Mobile Application UI Design

Hua Huang

Calendar Events

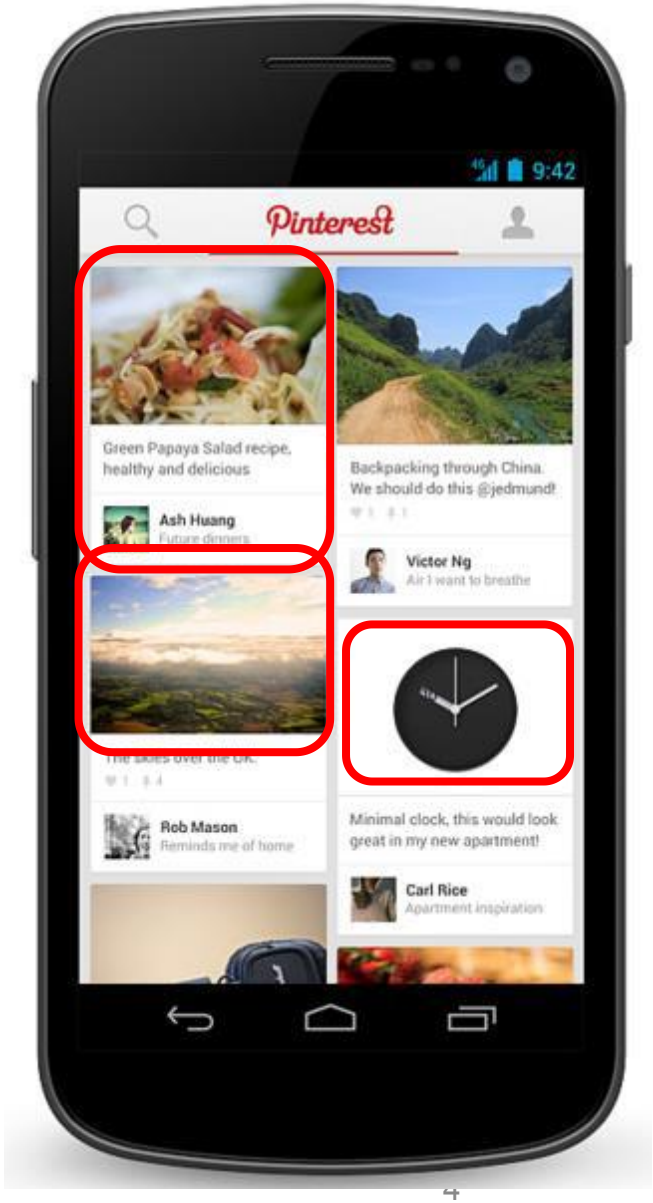
- Sept 24, Wednesday: class canceled due to travel
- Sept 26, first exam, covering the first eight lectures
  - Topics include Backgrounds, Android, UI, Activity, Service, Intents

Editing in Android Studio

Views

Android UI design involves arranging views on a screen

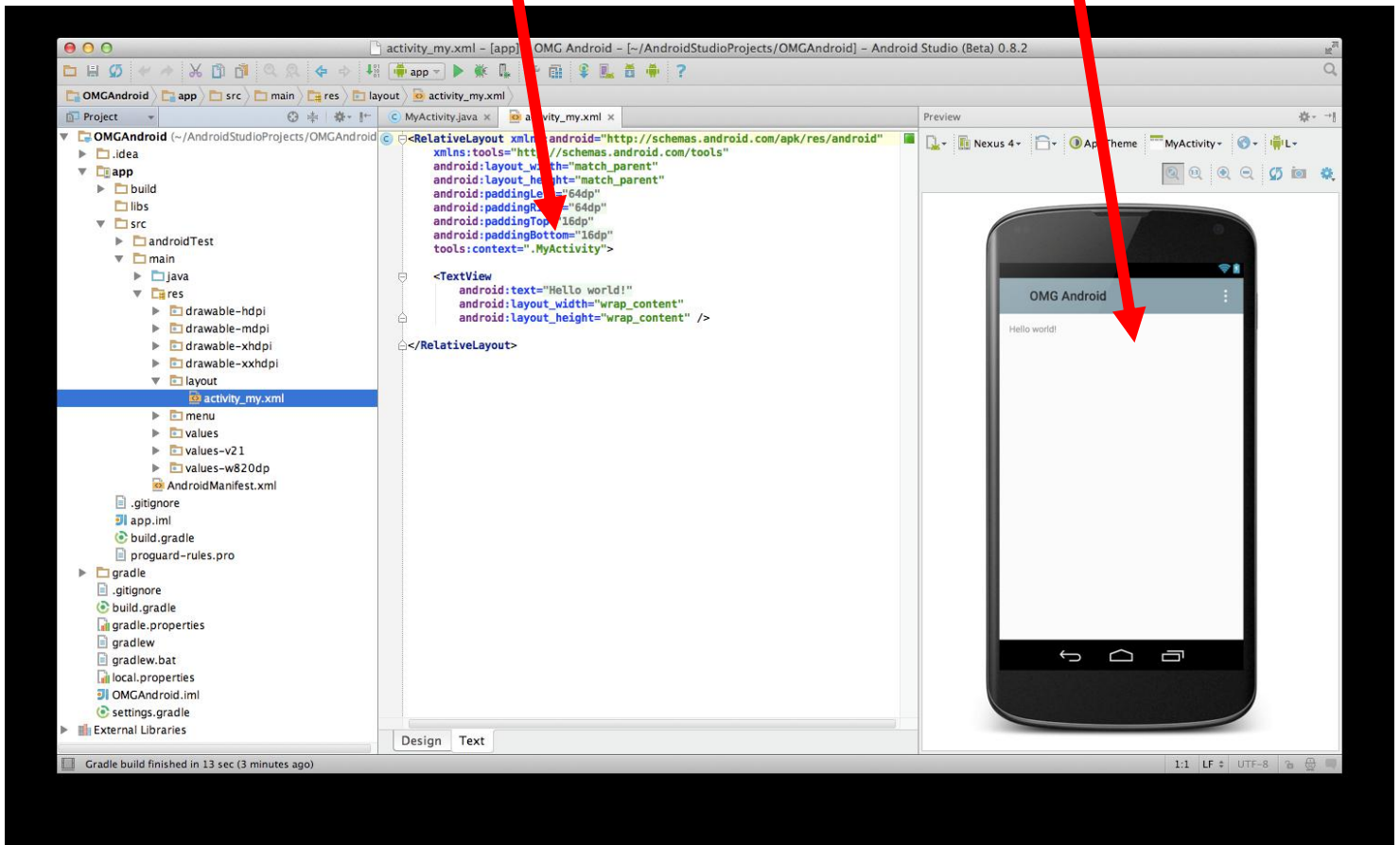
- **Views?** Rectangles containing texts, image, etc
- **Screen design:** Pick widgets, specify attributes (dimensions, margins, etc)



Editting Android UI

Can edit apps in:

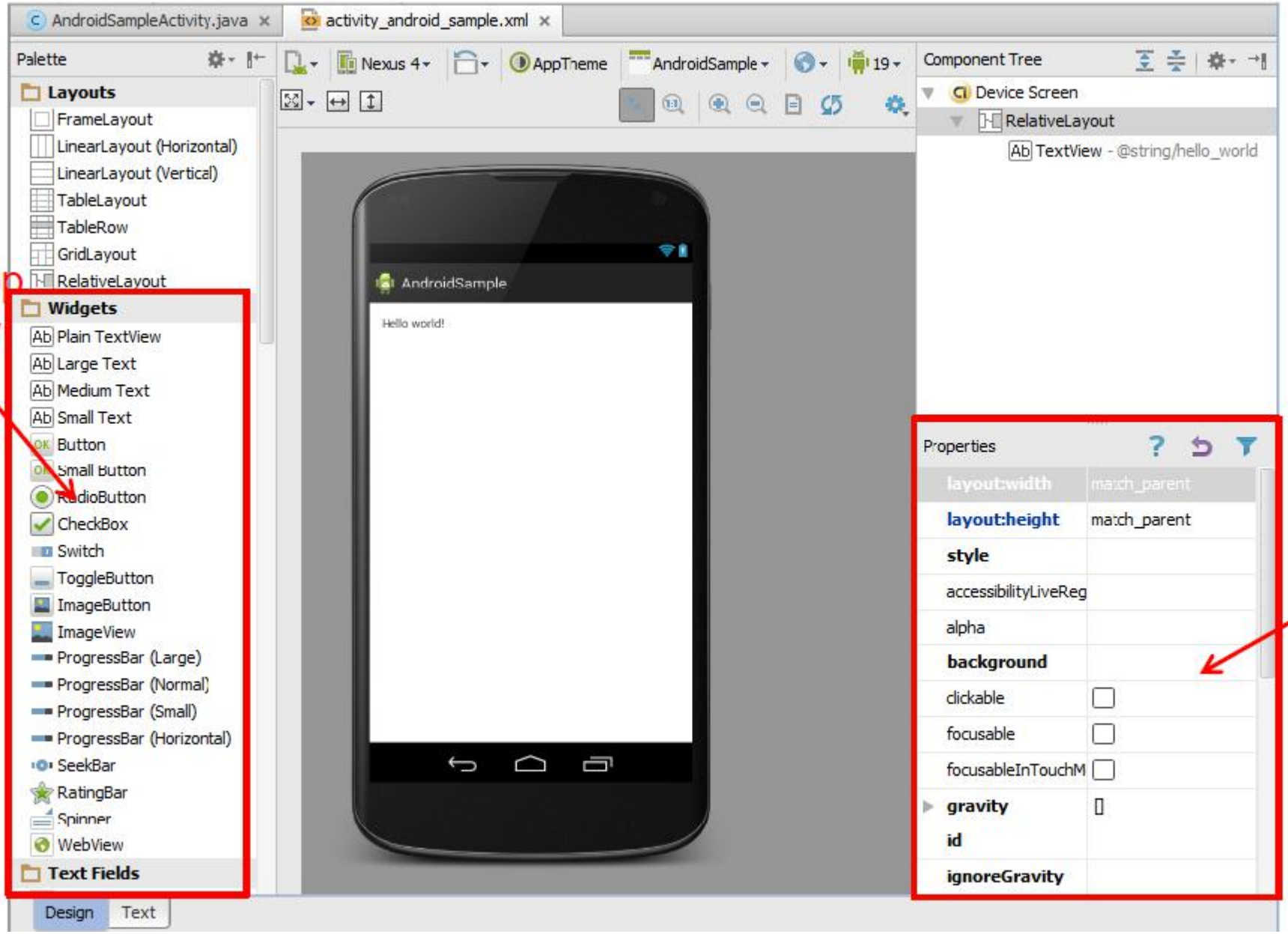
- **Text View:** edit XML directly
- **Design View:** or drag and drop widgets unto emulated phone



