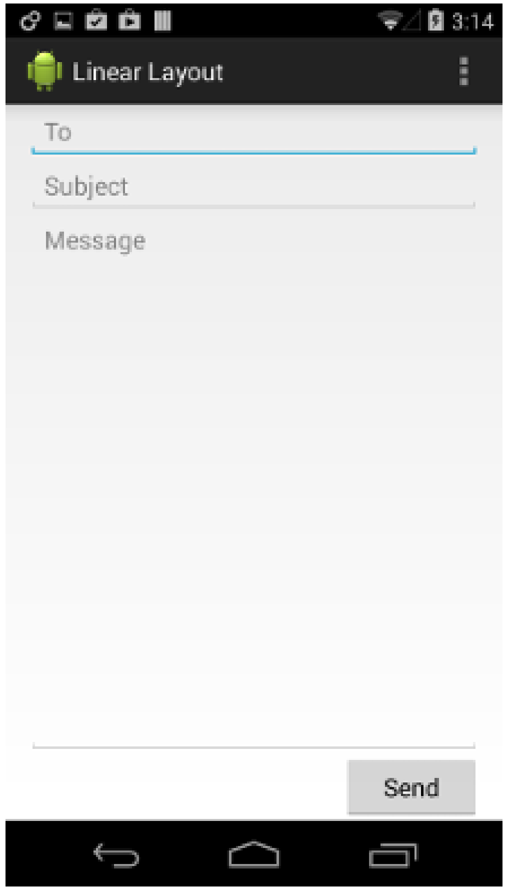
LinearLayout : Example

```
<LinearLayout ...
        android:orientation="horizontal"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4
                    Very Long Text" />
</LinearLayout>
```



LinearLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

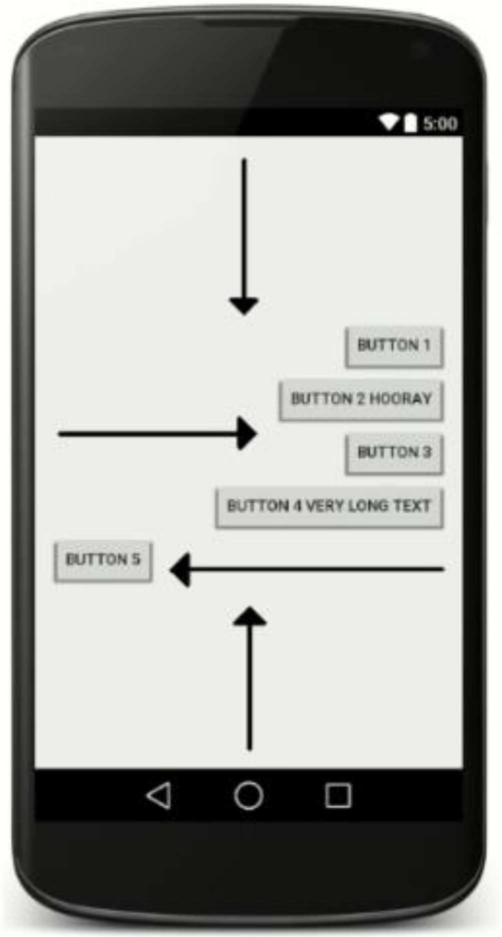


Gravity

Gravity : alignment direction that widgets are pulled.

- top , bottom , left , right , center
- Combine multiple with |
- Set gravity on the layout to adjust all widgets ;set **layout_gravity** on an individual widget.

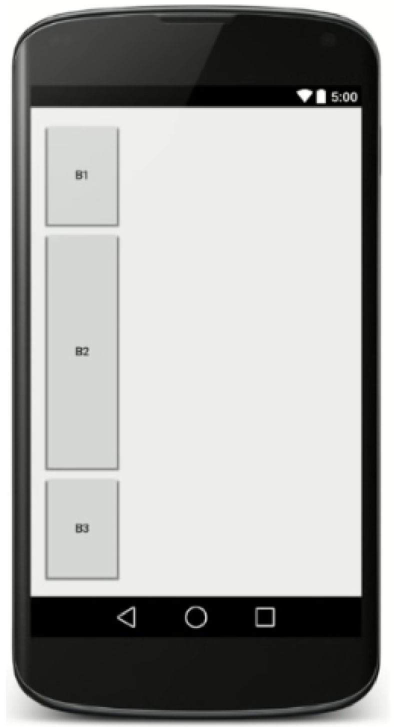
```
<LinearLayout ...
        android:orientation="vertical"
        android:gravity="center|right">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button 4 Long Text" />
    <Button ... android:text="Button 5"
            android:layout_gravity="left" />
</LinearLayout>
```



Weight

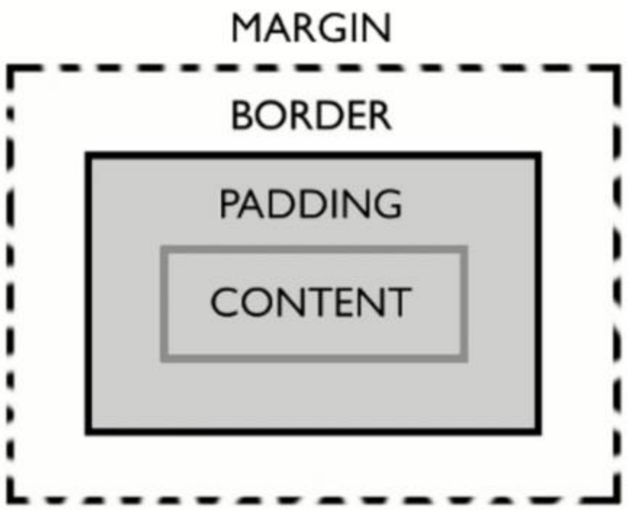
- Weight : relative element sizes by integers
  - Widget with weight k gets k/total fraction of total size.

```
<LinearLayout ...
        android:orientation="vertical">
    <Button ... android:text="B1"
                android:layout_weight="1" />
    <Button ... android:text="B2"
                android:layout_weight="3" />
    <Button ... android:text="B3"
                android:layout_weight="1" />
</LinearLayout>
```



Widget box model

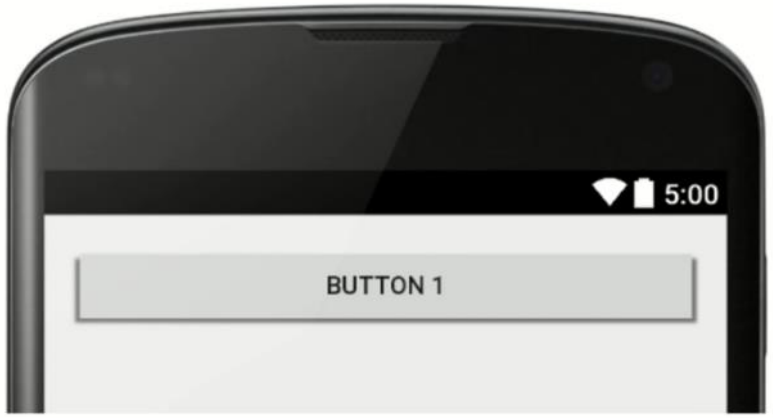
- **Content** : size of widget itself.
- **Padding** : artificial increase to widget size outside of content
- **Border** : outside padding , a line around edge of widget
- **Margin** : invisible separation from neighboring widget.



Sizing an individual widget

- Width and height of a widget can be :
  - *wrap_content* : *exactly large enough to fit the widget's content.*
  - *match_parent* : *as wide or tall as 100% of the screen or layout.*
  - *A specified fixed width such as 64dp (not usually recommended)*
    - **dp** : device pixel ; **dip** : device-independent pixel ; **sp** : scaling pixels.

