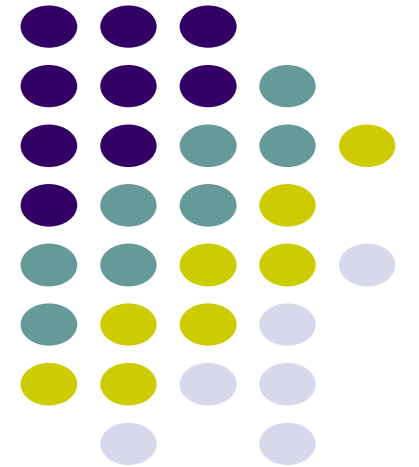


CS 528 Mobile and Ubiquitous Computing

Lecture 02a: Android UI Design

Emmanuel Agu



Quiz 1

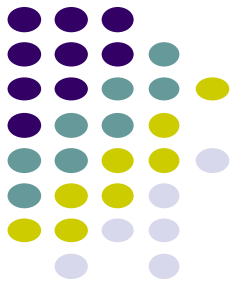
- No class Next week (Sept 7), Monday class schedule
- Quiz next class (Sept 14) at start of class
- Short/multiple choice questions
- Try to focus on understanding, not memorization
- Covers:
 - Lecture slides for lectures 1-2 (including today)
 - 3 code examples from books
 - **HFAD examples:** myFirstApp, Beer Advisor
 - **ANR example:** geoQuiz





Groups for Projects 2,3 and Final Project

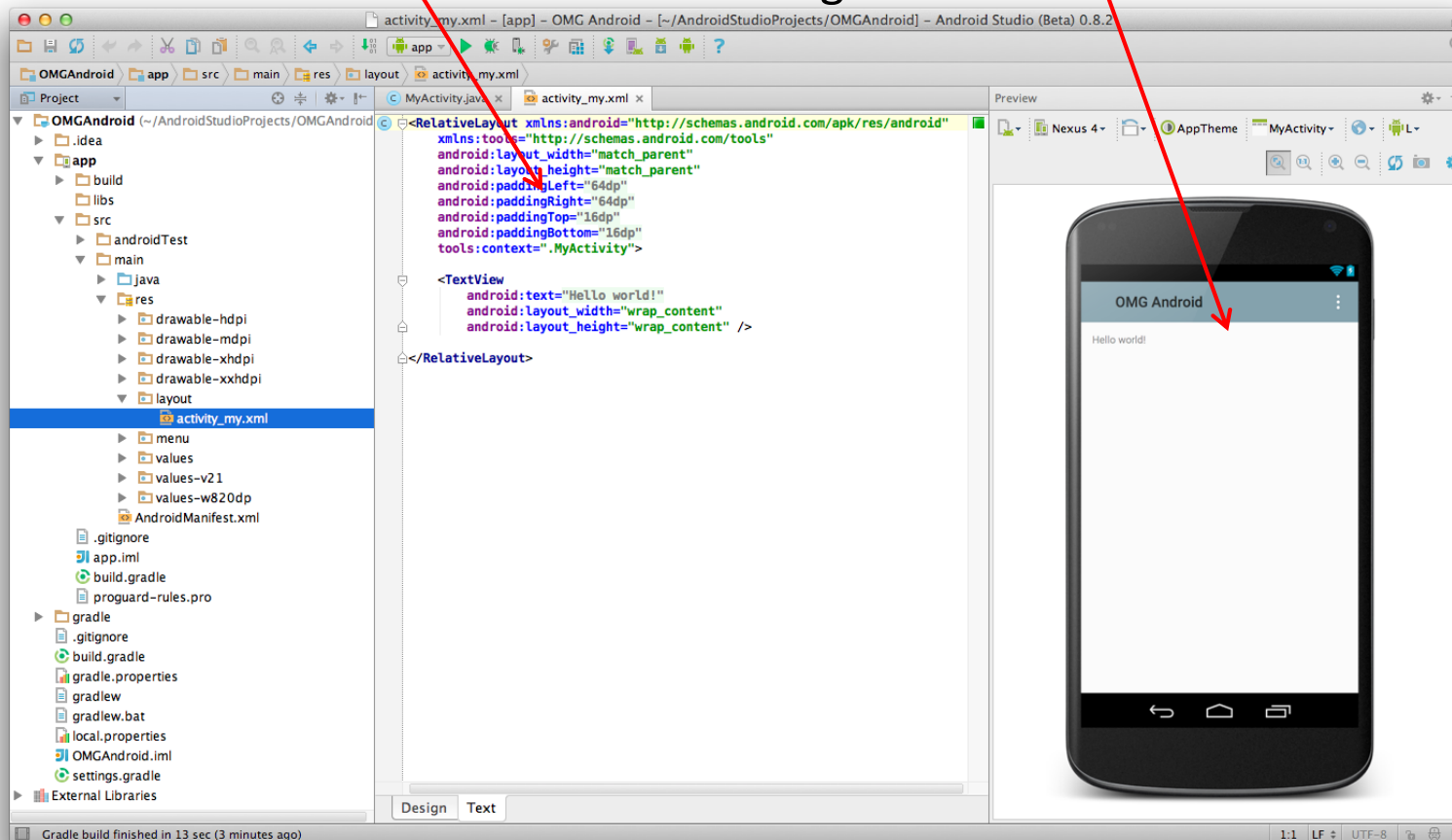
- Projects 2,3 and final project done in teams of 5!
- Deadline to form groups: by class time on Sept 21
- ALL members of the group should email me indicating their group
 - List all team members in their email to me
 - I will put students who don't have groups into groups
- I will posted list of all students in class, also set up Canvas Forum for discussion
- Q: How many students will likely form their own groups? Want to be put in teams?



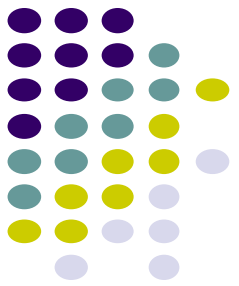
Editing in Android Studio

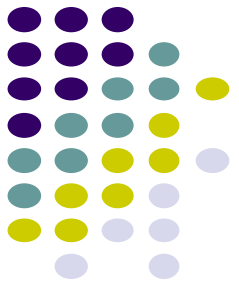
Recall: Editing Android

- Can edit apps in:
 - **Text View:** edit XML directly
 - **Design View:** or drag and drop widgets unto emulated phone
 - **Split View:** Combines both Text View and Design View in one screen



Note: Your Android Studio screen may Be different, depending On version





Android UI Design in XML

Recall: Files Hello World Android Project

XML file used to design Android UI

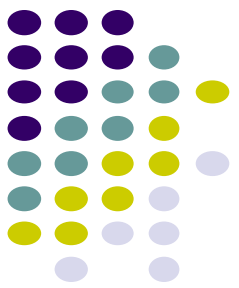
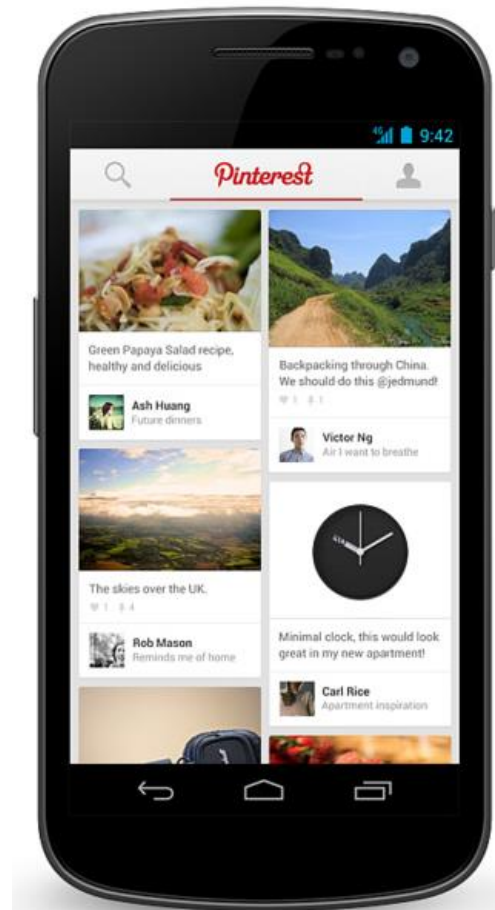
- 3 Files:

- **Activity_main.xml:** XML file specifying screen layout

- **MainActivity.Java or MainActivity.kt:** Java or kotlin code to define behavior, actions taken when button clicked (intelligence)