Design Option 1: Drag and Drop Widgets

- Drag and drop widgets in Android Studio Design View
- Edit widget properties (e.g. height, width, color, etc)

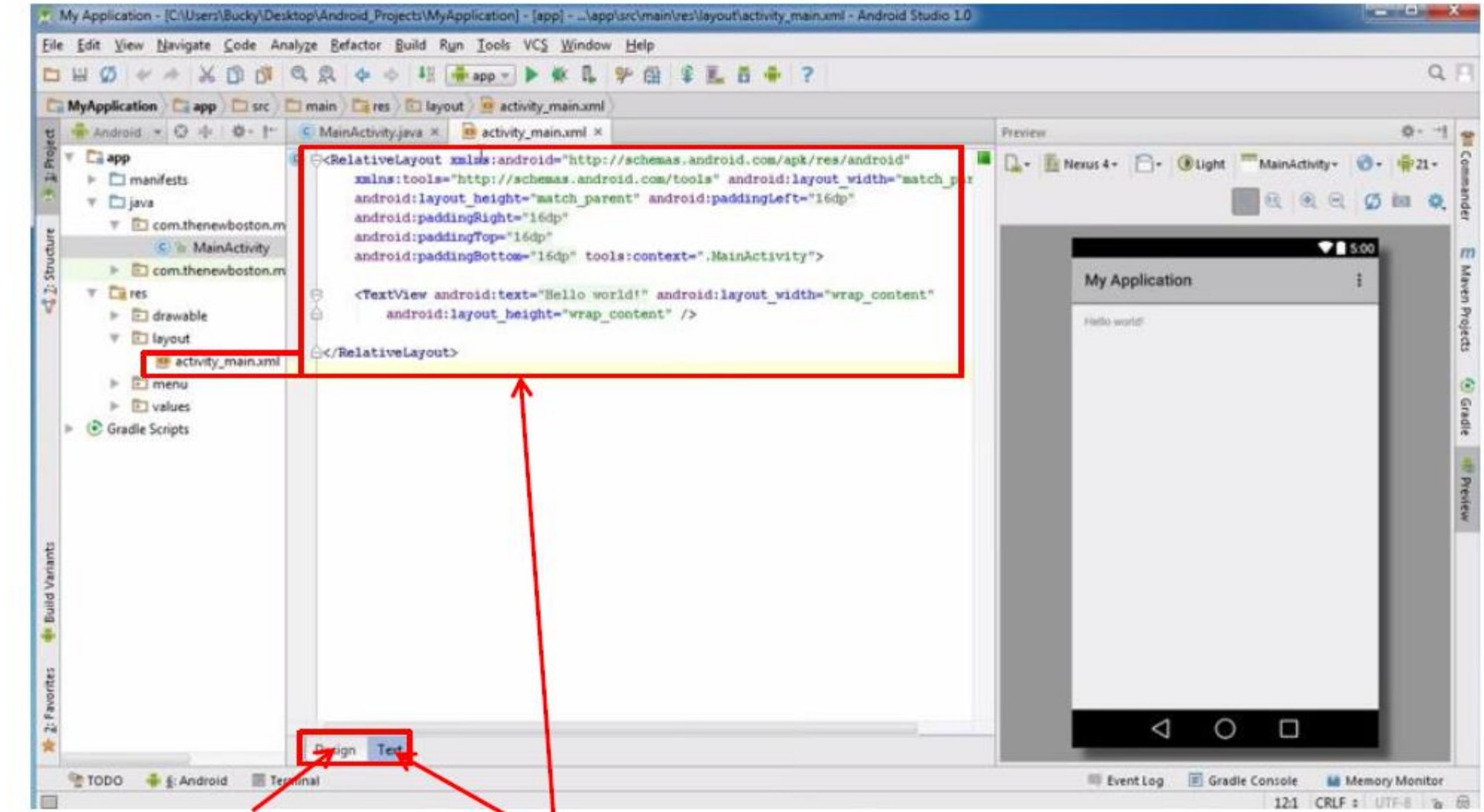
Drag and drop button or any other widget or view



Edit widget properties

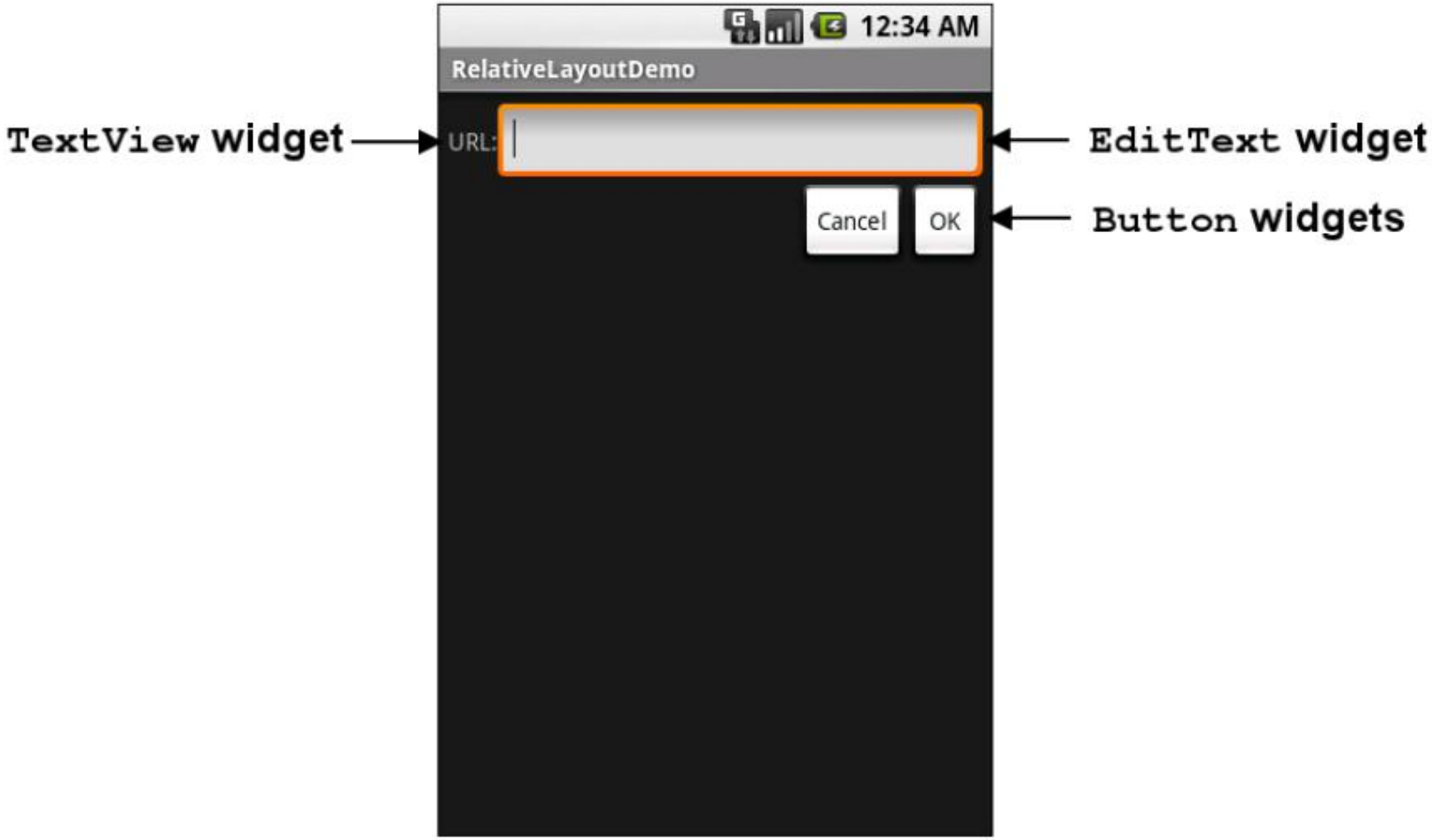
Design Option 2: Edit XML Directly

- **Text view:**Directly edit XML file defining screen (activity_main.xml)
- **Note:** dragging and dropping widgets in design view auto-generates corresponding XML in Text view