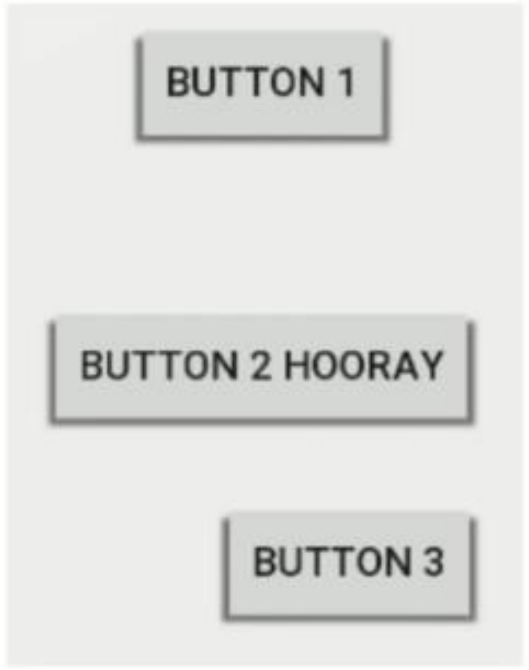
```
<Button ...
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```



Margin

- *Extra blank space outside widget.*
  - *set **layout_margin** to adjust all sides;*
  - *layout_marginTop,Bottom,Left,Right*

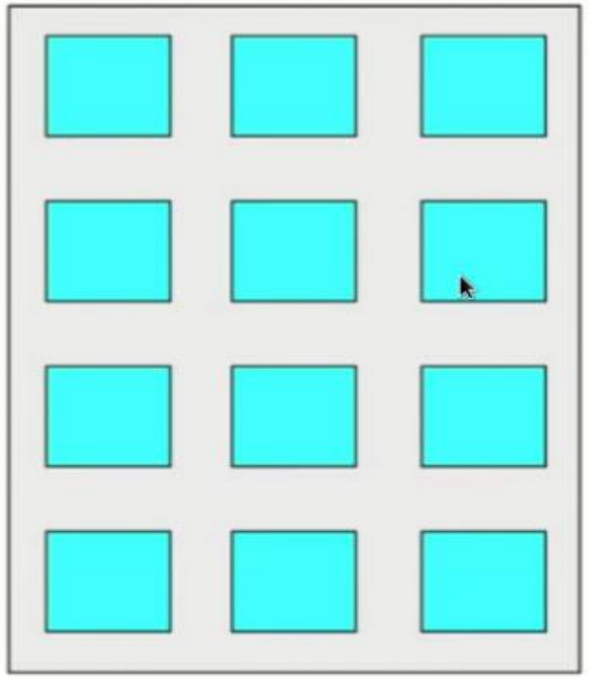
```
<LinearLayout ...
        android:orientation="vertical">
    <Button ... android:text="Button 1"
                android:layout_margin="50dp" />
    <Button ... android:text="Button 2 Hooray" />
    <Button ... android:text="Button 3"
                android:layout_marginLeft="30dp"
                android:layout_marginTop="40dp" />
</LinearLayout>
```



GridLayout

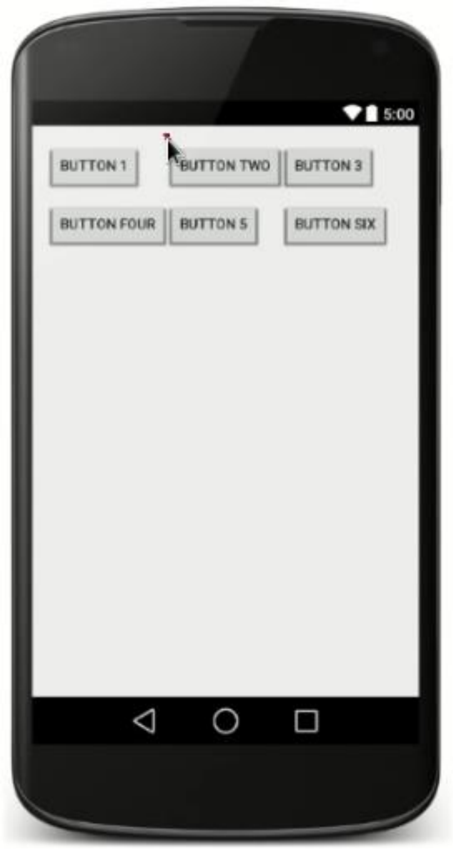
Lays out widgets / views in lines of rows and cols.

- Orientation attribute defines row/column order
- Introduces in Android 4; replaces older **TableLayout.**
- Each widget placed into “**next**” available row/columr
- unless given **layout_row** and **layout_column** attribute

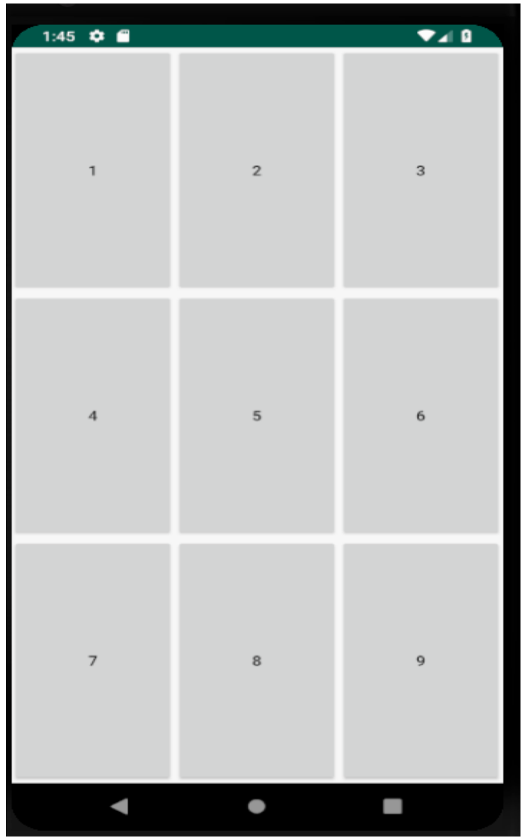

grid of 4 rows, 3 columns

GridLayout : Example

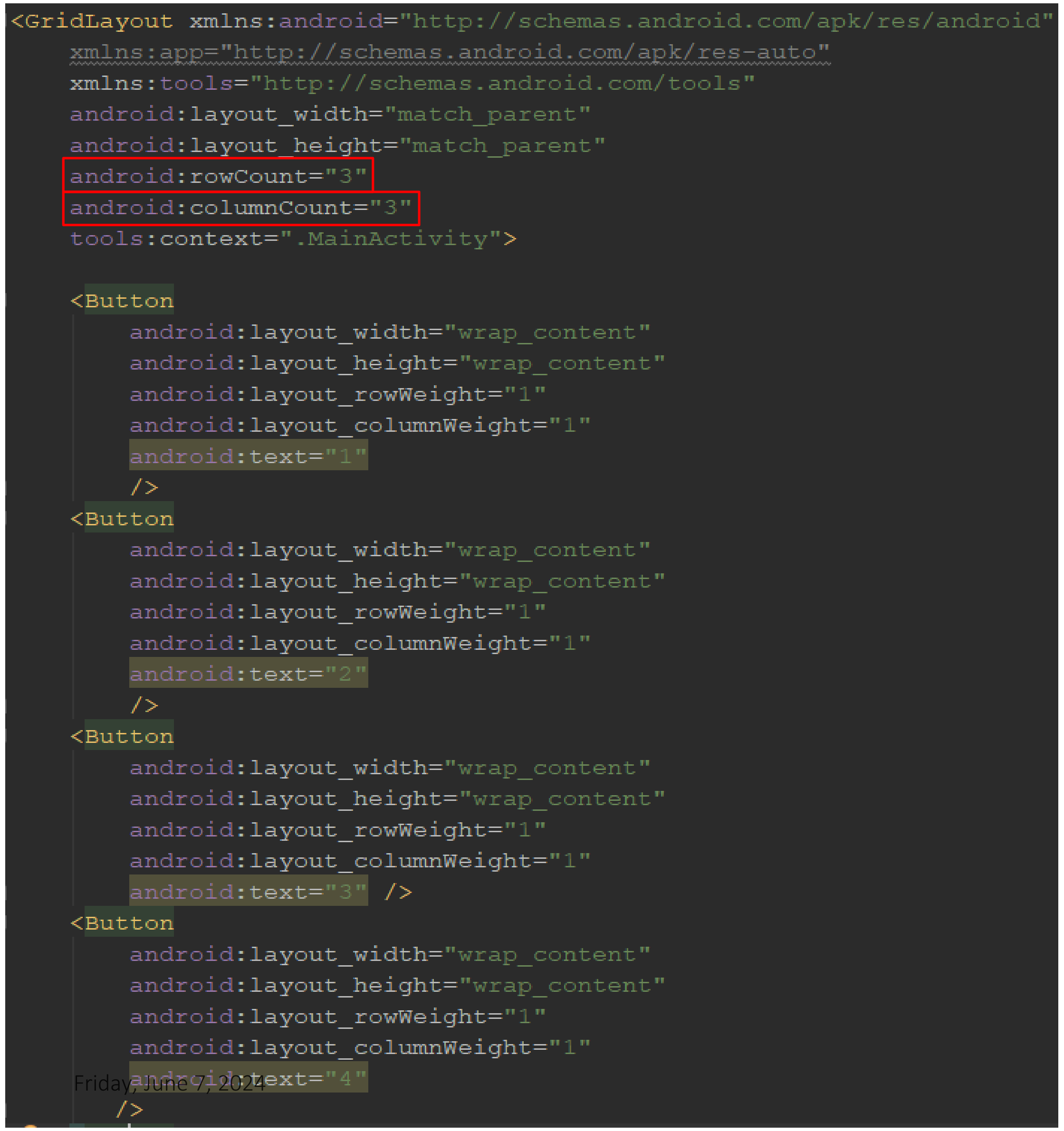
```
<GridLayout ...
        android:rowCount="2"
        android:columnCount="3"
        tools:context=".MainActivity">
    <Button ... android:text="Button 1" />
    <Button ... android:text="Button Two" />
    <Button ... android:text="Button 3" />
    <Button ... android:text="Button Four" />
    <Button ... android:text="Button 5" />
    <Button ... android:text="Button Six" />
</GridLayout>
```



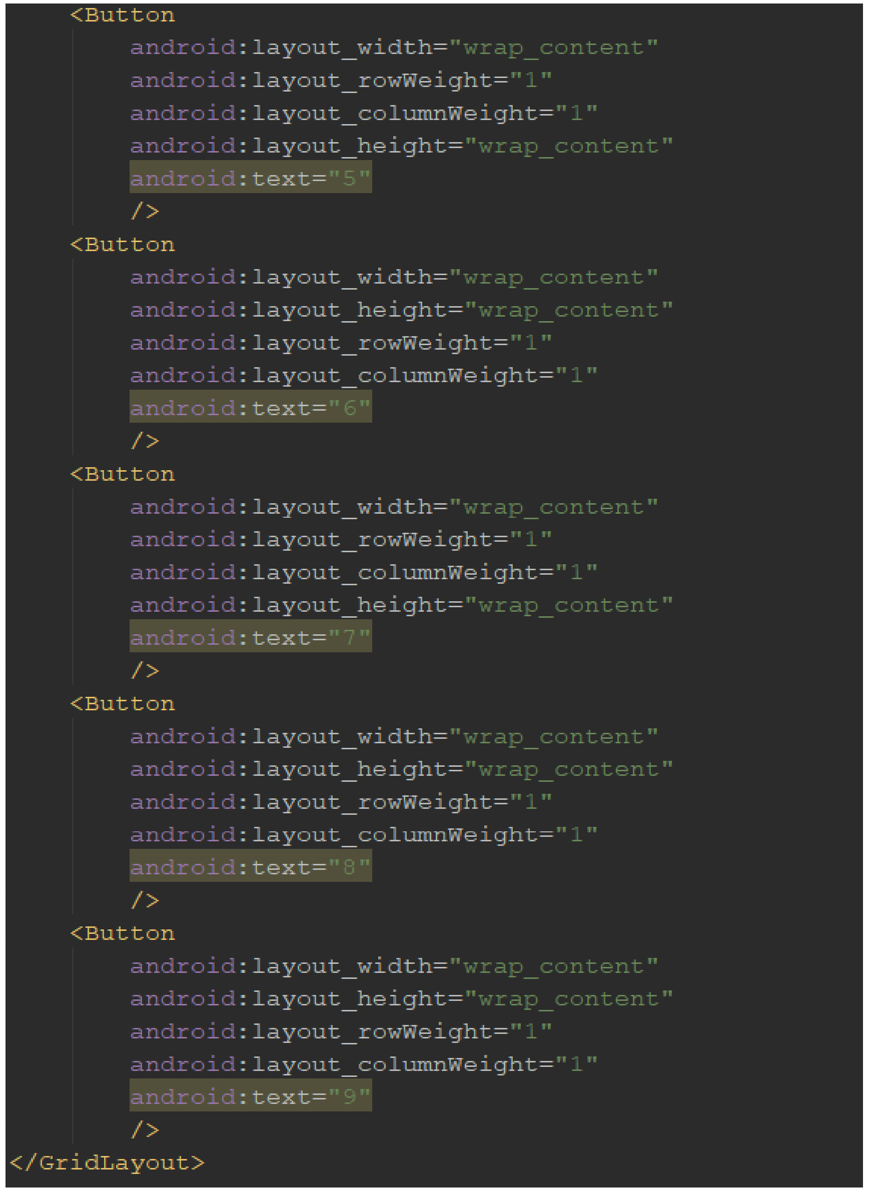
GridLayout : Exercise



GridLayout : Solution



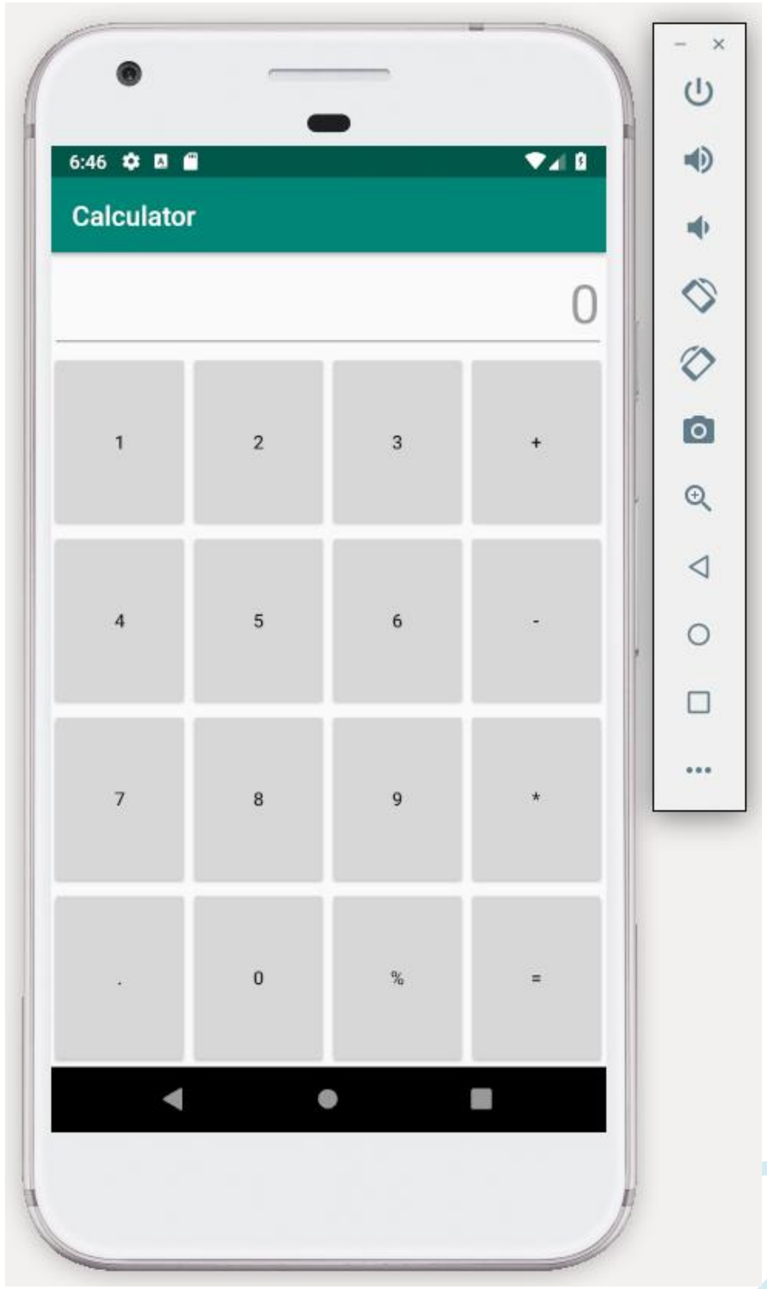
```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="3"
    android:columnCount="3"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="1"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="2"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="3"  />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="4"
       />
```



```
    <Button
        android:layout_width="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:layout_height="wrap_content"
        android:text="5"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="6"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:layout_height="wrap_content"
        android:text="7"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="8"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_rowWeight="1"
        android:layout_columnWeight="1"
        android:text="9"
        />
</GridLayout>
```

Calculator App

Calculator App interface is an example of Grid Layout.



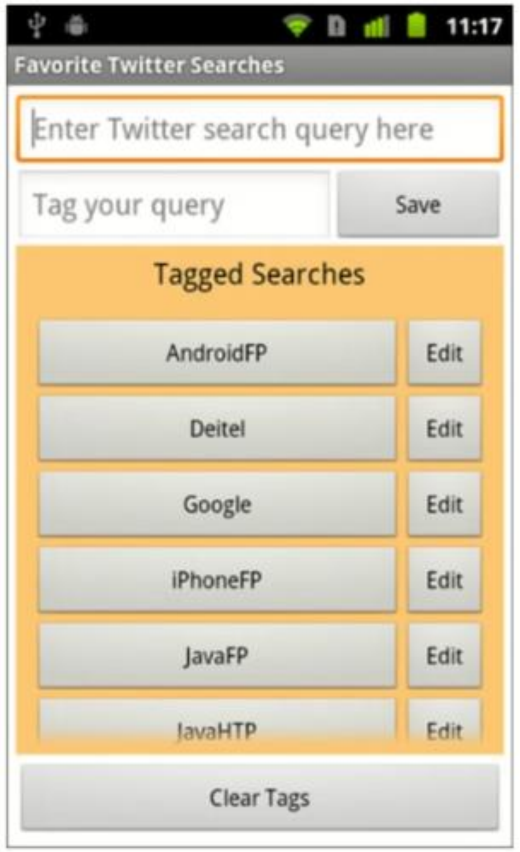
Friday, June 7, 2024

Nested Layout

Layout inside other layouts.

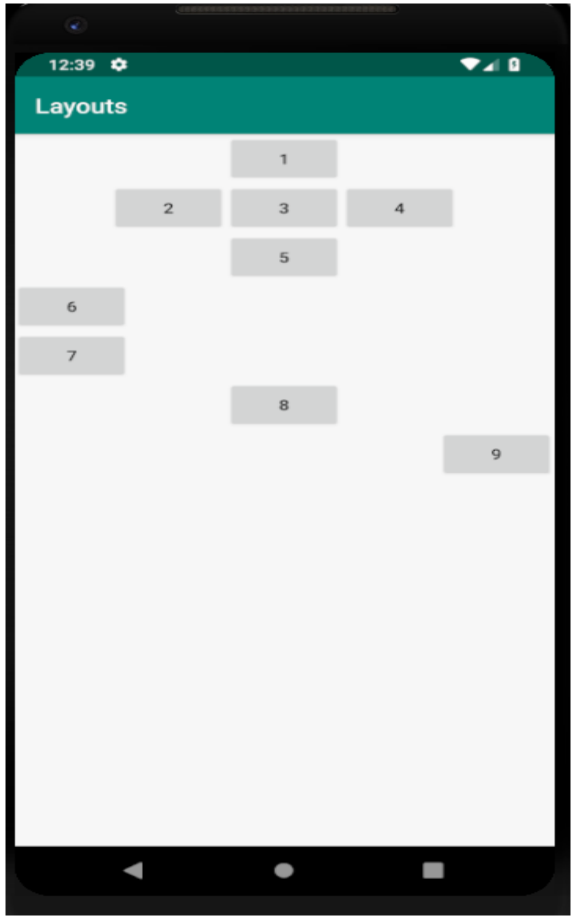
- Used to produce more complicated layouts.