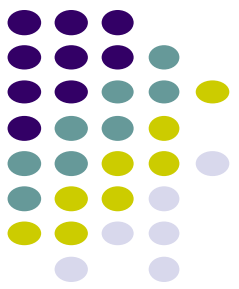
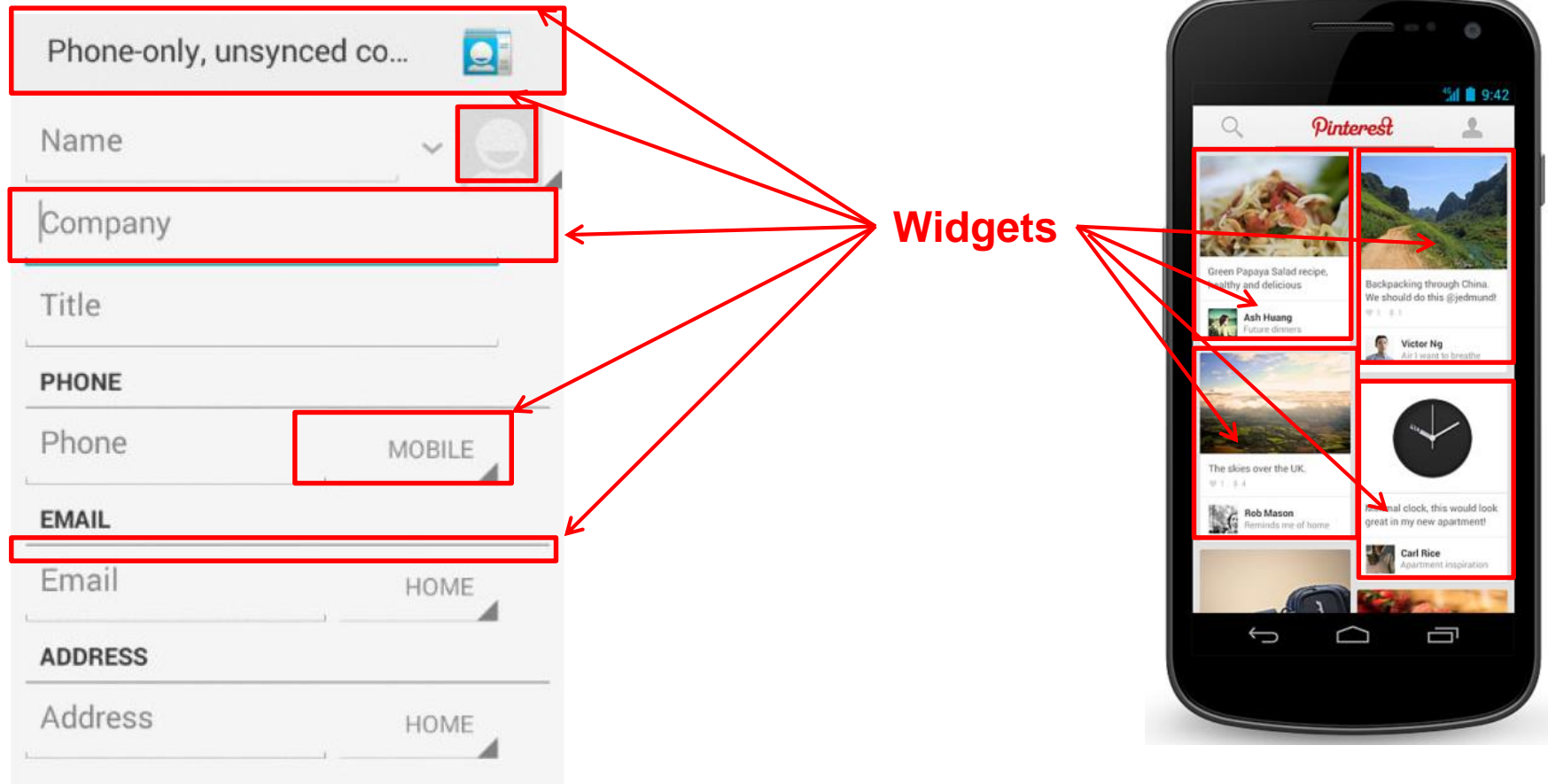
- **AndroidManifest.xml:**

- 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"



Recall: Widgets

- *Android UI design involves arranging widgets on a screen*
- **Widgets?** Rectangles containing texts, image, etc
- **Screen design:** Select widgets, specify attributes (dimensions, margins, etc)

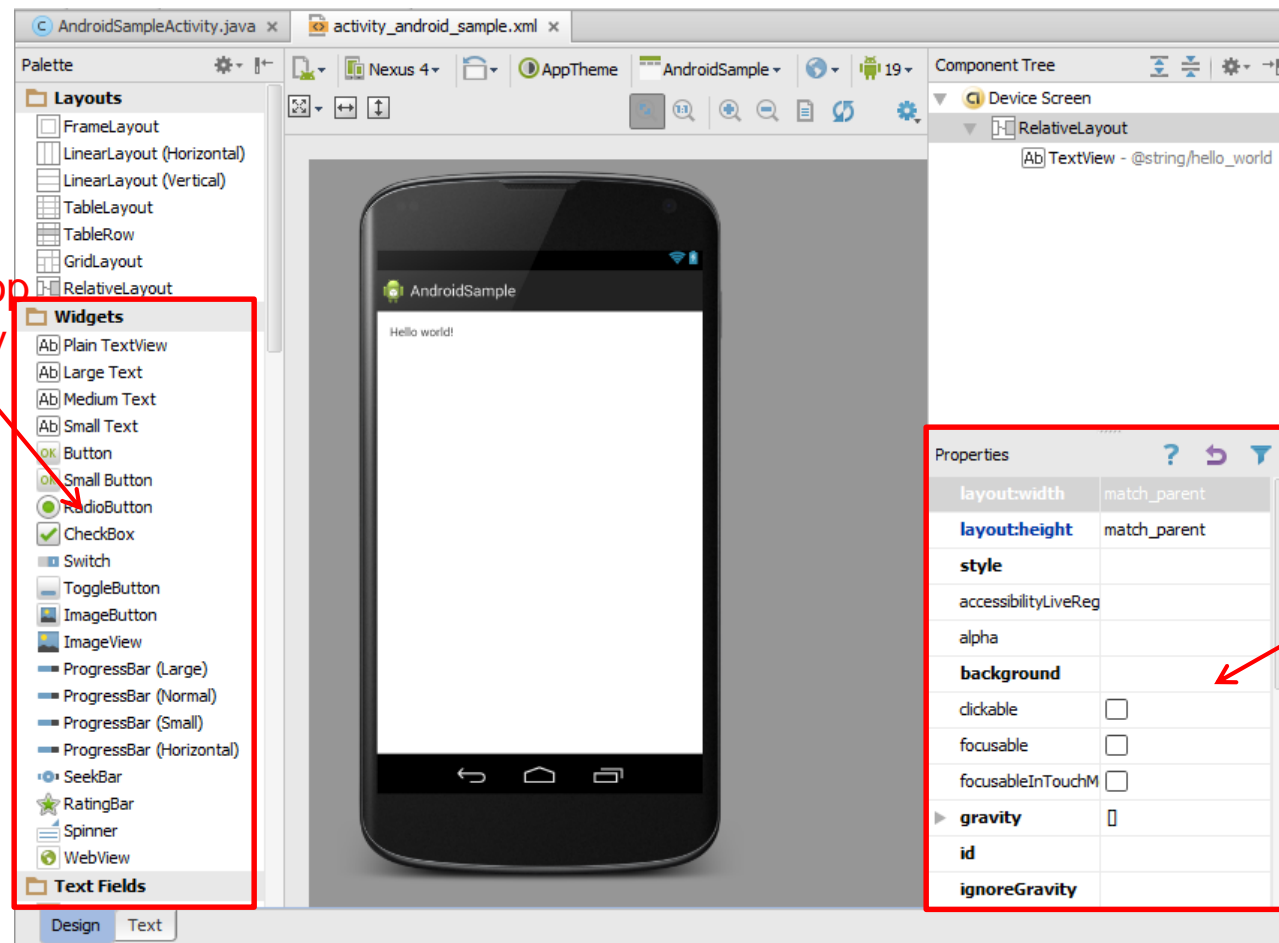




Recall: Design Option 1: Drag and Drop Widgets

- 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



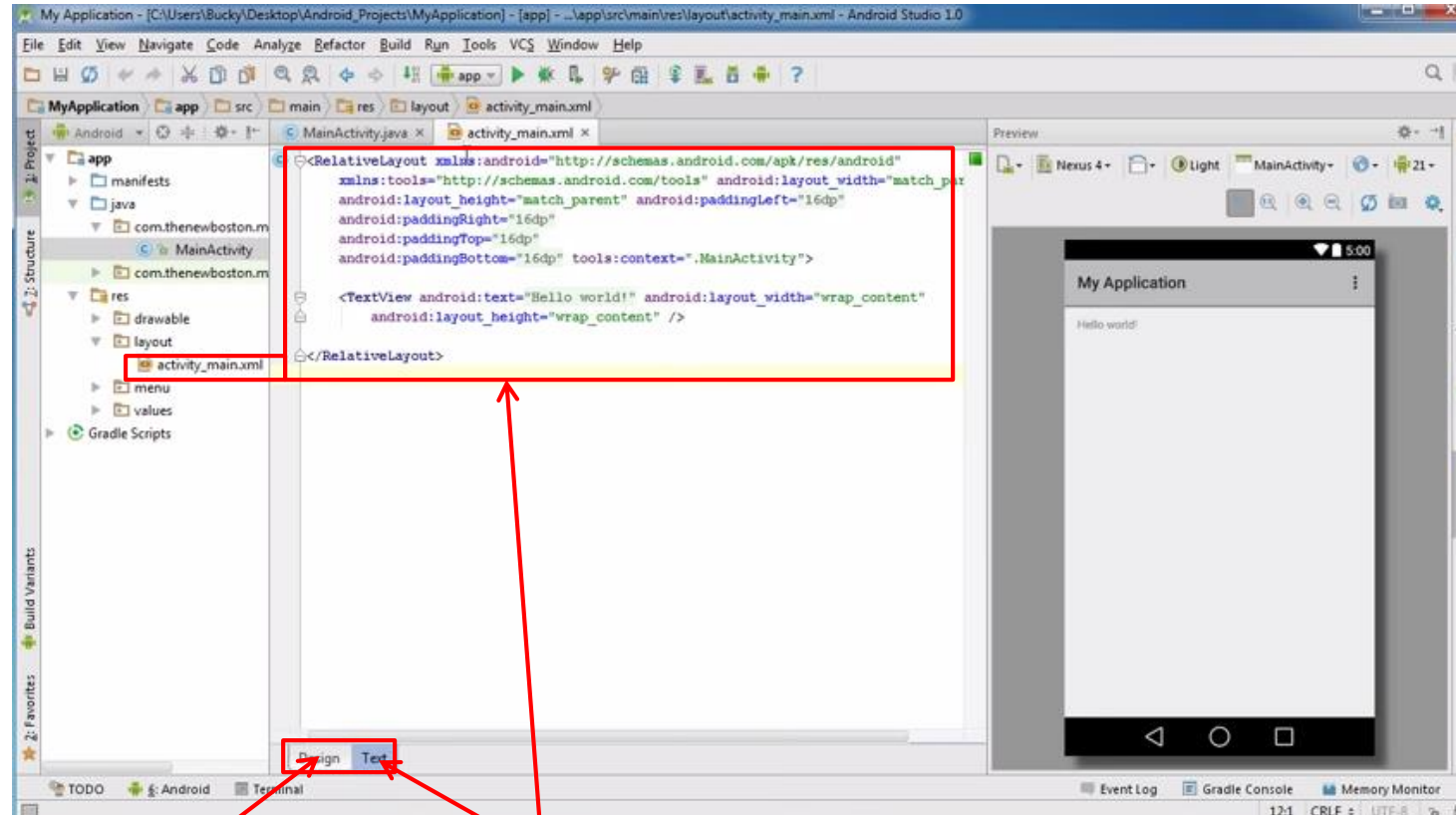
Note: Your Android
Studio screen may
Be different, depending
On version