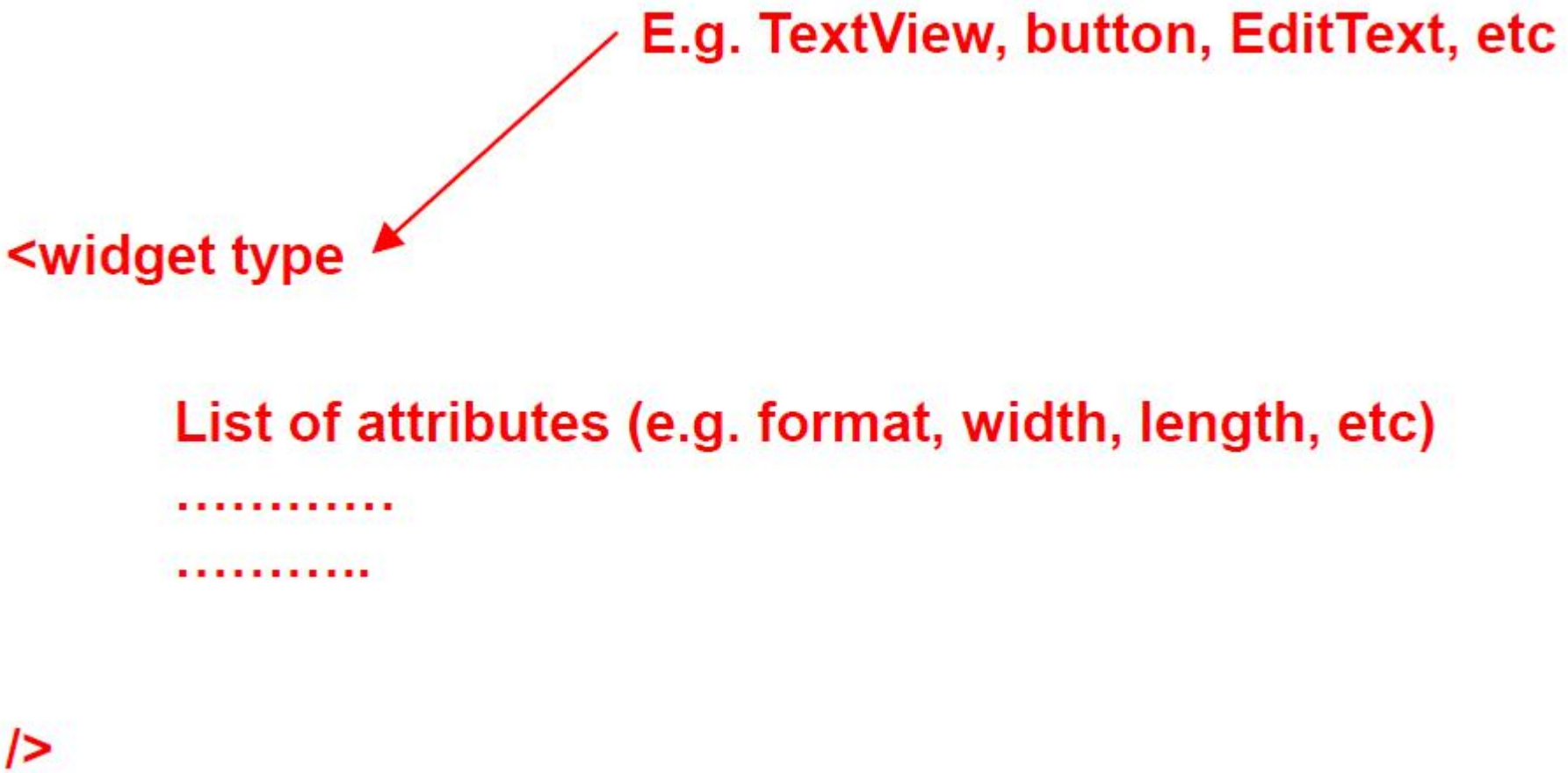

My Application - [C:\Users\Bucky\Desktop\Android_Projects\MyApplication] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 1.0
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication app src main res layout activity_main.xml
MainActivity.java
activity_main.xml
app
manifests
java
com.thenewboston.m
MainActivity
com.thenewboston.m
res
drawable
layout
activity_main.xml
menu
values
Gradle Scripts
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_par
android:layout_height="match_parent" android:paddingLeft="16dp"
android:paddingRight="16dp"
android:paddingTop="16dp"
android:paddingBottom="16dp" tools:context=".MainActivity">
<TextView android:text="Hello world!" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</RelativeLayout>
Design
Text
Preview
Nexus 4
Light
MainActivity
21
5:00
My Application
Hello world!
TODO
Android
Terminal
Event Log
Gradle Console
Memory Monitor
12:1
CRLF
Drag and drop widget
Edit XML

Android Views

- **TextView:** Text in a rectangle
- **EditText:** Text box for user to type in text
- **Button:** Button for user to click on



General Form of Widget Declaration in XML



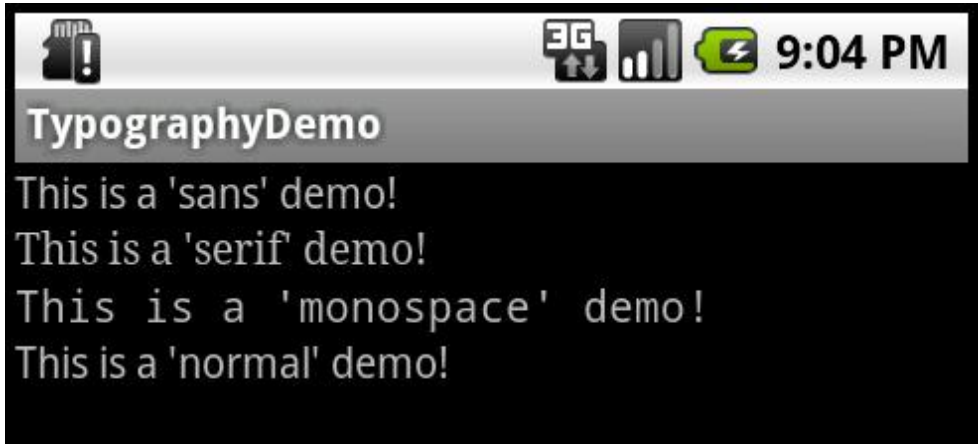
recall

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerHorizontal="true"
  android:layout_centerVertical="true"
  android:text="@string/hello_world"/>
```

TextView Widget

- Text in a rectangle
- Just displays text, no interaction

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is a 'sans' demo!"
    android:typeface="sans"
/>
```

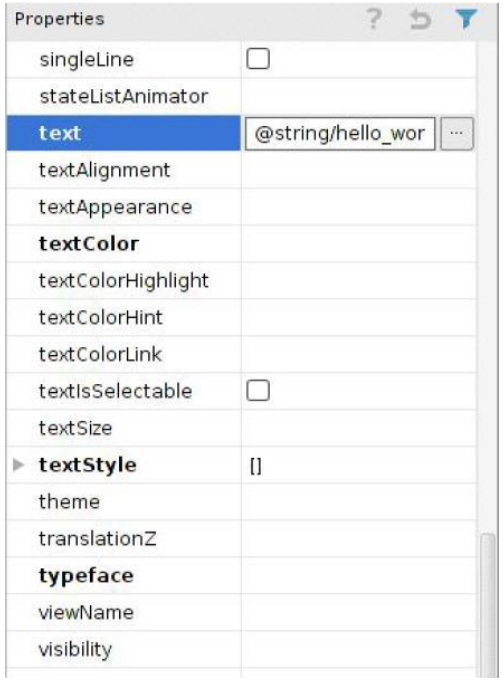


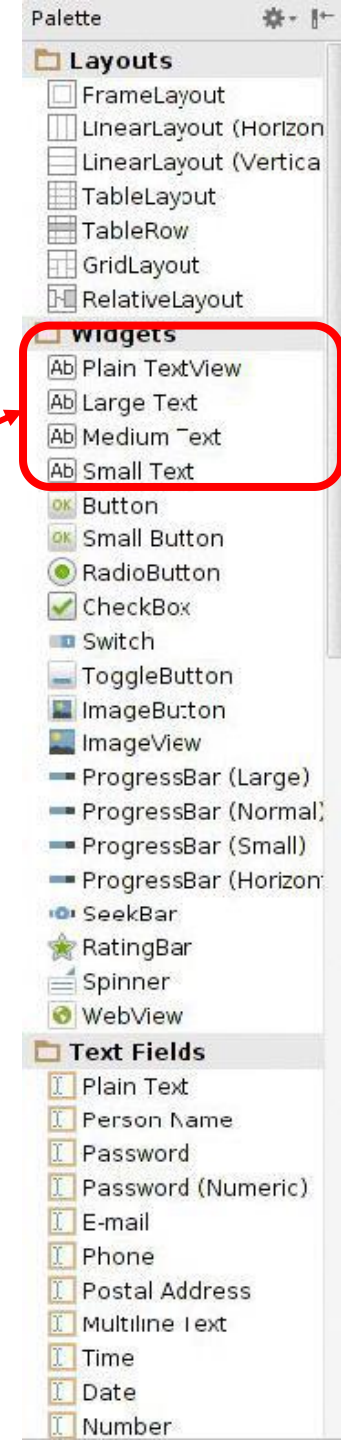
TextView Widget

- **Common attributes:**
  - typeface (android:typeface e.g monospace), bold, italic,
  - (android:textStyle),
  - text size,
  - text color (android:textColor e.g. #FF0000 for red),
  - width, height, padding,
  - background color
- Can also include links to email address, url, phone number,
- web, email, phone, map, etc

