CS 403X Mobile and Ubiquitous Computing

Lecture 4: Intro to Android Programming (Part 2)

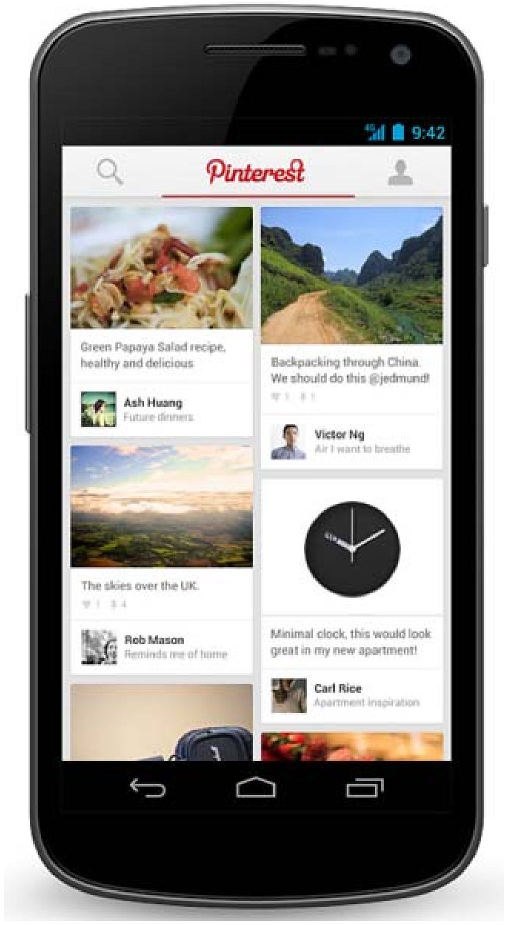
Emmanuel Agu

Android UI Design in XML

Recall: Files Hello World Android Project

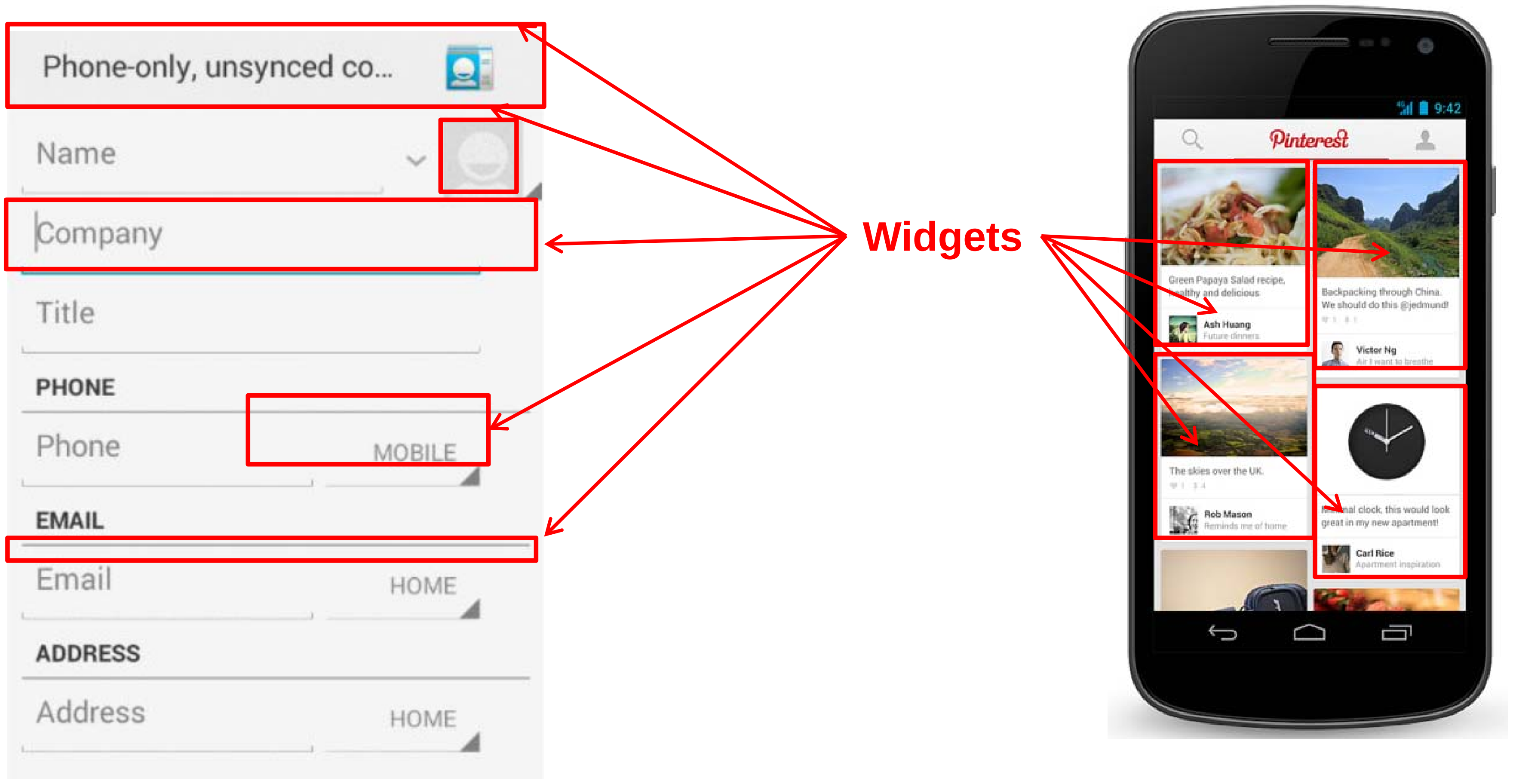
XML file used to design Android UI

- 3 Files:
  - **Activity_main.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all app components and screens
    - Like a table of contents for a book
    - E.g. Hello world program has 1 screen, so AndroidManifest.xml has 1 item listed
    - App starts running here (a bit like main( ) in C), launching activity with a tag "LAUNCHER"


9:42
Pinterest
Green Papaya Salad recipe, healthy and delicious
Ash Huang
Future dinners
Backpacking through China. We should do this @jedmund!
Victor Ng
Air I want to breathe
The skies over the UK.
Rob Mason
Reminds me of home
Minimal clock, this would look great in my new apartment!
Carl Rice
Apartment inspiration

Widgets

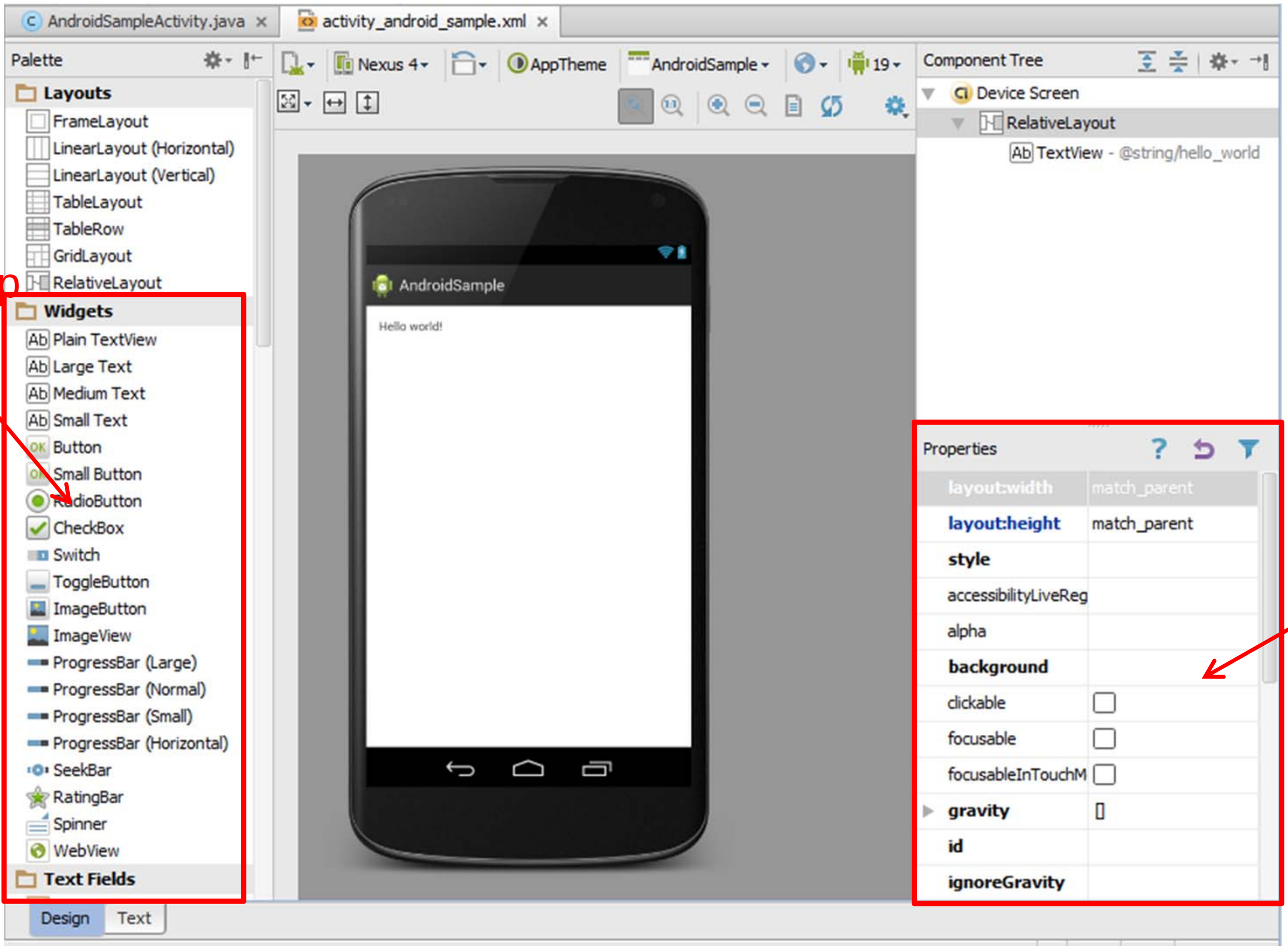
- ***Android UI design involves arranging widgets on a screen***
- Pick widgets, specify widget attributes (dimensions, margins, padding, etc)



Recall: Design Option 1: Add Widget in Design View

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

Drag and drop button or any other widget or view

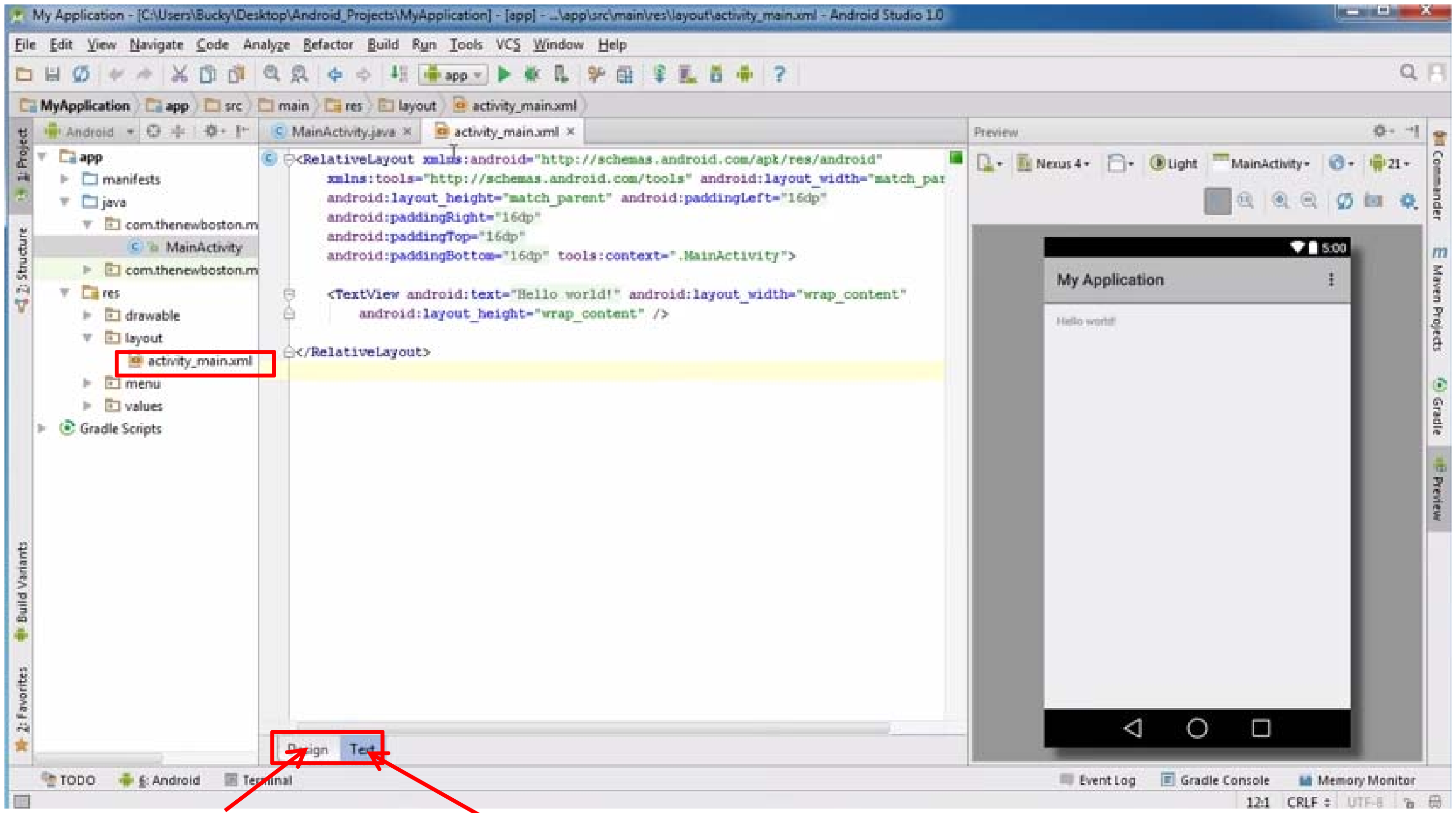


layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
clickable	☐
focusable	☐
focusableInTouchM	☐
gravity	[]
id	
ignoreGravity	

Edit widget properties

Recall: 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 generates related XML in Text view



```
<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>
```

