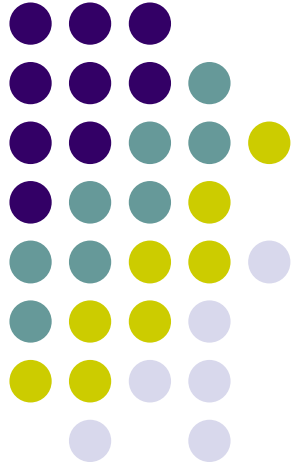
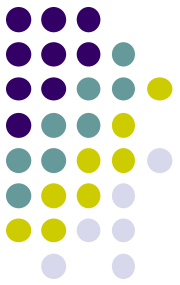


**CS 4518 Mobile and Ubiquitous  
Computing  
Lecture 3: Android UI Design in XML  
+ Examples**

---

**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:**

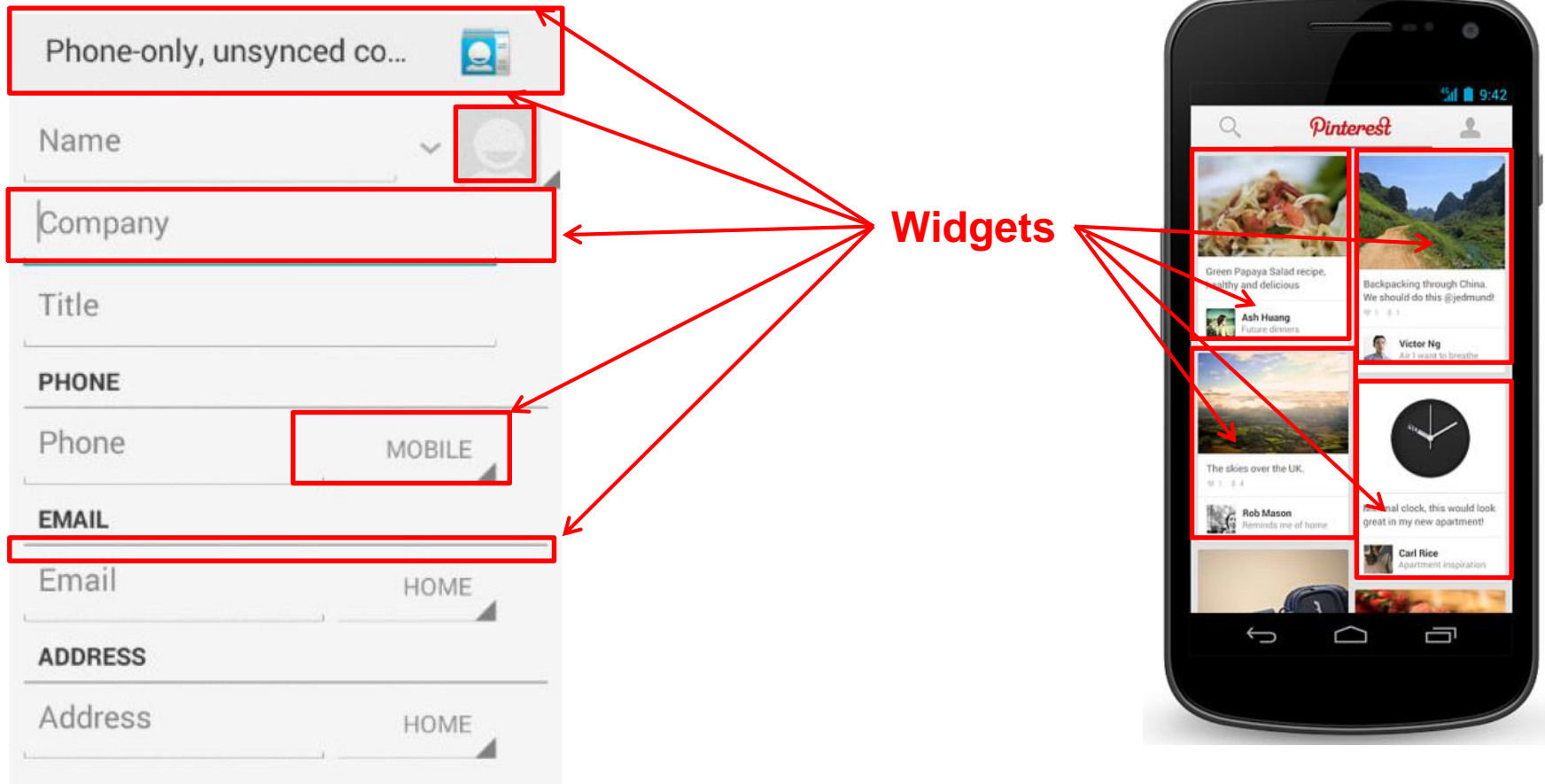
- Lists all app components and screens
- Like a table of contents for a book
- E.g. Hello world program has 1 screen, so AndroidManifest.xml has 1 item listed
- App starts running here (a bit like main( ) in C), launching activity with a tag "LAUNCHER"