TextView

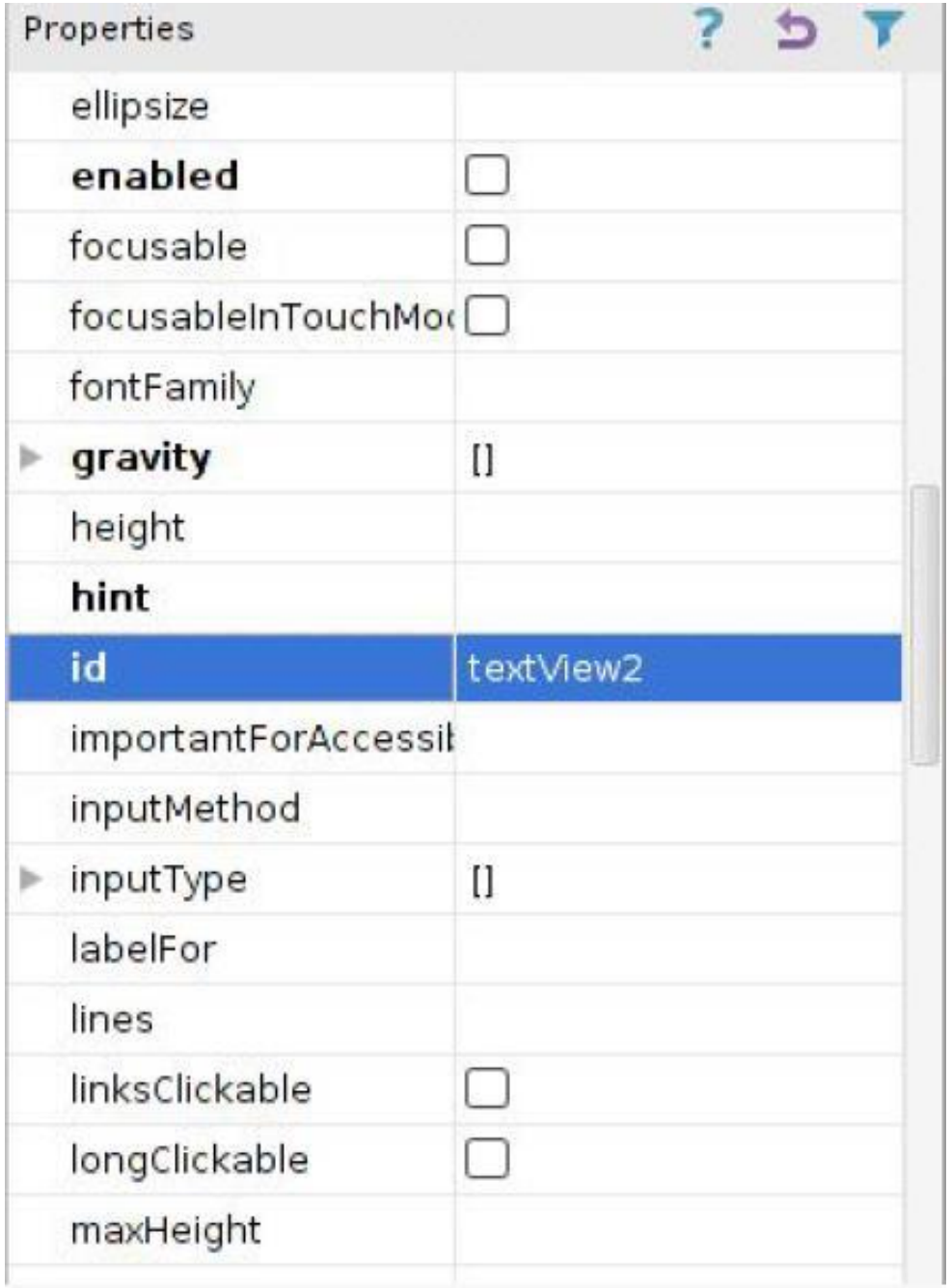
- TextView widget is available in widgets palette in Android Studio Layout editor
  - **Plain TextView**,**Large text, Medium text** and **Small text**
- After dragging Textviewwidget in, edit properties



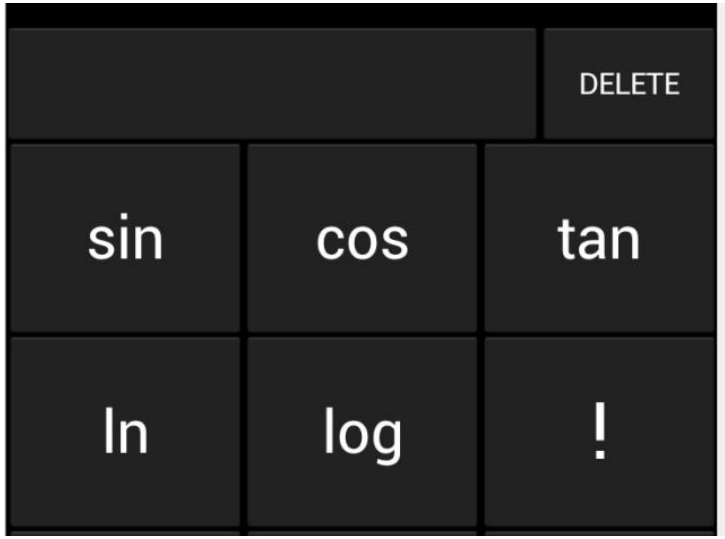


Widget ID

- Every widget has ID, stored in **android:id** attribute
- Using Widget ID declared in XML, widget can be referenced, modified in java code

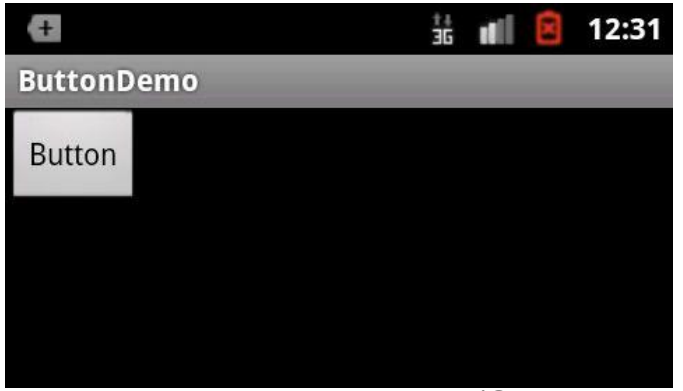


Button Widget



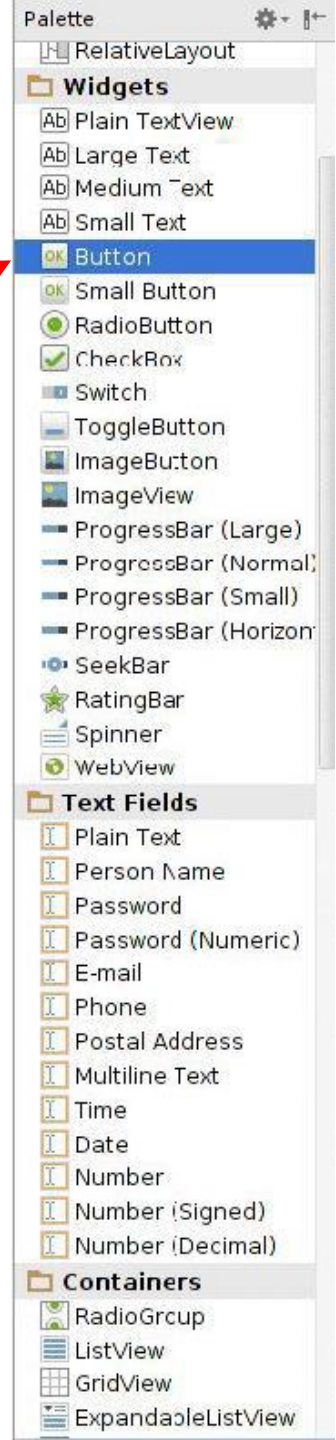
- Clickable Text or icon on a Widget (Button)
- Examples on the right
- Appearance can be customized
- Declared as subclass of TextView so similar attributes (e.g. width, height, etc)

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button"/>

</LinearLayout>
```



Button in Android Studio

- **Button** widget available in palette of Android Studio graphical layout editor
- Drag and drop button, edit its attributes



Responding to Button Clicks

- May want Button press to trigger some action. How?

1. In XML file (e.g. Activity_my.xml), set android:onClick attribute to specify method to be invoked

2. In Java file (e.g. MainActivity.java) declare method/handler to take desired action

Activity_my.xml