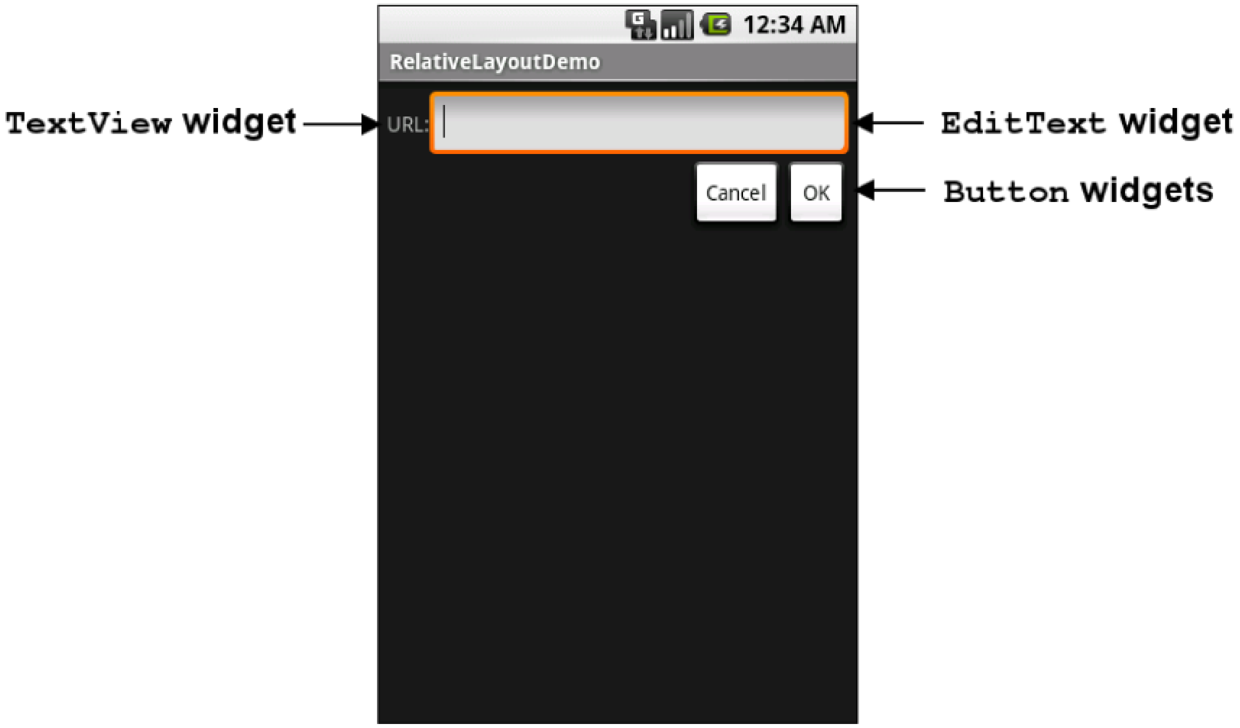
Drag and drop widget

Edit XML

Android Widgets

Example: Some Common Widgets

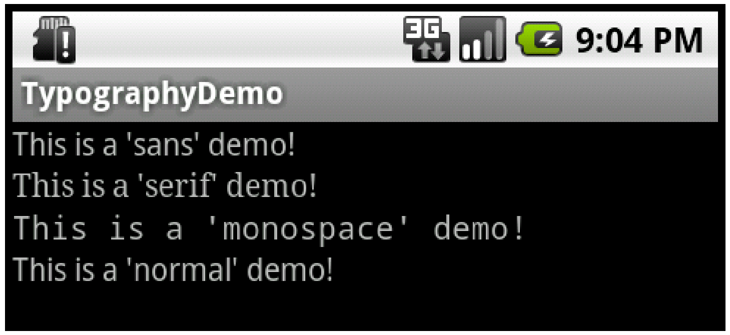
- **TextView:** Text in a rectangle
- **EditText:** Text box for user to type in text
- **Button:** Button for user to click on



TextView

- Text in a rectangle
- Display text, not for interaction

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



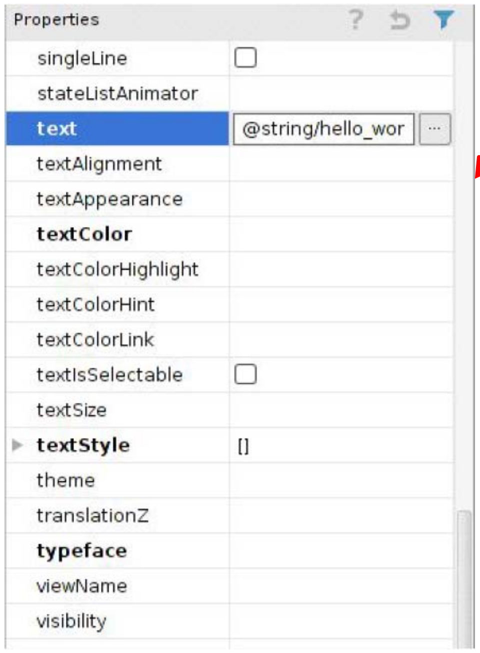
This is a 'sans' demo!
This is a 'serif' demo!
This is a 'monospace' demo!
This is a 'normal' demo!

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

TextView

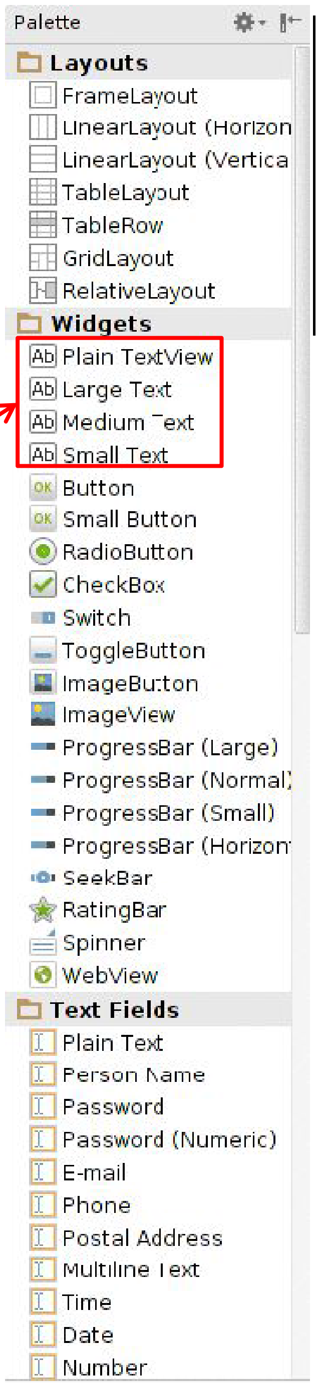
- TextView widget is available in widgets palette in Android Studio Layout editor

- **Plain TextView, Large text, Medium text** and **Small text** are all TextView widgets
- After dragging Textview in, edit properties



Properties

singleLine	☐
stateListAnimator	
text	@string/hello_wor
textAlignment	
textAppearance	
textColor	
textColorHighlight	
textColorHint	
textColorLink	
textIsSelectable	☐
textSize	
textStyle	[]
theme	
translationZ	
typeface	
viewName	
visibility	



Palette

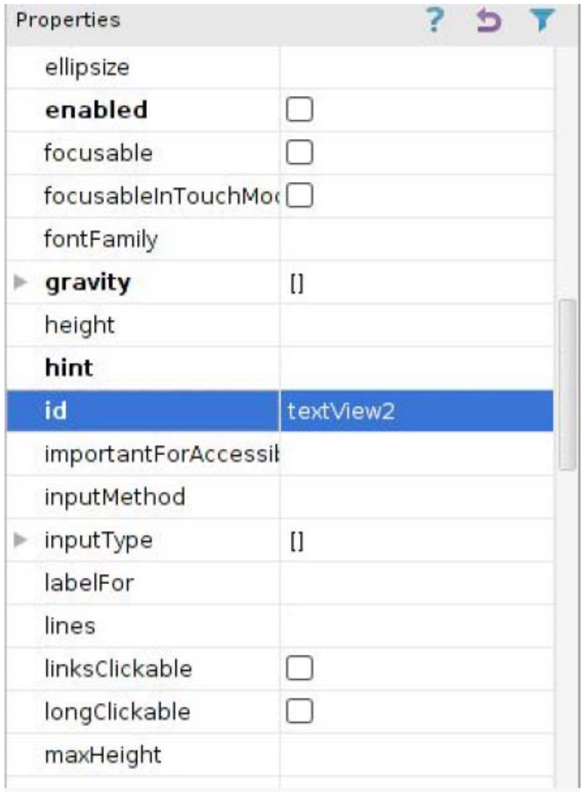
Layouts
- FrameLayout
- LinearLayout (Horizon
- LinearLayout (Vertica
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Widgets
- Plain TextView
- Large Text
- Medium Text
- Small Text
- Button
- Small Button
- RadioButton
- CheckBox
- Switch
- ToggleButton
- ImageButton
- ImageView
- ProgressBar (Large)
- ProgressBar (Normal)
- ProgressBar (Small)
- ProgressBar (Horizon
- SeekBar
- RatingBar
- Spinner
- WebView

Text Fields
- Plain Text
- Person Name
- Password
- Password (Numeric)
- E-mail
- Phone
- Postal Address
- Multiline Text
- Time
- Date
- Number

Widget ID

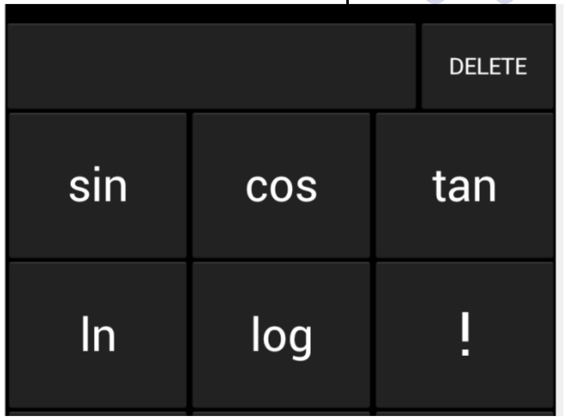
- Every widget has ID, stored in **android:id** attribute
- In java code, to manipulate declared in XML file, need to reference it using its ID (More later)
- Naming convention
  - First time use: @+id/xyx_name
  - Subsequent use: @id/xyz_name

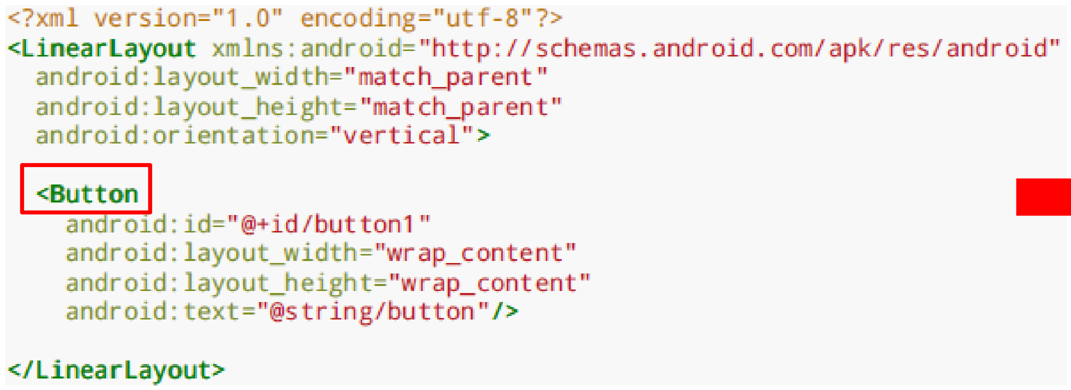

Properties

ellipsize	
enabled	☐
focusable	☐
focusableInTouchMo	☐
fontFamily	
gravity	[]
height	
hint	
id	textView2
importantForAccessil	
inputMethod	
inputType	[]
labelFor	
lines	
linksClickable	☐
longClickable	☐
maxHeight	

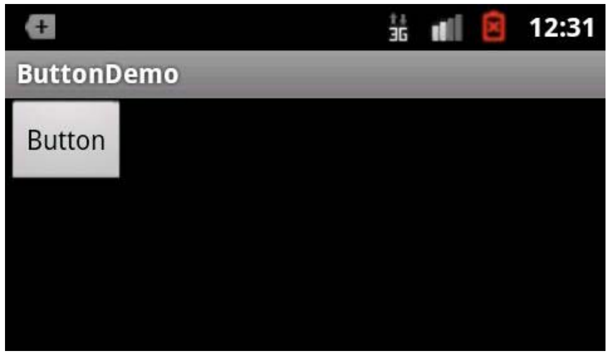
Button Widget

- Text or icon or both on View (Button)
- E.g. “Click Here”
- Appearance of buttons can be customized
- Declared as subclass of TextView so similar attributes (e.g. width, height, etc)



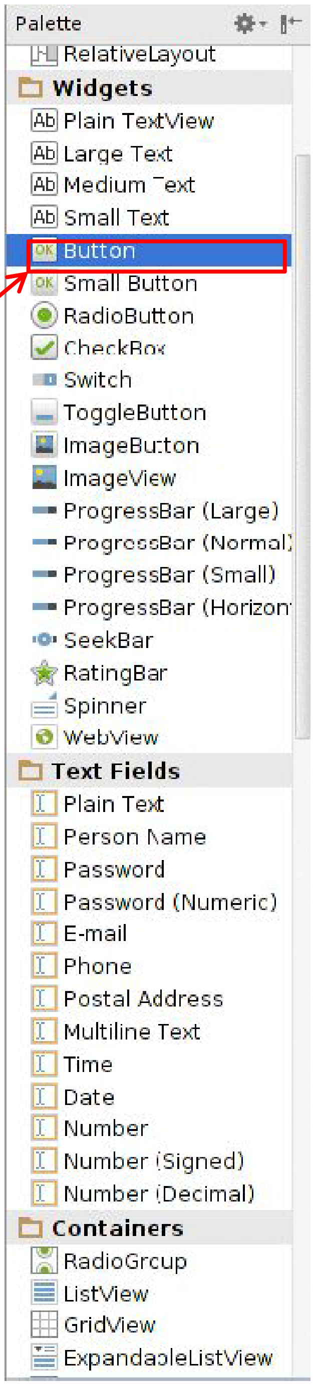


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

 <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**

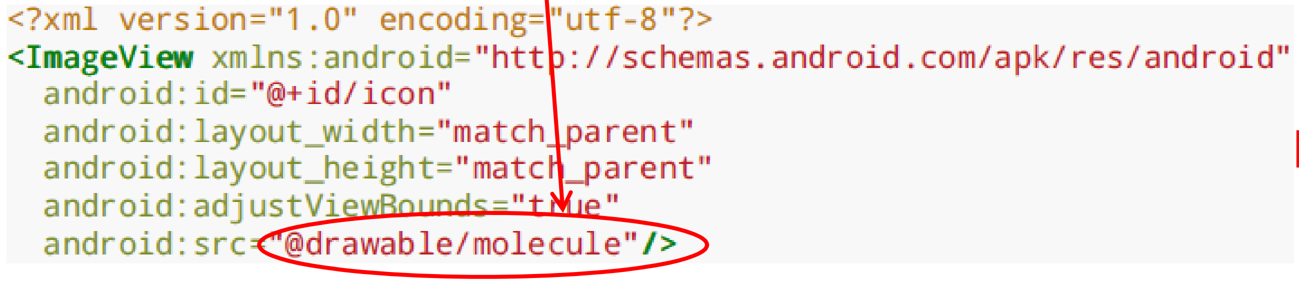
```
<Button
  android:onClick="someMethod"
  ...
/>
```

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

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

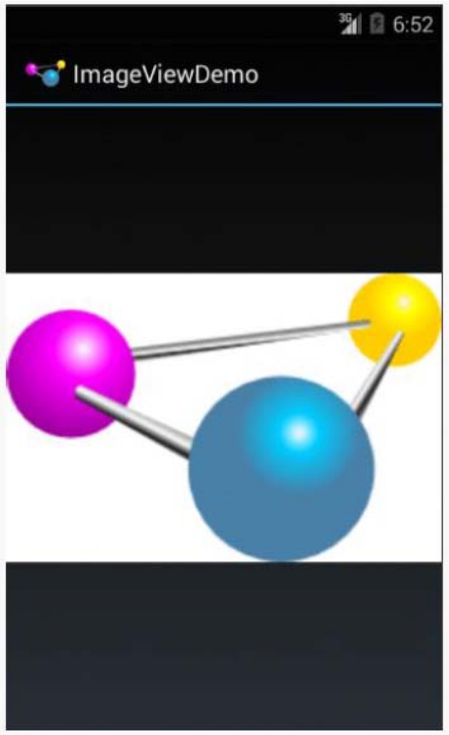
Embedding Images: ImageView and ImageButton