```
<OuterLayout ...>
    <InnerLayout ...>
        <Widget ... />
        <Widget ... />
    </InnerLayout>

    <InnerLayout ...>
        <Widget ... />
        <Widget ... />
    </InnerLayout>
</OuterLayout>
```

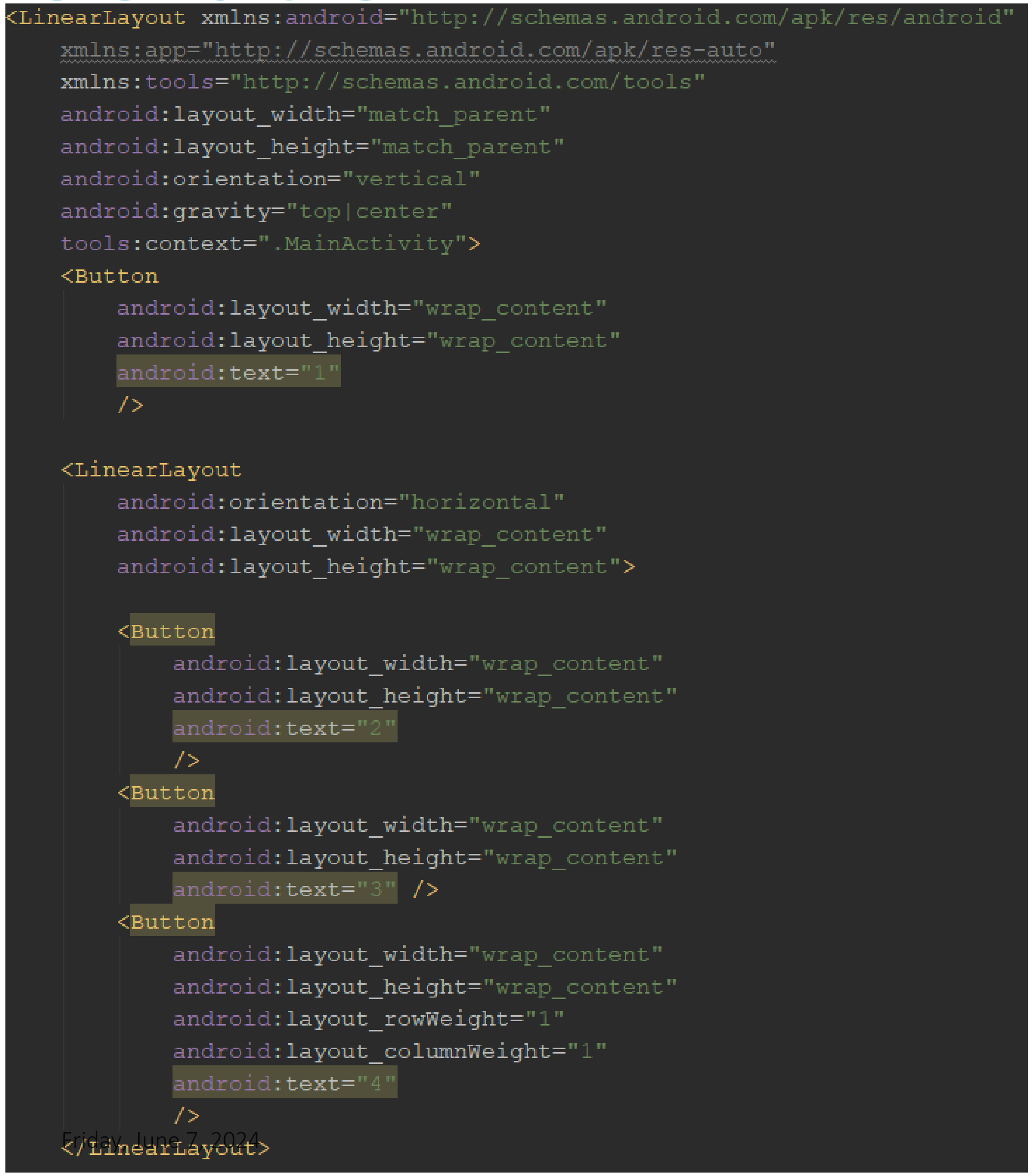


Nested Layout Exercise

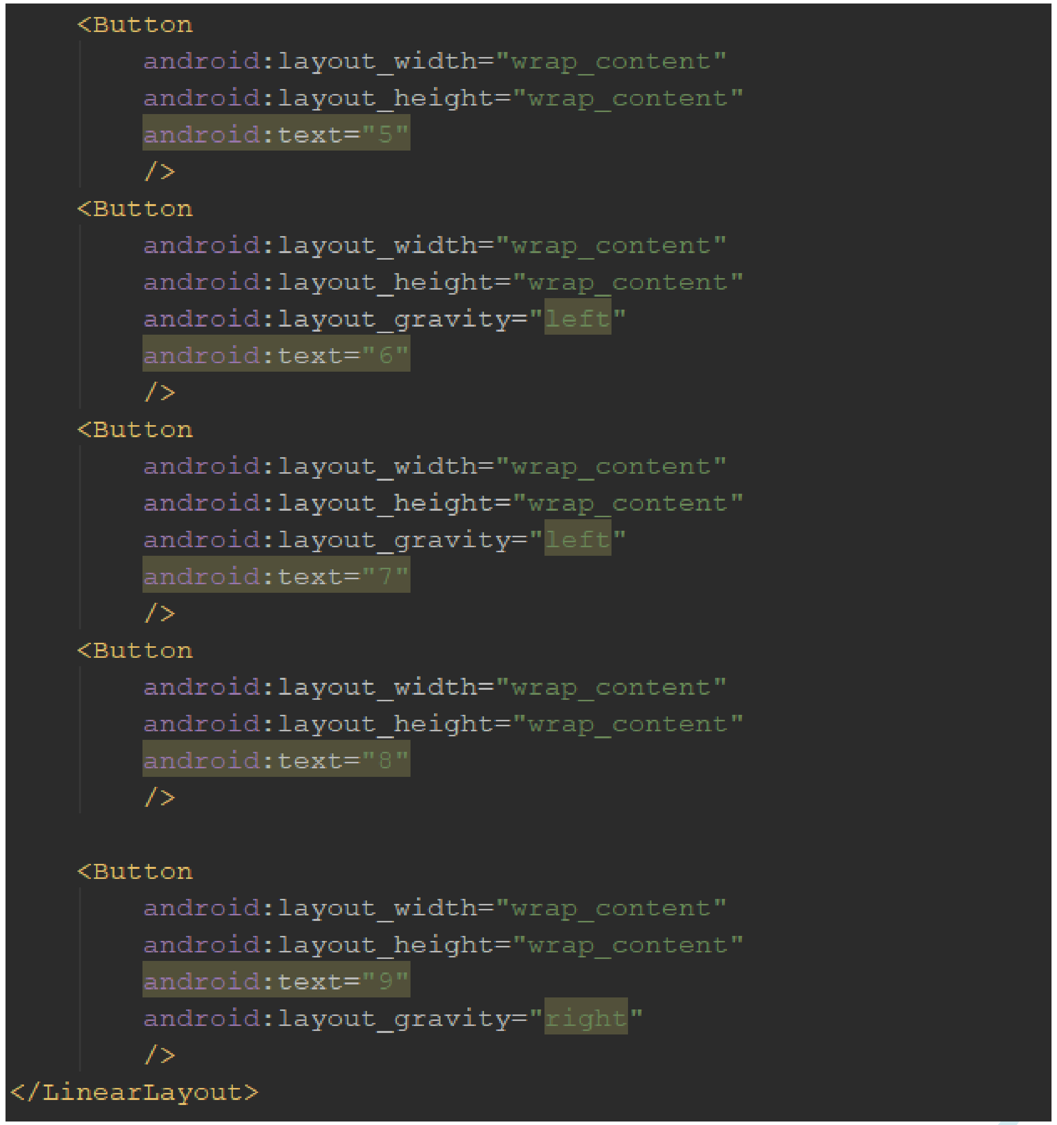
- Write the layout XML necessary to create the following app UI.
- How many overall layouts are needed ?
- Which widget go into which layout ?



Solution



```xml
<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"
    android:orientation="vertical"
    android:gravity="top|center"
    tools:context=".MainActivity">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        />

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="2"
            />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="3" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_rowWeight="1"
            android:layout_columnWeight="1"
            android:text="4"
            />
    </LinearLayout>
```



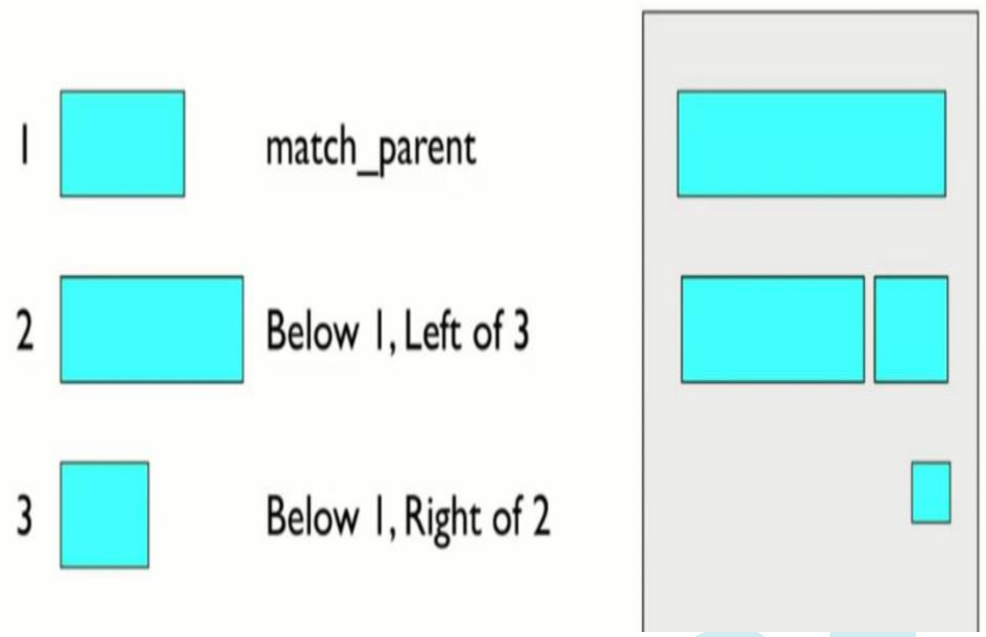
```xml
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="5"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:text="6"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:text="7"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="8"
        />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="9"
        android:layout_gravity="right"
        />
</LinearLayout>
```

RelativeLayout

Intended to reduce the need for nested layouts

•Each widget’s position and size are relative to each other views

- relative to “parent” (the activity itself).
- relative to other widgets/views.
- x-positions of reference : left , right , center.
- y-positions of reference top , bottom , center.



Friday, June 7, 2024

Relative anchor points

- **Properties of x/y relative to another widget:**
- Layout_below , above,toLeftOf,toRightOf,toEndOf
- *Set these to the ID of another widget in the format "@id/ID" (the given widget must have ID for this to work)*

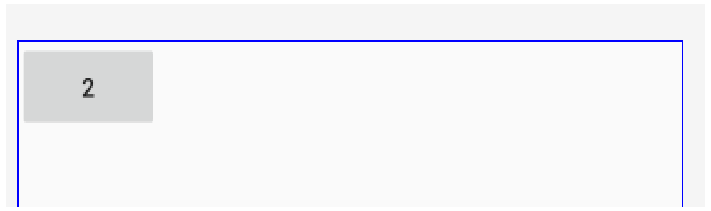
•Properties for x/y relative to layout container (the activity) :

- *Layout_alignParentTop,Bottom,Left,Right*
- Set these flags to a Boolean value of "true" to enable them.
- *Layout_centerHorizontal,Vertical,InParent*
- Set these flags to "true" to center the control within its parent in a dimension

Example - 1

```
<RelativeLayout 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=".RelativeActivity">
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="2"
        />
</RelativeLayout>
```

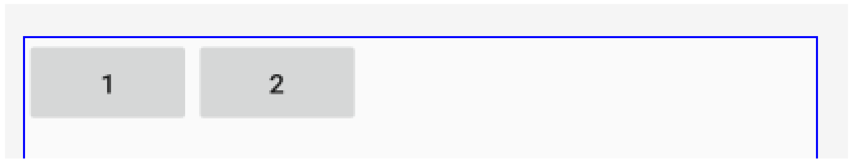
Button 1 has been overlapped by Button 2



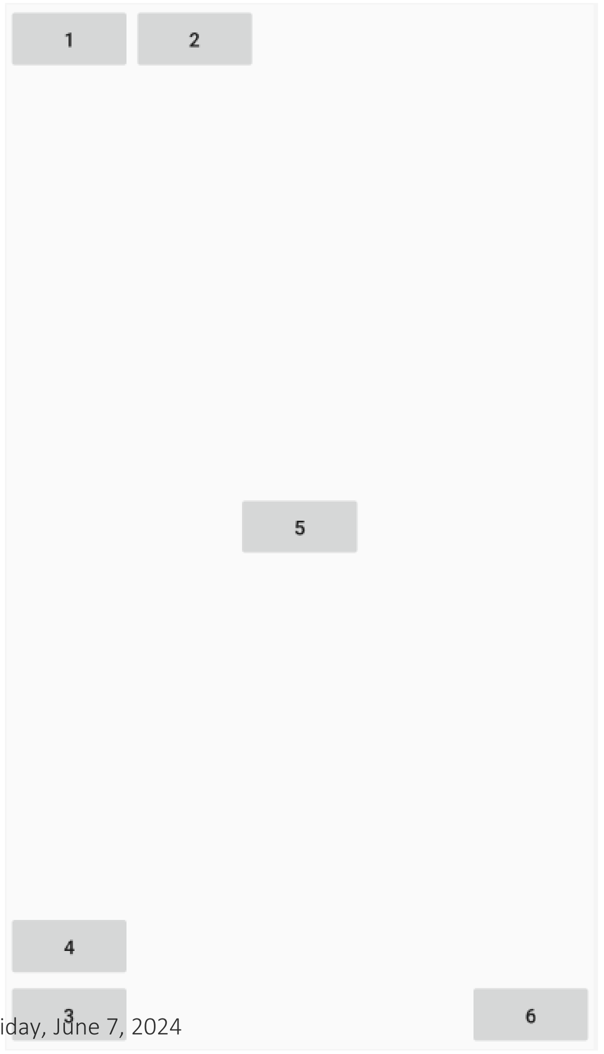
Friday, June 7, 2024

Example - 2