```
<Button
  android:onClick="someMethod"
  ...
/>
```

MainActivity.java

```
public void someMethod(View theButton) {
  // do something useful here
}
```

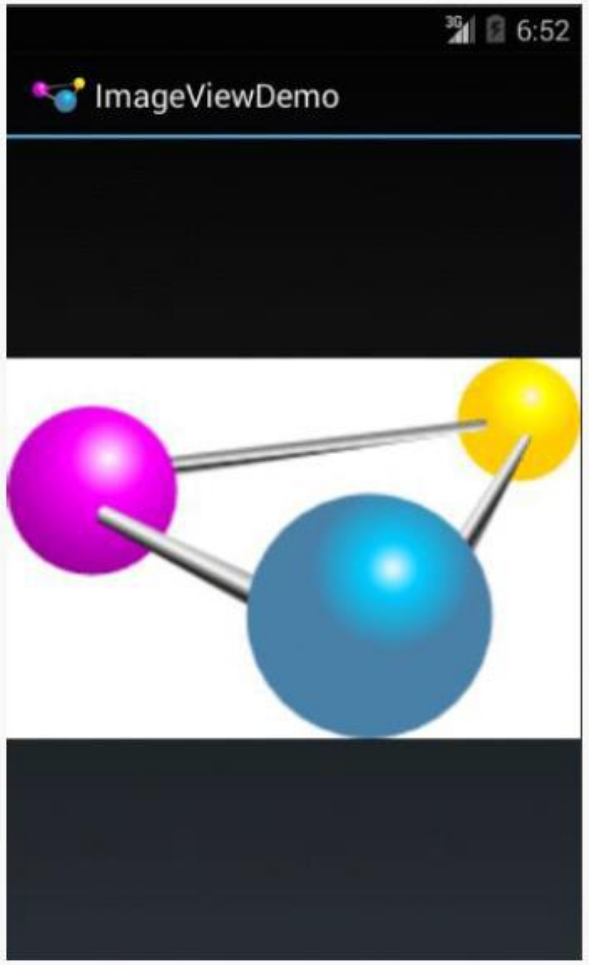
Embedding Images: ImageView and ImageButton

- **ImageView:** display image (not clickable)
- **ImageButton:** Clickable image

- Use **android:src** attribute to specify image source in **drawable** folder (e.g. **@drawable/molecule**)

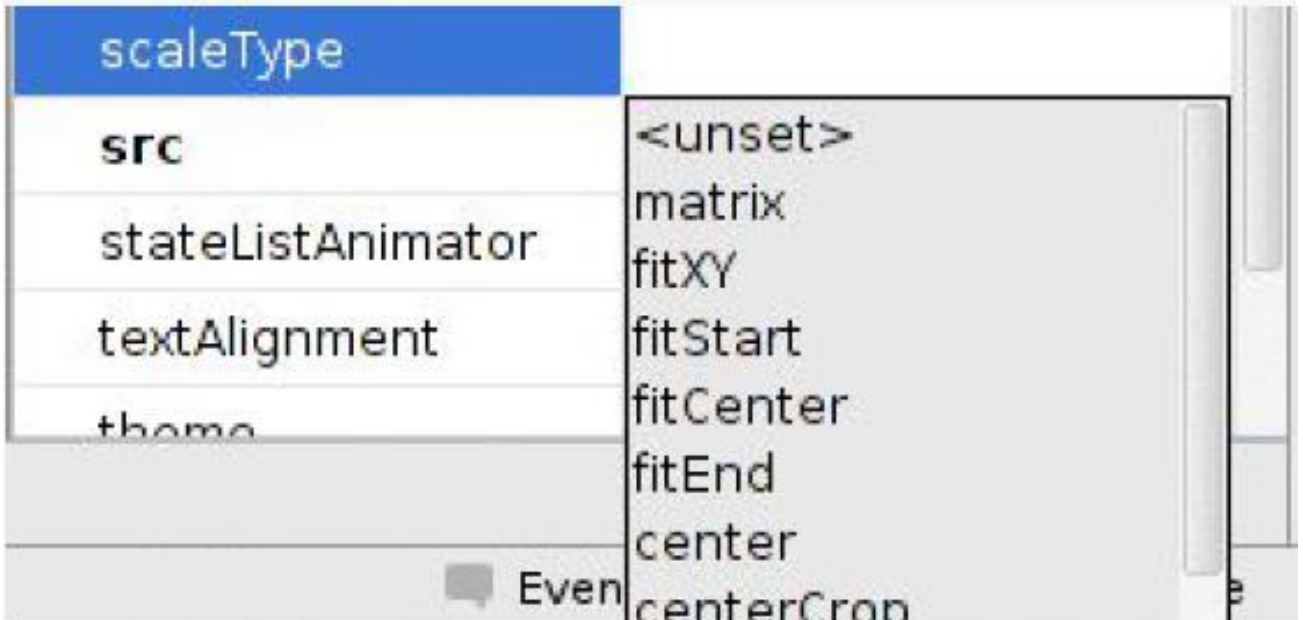
```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/icon"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:adjustViewBounds="true"
  android:src="@drawable/molecule"/>
```

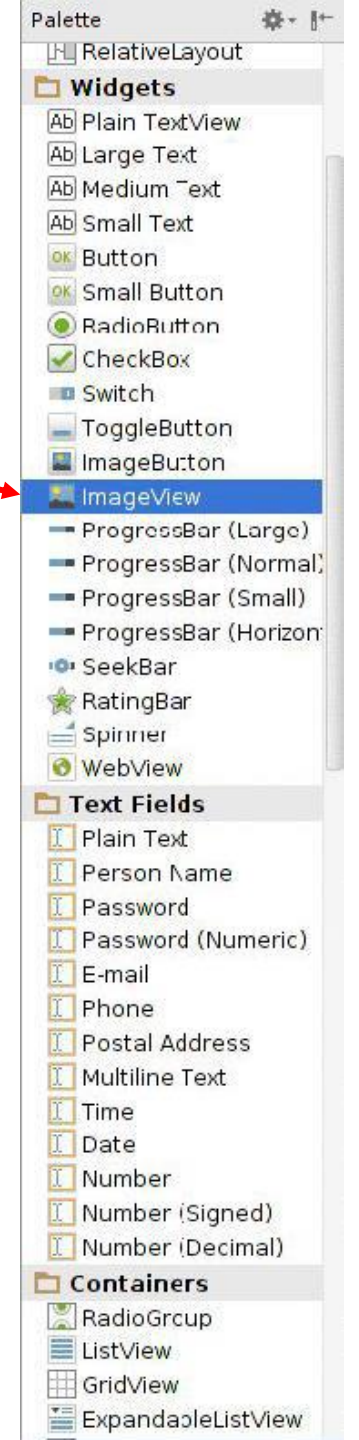
File molecule.png in drawable/ folder



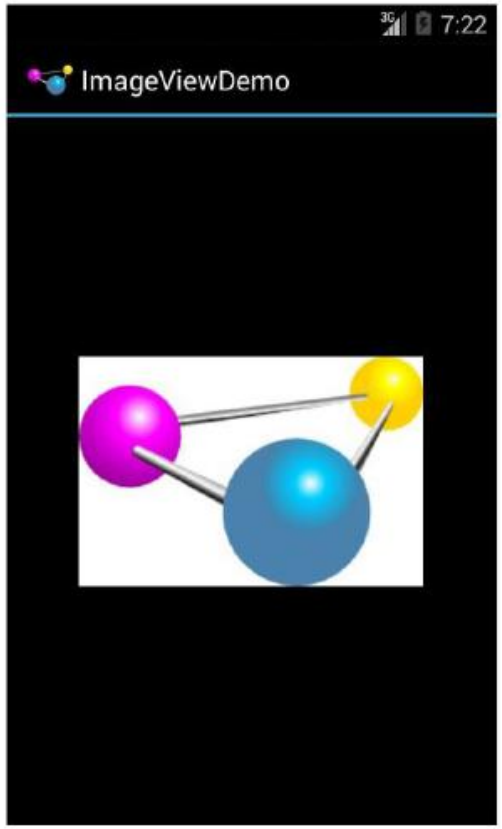
ImageView in Widgets Palette

- Can drag and drop ImageView from Widgets Palette
- Use pop-up menus (right-click) to specify:
  - **src:** choose image to be displayed
  - **scaleType:** choose how image should be scaled

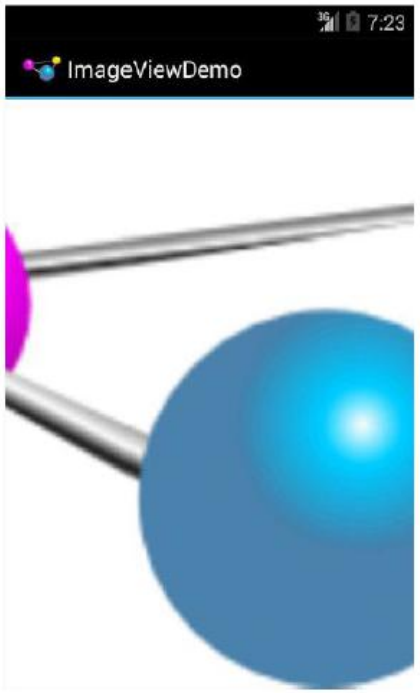




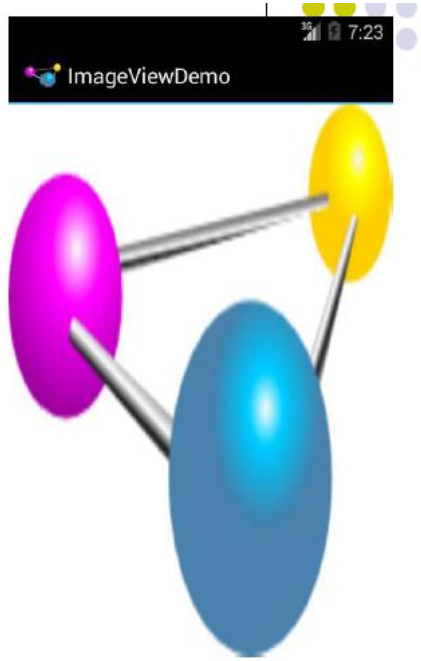
Options for Scaling Images (scaleType)



"center" centers image but does not scale it



"centerCrop" centers image, scales it (maintaining aspect ratio) so that shorter dimension fills available space, and crops longer dimension

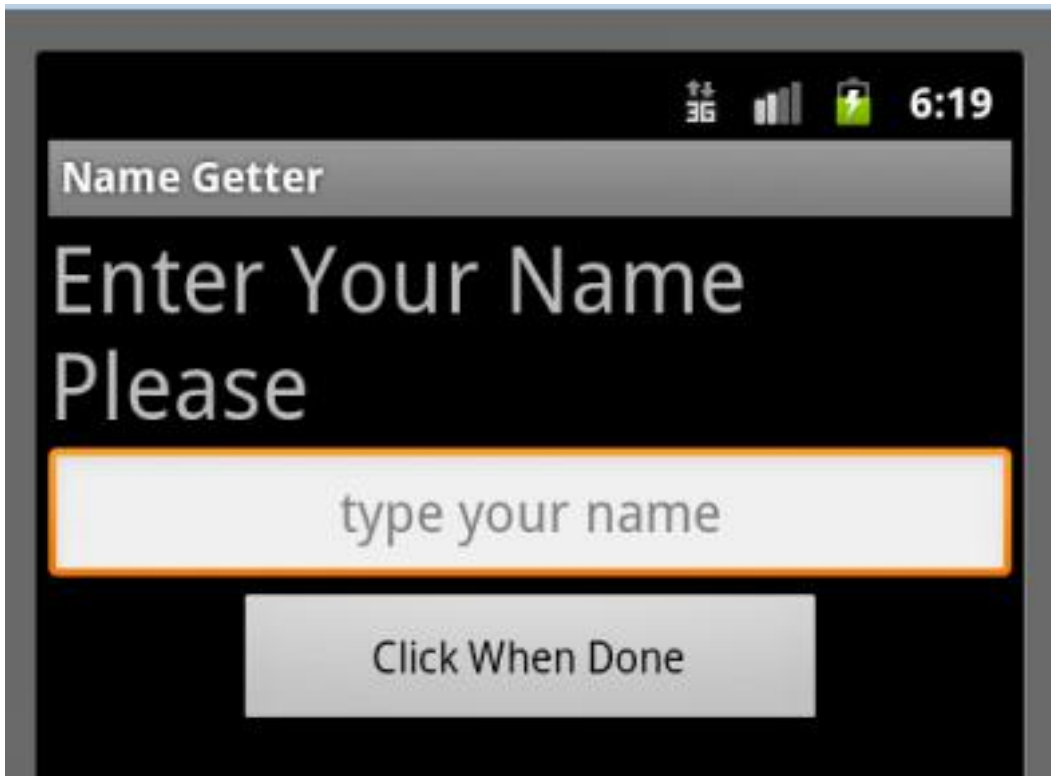


"fitXY" scales/distorts image to fit ImageView, ignoring aspect ratio

EditText Widget

- Widget with box for user input