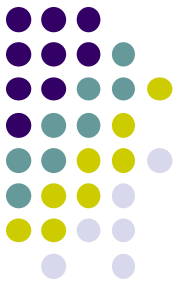


# Widgets



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

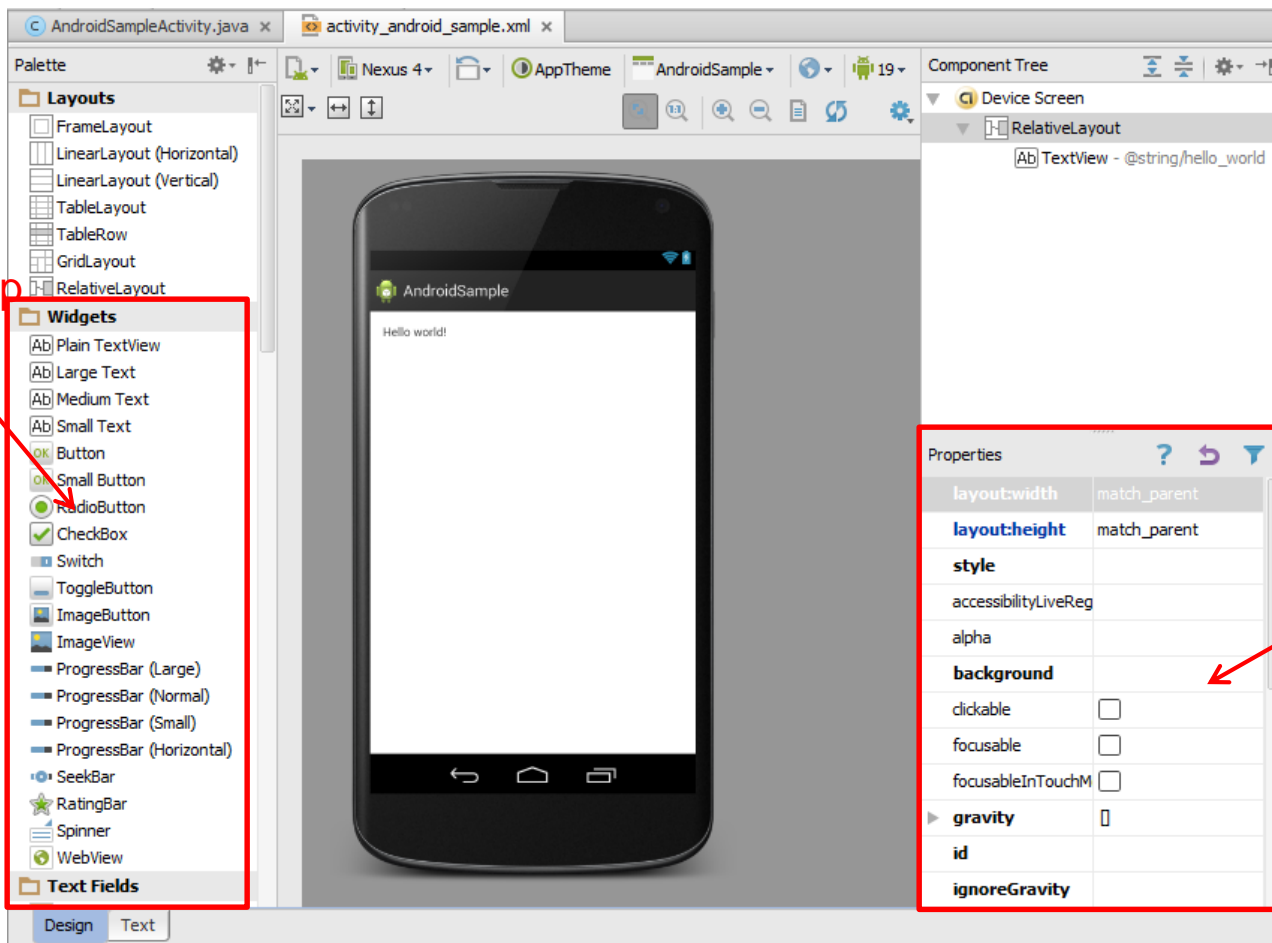




# Recall: Design Option 1: Drag and Drop Widgets

- 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

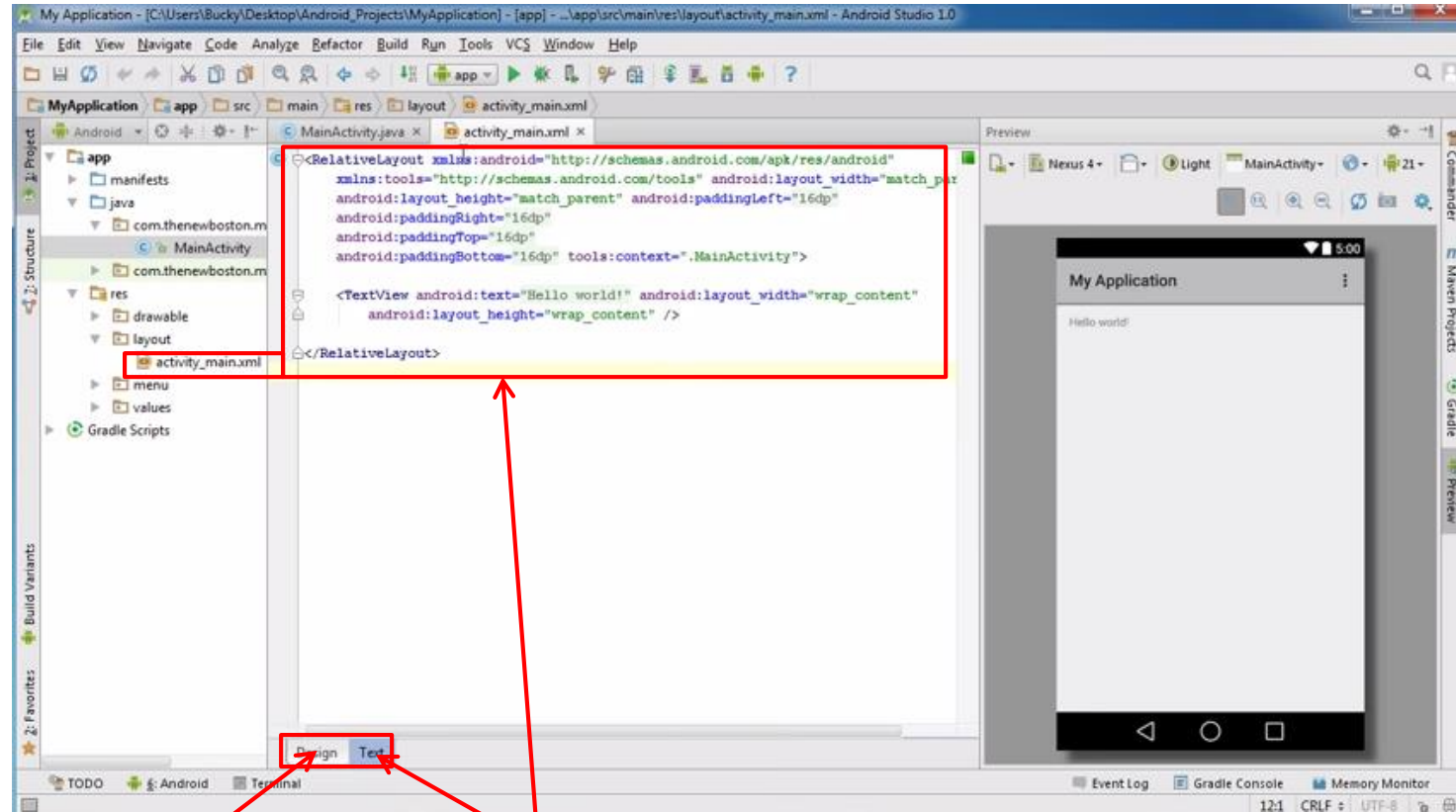


Edit widget properties

# Recall: Design Option 2: Edit XML Directly

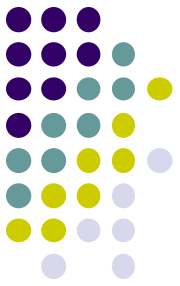


- **Text view:** Directly edit XML file defining screen (activity\_main.xml)
- **Note:** dragging and dropping widgets in design view auto-generates corresponding XML in Text view