```
<RelativeLayout 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=".RelativeActivity">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"/>
    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/btn1"
        android:text="2"/>
</RelativeLayout>
```



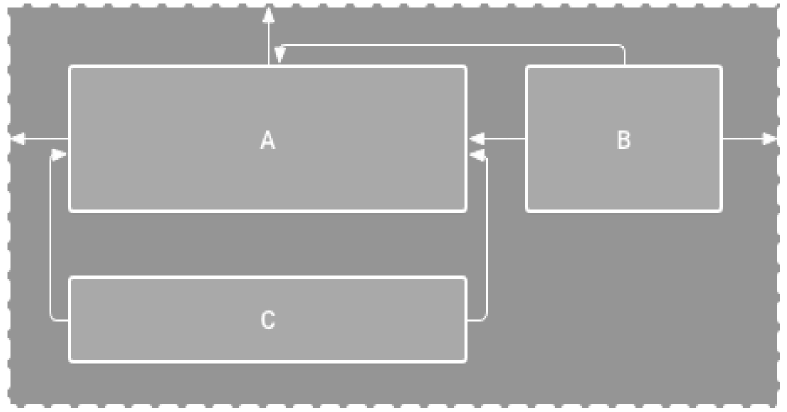
Example - 3

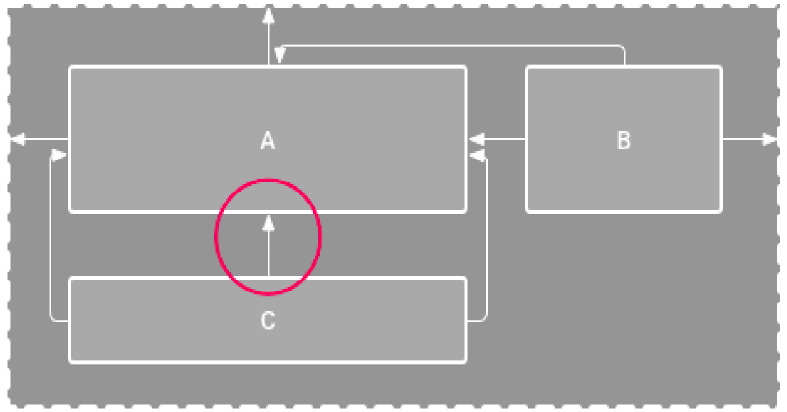


```
<Button
    android:id="@+id/btn3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:text="3"/>
<Button
    android:id="@+id/btn4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/btn3"
    android:text="4" />
<Button
    android:id="@+id/btn5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="5"/>
<Button
    android:id="@+id/btn6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:text="6"/>
```

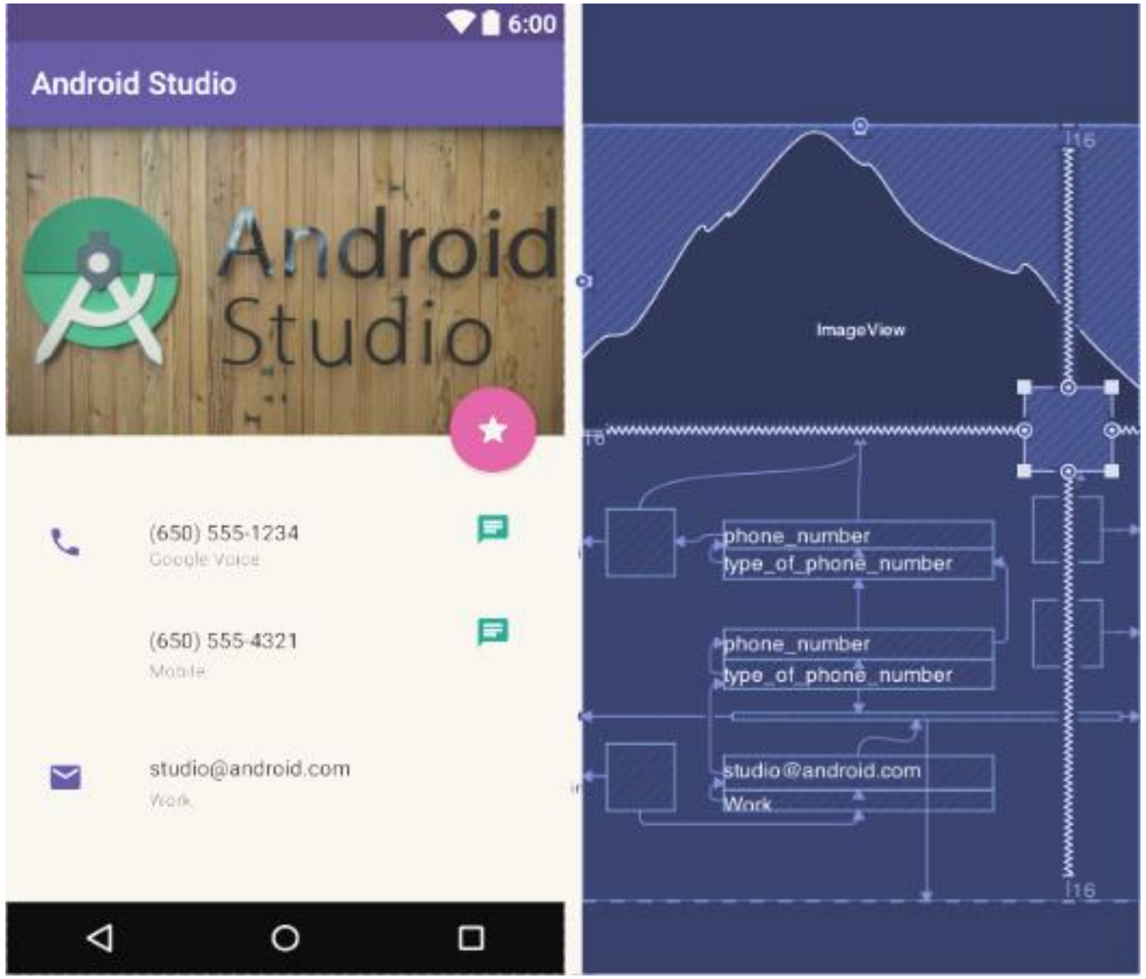
Constraint Layout

ConstraintLayout allows you to create large and complex layouts with a flat view hierarchy (no nested view groups). It's similar to RelativeLayout in that all views are laid out according to relationships between sibling views and the parent layout, but it's more flexible than RelativeLayout and easier to use with Android Studio's Layout Editor.

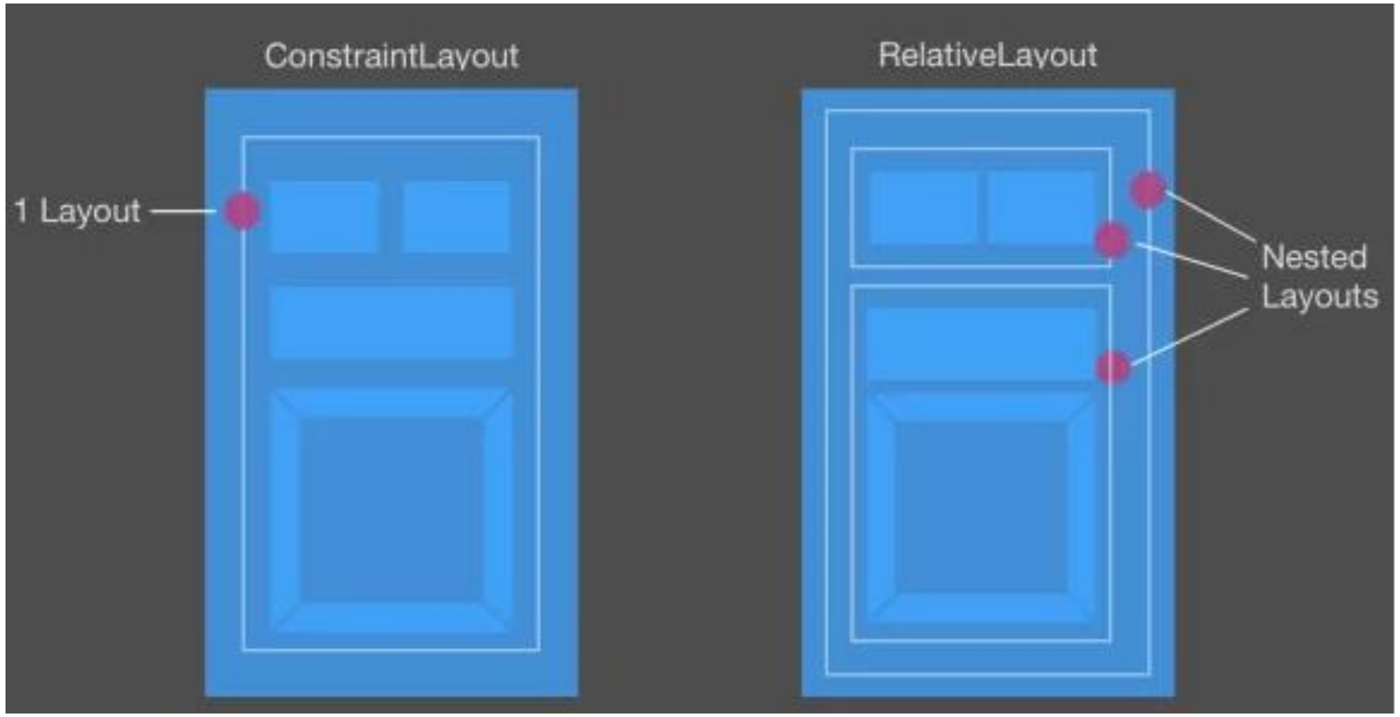




Constraint Layout



Constraint Layout vs Relative Layout



EditText

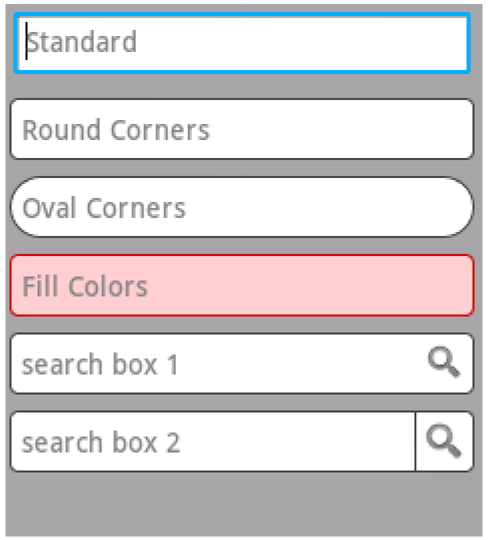
An editable text input box

android:id="@+id/*theID*"	unique ID for use in Java code
android:hint="*text*"	gray placeholder text shown before user types
android:inputType="*type*"	what kind of input is being typed; number, phone, date,time, ...
android:lines="*int*"	number of visible lines (rows) of input
android:maxLines="*int*"	max lines to allow user to type in the box
android:text="*text*"	initial text to put in box (default empty)
android:textSize="*size*"	size of font to use (e.g. "20dp")

key attributes in XML

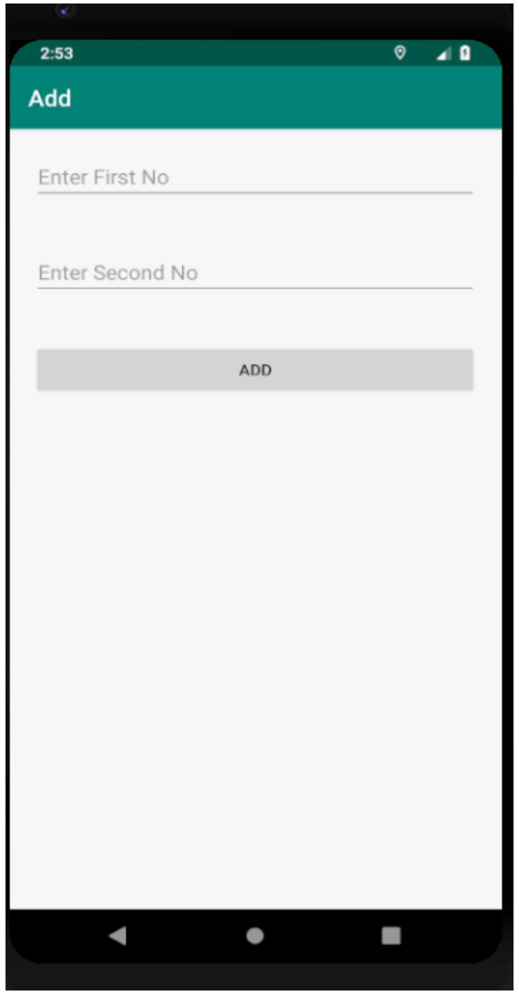
(other attributes: capitalize, digits, fontFamily, letterSpacing, lineSpacingExtra, minLines, numeric, password, phoneNumber, singleLine, textAllCaps, textColor, typeface)