```
<EditText
    android:id="@+id/edittext"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:gravity="center"
    android:inputType="textPersonName"
    android:hint="type your name" />
```



- Text fields can have different input types
  - e.g. number, date, password, or email address
- **android:inputType** attribute sets input type, affects
  - What type of keyboard pops up for user
  - E.g. if inputType is a number, numeric keyboard pops up

EditText Widget in Android Studio Palette

A section of Android Studio palette has EditText widgets (or text fields)

Text Fields
Section of Widget palette

EditText inputType menu

Some Other Available Widgets

MapView

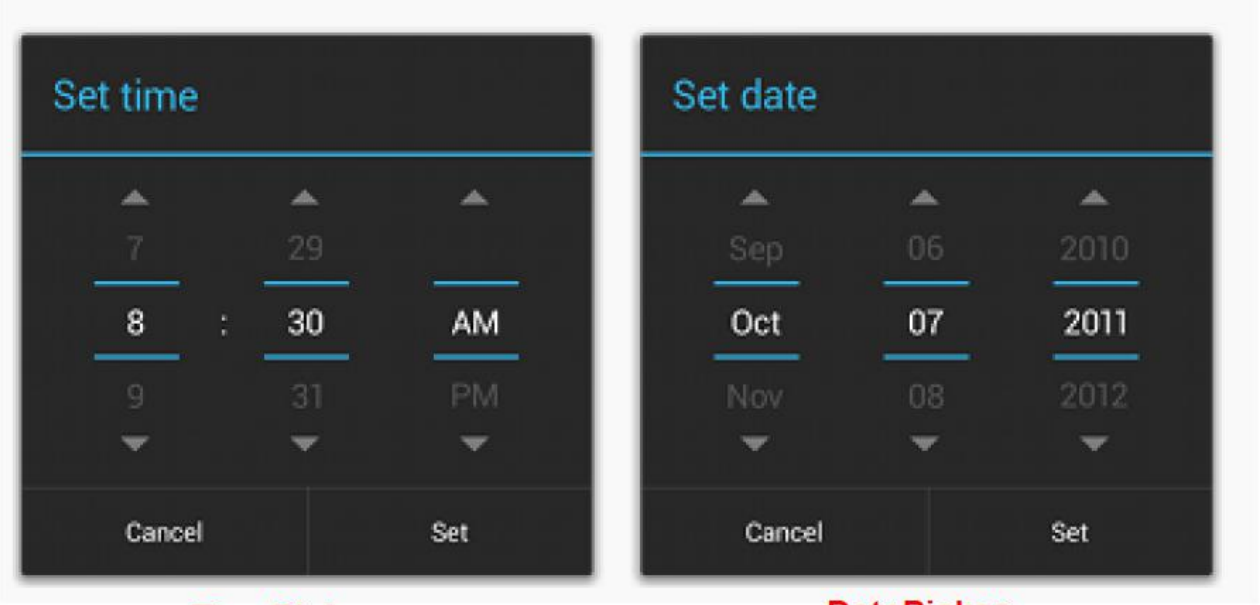
Rectangle that contains a map

WebView

Rectangle that contains a web page

Pickers

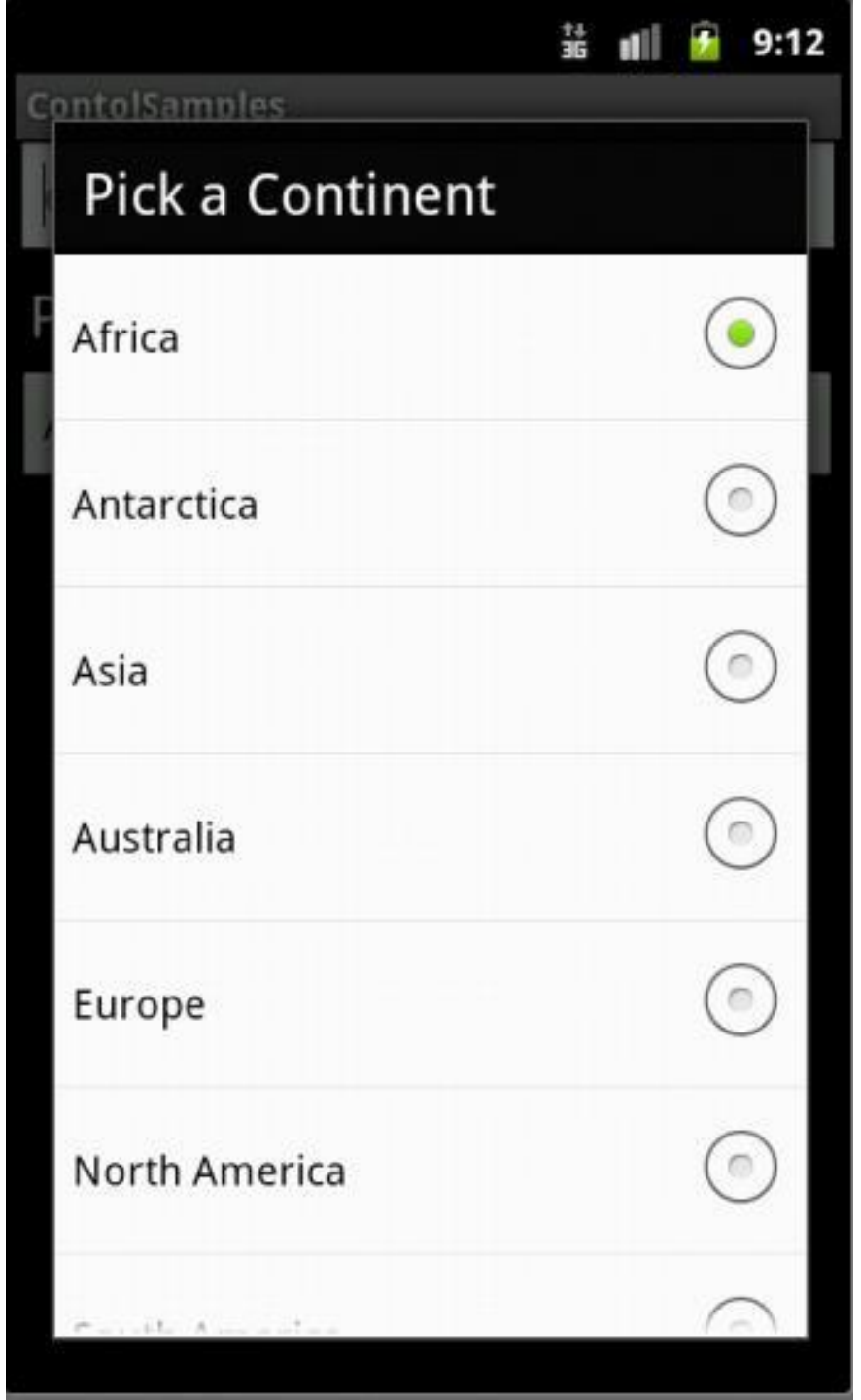
- **TimePicker:** Select a time
- **DatePicker:** Select a date
- Typically displayed in pop-up dialogs (**TimePickerDialog** or **DatePickerDialog**)



TimePicker DatePicker

Spinner Controls

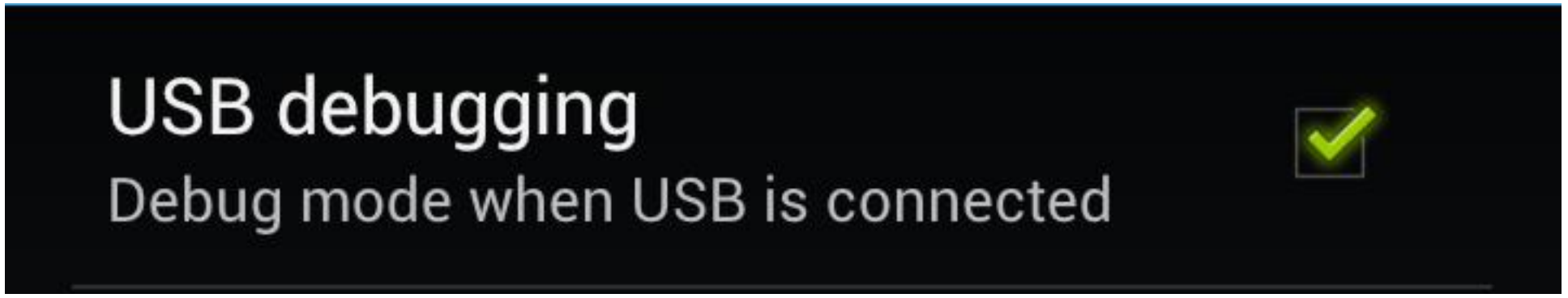
user **must** select one of a set of choices



Checkbox

- Checkbox has 2 states: checked and unchecked
- XML code to create Checkbox

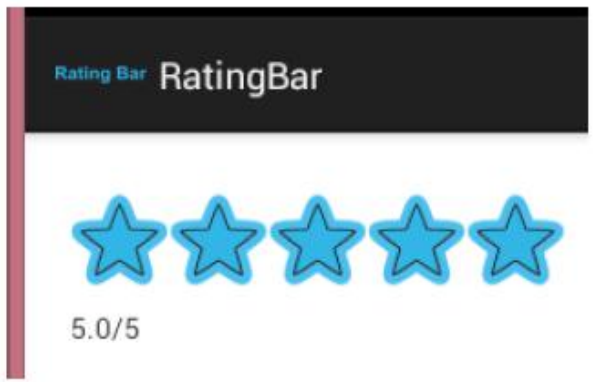
```
<?xml version="1.0" encoding="utf-8"?>
<CheckBox xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/check"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/unchecked"/>
```



Other Indicators & More Widgets

- ProgressBar

- RatingBar



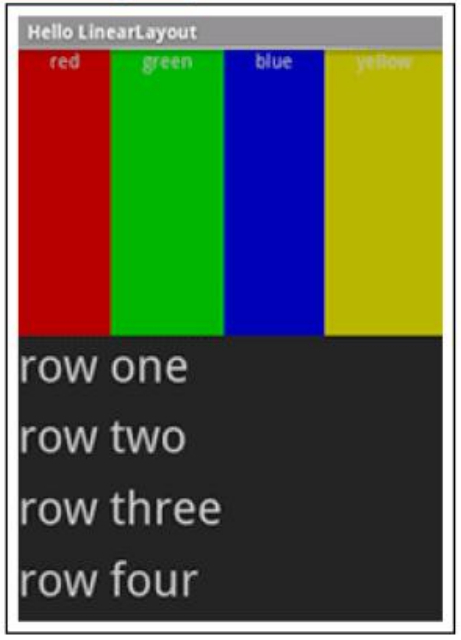
- Chronometer
- DigitalClock
- AnalogClock

Android Layouts

Android Layouts

- Layout? Pattern in which multiple widgets are arranged
- Layouts contain widgets
- In Android internal classes, widget is child of layout
- Layouts (XML files) stored in **res/layout**

LinearLayout



RelativeLayout

TableLayout

Views vs Layouts

Views

- text
- buttons
- images

...

Layouts

- linear layout
- grid layout
- relative layout

...

Some Layouts

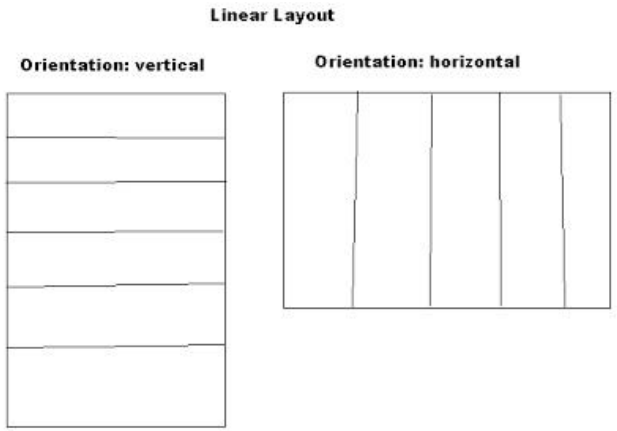
- FrameLayout,
- LinearLayout,
- TableLayout,
- GridLayout,
- RelativeLayout,
- ListView,
- GridView,
- ScrollView,
- DrawerLayout,
- ViewPager

LinearLayout

- aligns child elements (e.g. buttons, text boxes, pictures, etc.) in one direction

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.c
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
        android:background="#ff00ff"
    android:orientation="vertical" >
```