Drag and drop widget

Edit XML

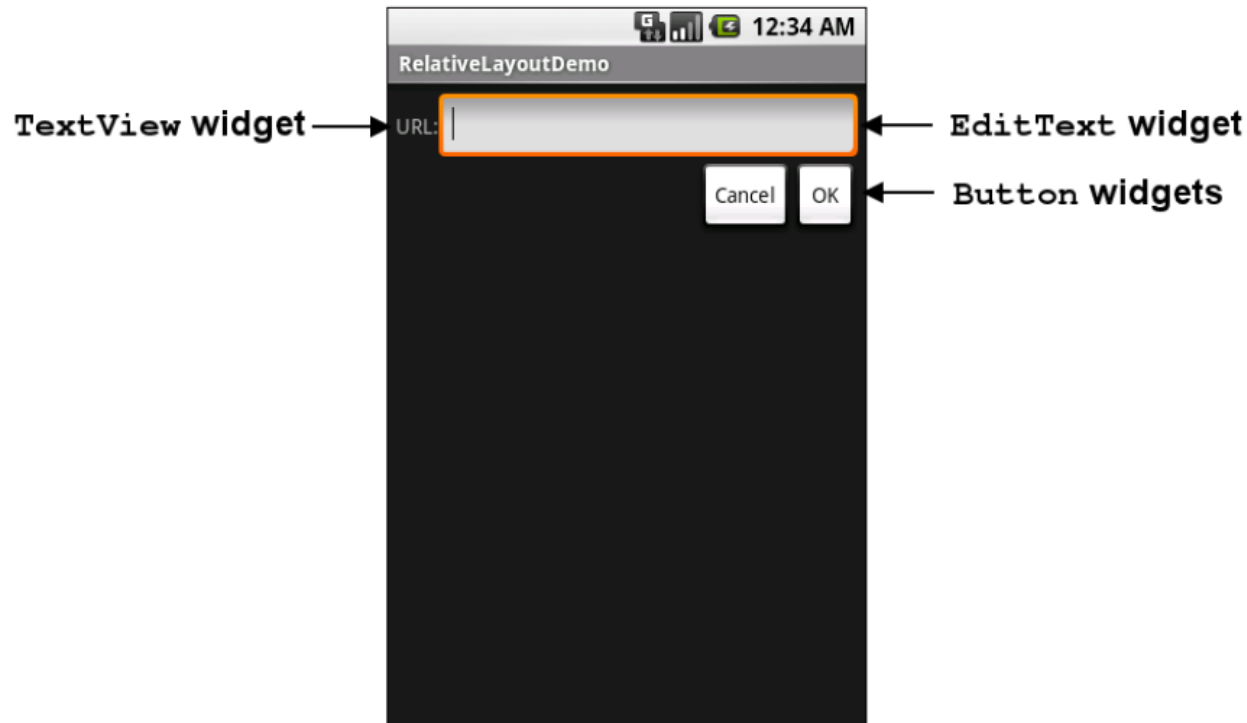


# Android Widgets



# Example: Some Common Widgets

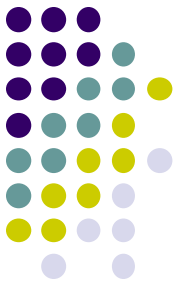
- **TextView:** Text in a rectangle
- **EditText:** Text box for user to type in text
- **Button:** Button for user to click on





# TextView Widget

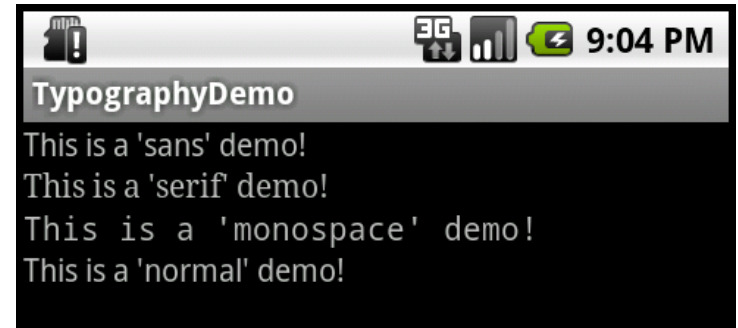
- Text in a rectangle
- Just displays text, no interaction