- **ImageView** and **ImageButton:** Image-based based analogs of TextView and Button
  - **ImageView:** display image
  - **ImageButton:** Clickable image
- Use **android:src** to specify image source in **drawable** folder (e.g. **@drawable/icon**)



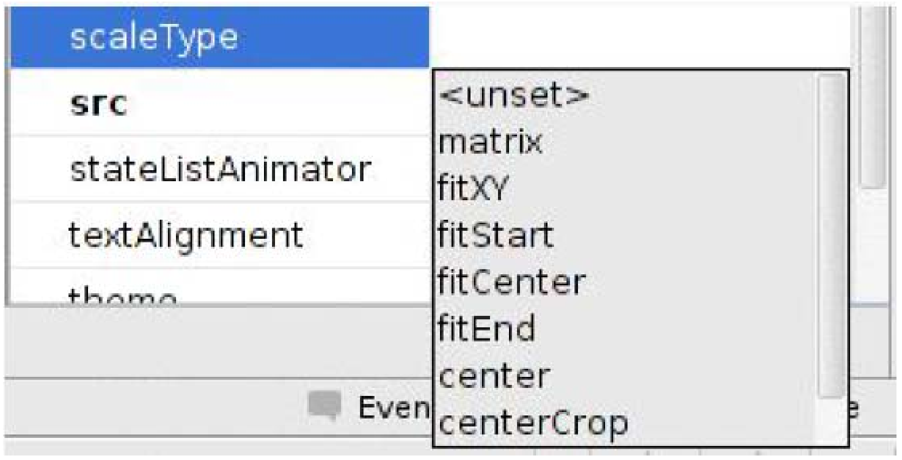
```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/icon"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:adjustViewBounds="true"
  android:src="@drawable/molecule"/>
```

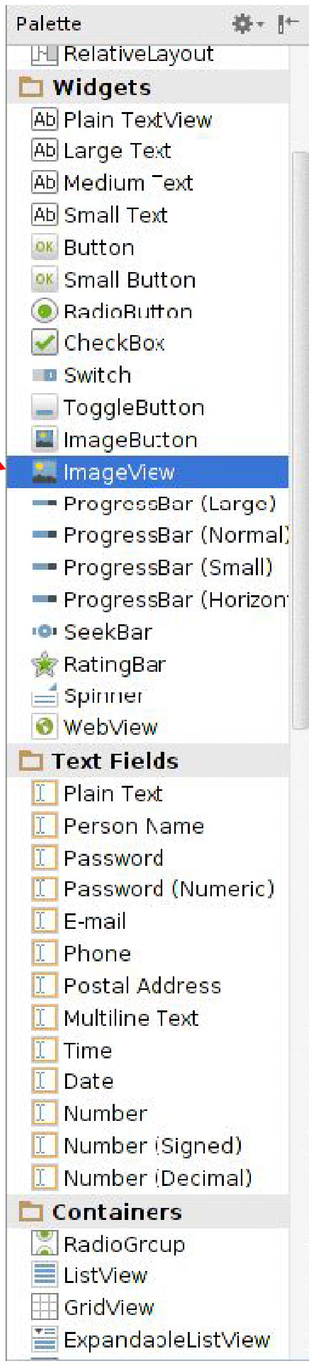
File molecule.png in drawable/ folder



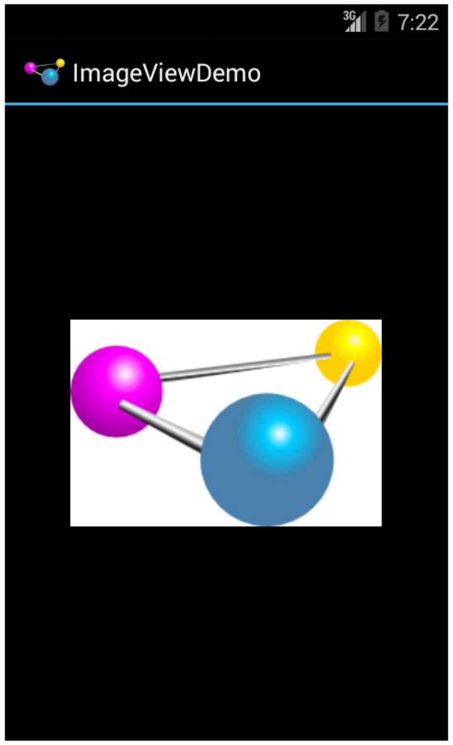
ImageView in Widgets Palette

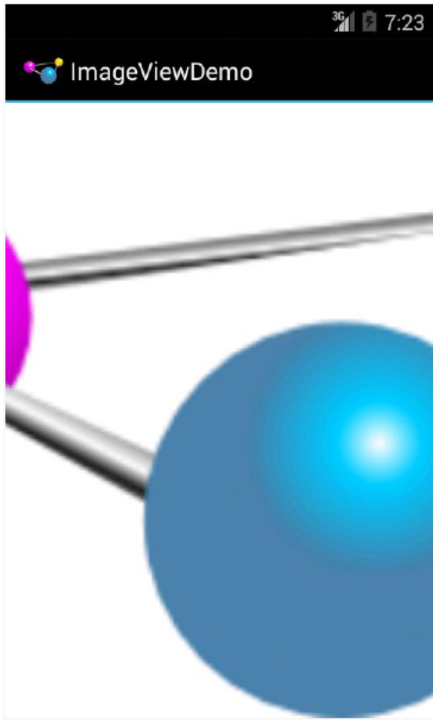
- Can drag and drop ImageView from Widgets Palette
- Can use menus (right-click) to specify:
  - **src:** to choose image to be displayed
  - **scaleType:** to choose how image should be scaled

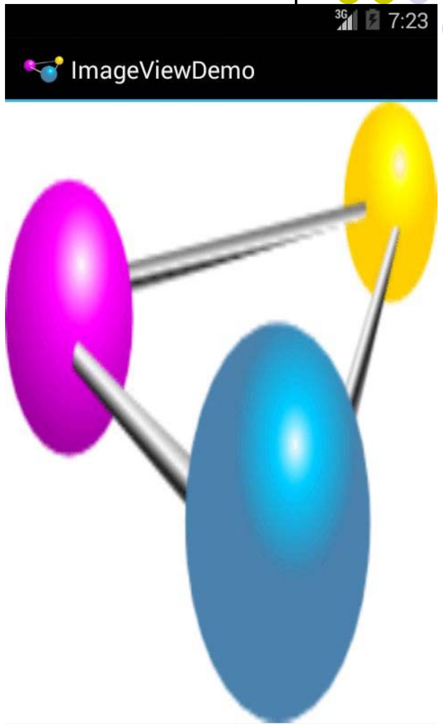




Options for Scaling Images (scaleType)







“center” centers image but does not scale it

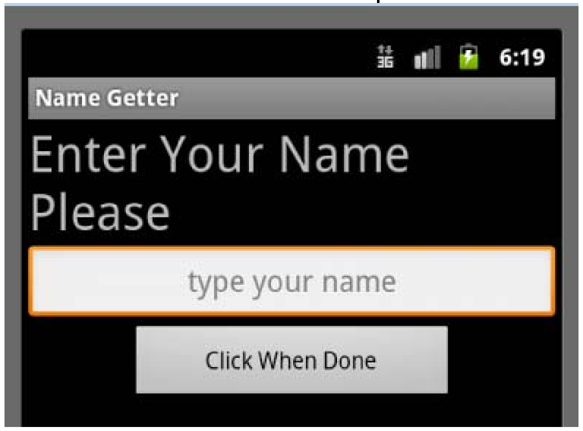
“centerCrop” centers images, scales it so that shortest dimension fills available space, and crops longer dimension

“fitXY” scales image to fit ImageView, ignoring aspect ratio (distorts)

EditText Widget

- UI Component used for user input
- Example:

```
<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" />
```

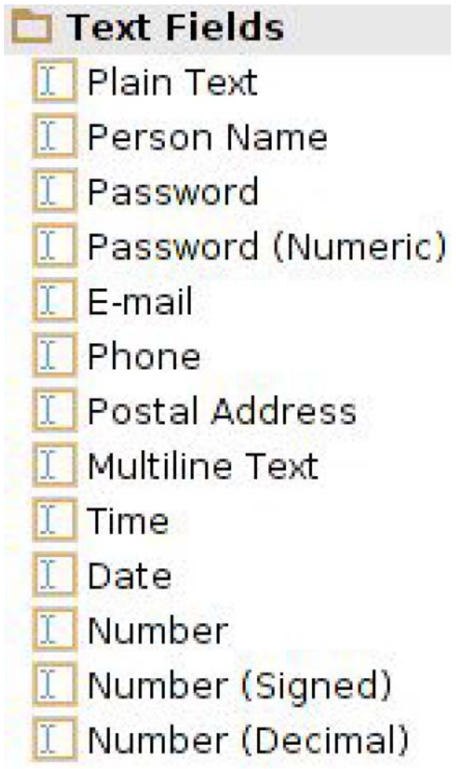


- Text fields can have different input types
  - e.g. number, date, password, or email address

- **android:inputType** attribute sets input type, affects
  - What type of keyboard pops up for user

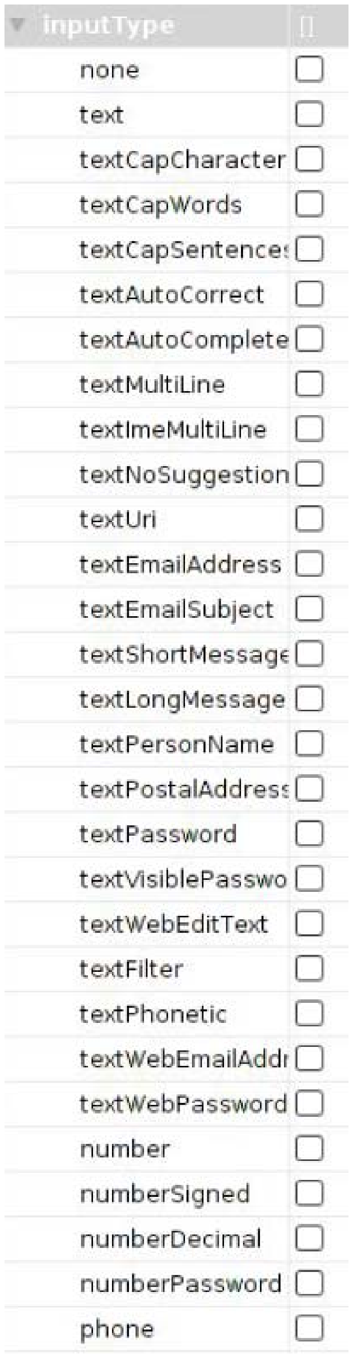
EditText Widget in Android Studio Palette

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



Text Fields
Plain Text
Person Name
Password
Password (Numeric)
E-mail
Phone
Postal Address
Multiline Text
Time
Date
Number
Number (Signed)
Number (Decimal)

Text Fields
Section of Widget palette

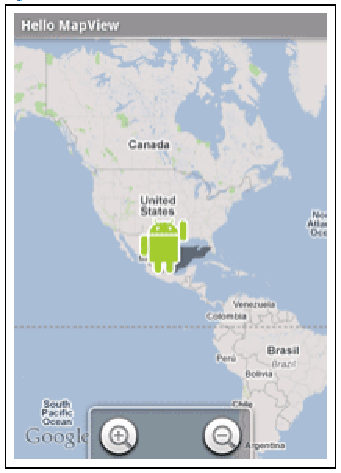


inputType
none
text
textCapCharacter
textCapWords
textCapSentence:
textAutoCorrect
textAutoComplete
textMultiLine
textImeMultiLine
textNoSuggestion
textUri
textEmailAddress
textEmailSubject
textShortMessage
textLongMessage
textPersonName
textPostalAddress
textPassword
textVisiblePasswo
textWebEditText
textFilter
textPhonetic
textWebEmailAddr
textWebPassword
number
numberSigned
numberDecimal
numberPassword
phone

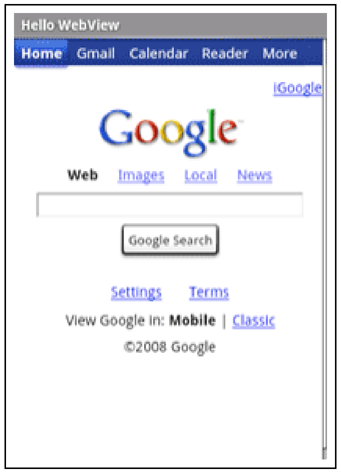
EditText inputType menu

Other Available Widgets

MapView



WebView


Hello WebView

Home Gmail Calendar Reader More

iGoogle

Google

Web Images Local News

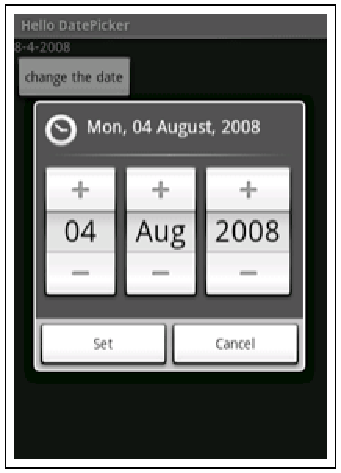
Google Search

Settings Terms

View Google in: **Mobile** | Classic

©2008 Google

DatePicker


Hello DatePicker

8-4-2008

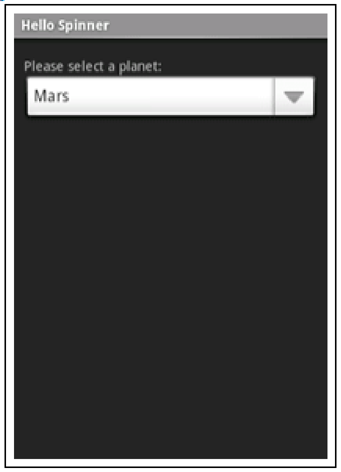
change the date

Mon, 04 August, 2008

04	Aug	2008

Set Cancel

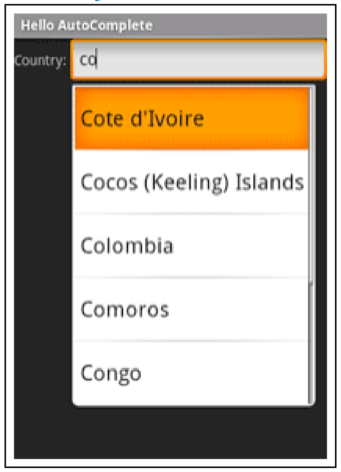
Spinner


Hello Spinner

Please select a planet:

Mars

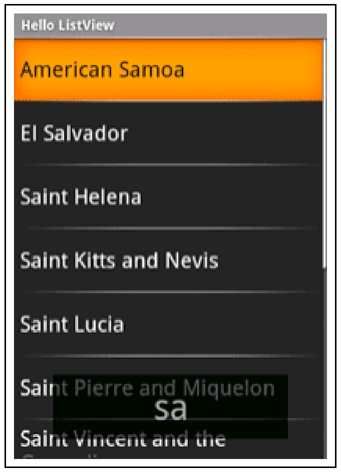
AutoComplete


Hello AutoComplete

Country: co

- Cote d'Ivoire
- Cocos (Keeling) Islands
- Colombia
- Comoros
- Congo

ListView

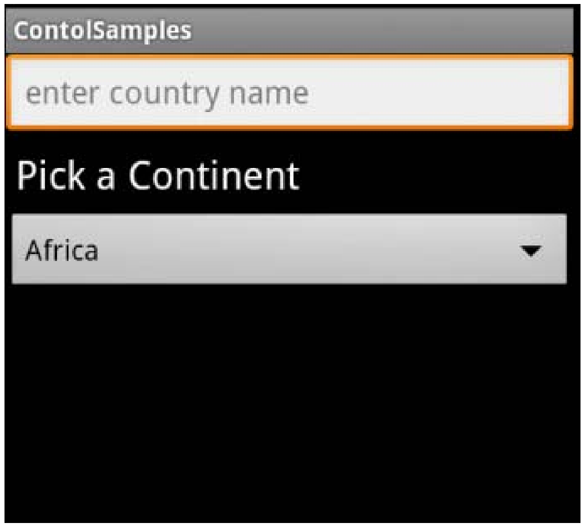

Hello ListView

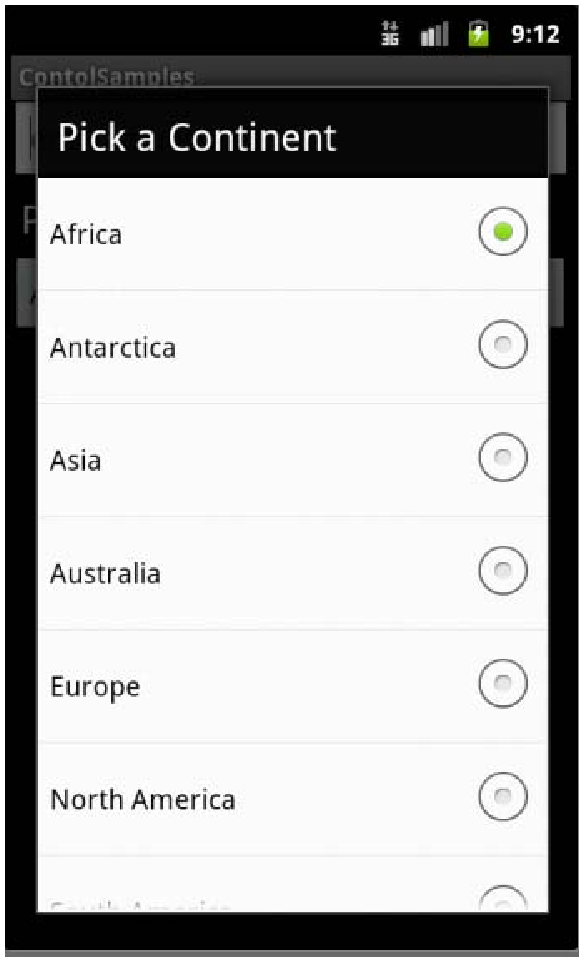
- American Samoa
- El Salvador
- Saint Helena
- Saint Kitts and Nevis
- Saint Lucia
- Saint Pierre and Miquelon
- Saint Vincent and the

sa

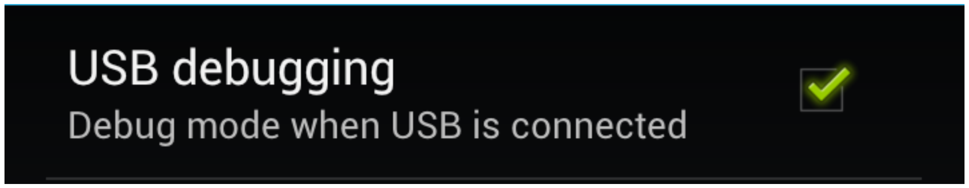
Spinner Controls

- Similar to auto complete, but user **must** select from a set of choices





Checkbox

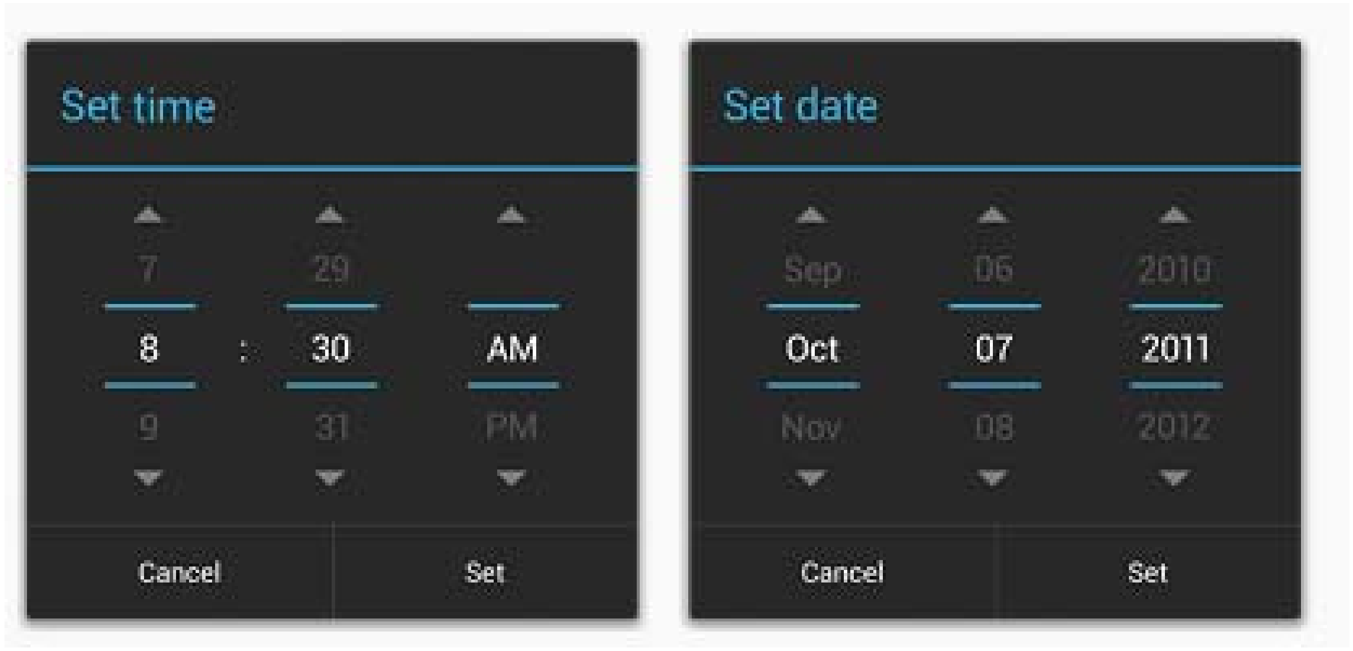

USB debugging
Debug mode when USB is connected

- Checkbox has 2 states: checked and unchecked
- Clicking on checkbox toggles between these 2 states
- Checkbox widget inherits from TextView, so its properties like android:textColor can be used to format checkbox
- 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"/>
```

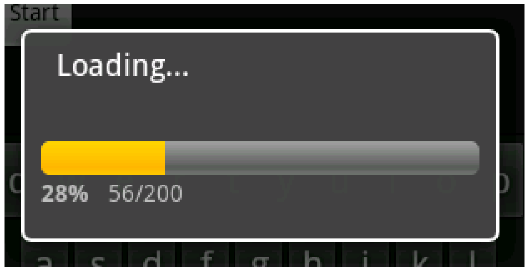
Pickers

- TimePicker and DatePicker
- Typically displayed in a TimePickerDialog or DatePickerDialog
  - Dialogs are small pop-up windows that appear in front of the current activity

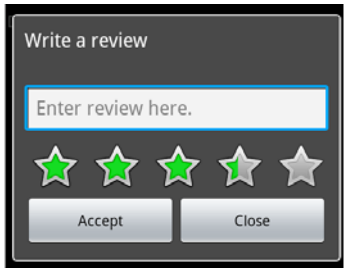


Indicators

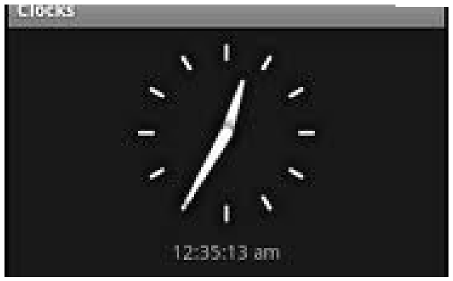
- ProgressBar



- RatingBar



- Chronometer
- DigitalClock
- AnalogClock

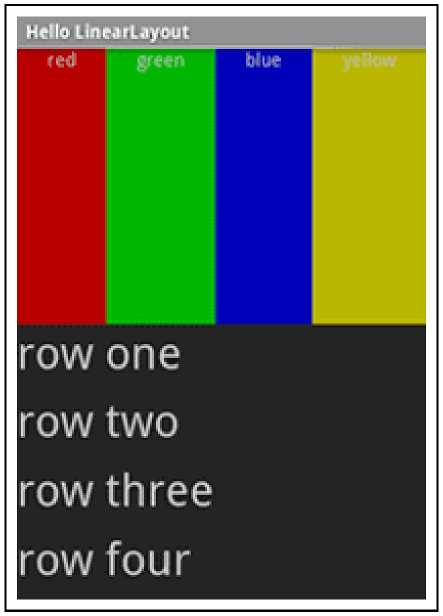


Android Layouts in XML

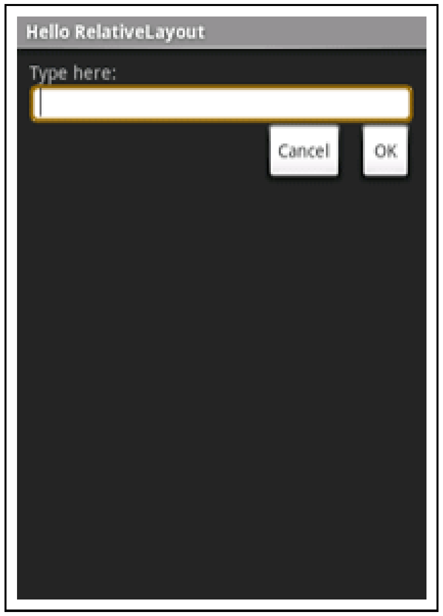
Android UI using XML Layouts

- Layout? Pattern in which multiple widgets are arranged
- In XML layout file, we have to choose a layout to use
- Layouts (XML files) stored in **res/layout**

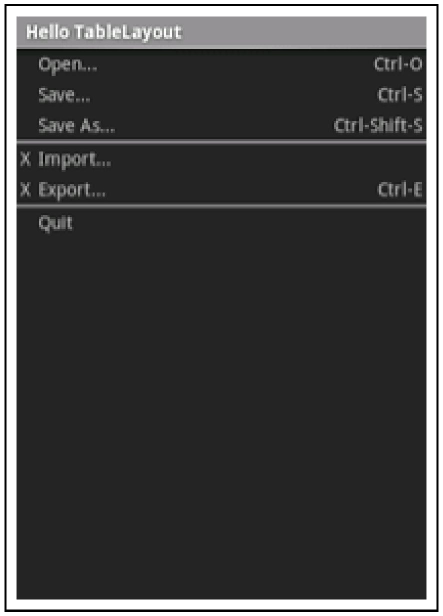
LinearLayout



RelativeLayout



TableLayout

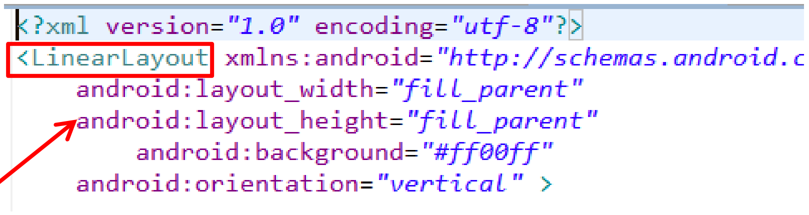


Some Layouts

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

LinearLayout

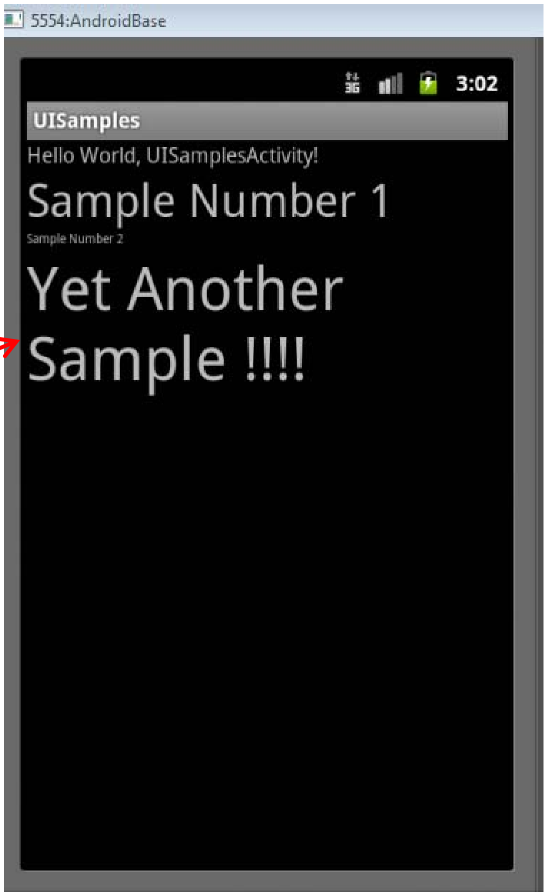
- aligns child elements (e.g. buttons, text boxes, pictures, etc.) in single direction
- Example:



```
<?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" >
```

Layout properties

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



5554:AndroidBase

3:02

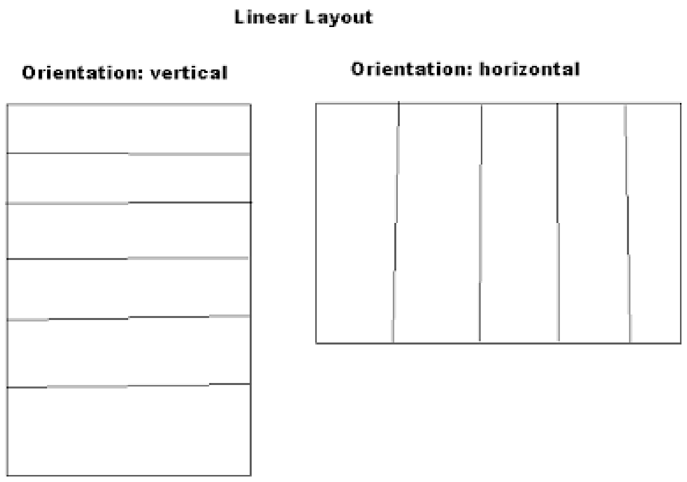
UISamples

Hello World, UISamplesActivity!

Sample Number 1

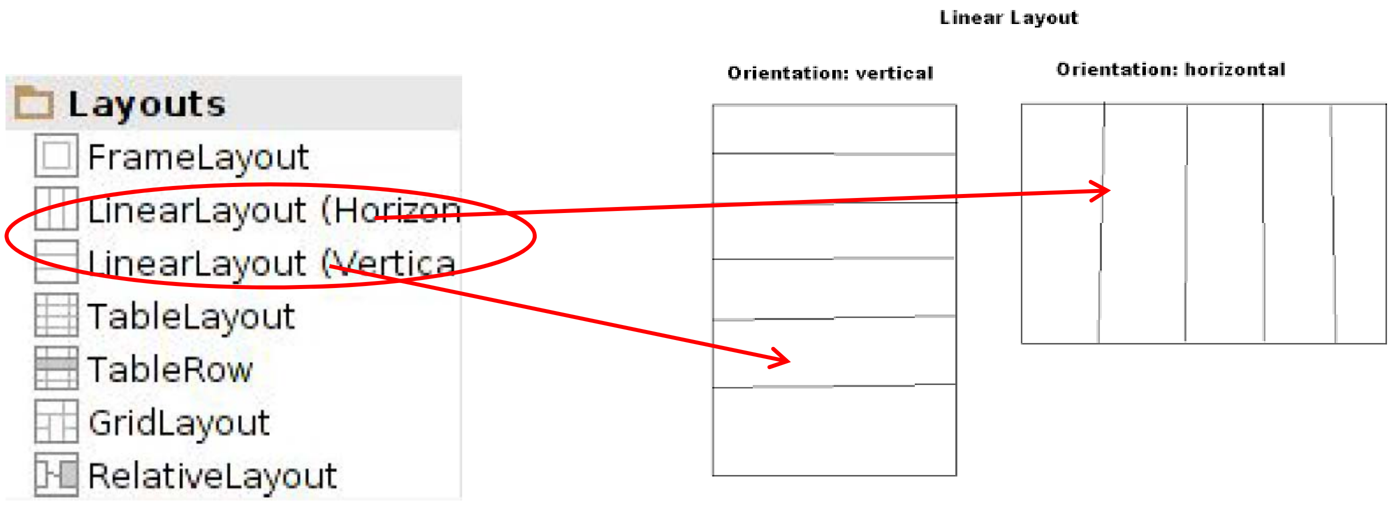
Sample Number 2

Yet Another Sample !!!!

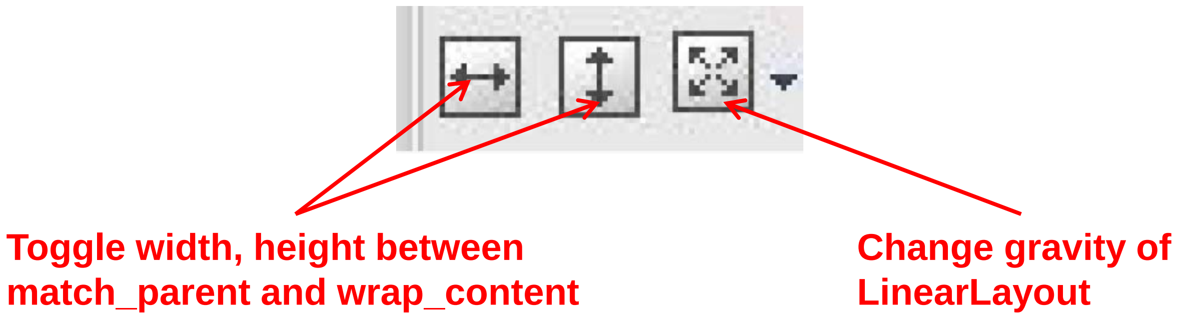


LinearLayout in Android Studio

- LinearLayout in Android Studio Graphical Layout Editor



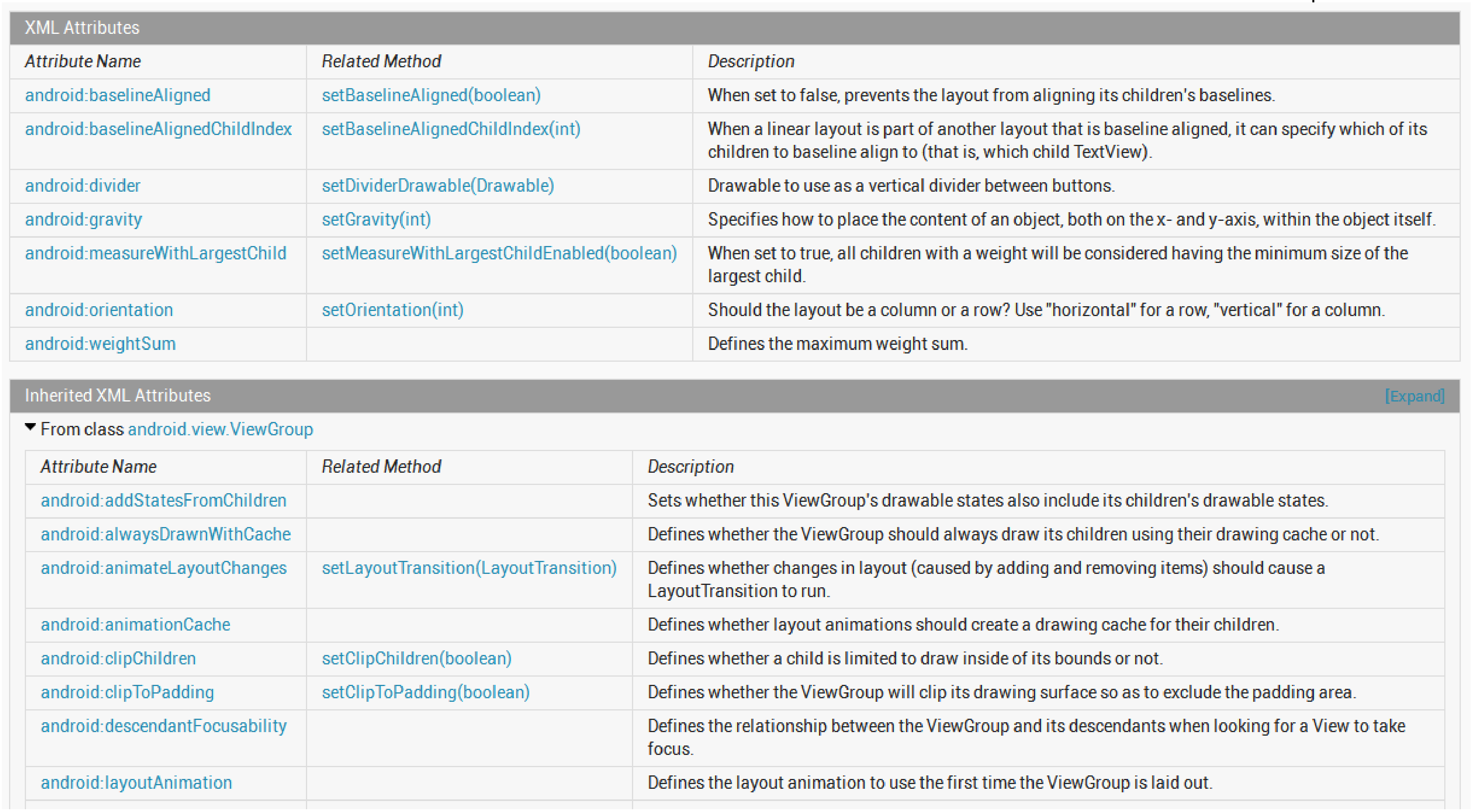
- After selecting LinearLayout, toolbars buttons to set parameters


Toggle width, height between match_parent and wrap_content
Change gravity of LinearLayout

Attributes

- Layouts have attributes (e.g. width, height, orientation)
- E.g. *android:orientation="vertical"*
- Attributes can be set:
  - In xml file
  - Using IDE (e.g. Android Studio)
  - In Java program
- Lots of attributes!

LinearLayout Attributes



XML Attributes

Attribute Name	Related Method	Description
android:baselineAligned	setBaselineAligned(boolean)	When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	setBaselineAlignedChildIndex(int)	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	setDividerDrawable(Drawable)	Drawable to use as a vertical divider between buttons.
android:gravity	setGravity(int)	Specifies how to place the content of an object, both on the x- and y-axis, within the object itself.
android:measureWithLargestChild	setMeasureWithLargestChildEnabled(boolean)	When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation	setOrientation(int)	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.

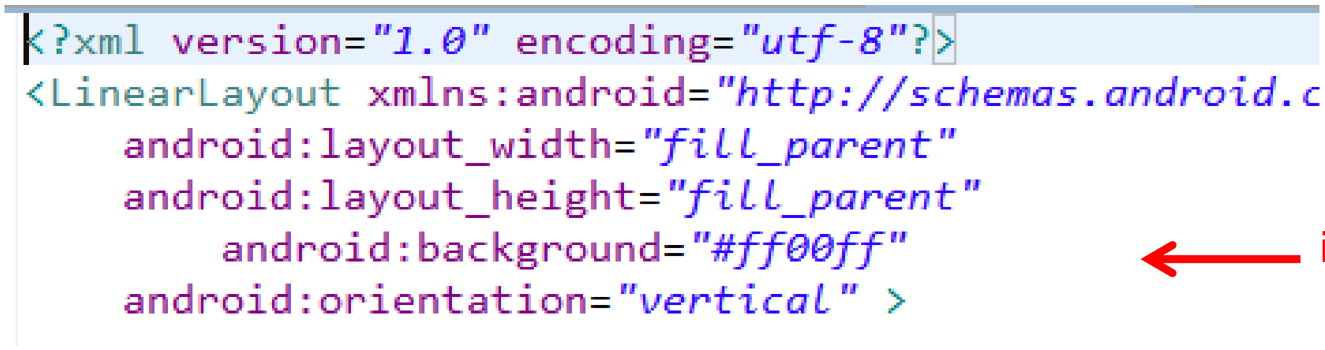
Inherited XML Attributes [Expand]

▼ From class android.view.ViewGroup

Attribute Name	Related Method	Description
android:addStatesFromChildren		Sets whether this ViewGroup's drawable states also include its children's drawable states.
android:alwaysDrawnWithCache		Defines whether the ViewGroup should always draw its children using their drawing cache or not.
android:animateLayoutChanges	setLayoutTransition(LayoutTransition)	Defines whether changes in layout (caused by adding and removing items) should cause a LayoutTransition to run.
android:animationCache		Defines whether layout animations should create a drawing cache for their children.
android:clipChildren	setClipChildren(boolean)	Defines whether a child is limited to draw inside of its bounds or not.
android:clipToPadding	setClipToPadding(boolean)	Defines whether the ViewGroup will clip its drawing surface so as to exclude the padding area.
android:descendantFocusability		Defines the relationship between the ViewGroup and its descendants when looking for a View to take focus.
android:layoutAnimation		Defines the layout animation to use the first time the ViewGroup is laid out.

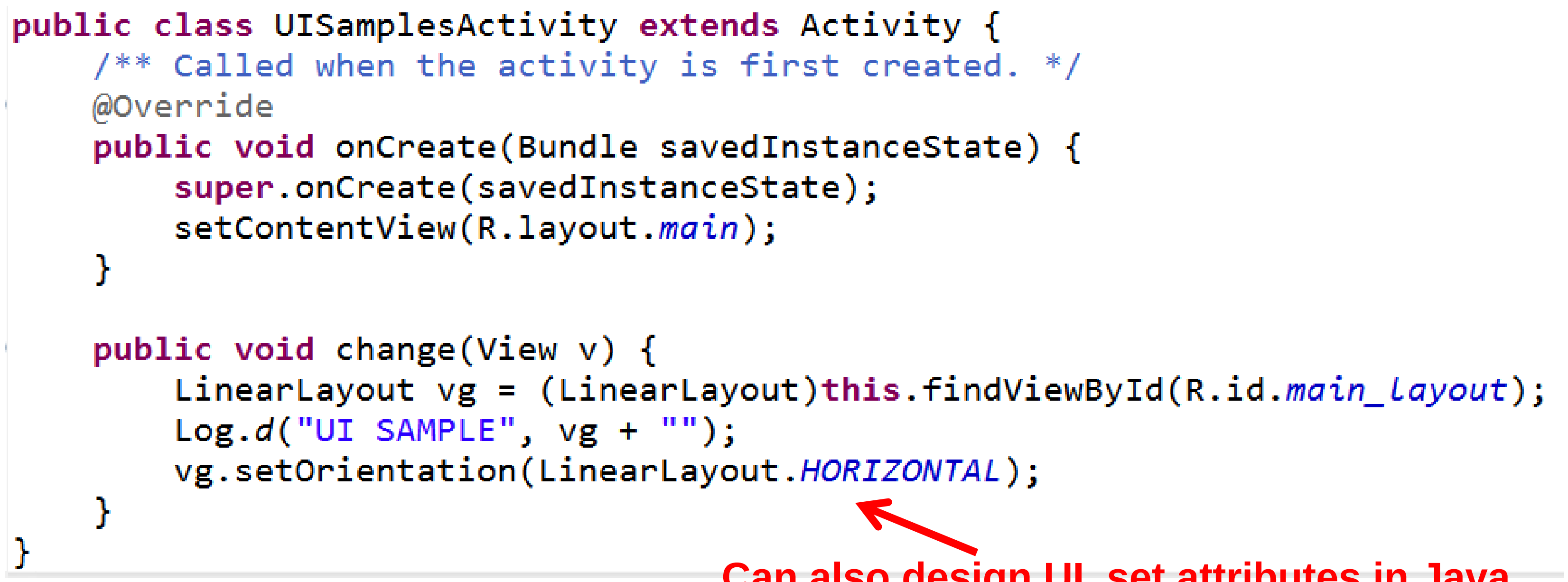
Can find complete list of attributes, possible values on Android Developer website

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

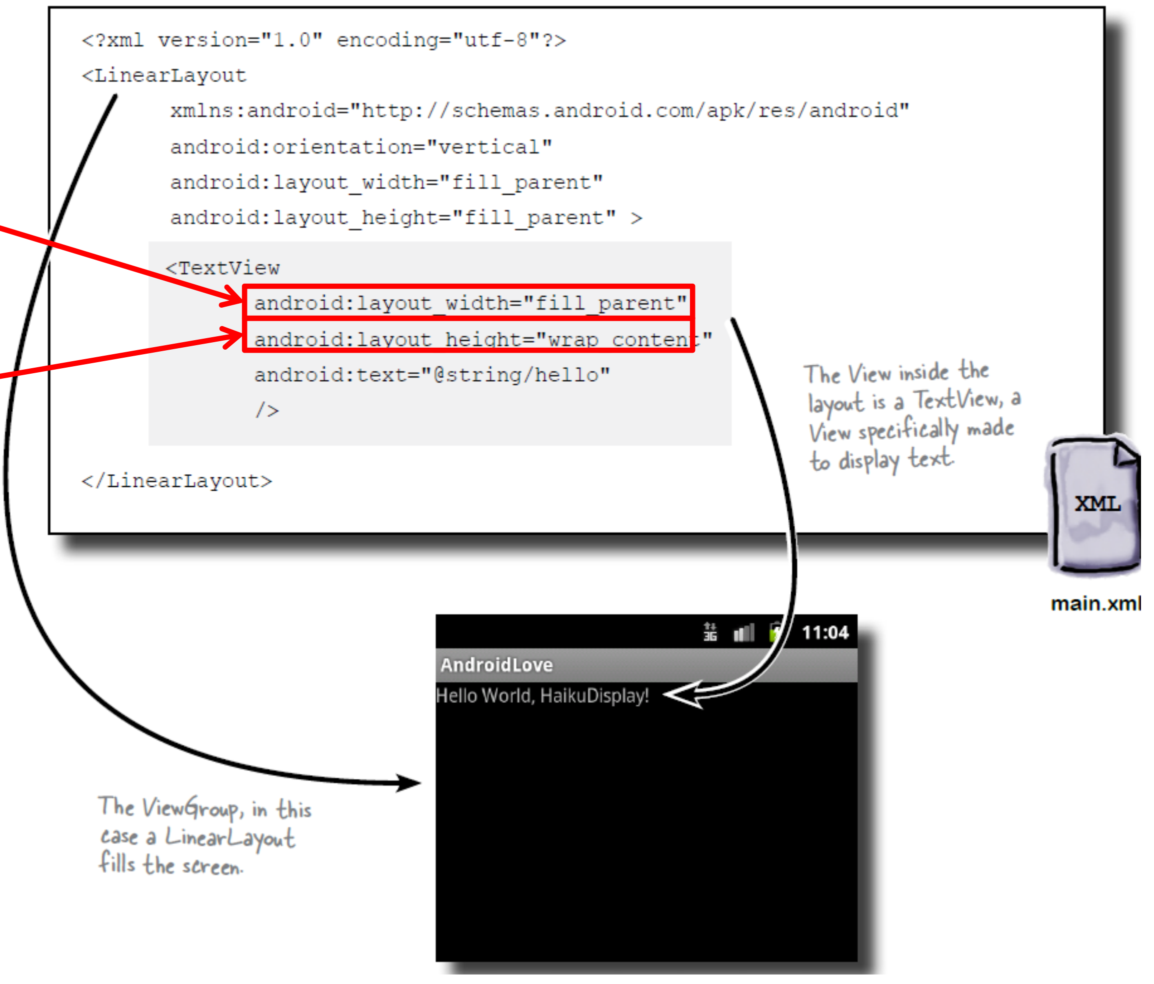