```
// to get the user's text in Java code
EditText myEditText = (EditText) findViewById(R.id.theID);
String text = myEditText.getText().toString();
...
```

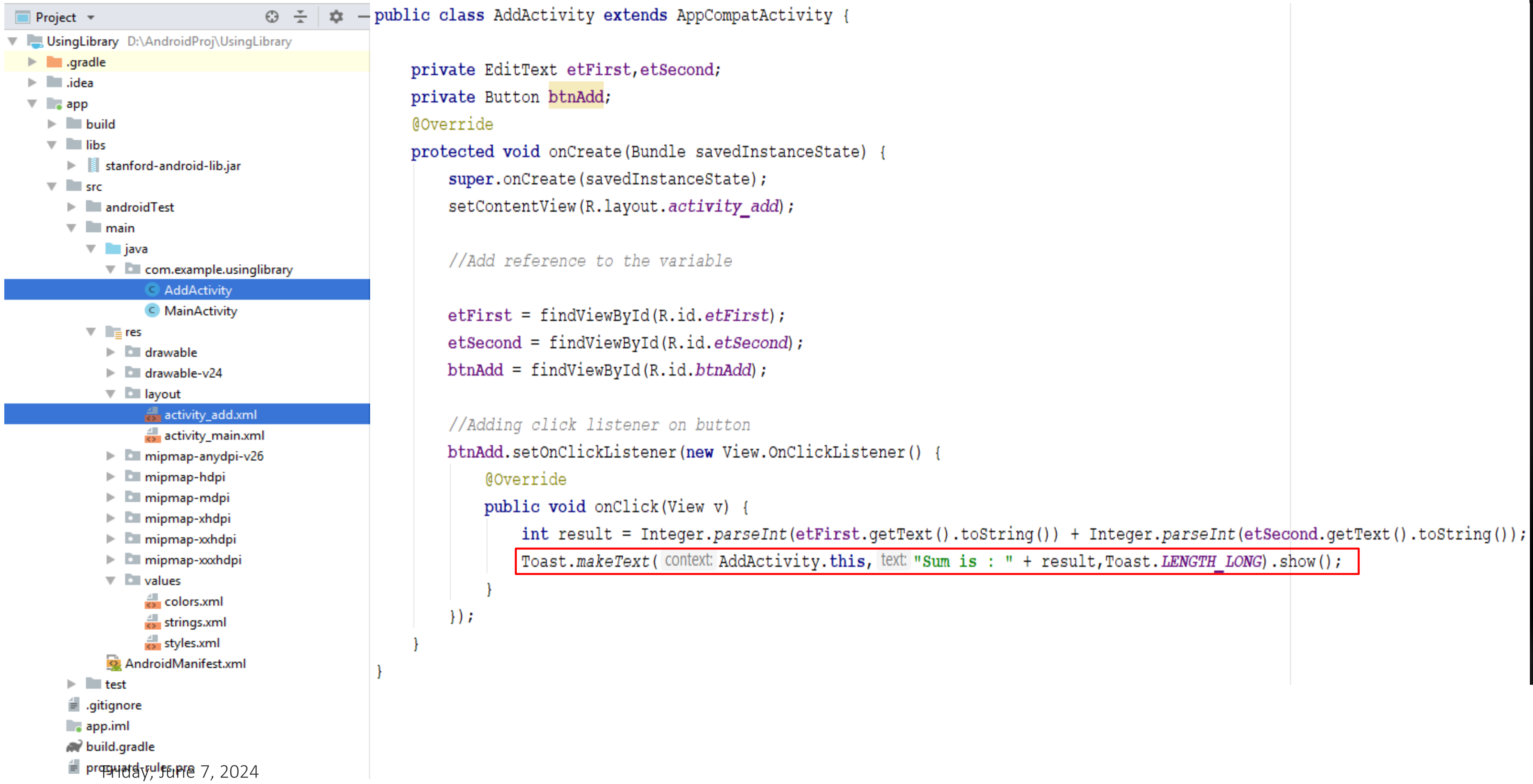


Hints

```
<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:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddActivity">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/etFirst"
        android:hint="Enter First No"
        android:layout_margin="20dp"
        />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/etSecond"
        android:hint="Enter Second No"
        android:layout_margin="20dp"
        />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btnAdd"
        android:text="ADD"
        android:layout_margin="20dp"
        />
</LinearLayout>
```

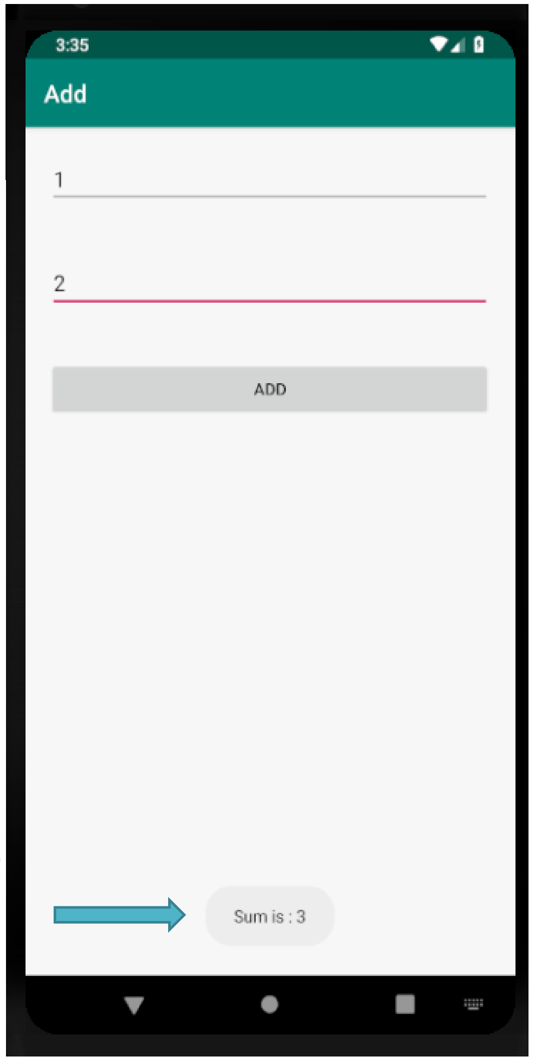


Java Code



```
Project
UsingLibrary  D:\AndroidProj\UsingLibrary
  .gradle
  .idea
  app
    build
    libs
      stanford-android-lib.jar
    src
      androidTest
      main
        java
          com.example.usinglibrary
            AddActivity
            MainActivity
        res
          drawable
          drawable-v24
          layout
            activity_add.xml
            activity_main.xml
          mipmap-anydpi-v26
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
            colors.xml
            strings.xml
            styles.xml
        AndroidManifest.xml
      test
    .gitignore
    app.iml
    build.gradle
    proguard-rules.pro
```

```java
public class AddActivity extends AppCompatActivity {

    private EditText etFirst,etSecond;
    private Button btnAdd;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);

        //Add reference to the variable

        etFirst = findViewById(R.id.etFirst);
        etSecond = findViewById(R.id.etSecond);
        btnAdd = findViewById(R.id.btnAdd);

        //Adding click listener on button
        btnAdd.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int result = Integer.parseInt(etFirst.getText().toString()) + Integer.parseInt(etSecond.getText().toString());
                Toast.makeText( context: AddActivity.this, text: "Sum is : " + result,Toast.LENGTH_LONG).show();
            }
        });
    }
}
```



Toast

- Android Toast can be used to display information for the short period of time.

A toast contains message to be displayed quickly and disappears after sometime.

- The android.widget.Toast class is the subclass of java.lang.Object class.
- You can also create custom toast as well for example toast displaying image.

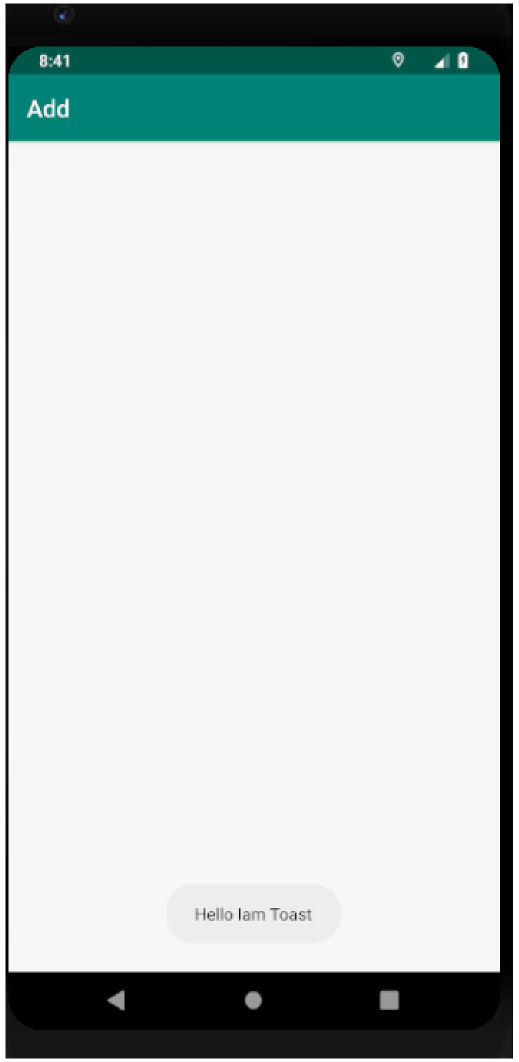
You can visit next page to see the code for custom toast.

```
Toast.makeText(context, text, duration).show();
```

```
Toast toast = Toast.makeText( context: ToastActivity.this   , text: "Hello Iam Toast", Toast.LENGTH_LONG);
toast.show();
```

OR

```
Toast.makeText( context: ToastActivity.this   , text: "Hello Iam Toast", Toast.LENGTH_LONG).show();
```



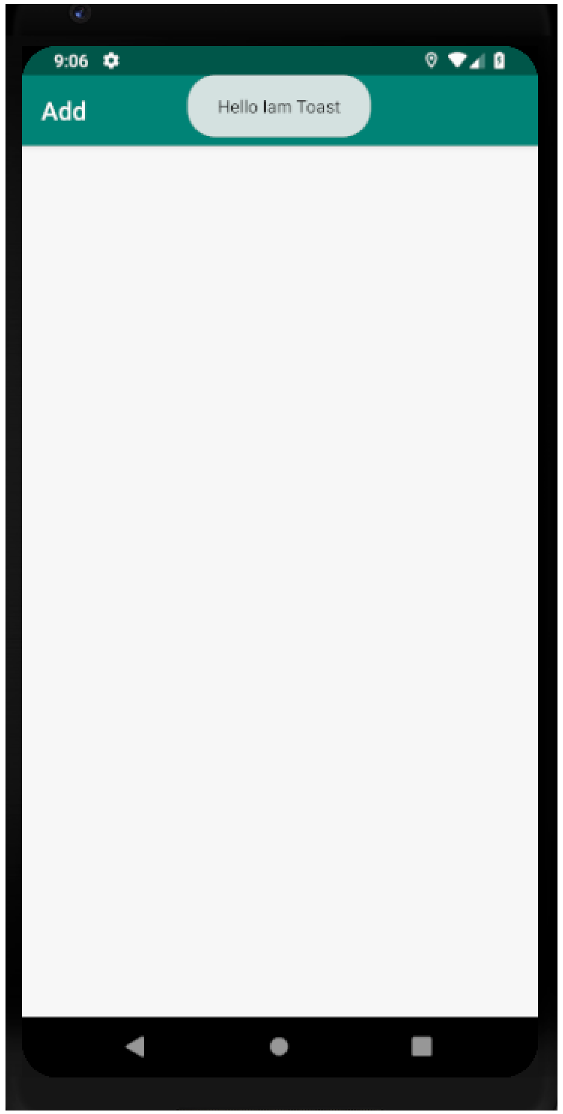
Positioning your Toast

A standard toast notification appears near the bottom of the screen, centered horizontally.

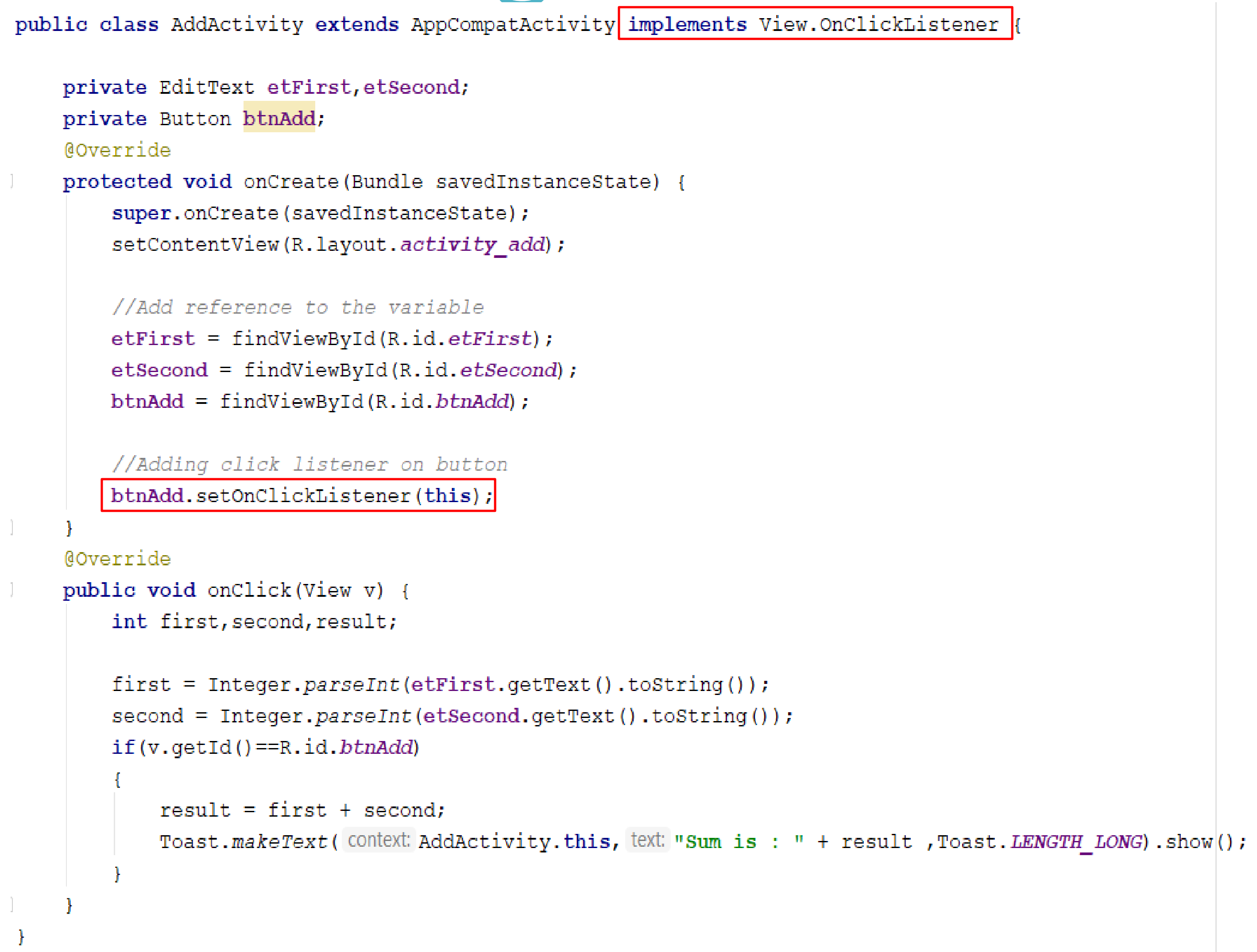
You can change this position with the `setGravity(int, int, int)` method. This accepts three parameters :

1. a Gravity constant.
2. an x-position offset
3. an y-position offset