## 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

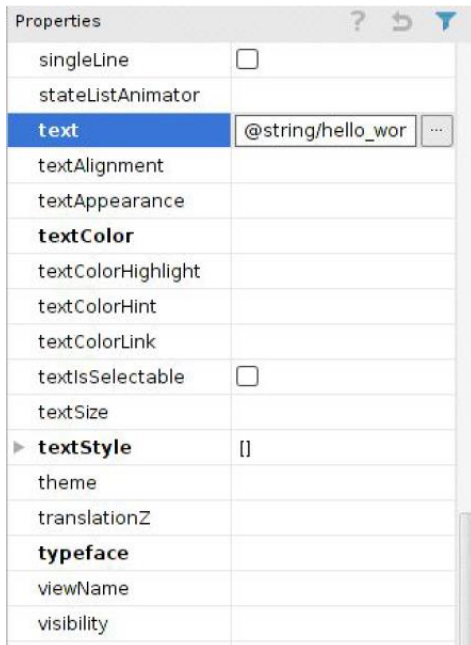
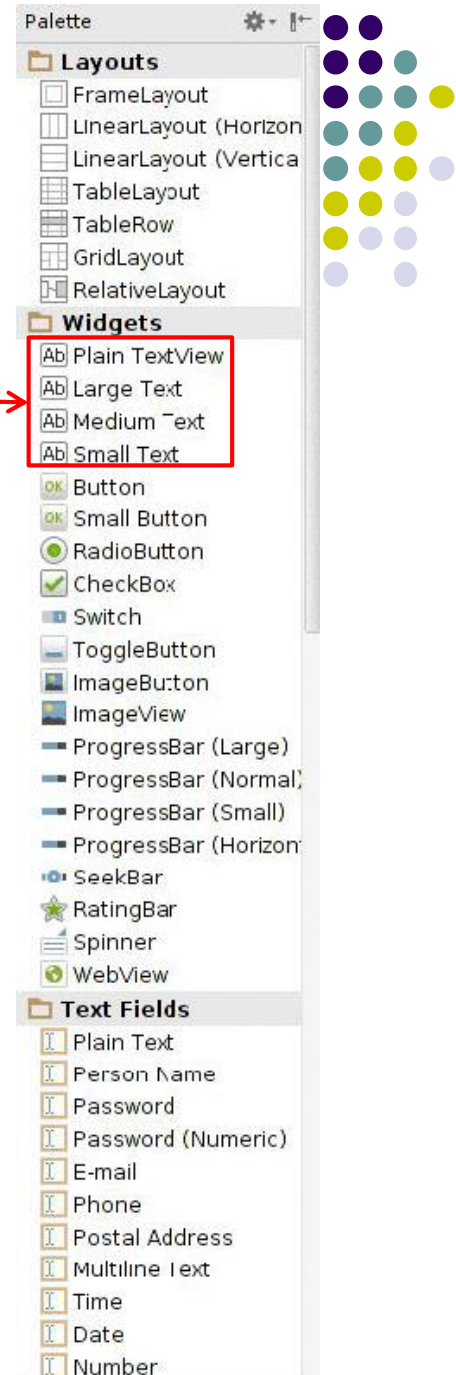


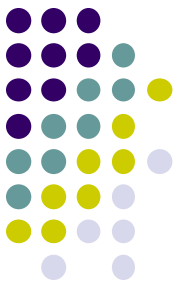
- **Common attributes:**

- 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

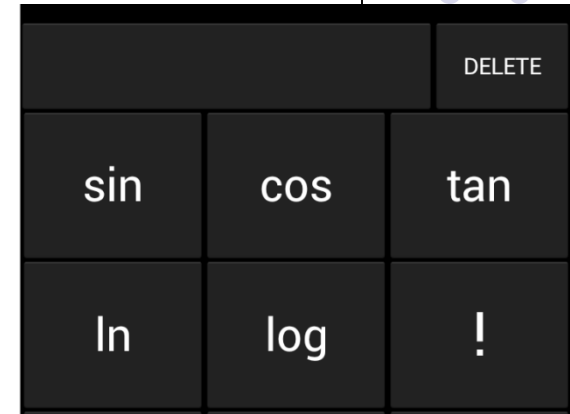
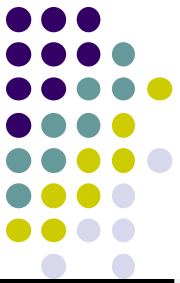
- Every widget has ID, stored in **android:id** attribute
- Using Widget ID declared in XML, widget can be referenced, modified in java code (More later)

The screenshot shows the 'Properties' window in an IDE, displaying a list of attributes for a widget. The 'id' attribute is highlighted in blue and set to 'textView2'. Other attributes include 'ellipsize', 'enabled', 'focusable', 'focusableInTouchMode', 'fontFamily', 'gravity', 'height', 'hint', 'importantForAccessibility', 'inputMethod', 'inputType', 'labelFor', 'lines', 'linksClickable', 'longClickable', and 'maxHeight'.

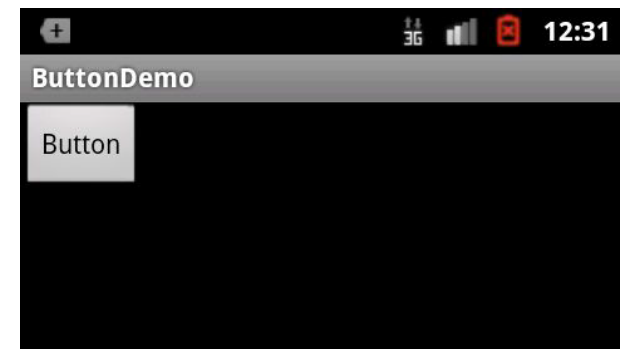
Properties	
ellipsize	
<b>enabled</b>	<input type="checkbox"/>
focusable	<input type="checkbox"/>
focusableInTouchMode	<input type="checkbox"/>
fontFamily	
▶ <b>gravity</b>	[]
height	
<b>hint</b>	
<b>id</b>	textView2
importantForAccessibility	
inputMethod	
▶ inputType	[]
labelFor	
lines	
linksClickable	<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 so similar attributes (e.g. width, height, etc)