```
public class UISamplesActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    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)

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**



```
<?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.
XML
main.xml
11:04
AndroidLove
Hello World, HaikuDisplay!
The ViewGroup, in this case a LinearLayout fills the screen.

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

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

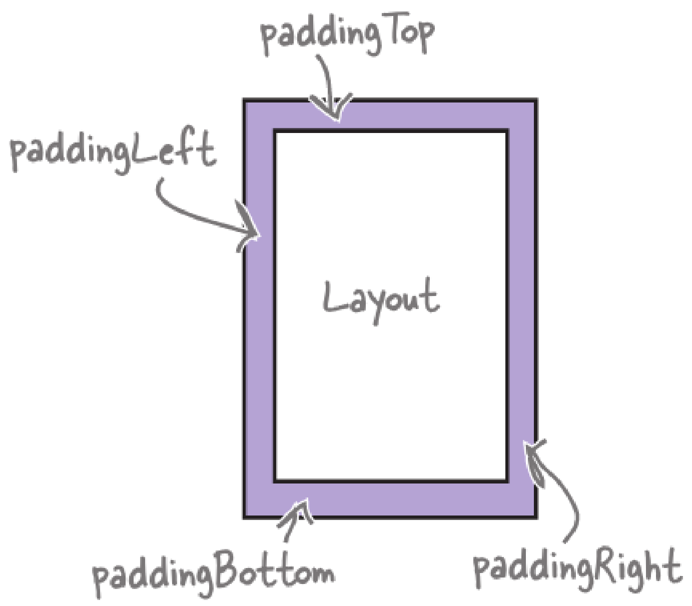
Screen (Hardware)
↑
Linear Layout
↑
TextView

Adding Padding

- Paddings sets space between layout sides and its parent

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

Add padding of 16dp.

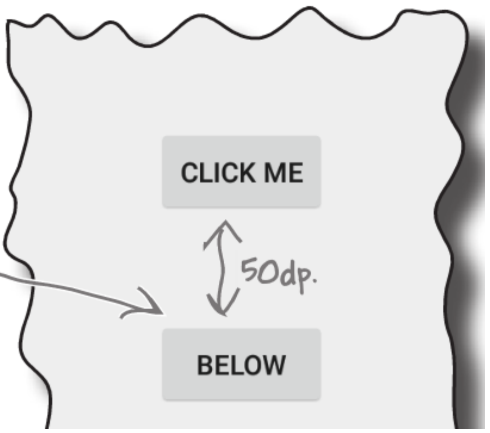


Setting Margins

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

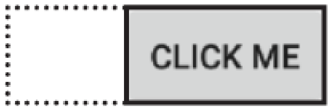
```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button_click_me"
        android:layout_below="@+id/button_click_me"
        android:layout_marginTop="50dp"
        android:text="@string/button_below" />
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

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

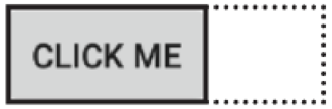


- Other options

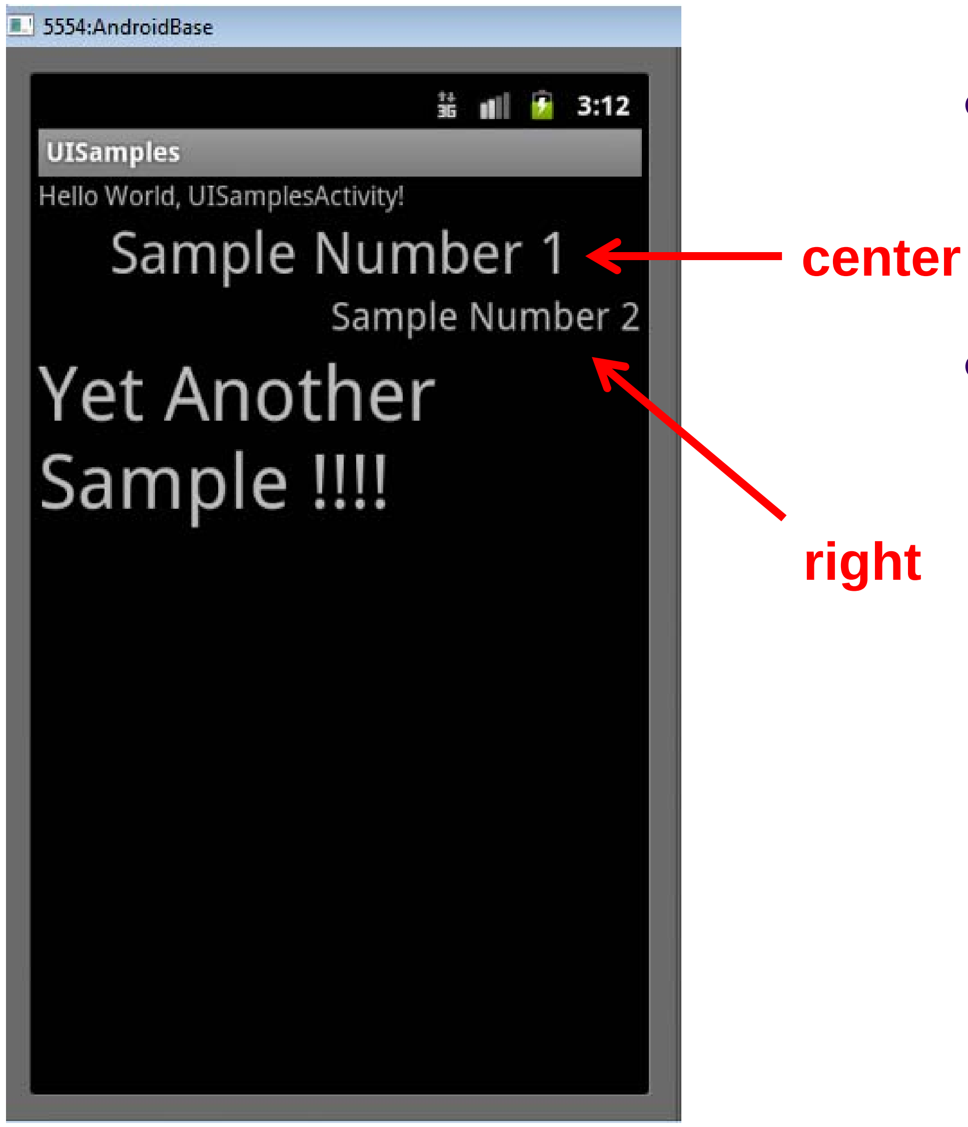
android:layout_marginLeft



android:layout_marginRight



Gravity Attribute


5554:AndroidBase

3:12

UISamples

Hello World, UISamplesActivity!

Sample Number 1

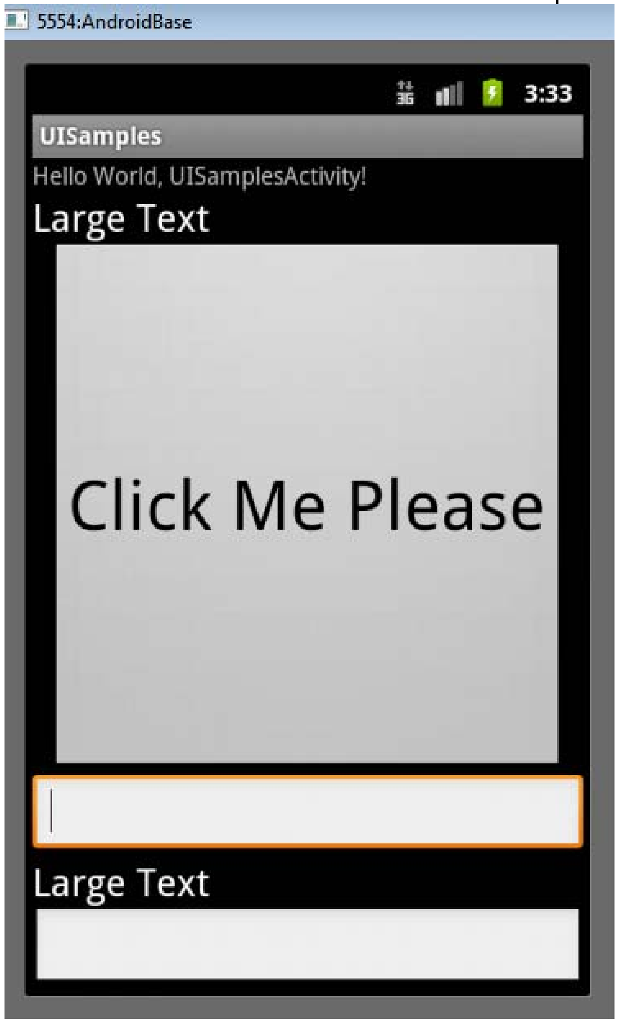
Sample Number 2

Yet Another Sample !!!!

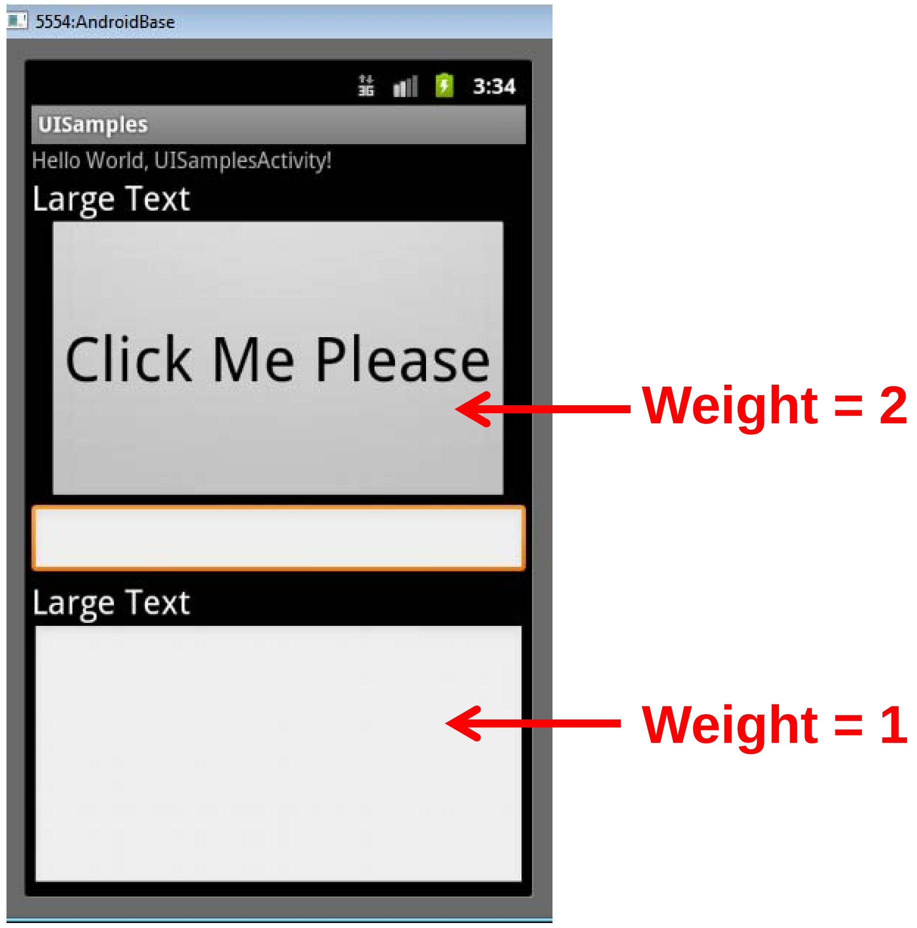
- By default, linearlayout left- and top-aligned
- Gravity attribute can change alignment :
  - e.g. android:gravity = “right”

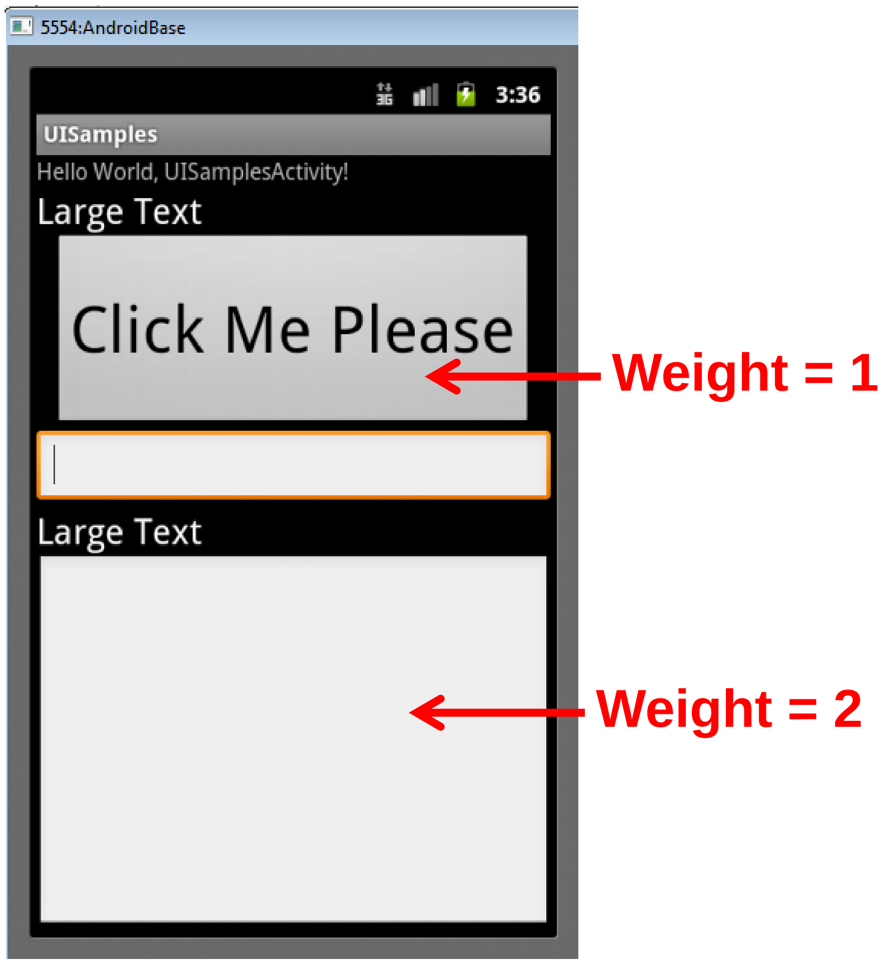
Layout Weight Attribute

- layout_weight attribute
  - Specifies "importance" of a view (i.e. button, text, etc)
  - Default = 0
  - Larger weights (layout_weight > 0) takes up more space