Edit widget
properties



Recall: Design Option 2: Edit XML Directly

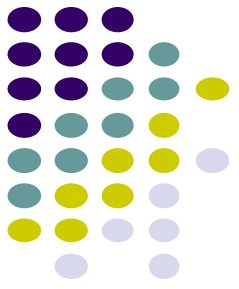
- **Text view:** Directly edit XML file defining screen (activity_main.xml)
- **Note:** dragging and dropping widgets in design view auto-generates corresponding XML in Text view



Drag and drop widget

Edit XML

Note: Your Android Studio screen may Be different, depending On version

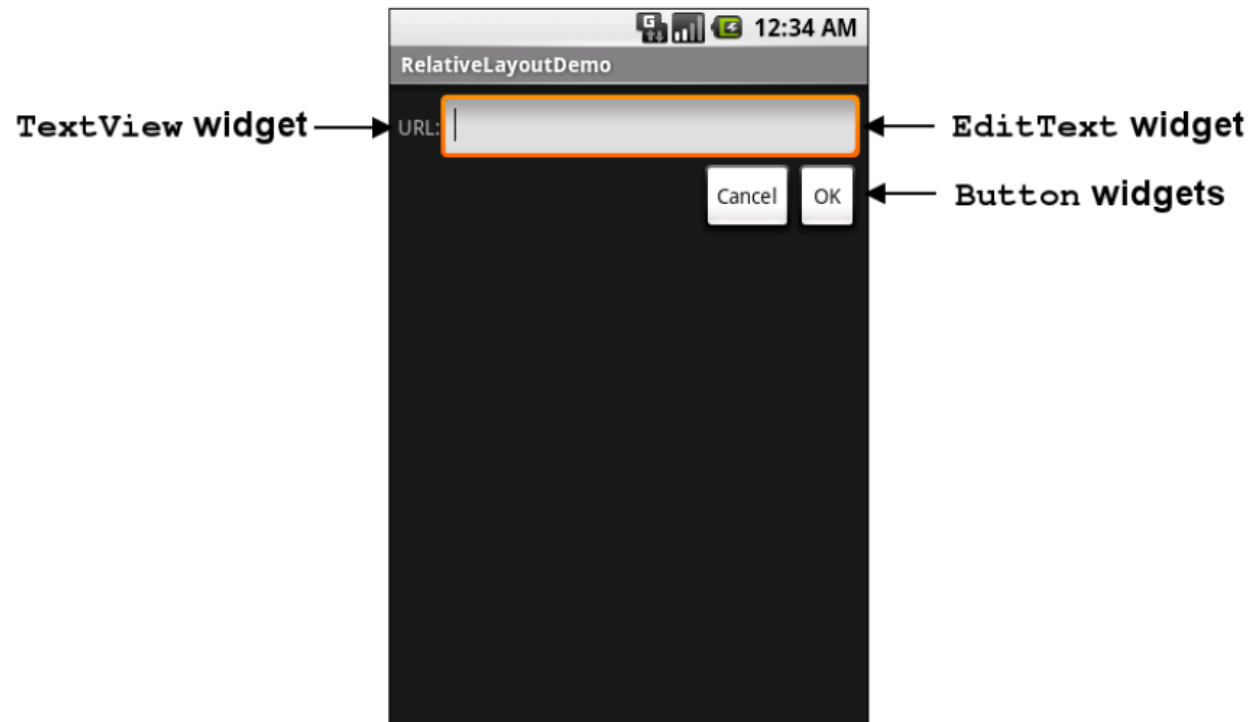


Android Widgets

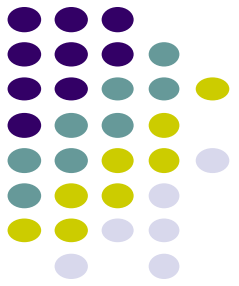



Example: Some Common Widgets

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



General Form of Widget Declaration



<widget type  **E.g. TextView, button, EditText, etc**

Widget attributes (e.g. format, width, length, etc)

.....

.....

/>

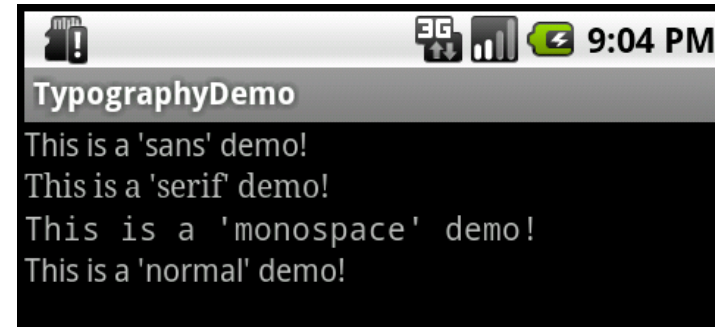
TextView Widget

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

XML code

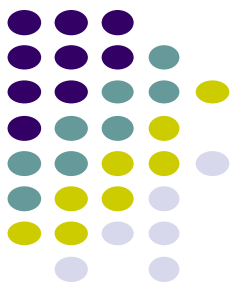
```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="This is a 'sans' demo!"  
    android:typeface="sans"  
>
```

TextView Widgets



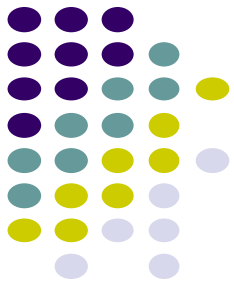
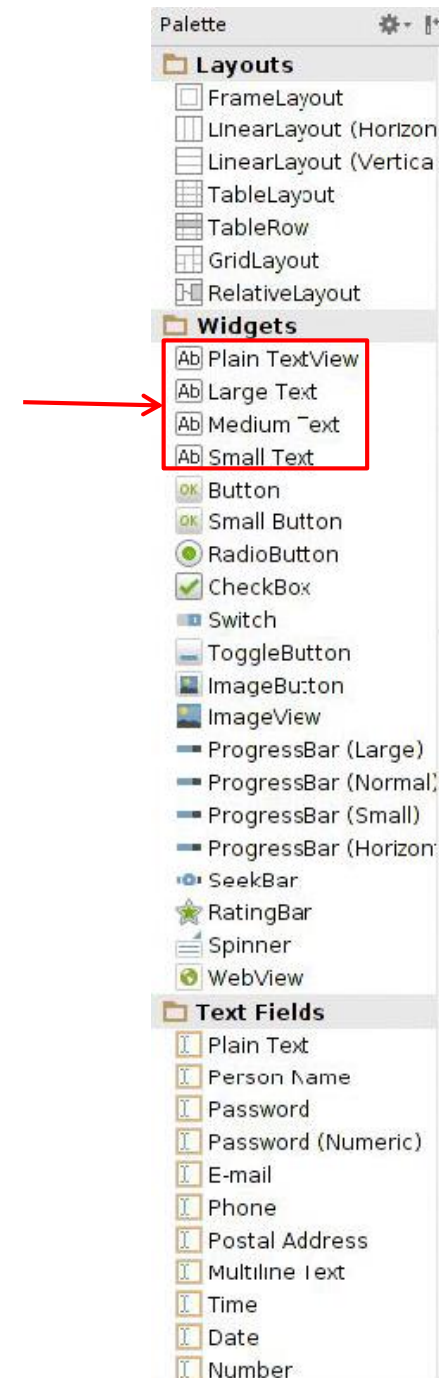
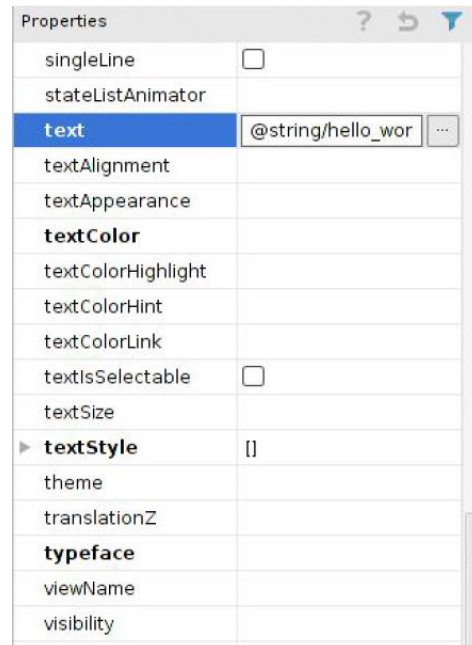
- **Common attributes:**

- 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 Textview widget in, edit properties





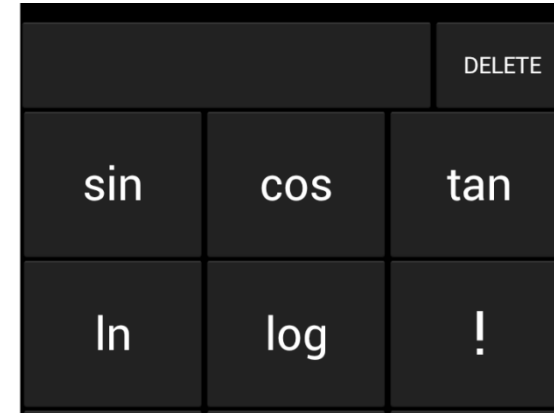
Widget ID

- Every widget has ID, stored in **android:id** attribute
- Using Widget ID declared in XML, widget can be referenced, modified in java/kotlin code (More later)