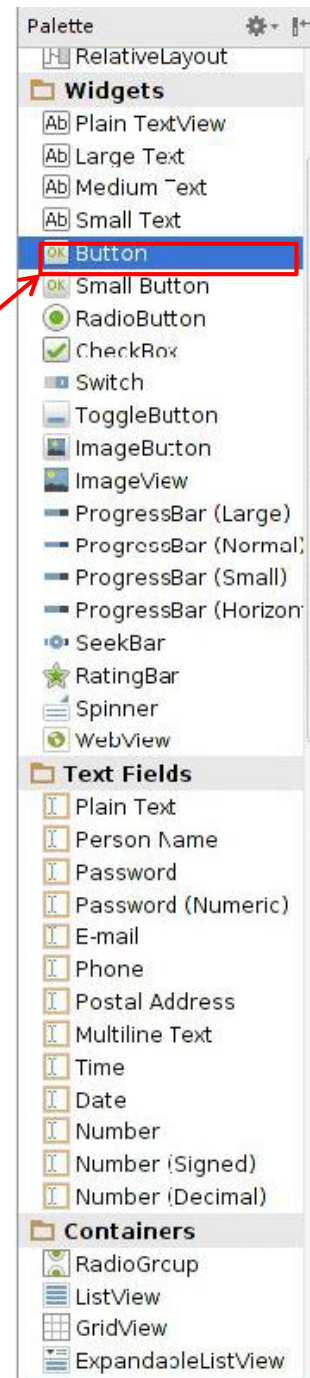


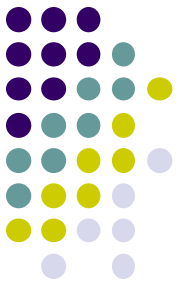
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"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