```
Toast toast = Toast.makeText( context: ToastActivity.this  , text: "Hello Iam Toast", Toast.LENGTH_LONG);
toast.show();
toast.setGravity( gravity: Gravity.TOP|Gravity.CENTER,  xOffset: 0,  yOffset: 0);
```



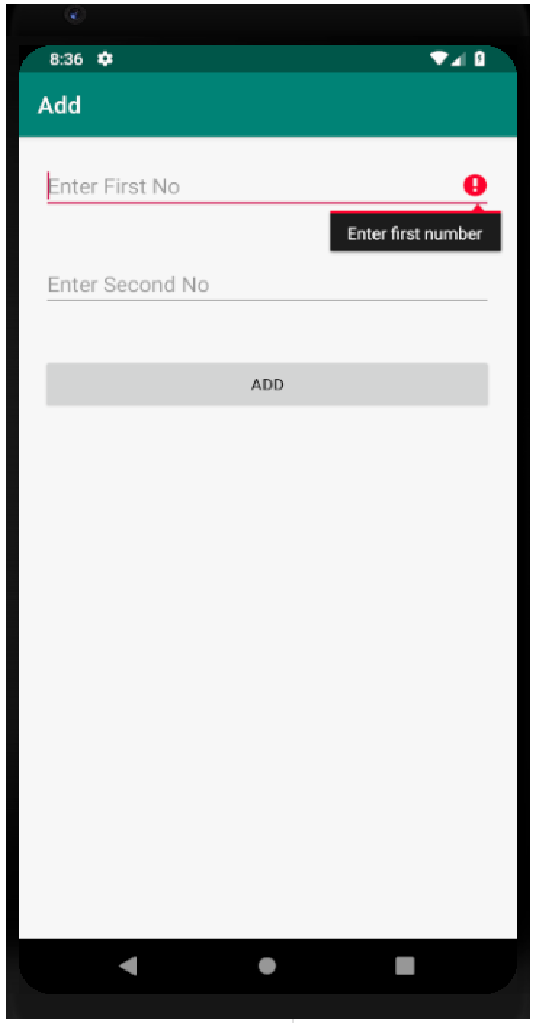
Click Event Using Interface



```
public class AddActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText etFirst,etSecond;
    private Button btnAdd;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add);

        //Add reference to the variable
        etFirst = findViewById(R.id.etFirst);
        etSecond = findViewById(R.id.etSecond);
        btnAdd = findViewById(R.id.btnAdd);

        //Adding click listener on button
        btnAdd.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        int first,second,result;

        first = Integer.parseInt(etFirst.getText().toString());
        second = Integer.parseInt(etSecond.getText().toString());
        if(v.getId()==R.id.btnAdd)
        {
            result = first + second;
            Toast.makeText( context: AddActivity.this, text: "Sum is : " + result ,Toast.LENGTH_LONG).show();
        }
    }
}
```

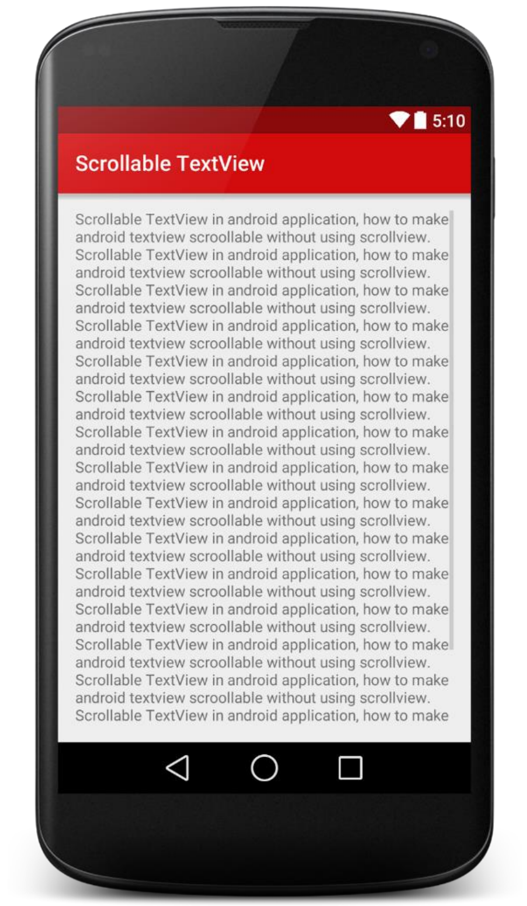
Validation in EditText

```
private boolean validate() {
    boolean flag = true;
    if (TextUtils.isEmpty(etFirst.getText().toString())) {
        etFirst.setError("Enter first number");
        etFirst.requestFocus();
        flag = false;
    } else if (TextUtils.isEmpty(etSecond.getText().toString())) {
        etSecond.setError("Enter second number");
        etSecond.requestFocus();
        flag = false;
    }
    return flag;
}
@Override
public void onClick(View v) {
    int first, second, result;
    if (validate()) {
        first = Integer.parseInt(etFirst.getText().toString());
        second = Integer.parseInt(etSecond.getText().toString());
        if (v.getId() == R.id.btnAdd) {
            result = first + second;
            Toast.makeText( context: AddActivity.this,  text: "Sum is : " + result, Toast.LENGTH_LONG).show();
        }
    }
}
```

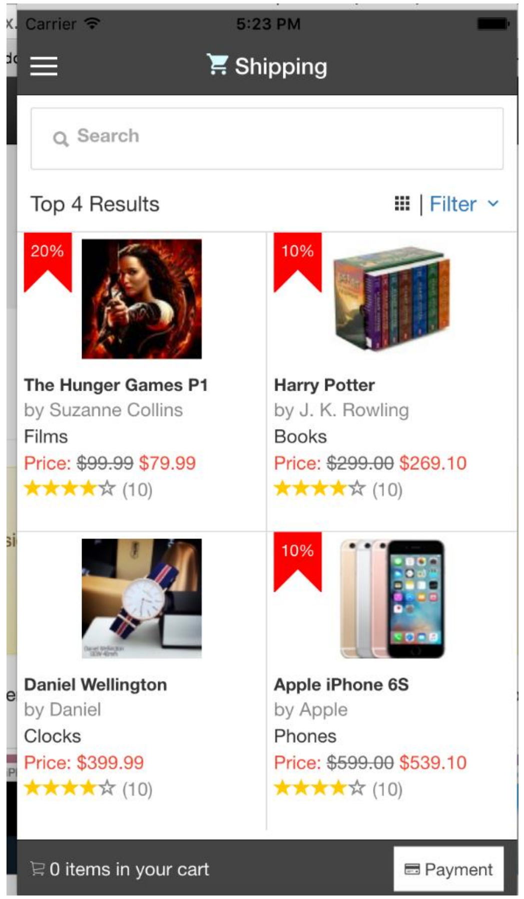


ScrollView

A container with scrollbars around anther widget or container



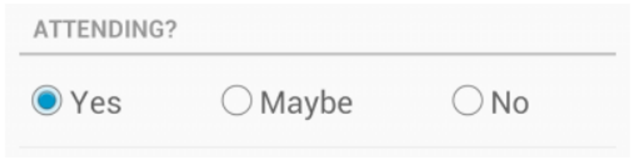
Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make android textview scrollable without using scrollview. Scrollable TextView in android application, how to make



```
<LinearLayout ...>
      ...
    <ScrollView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        <TextView ... android:id="@+id/turtle_info" />
    </ScrollView>
</LinearLayout>
```

RadioButton and RadioButton Group

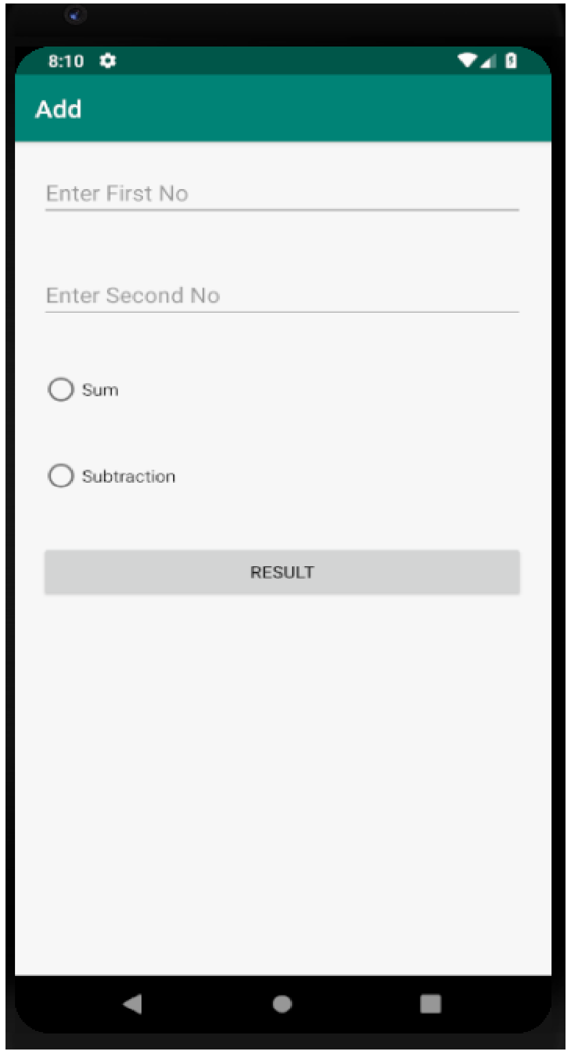
- Radio buttons allow the user to select one option from a set.
- You should use radio buttons for optional sets that are mutually exclusive if you think that the user needs to see all available options side-by-side. If it's not necessary to show all options side-by-side, use a spinner instead.
- To create each radio button option, create a Radio button in your layout.



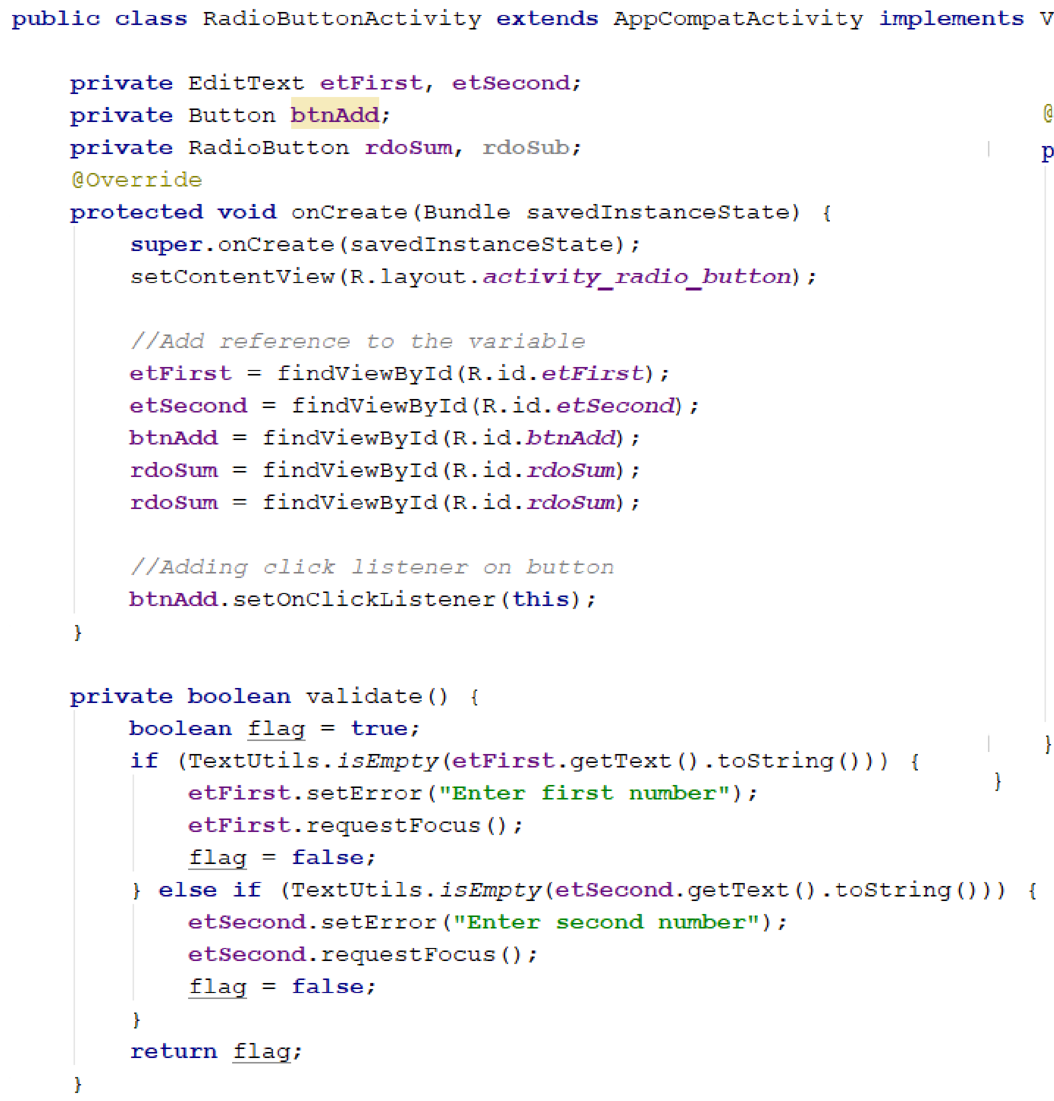
- However, because radio buttons are mutually exclusive, you must group them together inside a RadioGroup. By grouping them together, the system ensures that only one radio button can be selected at a time.

RadioButton : Example

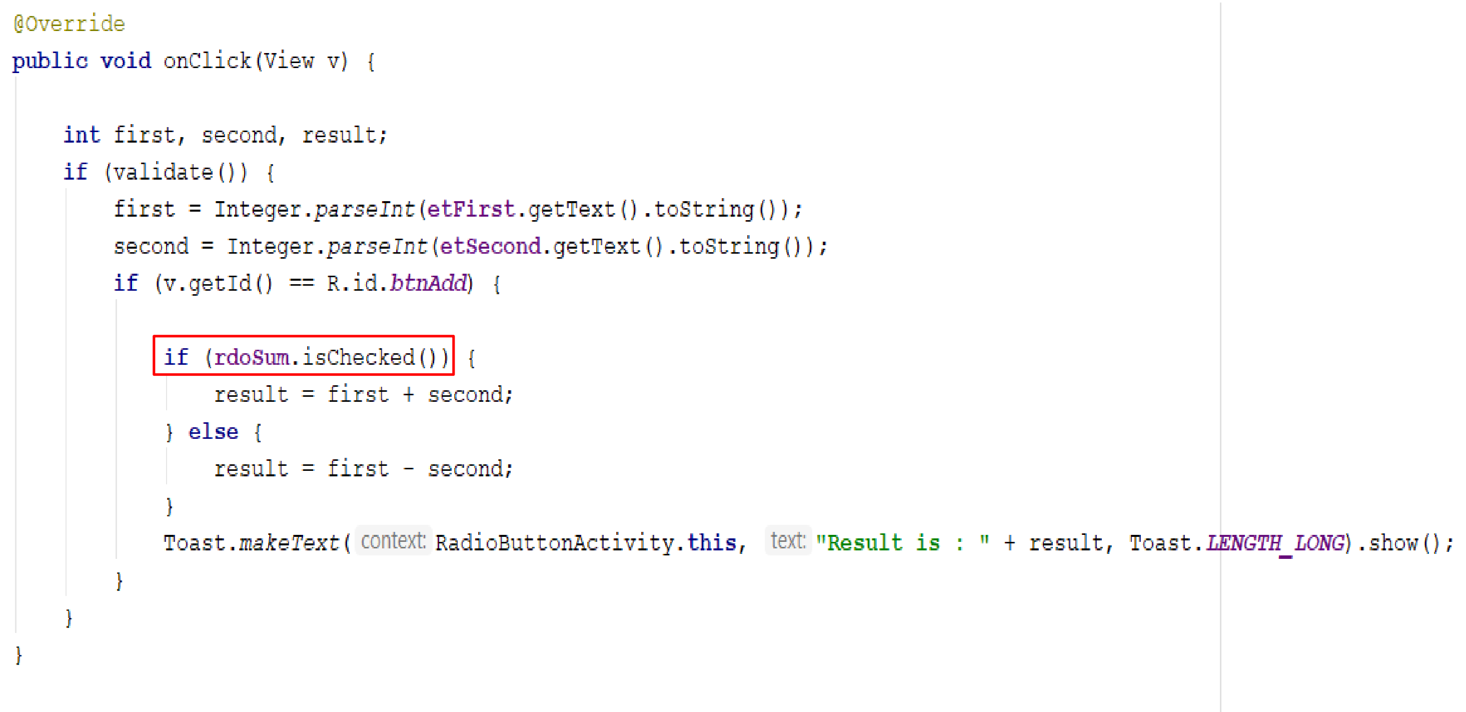
```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/etFirst"
    android:hint="Enter First No"
    android:layout_margin="20dp"
    />
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/etSecond"
    android:hint="Enter Second No"
    android:layout_margin="20dp"
    />
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <RadioButton
        android:id="@+id/rdoSum"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="Sum" />
    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/rdoSub"
        android:layout_margin="20dp"
        android:text="Subtraction"
        />
</RadioGroup>
<Button
    android:id="@+id/btnAdd"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:text="Result" />
```



RadioButton : Java code