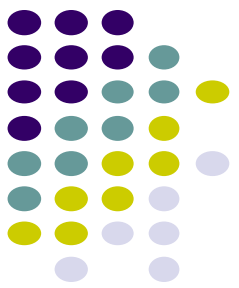
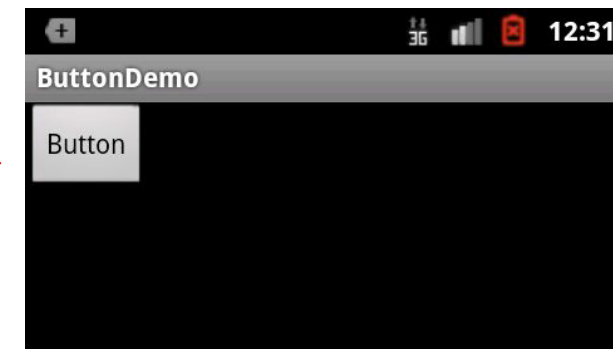
Properties	
ellipsize	
enabled	<input type="checkbox"/>
focusable	<input type="checkbox"/>
focusableInTouchMode	<input type="checkbox"/>
fontFamily	
▶ gravity	[]
height	
hint	
id	textView2
importantForAccessibility	
inputMethod	
▶ inputType	[]
labelFor	
lines	
linksClickable	<input type="checkbox"/>
longClickable	<input type="checkbox"/>
maxHeight	

Button Widget

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

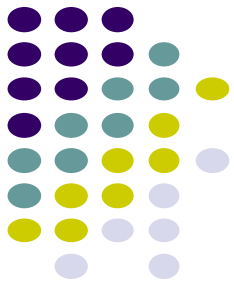


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        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

- How to trigger some action when a button is clicked?

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

Activity_my.xml

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

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

MainActivity.kt

... declare `someMethod` function

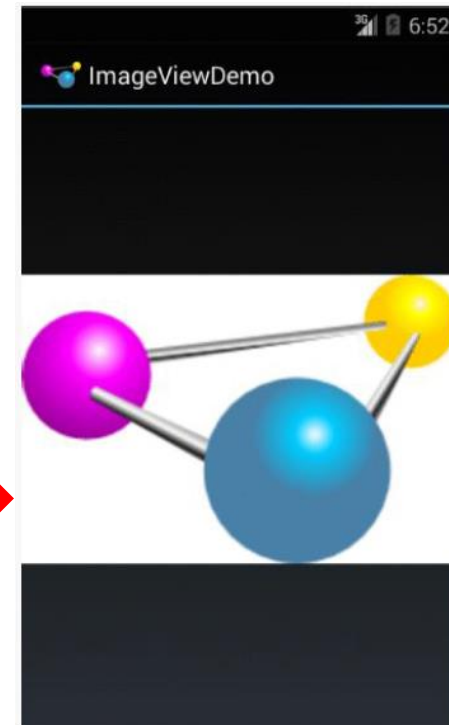
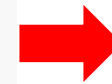


Embedding Images: ImageView and ImageButton

- **ImageView:** display image (not clickable)
- **ImageButton:** Clickable image
- Use **android:src** attribute to specify image source in **drawable** folder (e.g. **@drawable/icon**)

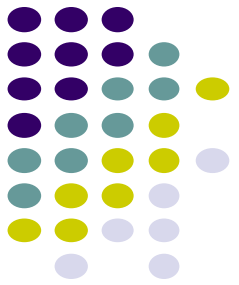
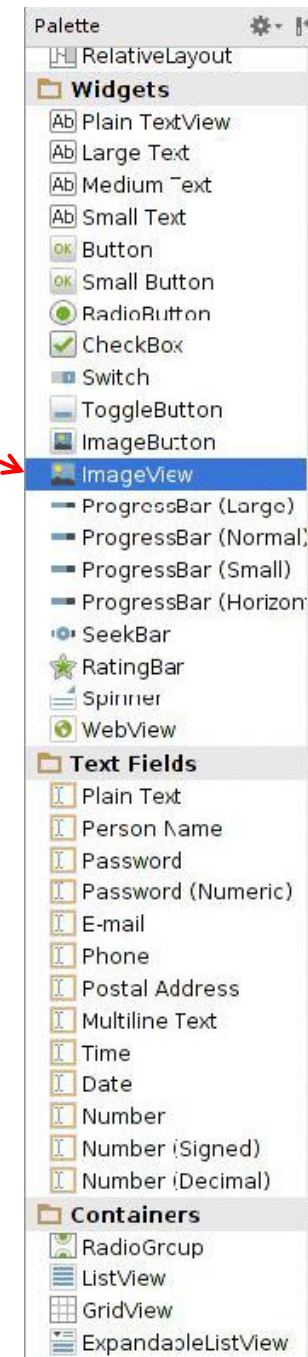
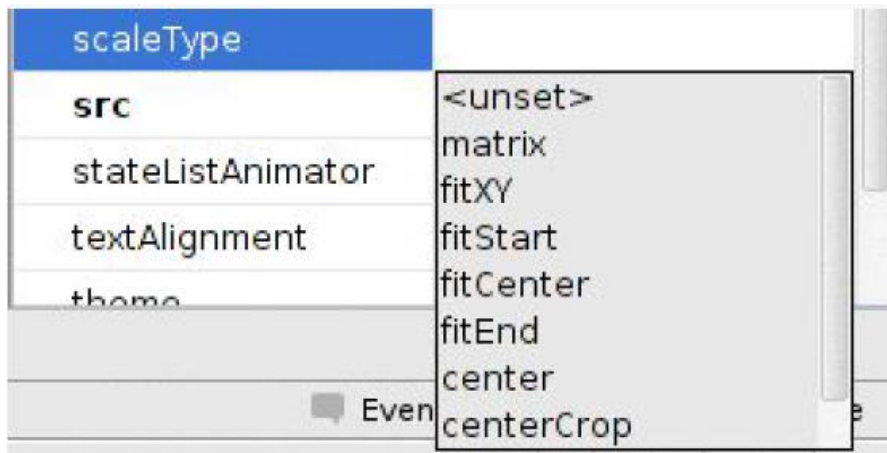
```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/icon"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:adjustViewBounds="true"
    android:src="@drawable/molecule" />
```

File molecule.png in drawable/ folder



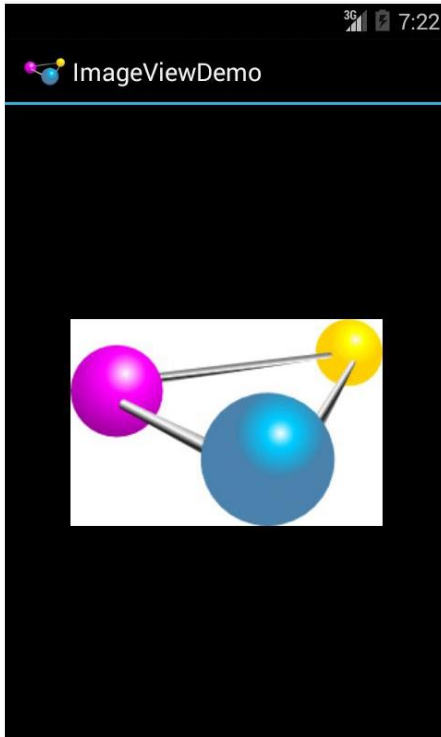
ImageView in Widgets Palette

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

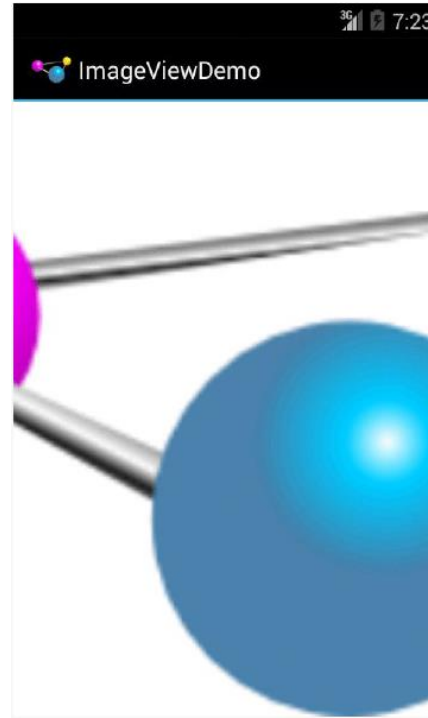




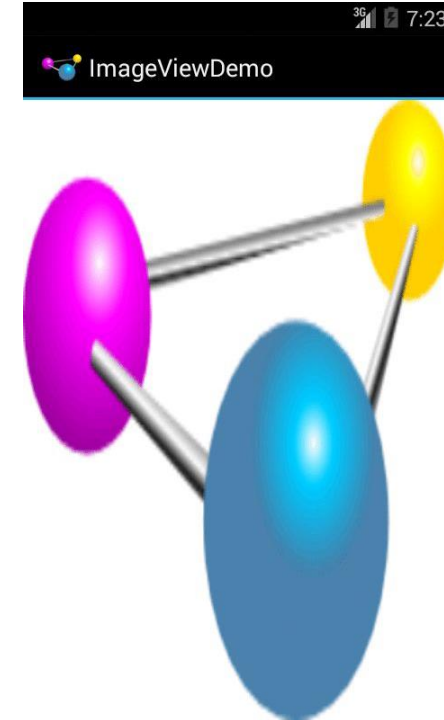
Options for Scaling Images (scaleType)