- orientation attribute defines direction (vertical or horizontal):
  - E.g. android:orientation=*"vertical"*



Layout Width and Height Attributes

- **wrap_content:** widget as wide/high as its content (e.g. text)
- **match_parent:** widget as wide/high as its parent layout box
- **fill_parent:** older form of **match_parent**

Text widget width should be as wide as Its parent (the layout)

Text widget height should Be as wide as the content (text)

```
<?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"
        />

</LinearLayout>
```

The View inside the layout is a TextView, a View specifically made to display text.

main.xml

Hierarchy

Screen (Hardware)

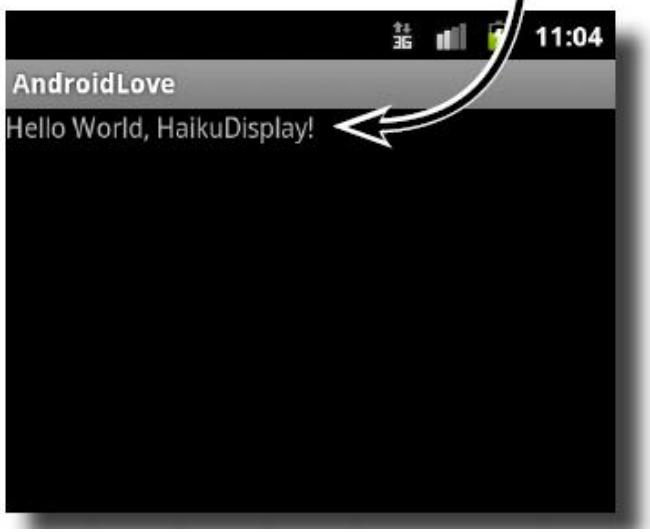
↑

Linear Layout

↑

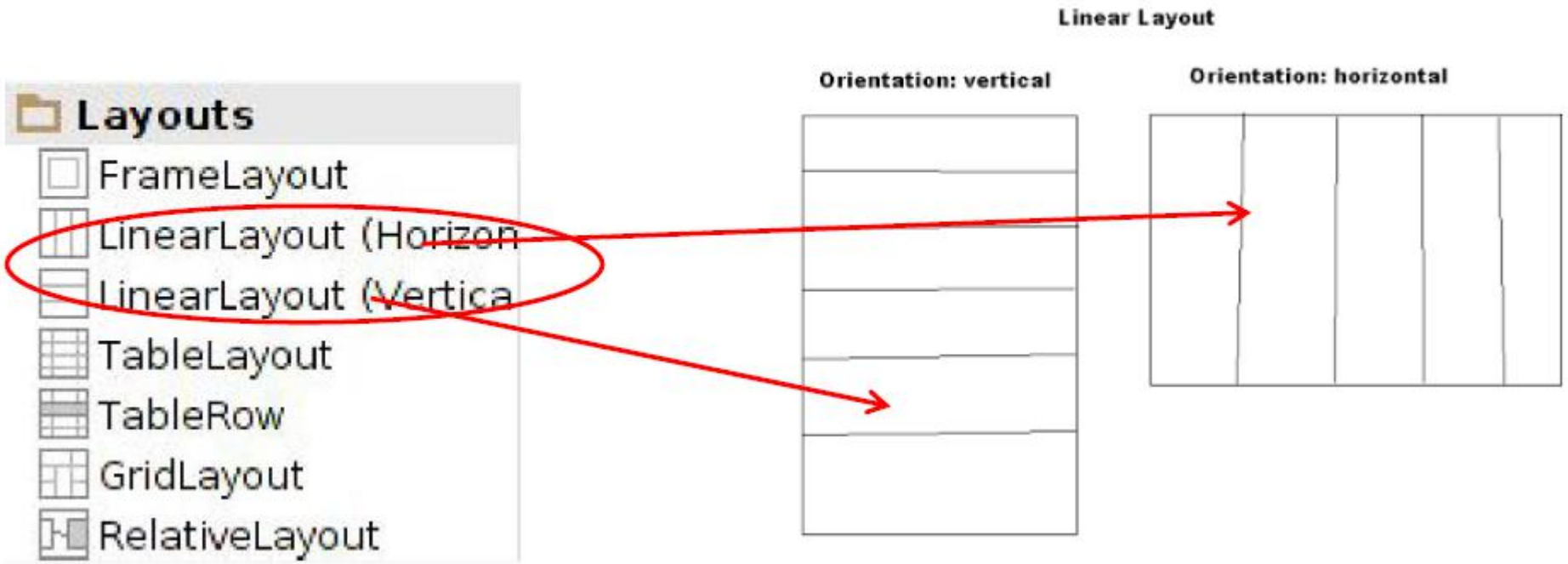
TextView

The ViewGroup, in this case a LinearLayout fills the screen.



LinearLayout in Android Studio

LinearLayout in Android Studio Graphical Layout Editor



LinearLayout in Android Studio

- After selecting LinearLayout, toolbars buttons to set parameters

Setting Attributes

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.c
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
        android:background="#ff00ff"
    android:orientation="vertical" >
```

in layout xml file

```
public class UISamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void change(View v) {
        LinearLayout vg = (LinearLayout)this.findViewById(R.id.main_layout);
        Log.d("UI SAMPLE", vg + "");
        vg.setOrientation(LinearLayout.HORIZONTAL);
    }
}
```

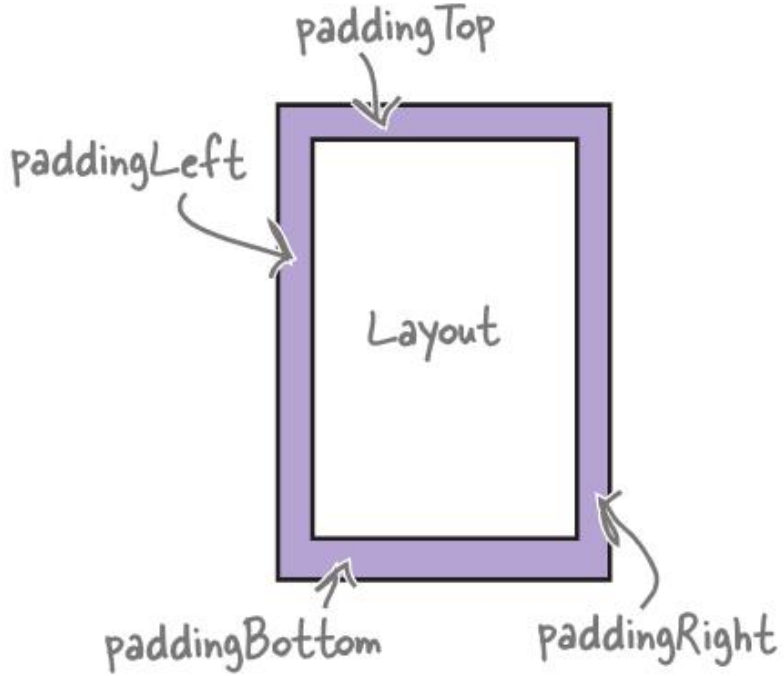
Can also design UI, set attributes in Java program (e.g. ActivityMain.java) (More later)

Adding Padding

- Paddings sets space between layout sides and its parent (e.g. the screen)