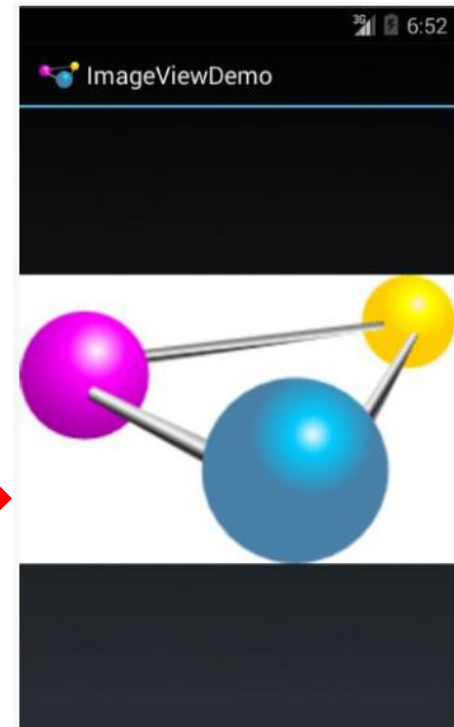


# Embedding Images: ImageView and ImageButton

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

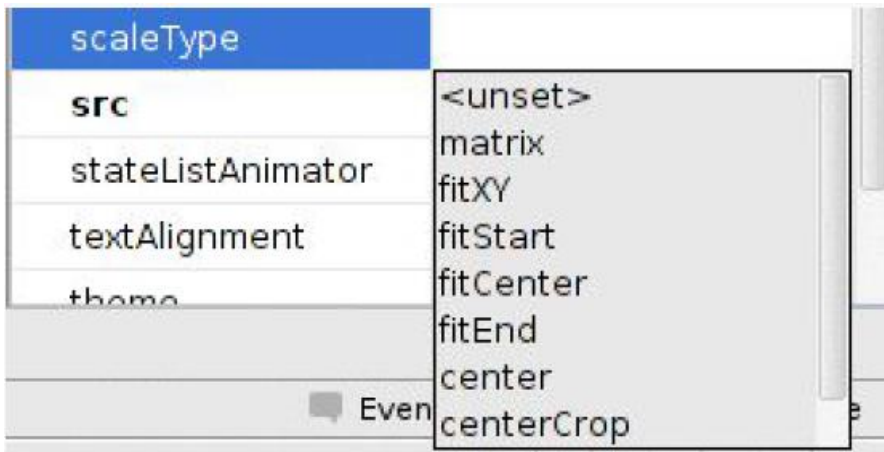
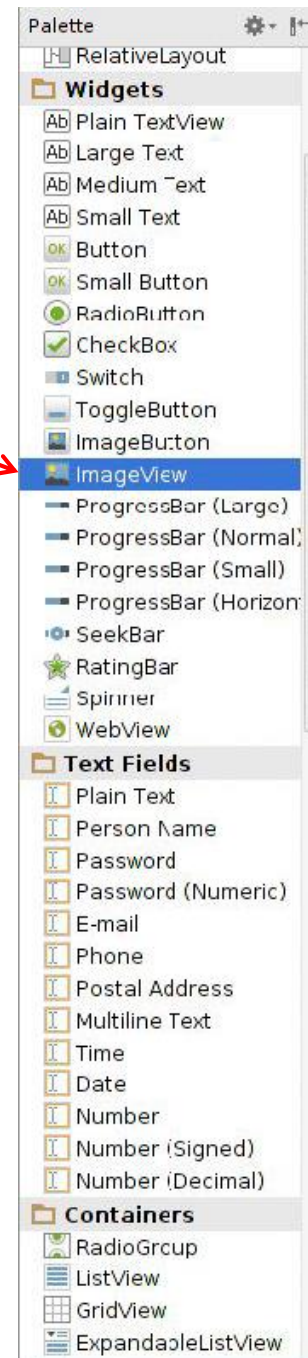
```
<?xml version="1.0" encoding="utf-8"?>  
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"  
  android:id="@+id/icon"  
  android:layout_width="match_parent"  
  android:layout_height="match_parent"  
  android:adjustViewBounds="true"  
  android:src="@drawable/molecule" />
```

File molecule.png in drawable/ folder

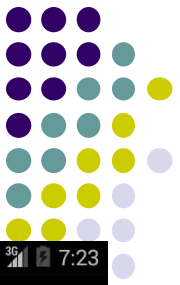


# ImageView in Widgets Palette

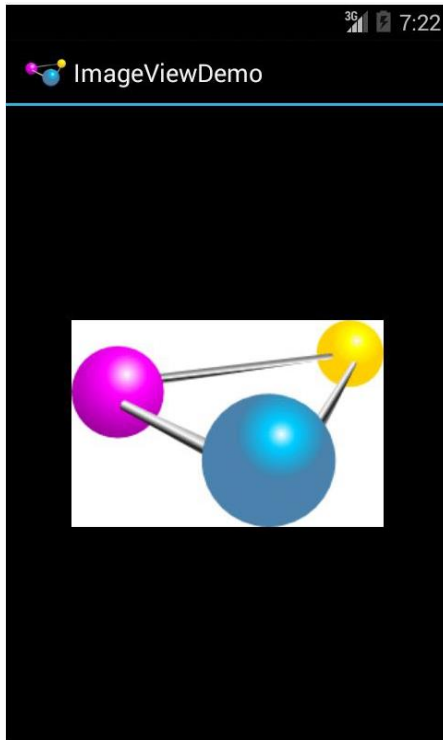
- Can drag and drop ImageView from Widgets Palette
- Use pop-up menus (right-click) to specify:
  - **src**: choose image to be displayed
  - **scaleType**: choose how image should be scaled



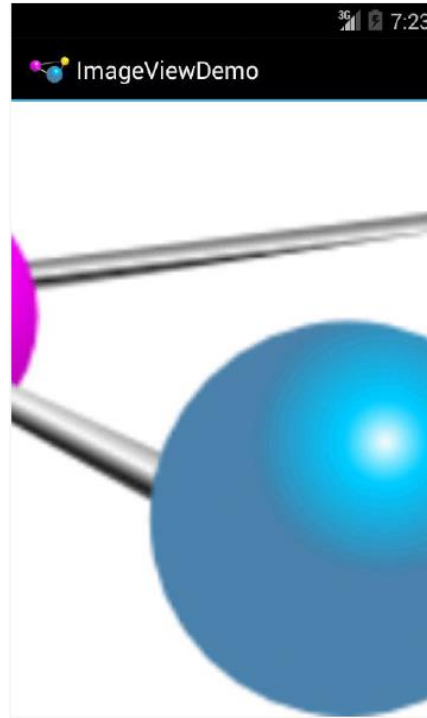




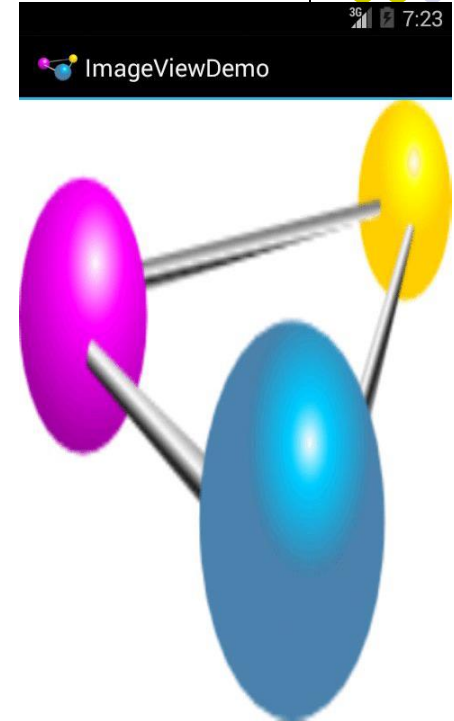
# Options for Scaling Images (scaleType)