“**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

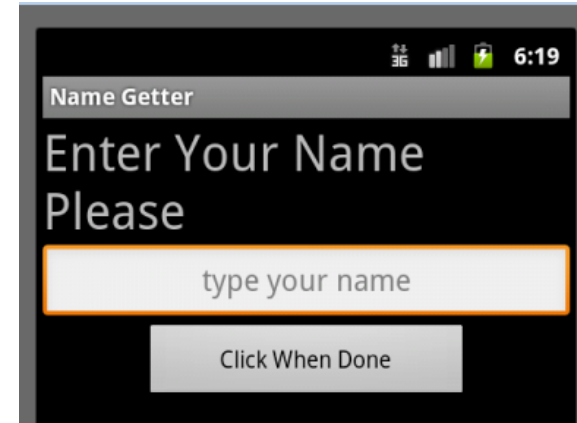


EditText Widget

- Widget with box for user input
- 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
 - E.g. if inputType is a number, numeric keyboard pops up



EditText Widget in Android Studio Palette

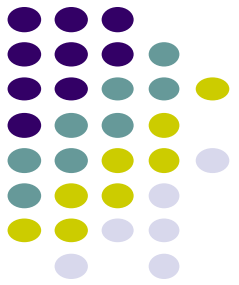
- A section of Android Studio palette has EditText widgets (or text fields)

Text Fields
Section of Widget
palette

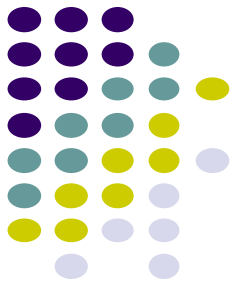


inputType	
none	<input type="checkbox"/>
text	<input type="checkbox"/>
textCapCharacter	<input type="checkbox"/>
textCapWords	<input type="checkbox"/>
textCapSentences	<input type="checkbox"/>
textAutoCorrect	<input type="checkbox"/>
textAutoComplete	<input type="checkbox"/>
textMultiLine	<input type="checkbox"/>
textimeMultiLine	<input type="checkbox"/>
textNoSuggestion	<input type="checkbox"/>
textUri	<input type="checkbox"/>
textEmailAddress	<input type="checkbox"/>
textEmailSubject	<input type="checkbox"/>
textShortMessage	<input type="checkbox"/>
textLongMessage	<input type="checkbox"/>
textPersonName	<input type="checkbox"/>
textPostalAddress	<input type="checkbox"/>
textPassword	<input type="checkbox"/>
textVisiblePasswo	<input type="checkbox"/>
textWebEditText	<input type="checkbox"/>
textFilter	<input type="checkbox"/>
textPhonetic	<input type="checkbox"/>
textWebEmailAddr	<input type="checkbox"/>
textWebPassword	<input type="checkbox"/>
number	<input type="checkbox"/>
numberSigned	<input type="checkbox"/>
numberDecimal	<input type="checkbox"/>
numberPassword	<input type="checkbox"/>
phone	<input type="checkbox"/>

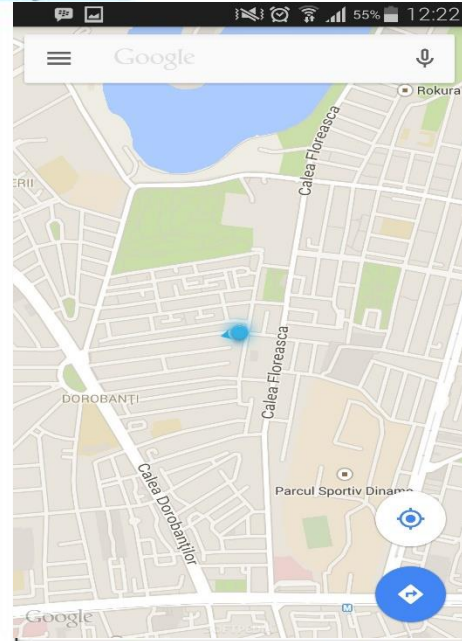
EditText
inputType menu



Some Other Available Widgets



MapView



**Rectangle that
contains a map**

WebView

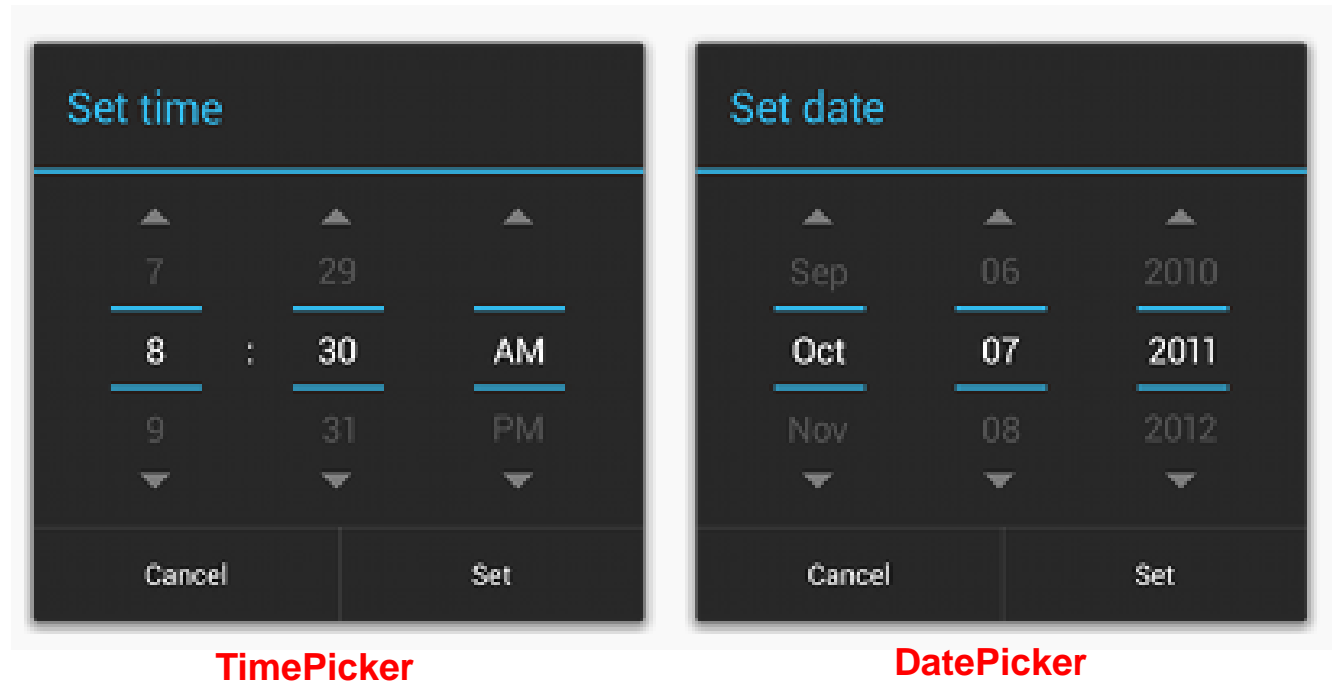


**Rectangle that
contains a web page**



Pickers

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



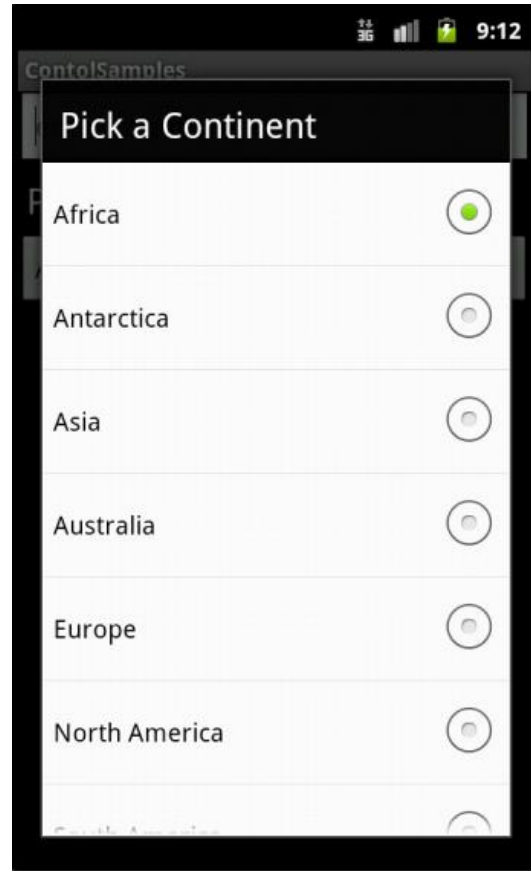
TimePicker

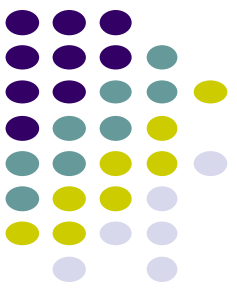
DatePicker

Spinner Controls

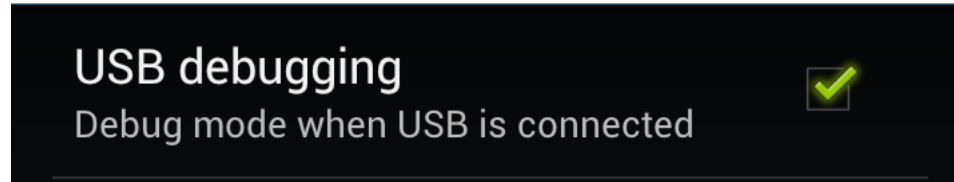


- user **must** select one of a set of choices



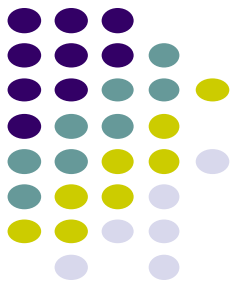


Checkbox



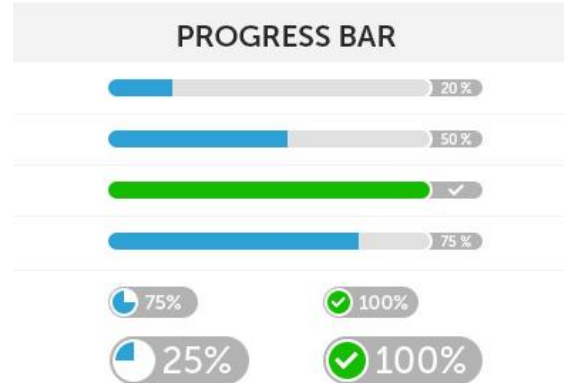
- Checkbox has 2 states: checked and unchecked
- XML code to create Checkbox

```
<?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"/>
```