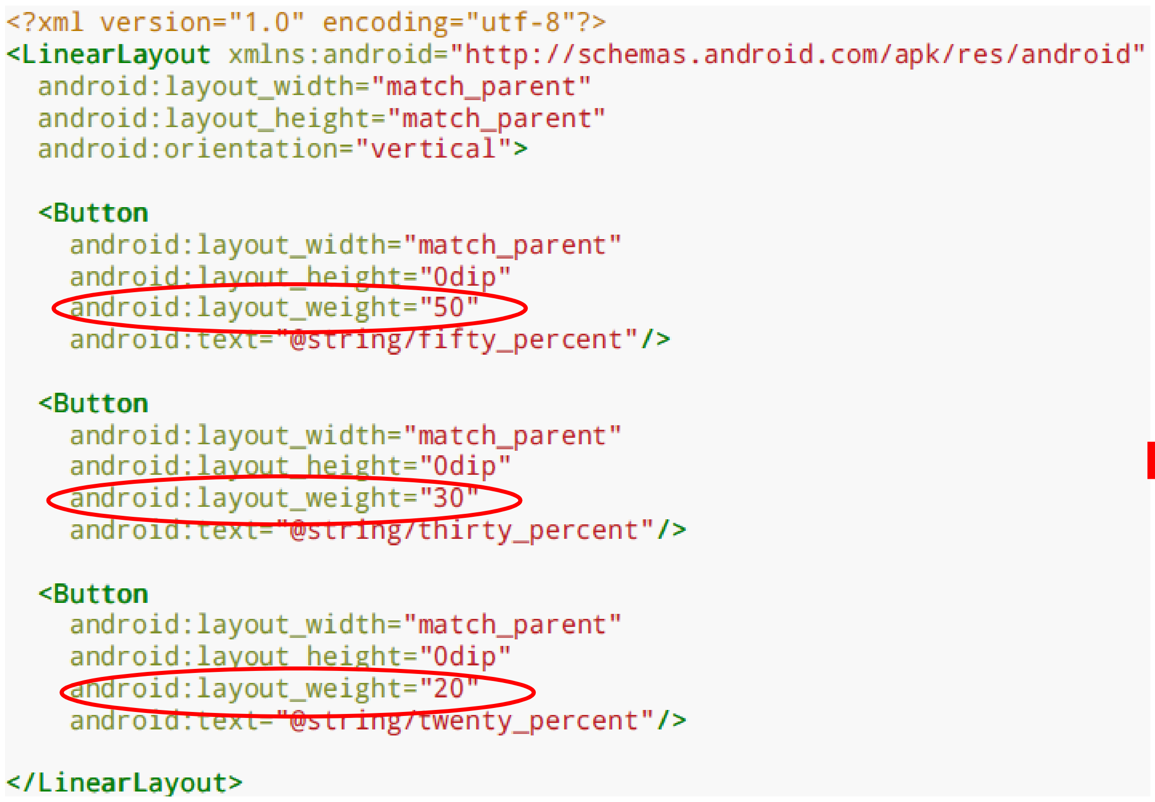
Another Weight Example



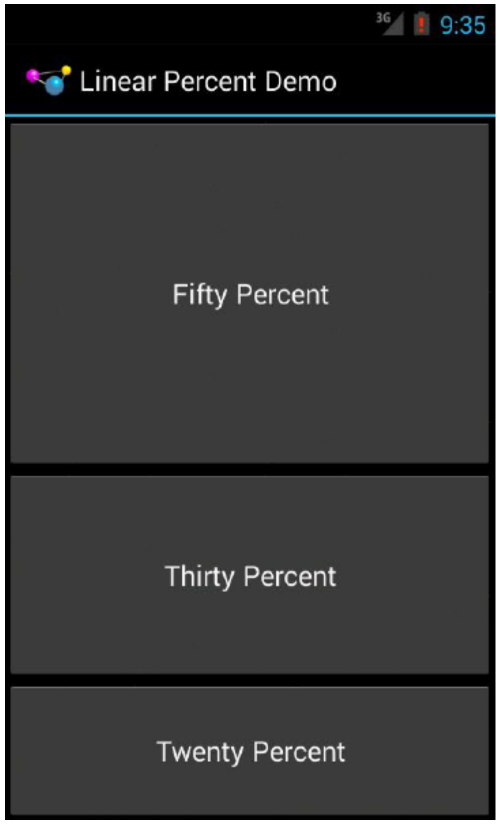


Linear Layout

- Can set width, height = 0 then
  - weight = percent of height/width you want element to cover



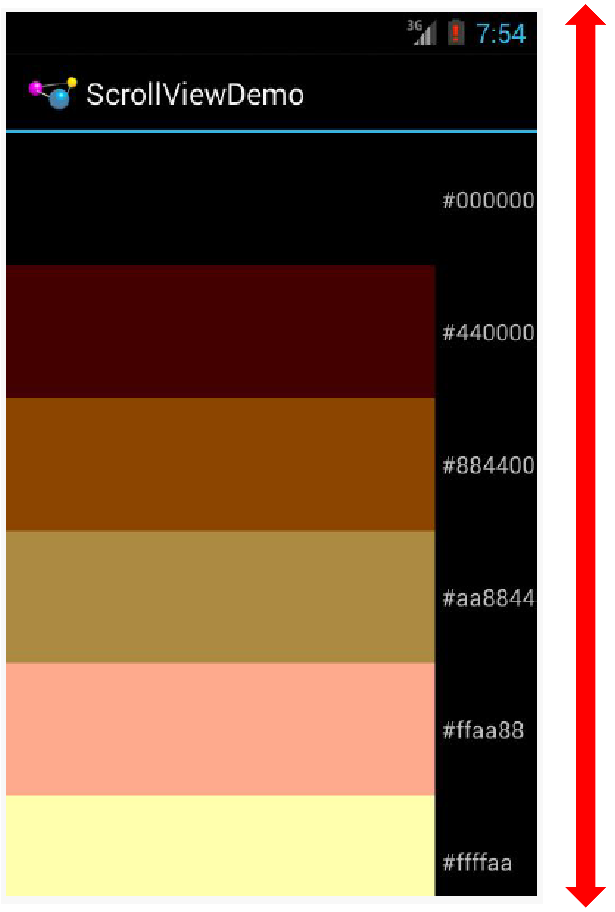
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <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>
```



Scrolling

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

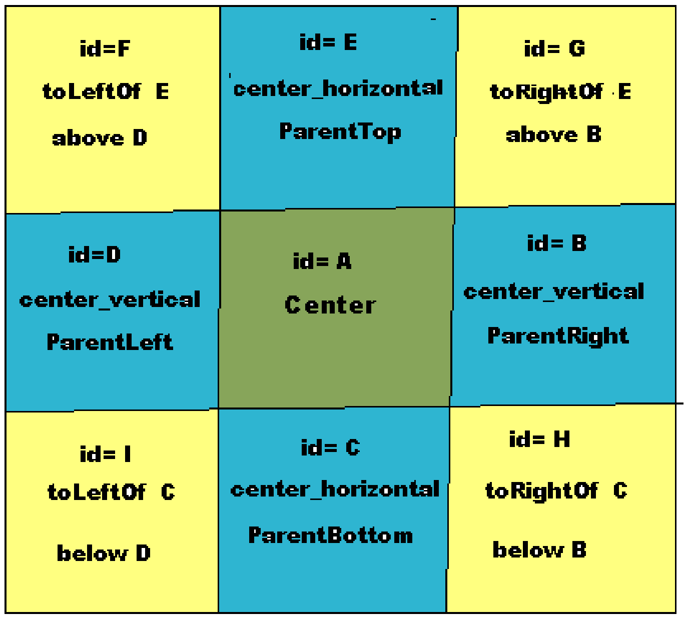
```
<ScrollView
    ...>
    <LinearLayout>
       ....
       <!-- you can have as many Views in here as you want -->
    </LinearLayout>
</ScrollView>
```

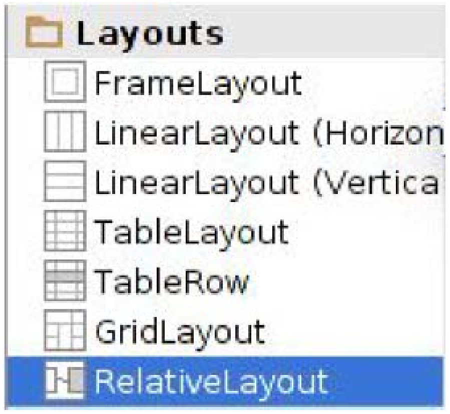


RelativeLayout

- First element listed is placed in "center"
- Positions of children specified relative to parent or to each other.

Relative Layout

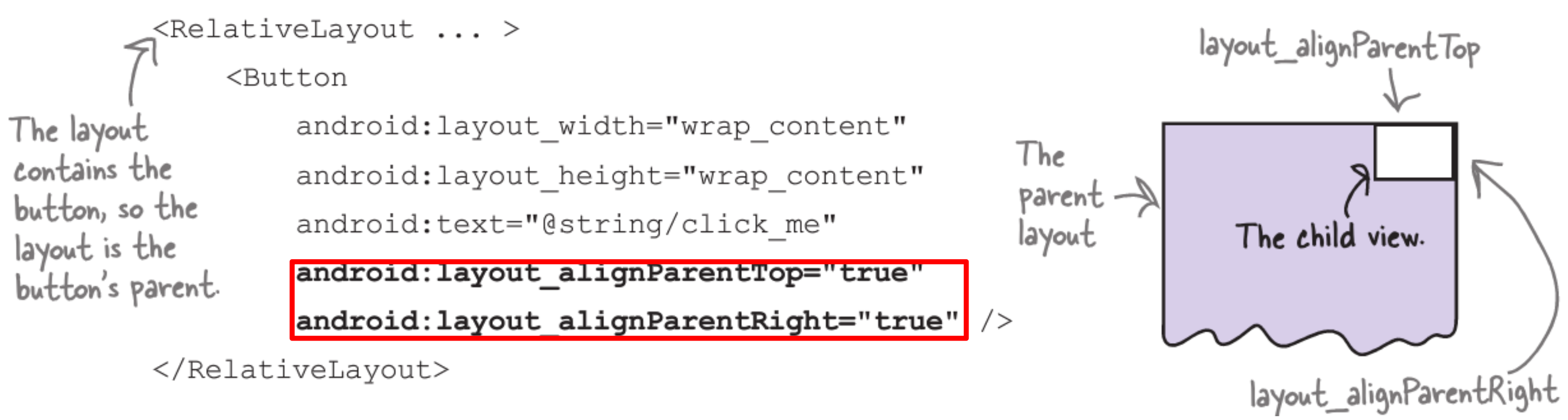




**RelativeLayout available
In Android Studio palette**

Positioning Views Relative to Parent Layout

- Can position a view (e.g. button, TextView) relative to its parent
- Example: Button aligned to top, right in a Relative Layout



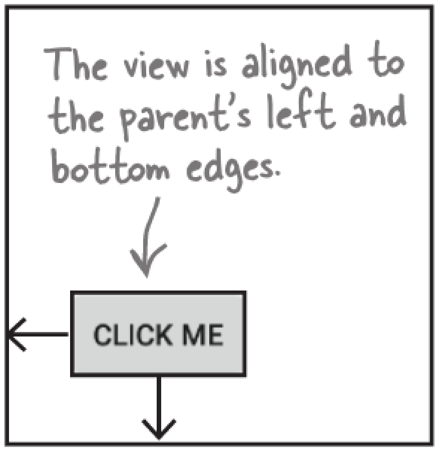
```
<RelativeLayout ... >
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/click_me"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true" />
</RelativeLayout>
```

Examples: Positioning a Button Relative to Parent Layout

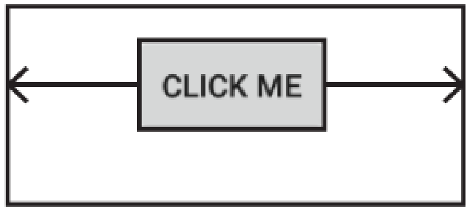
- Align to parent bottom and left

android:
layout_alignParentBottom

android:
layout_alignParentLeft



android:
layout_centerHorizontal



See Head First Android Development page 169 for more examples

Positioning Views Relative to Other Views

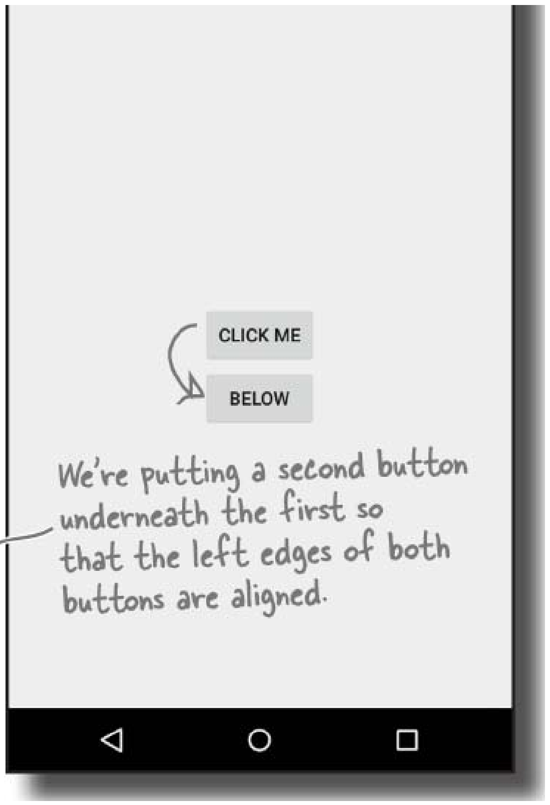
- The anchor view has to be assigned an ID using **android:id**
- **E.g.** Relative layout with 2 buttons (1 centered in layout middle, second button underneath first button)

Assign anchor button an ID

Align second button with first button's left and below it

We're using this button as an anchor for the second one, so it needs an ID.

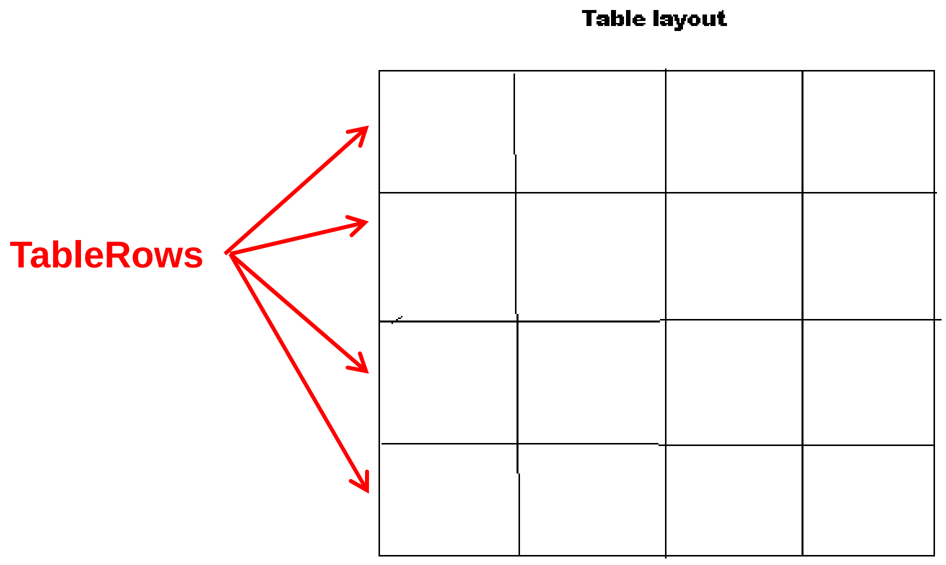
```
<RelativeLayout ... >
    <Button
        android:id="@+id/button_click_me"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="@string/click_me" />