```java
public class RadioButtonActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText etFirst, etSecond;
    private Button btnAdd;
    private RadioButton rdoSum, rdoSub;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_radio_button);

        //Add reference to the variable
        etFirst = findViewById(R.id.etFirst);
        etSecond = findViewById(R.id.etSecond);
        btnAdd = findViewById(R.id.btnAdd);
        rdoSum = findViewById(R.id.rdoSum);
        rdoSum = findViewById(R.id.rdoSum);

        //Adding click listener on button
        btnAdd.setOnClickListener(this);
    }

    private boolean validate() {
        boolean flag = true;
        if (TextUtils.isEmpty(etFirst.getText().toString())) {
            etFirst.setError("Enter first number");
            etFirst.requestFocus();
            flag = false;
        } else if (TextUtils.isEmpty(etSecond.getText().toString())) {
            etSecond.setError("Enter second number");
            etSecond.requestFocus();
            flag = false;
        }
        return flag;
    }
```

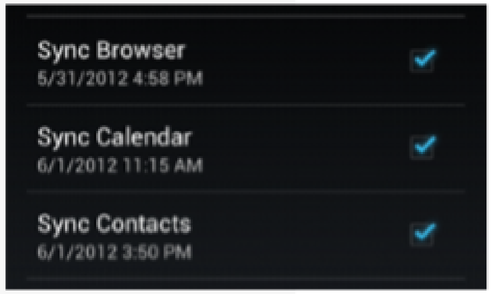


```java
    @Override
    public void onClick(View v) {

        int first, second, result;
        if (validate()) {
            first = Integer.parseInt(etFirst.getText().toString());
            second = Integer.parseInt(etSecond.getText().toString());
            if (v.getId() == R.id.btnAdd) {

                if (rdoSum.isChecked()) {
                    result = first + second;
                } else {
                    result = first - second;
                }
                Toast.makeText( context: RadioButtonActivity.this,  text: "Result is : " + result, Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

CheckBox

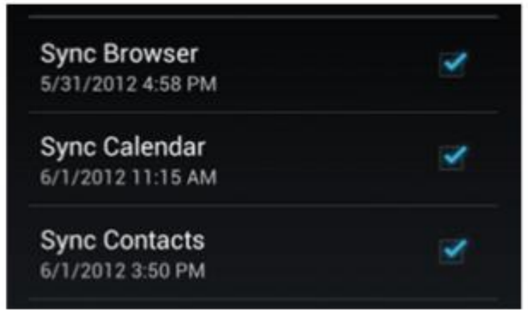
- In computing, a checkbox is a graphical user interface element that permits the user to make multiple selections from a number of options or to have the user answer yes or no on a simple yes/no question.

```xml
<?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">
    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
```


Sync Browser ✓
5/31/2012 4:58 PM
Sync Calendar ✓
6/1/2012 11:15 AM
Sync Contacts ✓
6/1/2012 3:50 PM

CheckBox

```
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

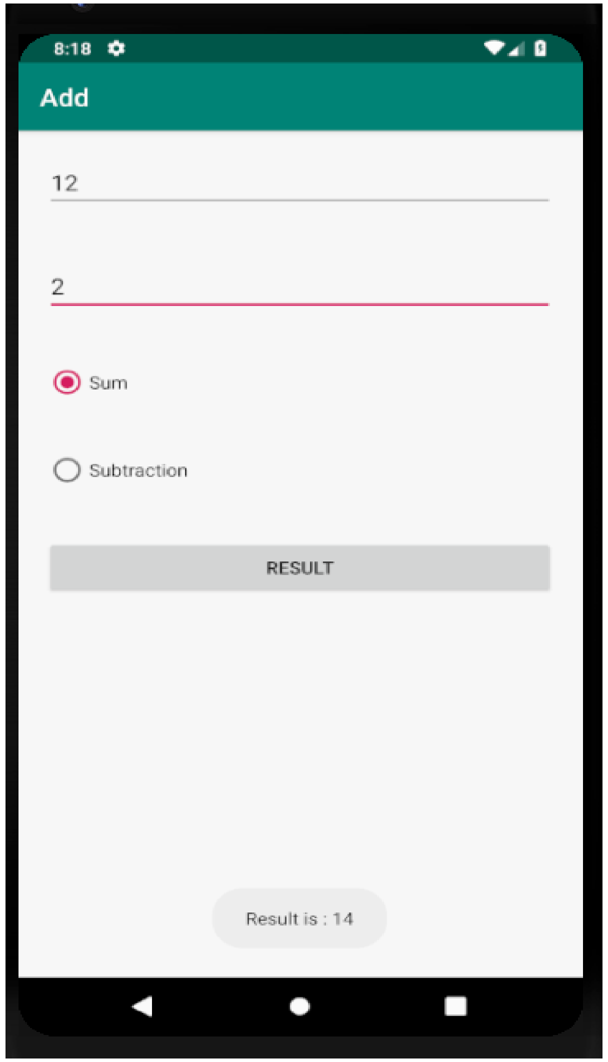


Sync Browser ✓
5/31/2012 4:58 PM

Sync Calendar ✓
6/1/2012 11:15 AM

Sync Contacts ✓
6/1/2012 3:50 PM

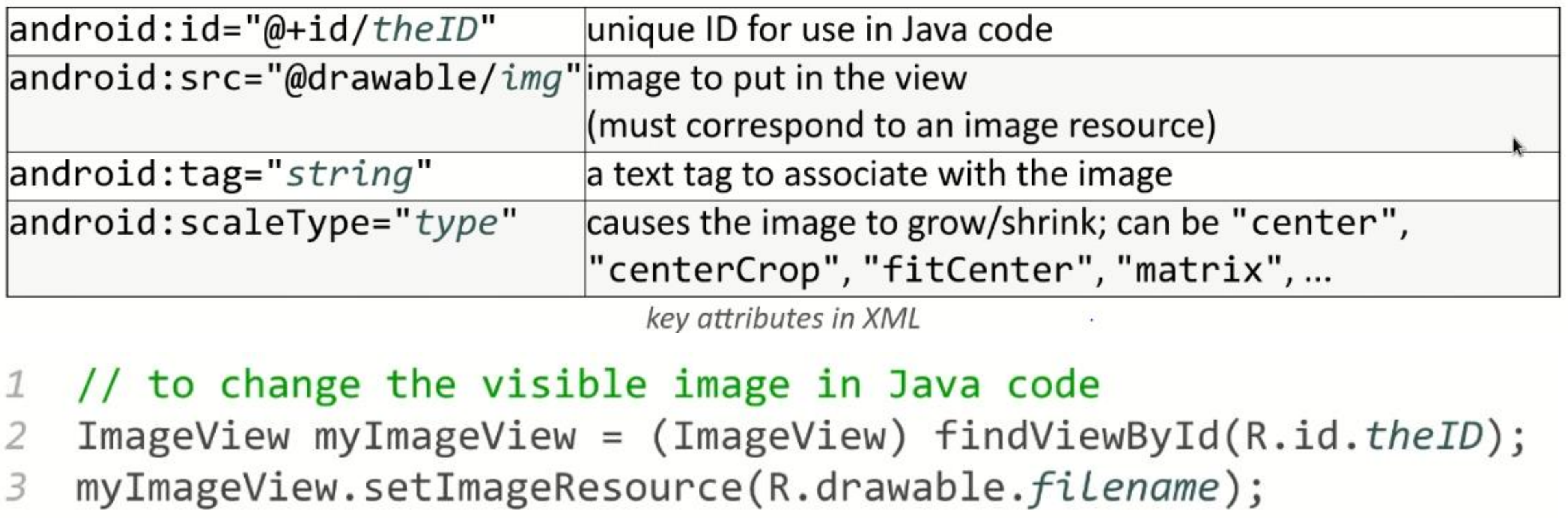
Output



Result is : 14

ImageView

ImageView is used to display an image file in application.

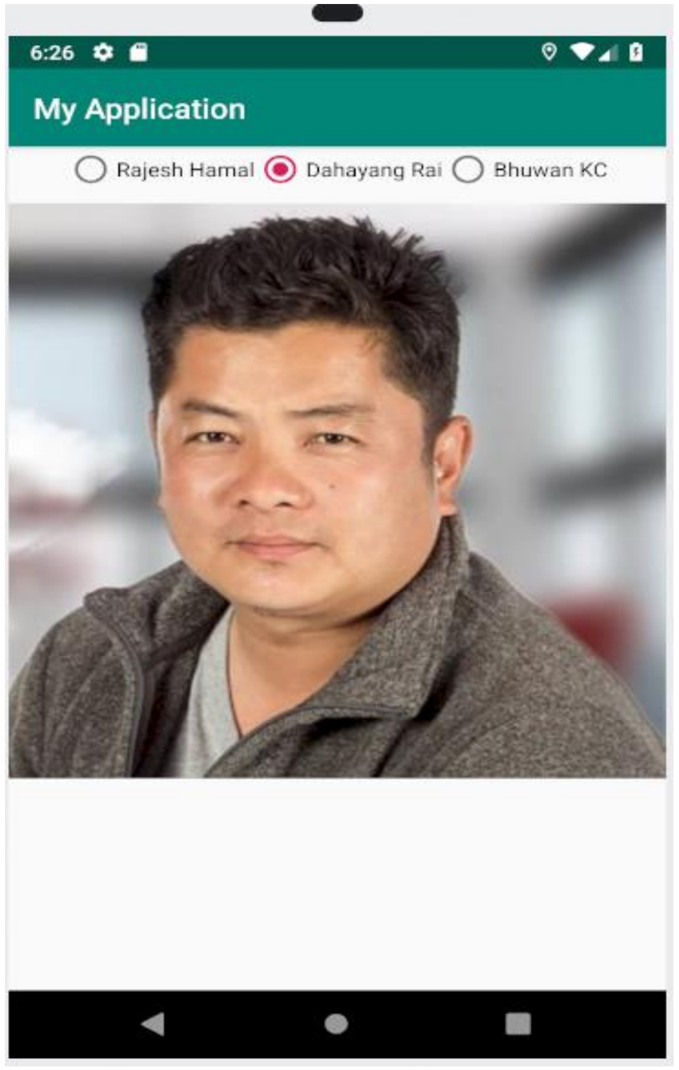


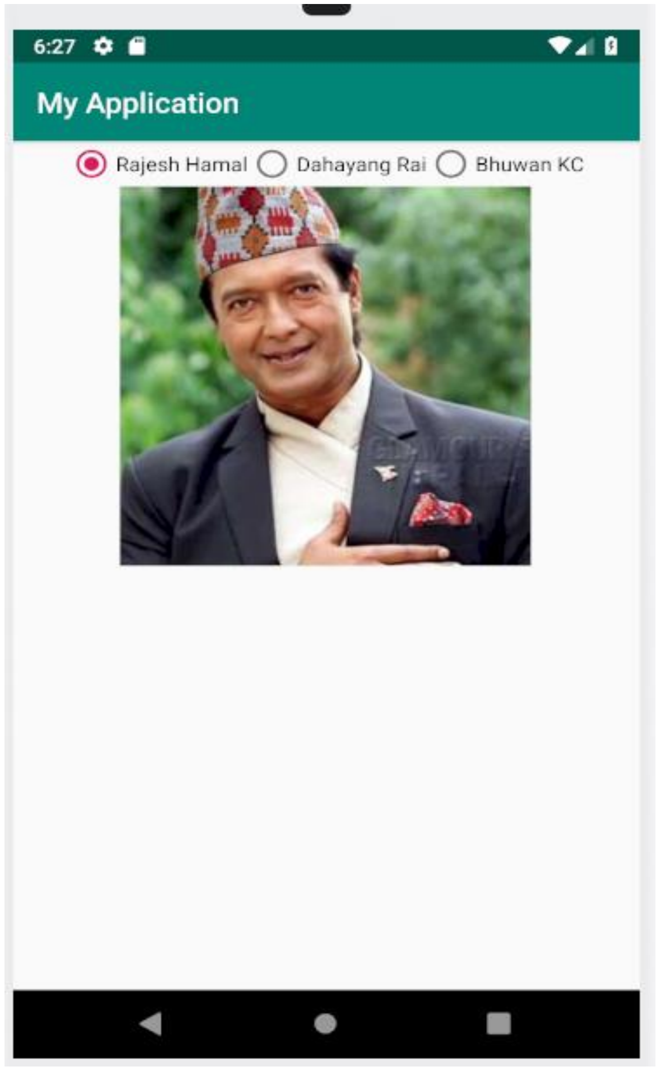
`android:id="@+id/theID"`	unique ID for use in Java code
`android:src="@drawable/img"`	image to put in the view (must correspond to an image resource)
`android:tag="string"`	a text tag to associate with the image
`android:scaleType="type"`	causes the image to grow/shrink; can be `"center"`, `"centerCrop"`, `"fitCenter"`, `"matrix"`, ...

key attributes in XML