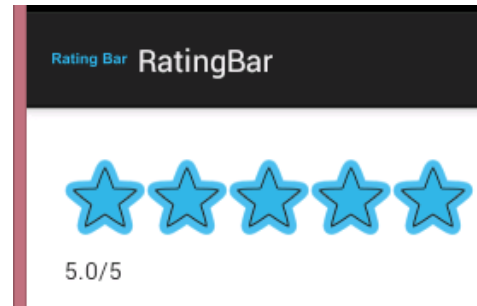


Other Indicators & More Widgets

- ProgressBar

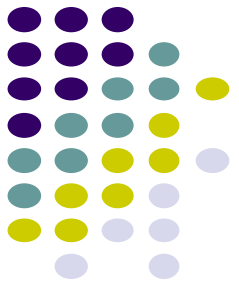


- RatingBar



- Chronometer
- DigitalClock
- AnalogClock





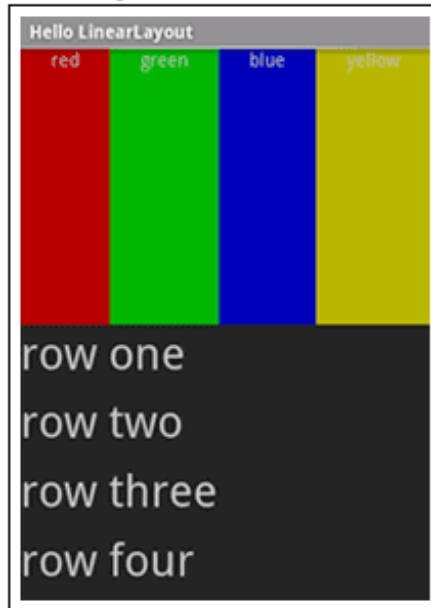
Android Layouts in XML



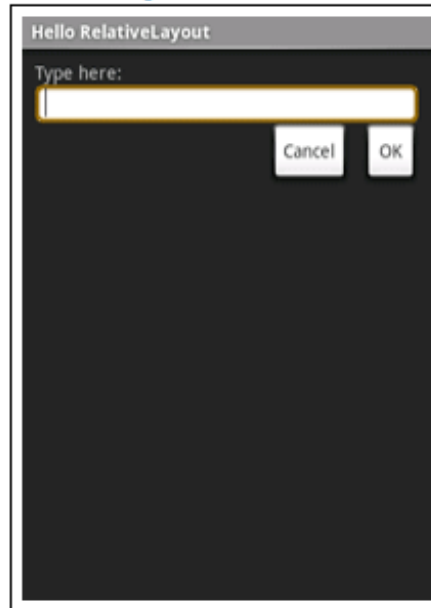
Android UI using XML Layouts

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

LinearLayout



RelativeLayout

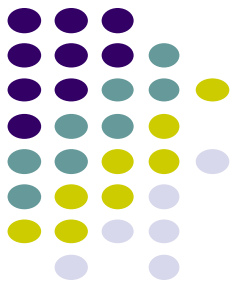


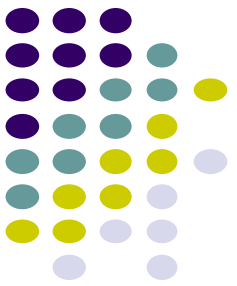
TableLayout



Some Layouts

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





LinearLayout

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

- Example:

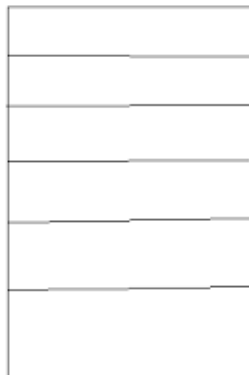
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.c
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff00ff"
    android:orientation="vertical" >
```

Layout
properties

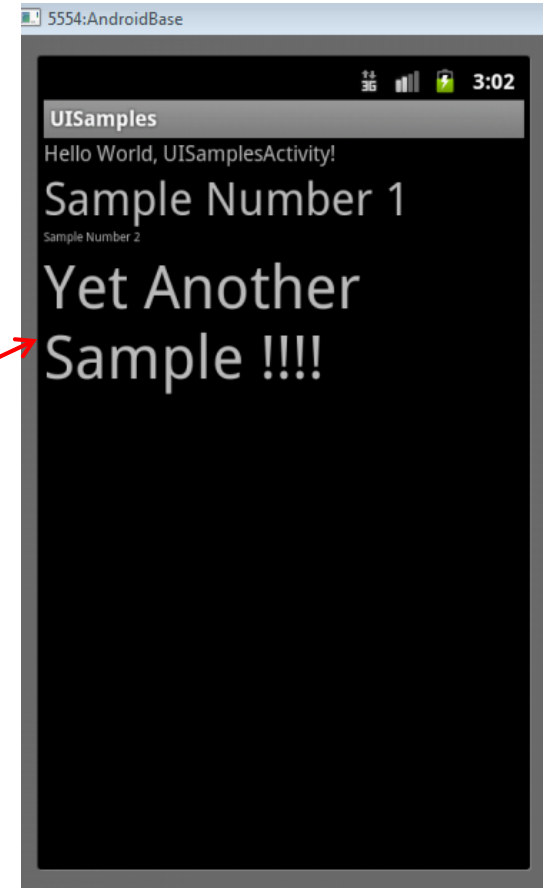
- orientation attribute defines direction (vertical or horizontal):
 - E.g. `android:orientation="vertical"`

Linear Layout

Orientation: vertical



Orientation: horizontal





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**

TextView widget width
should be as wide as
its parent (the layout)

TextView widget height
should be as wide as
the content (text)

Hierarchy
Screen (Hardware)
↑
Linear Layout
↑
TextView