```
<RelativeLayout ...
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp">
    ...
</RelativeLayout>
```

Add padding of 16dp.

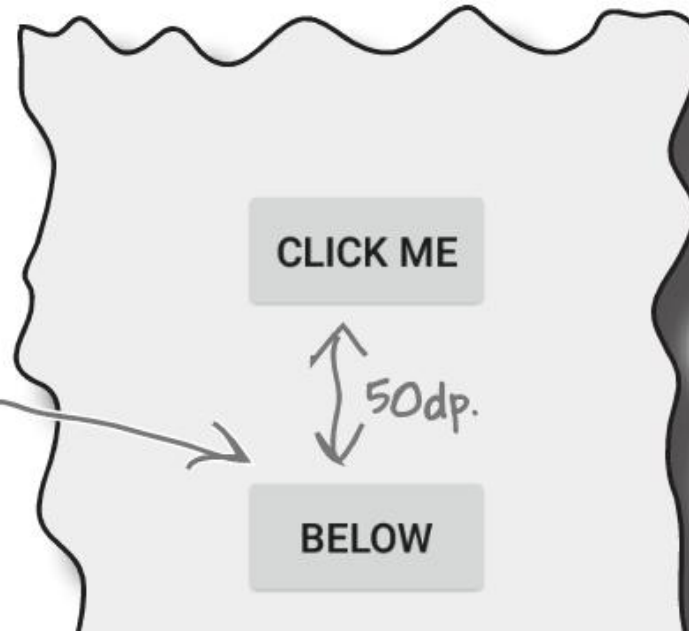


Setting Margins

- Can increase gap (margin) between adjacent widgets
- E.g. To add margin between two buttons, in declaration of bottom button

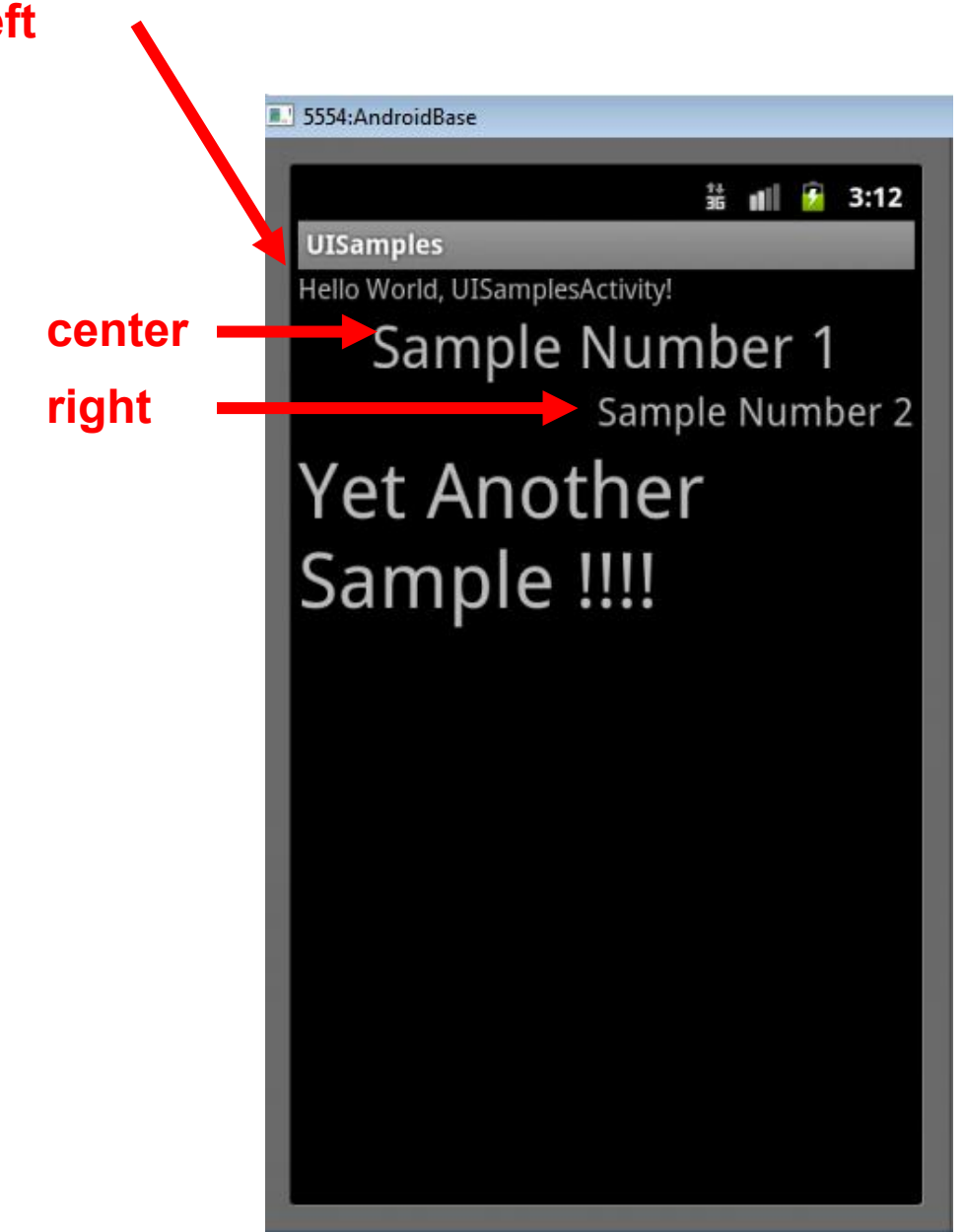
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/button_click_me"
    android:layout_below="@+id/button_click_me"
    android:layout_marginTop="50dp"
    android:text="@string/button_below" />
</RelativeLayout>
```

Adding a margin to the top of the bottom button adds extra space between the two views.



Gravity Attribute

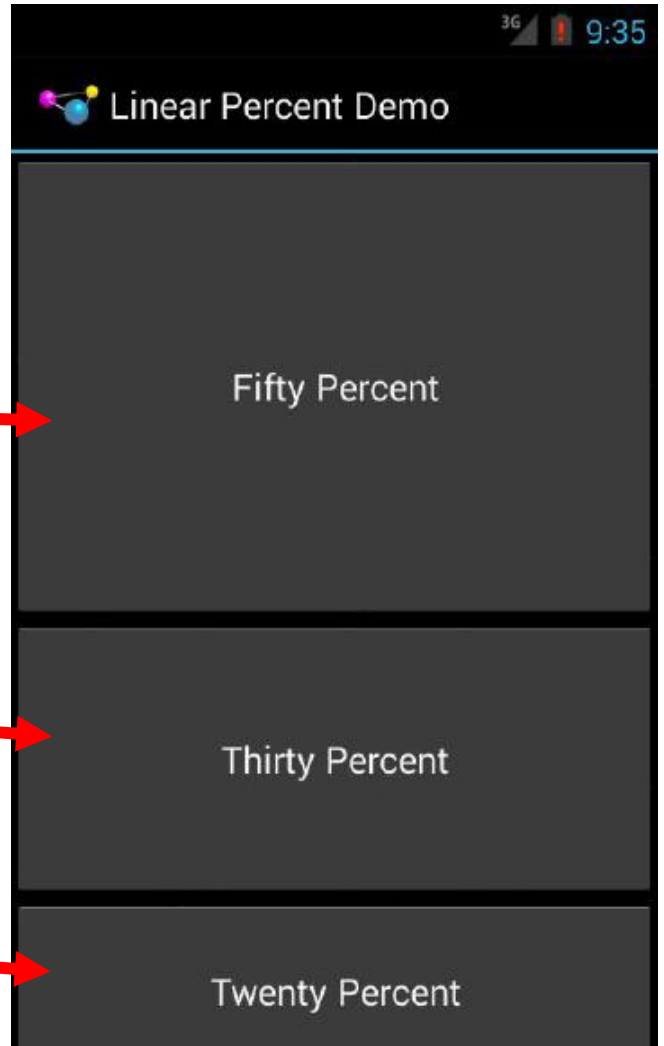
- By default, linearlayout left-and top-aligned
- Gravity attribute changes alignment :
  - e.g. android:gravity = “right”



Linear Layout Weight Attribute

- Specifies "importance“, larger weights takes up more space

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <Button
    android:layout_width="match_parent"
    android:layout_height="0dip"
    android:layout_weight="50"
    android:text="@string/fifty_percent"/>

  <Button
    android:layout_width="match_parent"
    android:layout_height="0dip"
    android:layout_weight="30"
    android:text="@string/thirty_percent"/>

  <Button
    android:layout_width="match_parent"
    android:layout_height="0dip"
    android:layout_weight="20"
    android:text="@string/twenty_percent"/>

</LinearLayout>
```



LinearLayout Attributes

XML attributes	
android:baselineAligned	When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider	Drawable to use as a vertical divider between buttons.
android:gravity	Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:measureWithLargestChild	When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation	Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum	Defines the maximum weight sum.

Scrolling

- Phone screens are small, scrolling content helps
- Examples: Scroll through
  - large image
  - Linear Layout with lots of elements
- Views for Scrolling:
  - **ScrollView**for vertical scrolling
  - **HorizontalScrollView**
- Rules:
  - Only one direct child View
  - Child could have many children of its own

```
<ScrollView
    ...>
    <LinearLayout>
       ....
       <!-- you can have as many Views in here as you want -->
    </LinearLayout>
</ScrollView>
```