    <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:text="@string/new_button_text" />
</RelativeLayout>
```

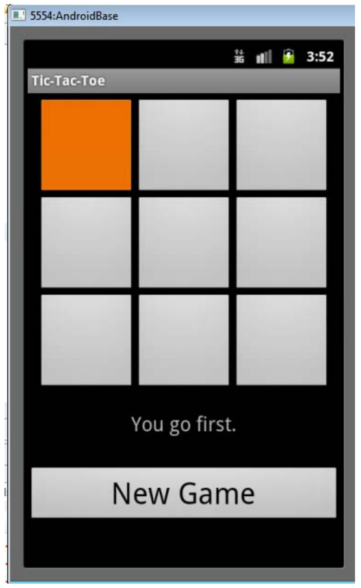


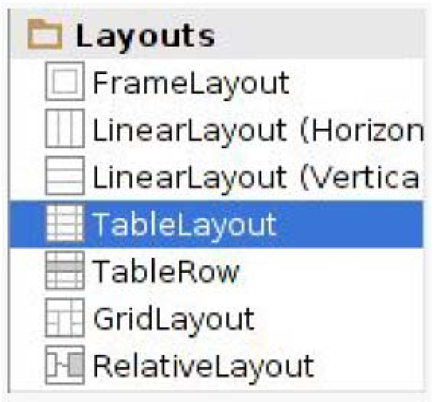
We're putting a second button underneath the first so that the left edges of both buttons are aligned.

Table Layout

- Specify number of rows and columns of views.
- Available in Android Studio palette

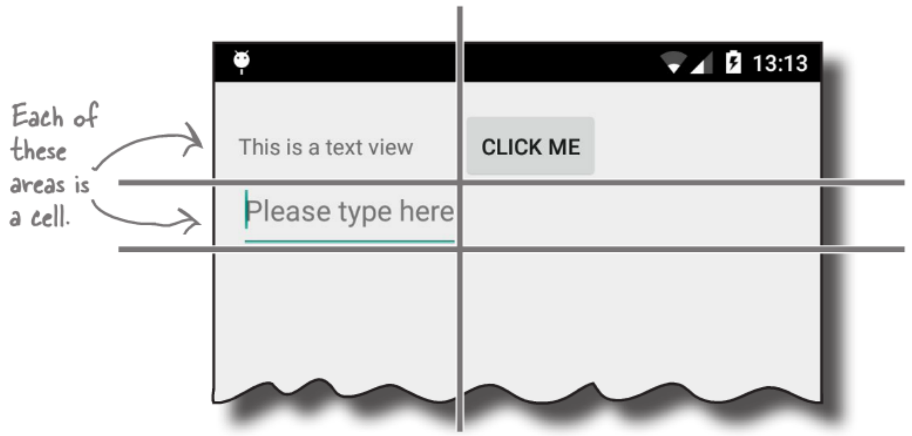






GridLayout

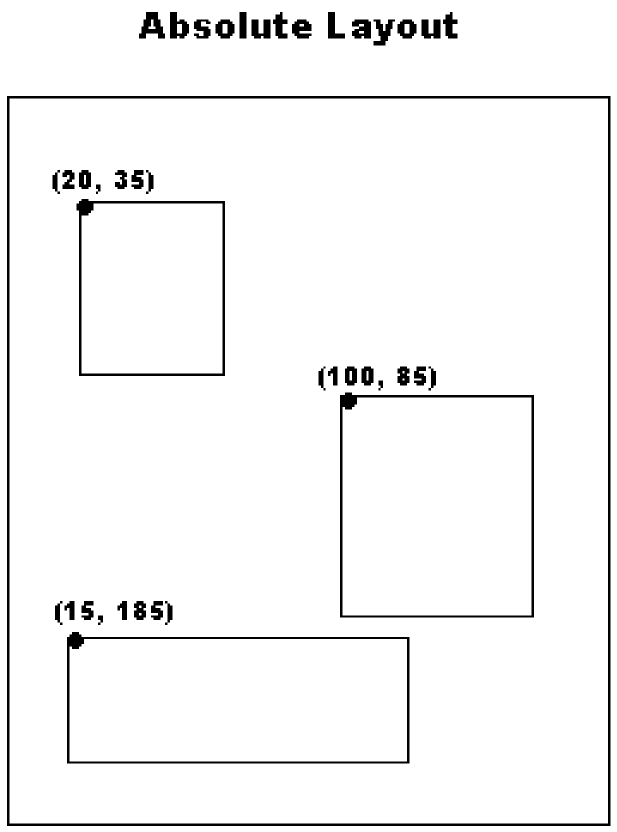
- Added in Android 4.0 (2011)
- In TableLayout, Rows can span multiple columns only
- In GridLayout, child views/controls can span multiple rows **AND** columns
  - different from TableLayout
- Gives greater design flexibility



- See section “GridLayout Displays Views in a Grid” in Head First Android Development (pg 189)

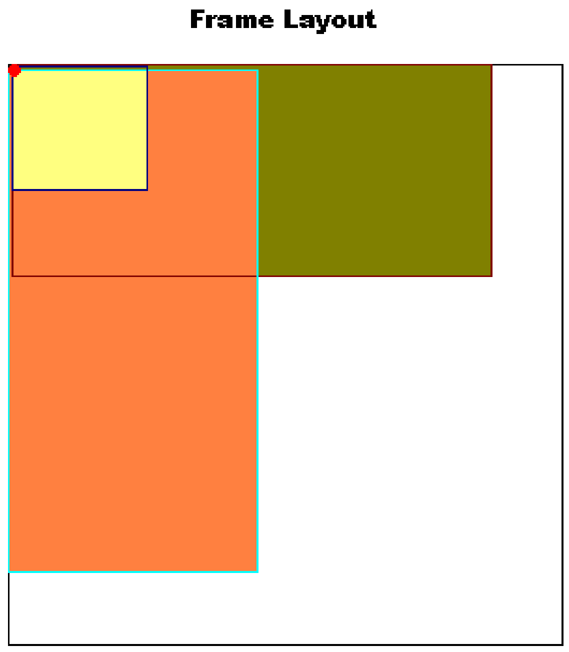
Absolute Layout

- Allows specification of exact x,y coordinates of layout's children.
- Less flexible, harder to maintain

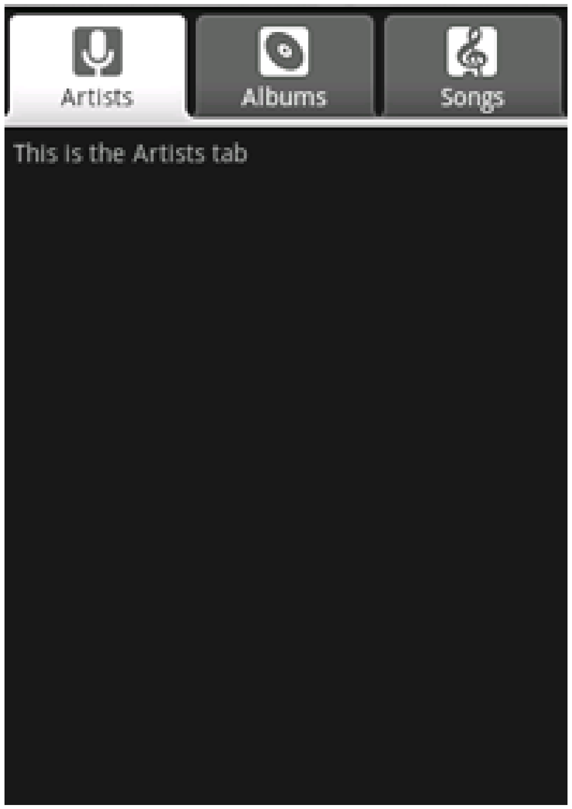


FrameLayout

- FrameLayout
  - simplest type of layout object
  - fill with single object (e.g a picture)
  - child elements pinned to top left corner of screen, cannot be moved
  - adding a new element / child draws over the last one



Other Layouts - Tabbed Layouts



This is the Artists tab

Android UI Youtube Tutorials

Tutorial 11: Designing the User Interface

- Tutorial 11: Designing the User Interface [6:19 mins]
  - https://www.youtube.com/watch?v=72mf0rmjNAA

- Main Topics
  - Designing the User interface
  - Manually adding activity
  - Dragging in widgets
  - Changing the text in widgets

Tutorial 12: More on User Interface

- Tutorial 12: More on User Interface [10:24 mins]
  - https://www.youtube.com/watch?v=72mf0rmjNAA

- Main Topics
  - Changing text in widgets
  - Changing strings from hardcoded to resources (variables)

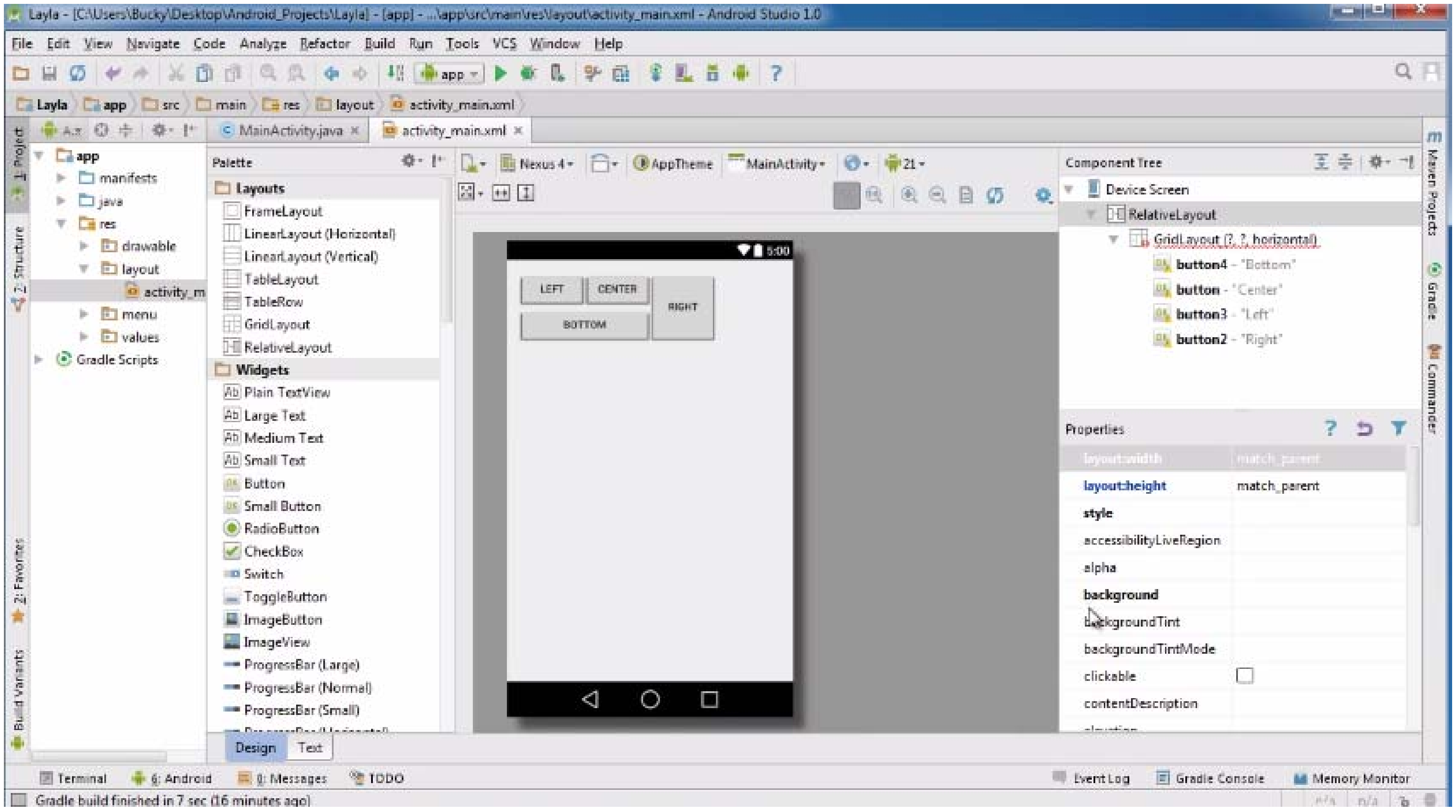
Tutorial 17: GridLayout

- Tutorial 17: GridLayout [9:40 mins]
  - https://www.youtube.com/watch?v=4bXOr5Rk1dk

- Main Topics
  - Creating GridLayout: Layout that places its children in a grid
  - Add widgets (buttons) to GridLayout
  - Format width, height, position of widgets

Create Grid Layout, Add & Format Widgets

- Add widgets (buttons) to GridLayout
- Format width, height, position of widgets


Layla - [C:\Users\Bucky\Desktop\Android_Projects\Layla] - [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
Layla app src main res layout activity_main.xml
MainActivity.java activity_main.xml
app manifests java res drawable layout activity_m menu values Gradle Scripts
Palette
Layouts FrameLayout LinearLayout (Horizontal) LinearLayout (Vertical) TableLayout TableRow GridLayout RelativeLayout
Widgets Plain TextView Large Text Medium Text Small Text Button Small Button RadioButton CheckBox Switch ToggleButton ImageButton ImageView ProgressBar (Large) ProgressBar (Normal) ProgressBar (Small)
Nexus 4 AppTheme MainActivity 21
5:00
LEFT CENTER RIGHT BOTTOM
Design Text
Terminal 6: Android 0: Messages TODO Event Log Gradle Console Memory Monitor

Component Tree

- Device Screen
  - RelativeLayout
    - GridLayout (?, ?, horizontal)
      - button4 - "Bottom"
      - button - "Center"
      - button3 - "Left"
      - button2 - "Right"

Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	☐
contentDescription	

Gradle build finished in 7 sec (16 minutes ago)

References

- Busy Coder’s guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014
- Android App Development for Beginners videos by Bucky Roberts (thenewboston)