```
<?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"
    />

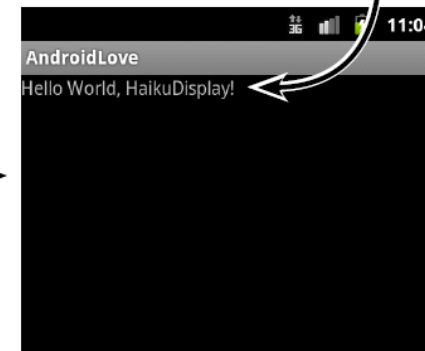
</LinearLayout>
```

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

XML

main.xml

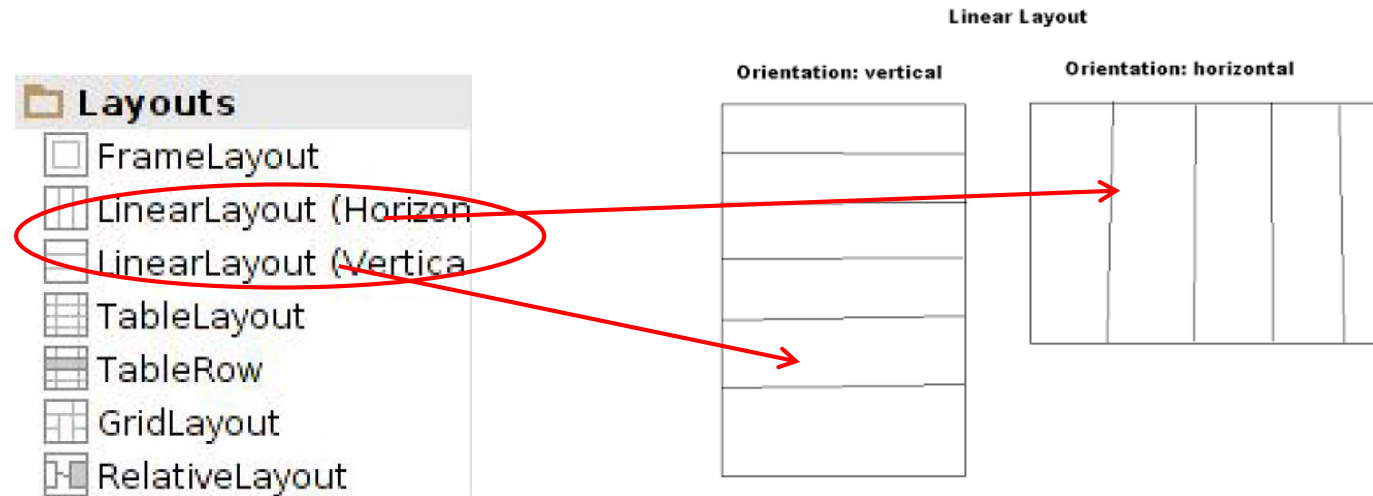
The ViewGroup, in this case a LinearLayout fills the screen.



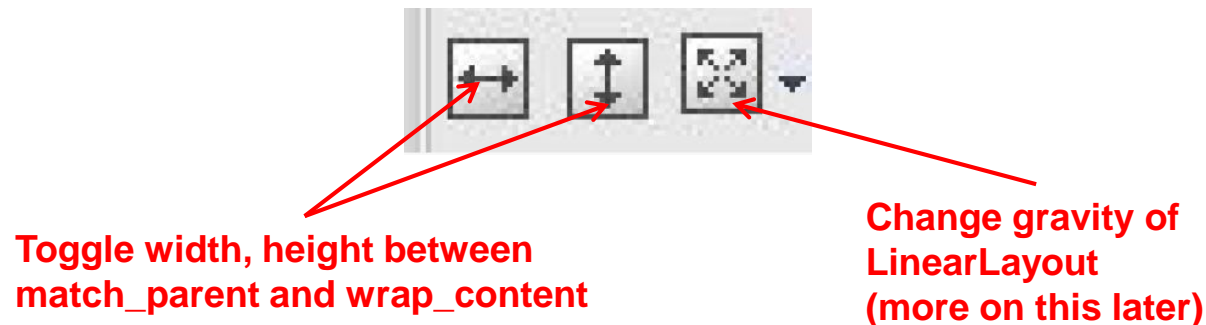


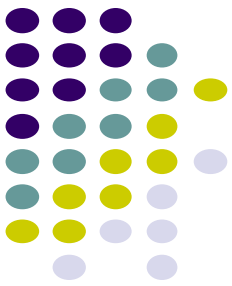
LinearLayout in Android Studio

- LinearLayout in Android Studio Graphical Layout Editor



- After selecting LinearLayout, toolbars buttons to set parameters





LinearLayout Attributes

- Layout attributes usually set in XML file
- But can also be set in code in kotlin or java file

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

Ref: <https://developer.android.com/reference/android/widget/LinearLayout>

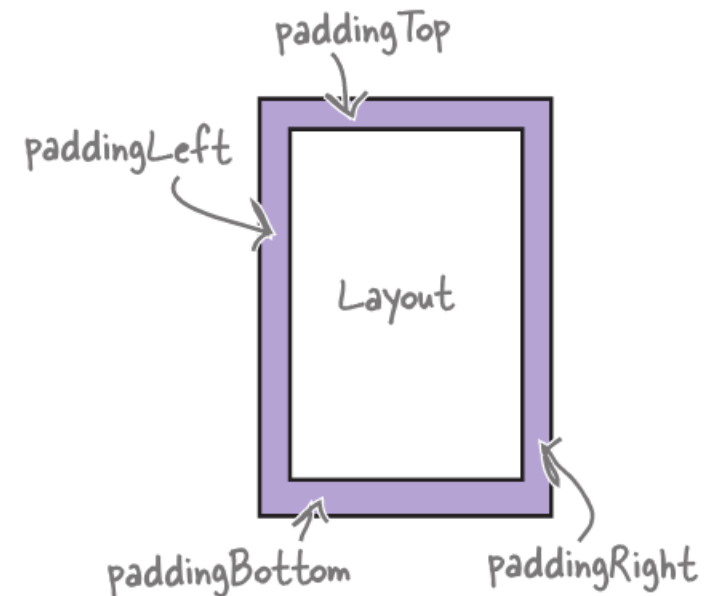


Adding Padding

- 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.



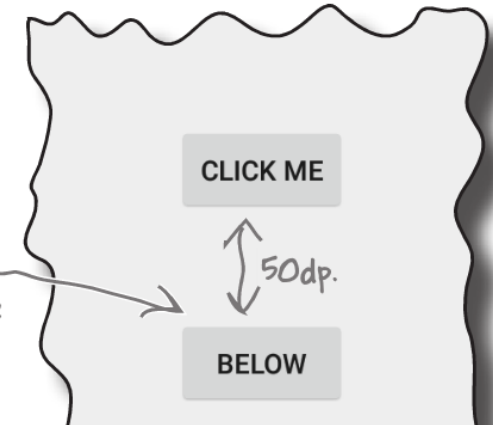


Setting Margins

- 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.



- Other options

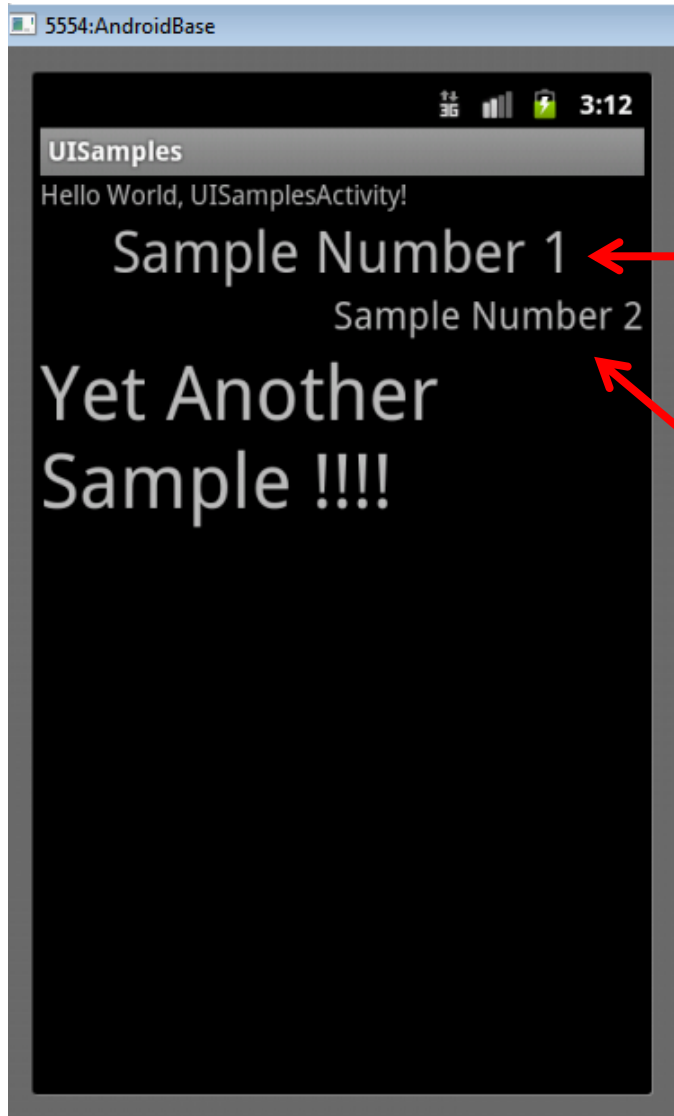
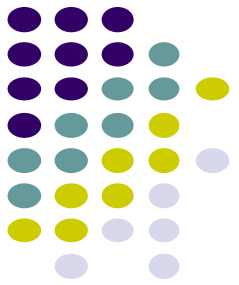
`android:layout_marginLeft`



`android:layout_marginRight`



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

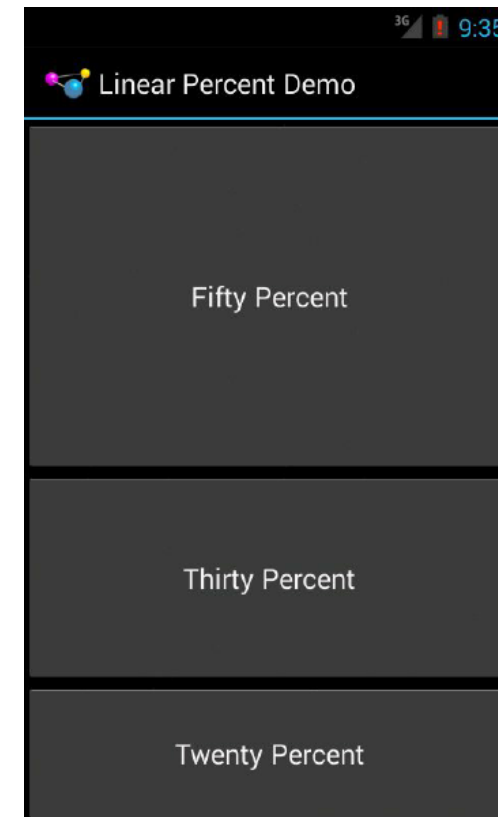
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <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"/>

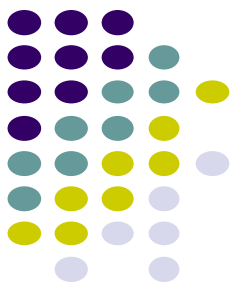
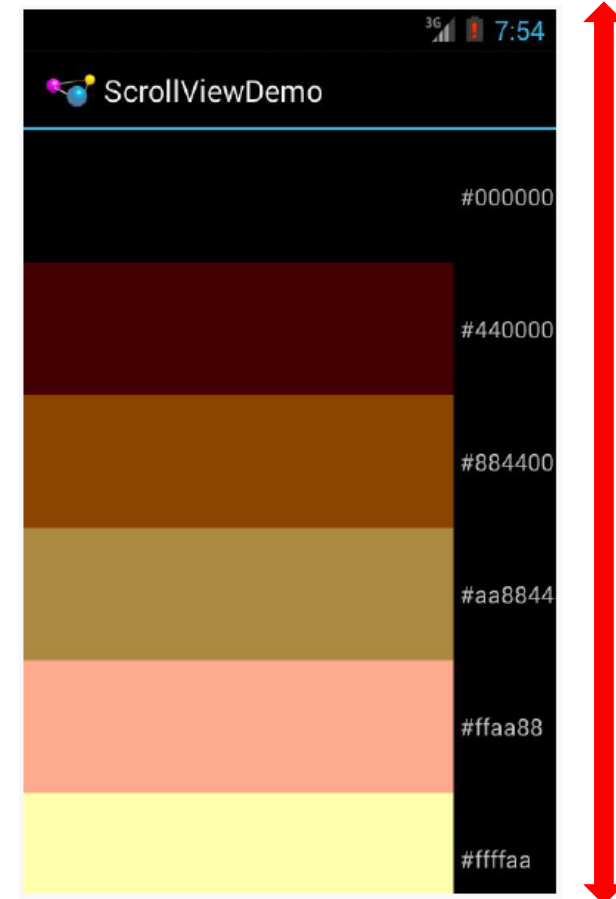
</LinearLayout>
```