```
// to change the visible image in Java code
ImageView myImageView = (ImageView) findViewById(R.id.theID);
myImageView.setImageResource(R.drawable.filename);
```

Example


6:26
My Application
○ Rajesh Hamal
● Dahayang Rai
○ Bhuwan KC


6:27
My Application
● Rajesh Hamal
○ Dahayang Rai
○ Bhuwan KC

Code

```xml
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center"
    android:id="@+id/rdogrp">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Rajesh Hamal"
        android:id="@+id/rdoRajesh"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dahayang Rai"
        android:id="@+id/rdoDahayang"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bhuwan KC"
        android:id="@+id/rdoBhuwan"/>
</RadioGroup>

<ImageView
    android:layout_below="@id/rdogrp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/imgHero"
    />
```

```java
public class ImgViewActivity extends AppCompatActivity implements View.OnClickListener {

    RadioButton rdoRajesh,rdoDahayang,rdoBhuwan;
    ImageView imgHero;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_img_view);

        rdoRajesh = findViewById(R.id.rdoRajesh);
        rdoBhuwan =findViewById(R.id.rdoBhuwan);
        rdoDahayang =findViewById(R.id.rdoDahayang);

        imgHero = findViewById(R.id.imgHero);

        rdoDahayang.setOnClickListener(this);
        rdoBhuwan.setOnClickListener(this);
        rdoRajesh.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch(v.getId())
        {
            case R.id.rdoBhuwan :
                imgHero.setImageResource(R.drawable.bhuwan);
                break;

            case R.id.rdoDahayang :
                imgHero.setImageResource(R.drawable.dayahang);
                break;

            case R.id.rdoRajesh :
                imgHero.setImageResource(R.drawable.rajesh);
                break;
        }
    }
}
```

Resources

- In the project directory structure :
  - *res/type/name.extension*
  - *Example : res/drawable/Pikachu.png*
- Referring to a resource , in the XML :
  - @type/name
  - Example : @drawable/pikachu
- Referring to a resource ID , in the java code
  - R.type.name
  - Example : R.Drawable.pikachu

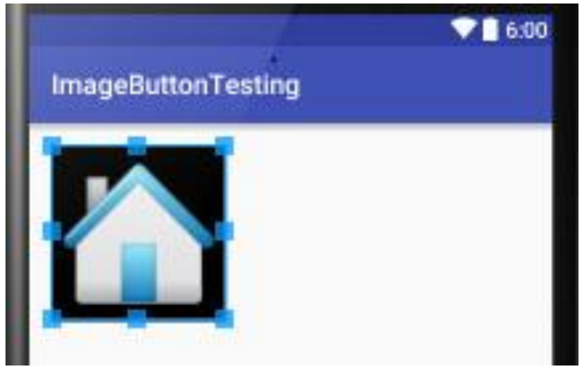
ImageButton

A clickable widget with an image label

`android:clickable="bool"`	set to `false` to disable the button
`android:id="@+id/theID"`	unique ID for use in Java code
`android:onClick="function"`	function to call in activity when clicked (must be `public`, `void`, and take a `View` arg)
`android:src="@drawable/img"`	image to put in the button (must correspond to an image resource)

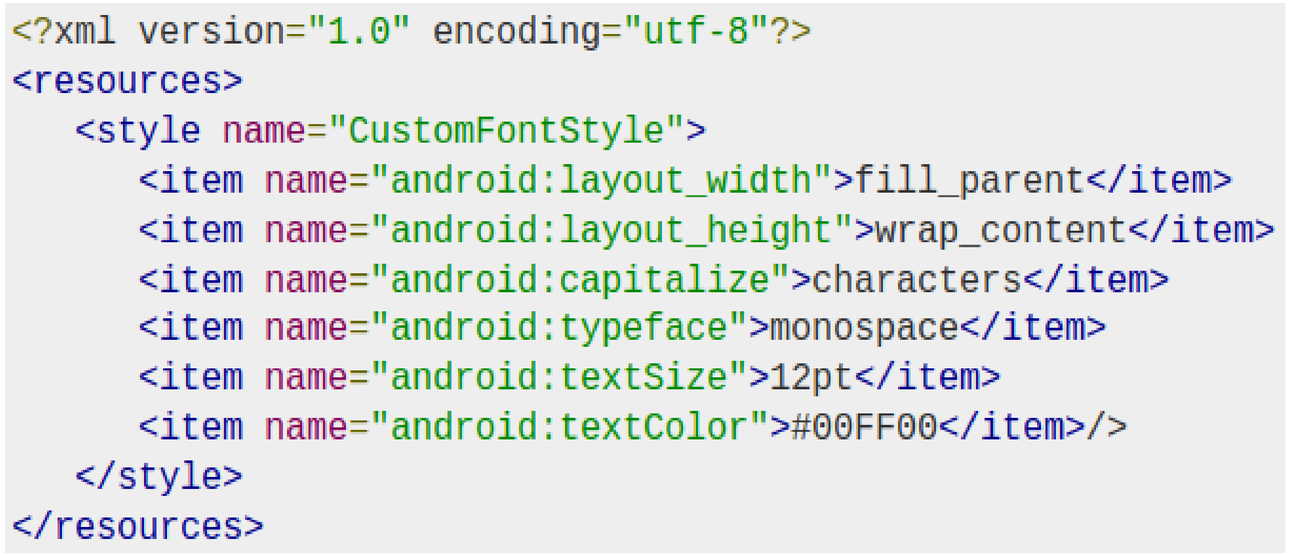
key attributes in XML

```
<ImageButton
    android:id="@+id/simpleImageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/home"
    android:background="#000"/><!-- black background color for image button-->
```



Defining Style

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.

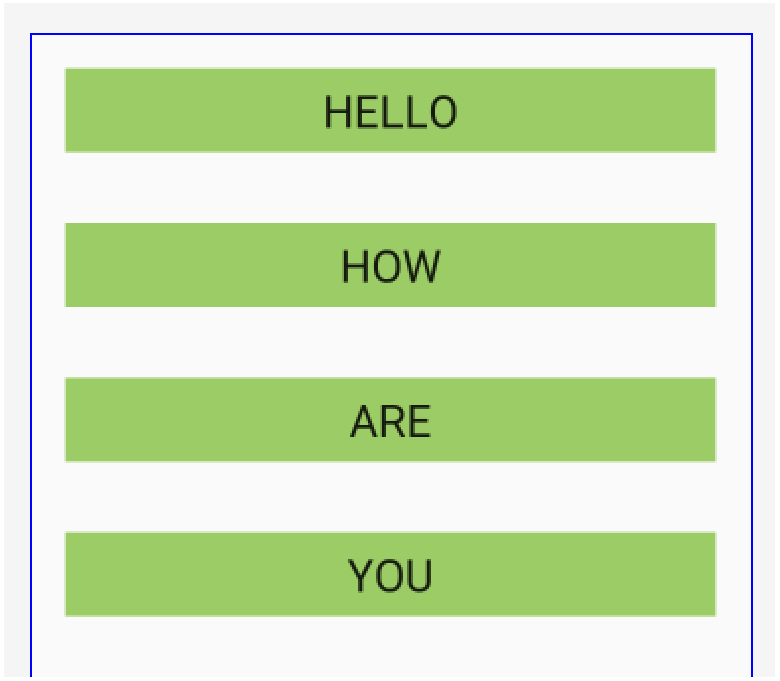


```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomFontStyle">
      <item name="android:layout_width">fill_parent</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:capitalize">characters</item>
      <item name="android:typeface">monospace</item>
      <item name="android:textSize">12pt</item>
      <item name="android:textColor">#00FF00</item>/>
   </style>
</resources>
```

Using Style

```xml
<?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:id="@+id/text_id"
      style="@style/CustomFontStyle"
      android:text="@string/hello_world" />

</LinearLayout>
```

Example



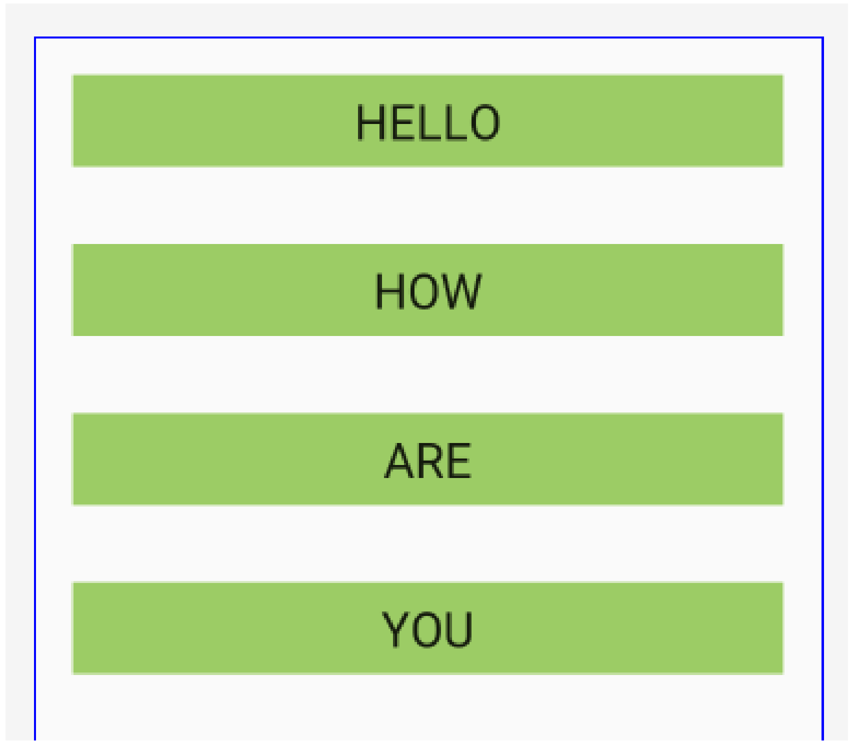
```
<Button
    android:text="Hello"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:textSize="25sp"
    android:fontFamily="sans-serif"
    android:background="#9CCC65"
    />
<Button
    android:text="How"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:textSize="25sp"
    android:fontFamily="sans-serif"
    android:background="#9CCC65"
    />
<Button
    android:text="Are"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:textSize="25sp"
    android:fontFamily="sans-serif"
    android:background="#9CCC65"
    />
<Button
    android:text="You"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="20dp"
    android:textSize="25sp"
    android:fontFamily="sans-serif"
    android:background="#9CCC65"
    />
```

Style.xml

```xml
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="ButtonStyle">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_margin">20sp</item>
        <item name="android:textSize">25sp</item>
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:background">#9CCC65</item>
    </style>

</resources>
```

Activity.xml

```
<?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"
    android:orientation="vertical"
    tools:context=".StyleActivity">

    <Button
        android:text="Hello"
        style="@style/ButtonStyle"
        />
    <Button
        android:text="How"
        style="@style/ButtonStyle"
        />
    <Button
        android:text="Are"
        style="@style/ButtonStyle"
        />
    <Button
        android:text="You"
        style="@style/ButtonStyle"
        />

</LinearLayout>
```


HELLO
HOW
ARE
YOU