Scrolling

- Phone screens are small, scrolling content helps
- Examples: Scroll through
 - large image
 - Linear Layout with lots of elements
- Views for Scrolling:
 - **ScrollView** for vertical scrolling
 - **HorizontalScrollView**
- Rules:
 - Can have 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>
```

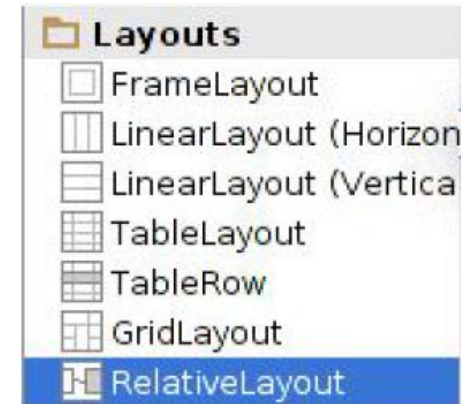
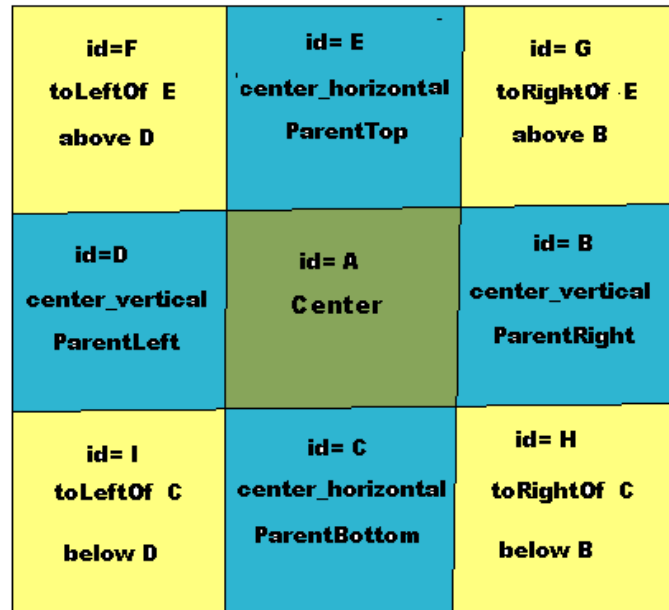




RelativeLayout

- 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

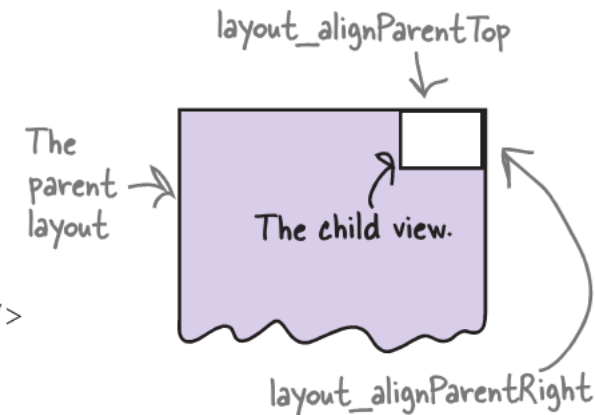


Positioning Views Relative to Parent Layout

- 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>
```

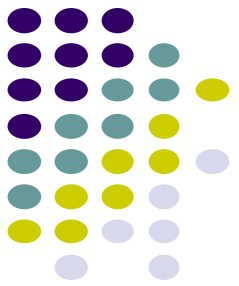
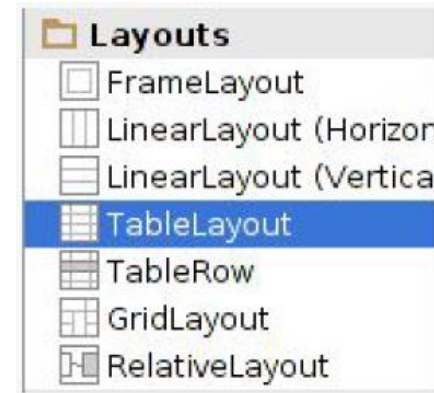
The layout contains the button, so the layout is the button's parent.



See Head First Android Development (2nd edition) page 169-220 for more examples

Table Layout

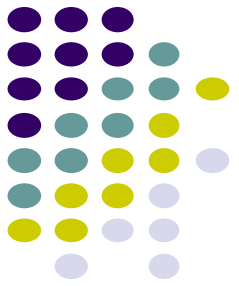
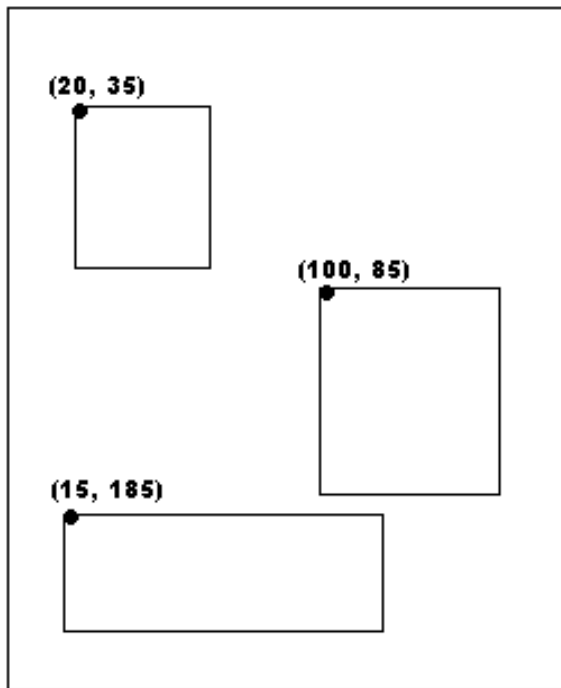
- Specify number of rows and columns of views.
- Available in Android Studio palette

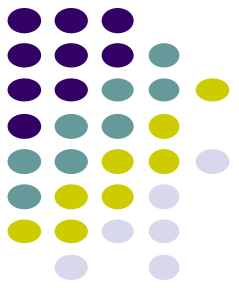


Absolute Layout

- Allows specification of exact x,y coordinates of layout's children.

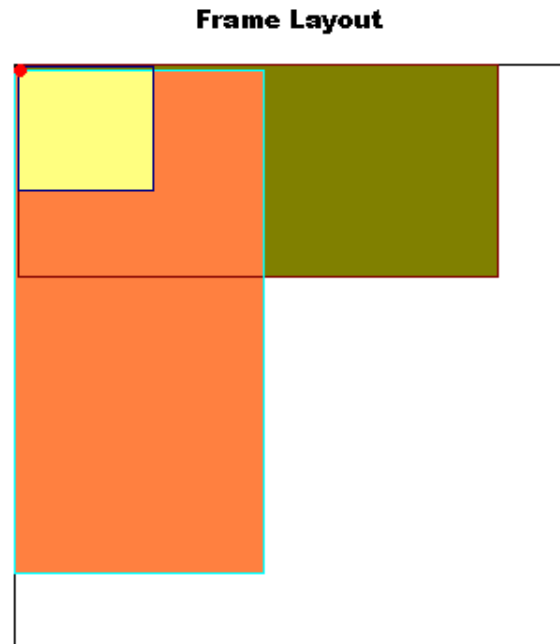
Absolute Layout



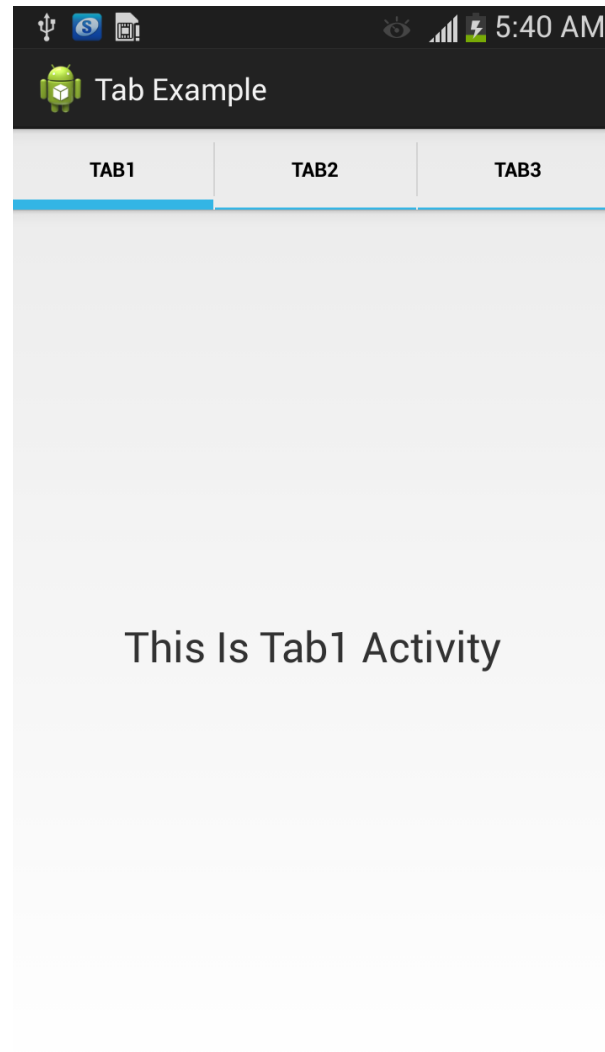
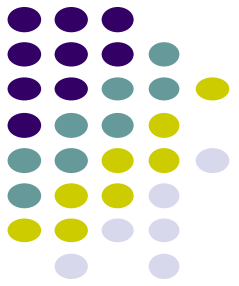


FrameLayout

- child elements pinned to top left corner of layout
- adding a new element / child draws over the last one



Other Layouts: Tabbed Layouts





Android Example: My First App

(Read: Head First Android, Chapter 1 & 2)



References

- Android App Development for Beginners videos by Bucky Roberts (thenewboston)
- Head First Android, 2nd and 3rd edition
- Android Nerd Ranch, 5th Edition
- Ask A Dev, Android Wear: What Developers Need to Know, <https://www.youtube.com/watch?v=zTS2NZpLyQg>
- Ask A Dev, Mobile Minute: What to (Android) Wear, https://www.youtube.com/watch?v=n5Yjzn3b_aQ