“**center**” centers image but does not scale it



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



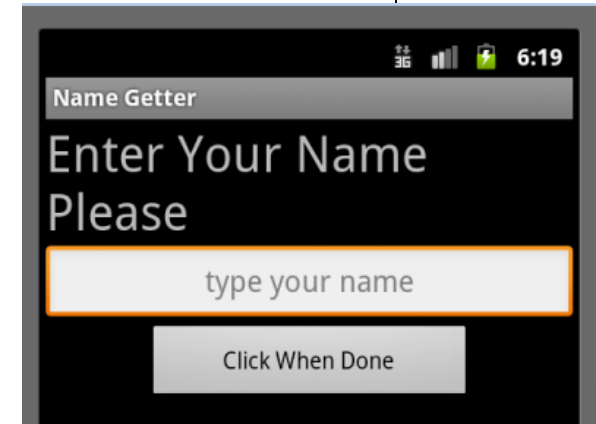
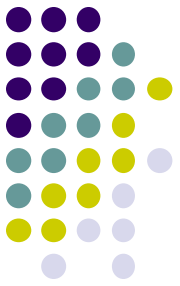
“**fitXY**” scales/distorts image to fit ImageView, ignoring aspect ratio

# EditText Widget

- Widget with box for user input
- Example:

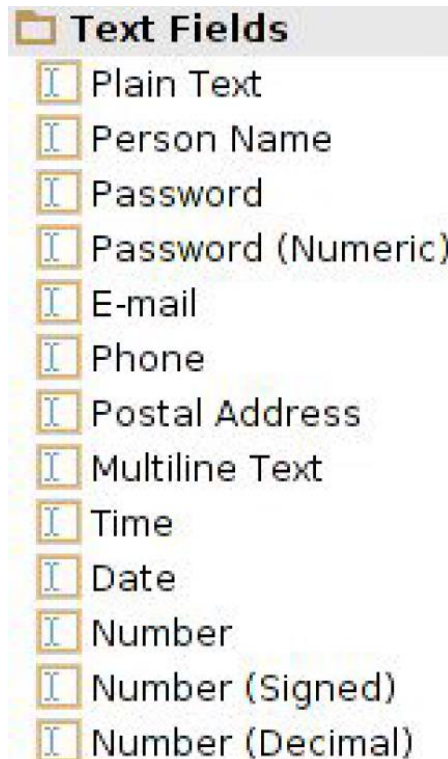
```
<EditText  
    android:id="@+id/edittext"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:gravity="center"  
    android:inputType="textPersonName"  
    android:hint="type your name" />
```

- 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)

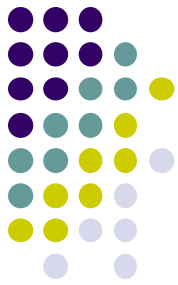


**Text Fields**  
Section of Widget  
palette

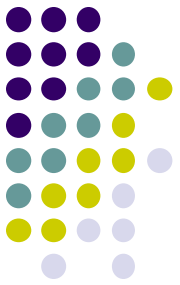
A screenshot of the EditText widget's 'inputType' menu. The menu is a list of input types, each with a small icon of a text field and a checkbox to its right:

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"/>

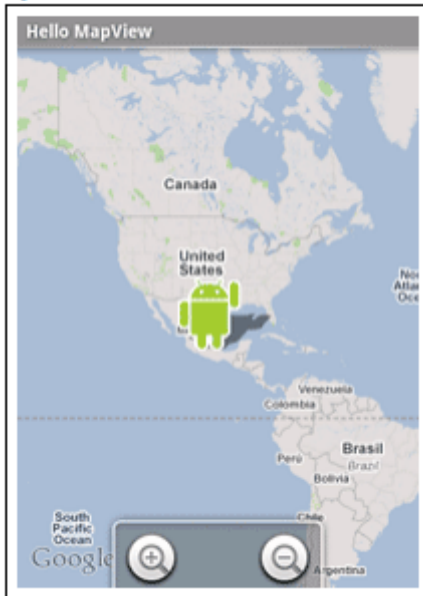
**EditText**  
**inputType** menu



# Some Other Available Widgets



MapView



WebView



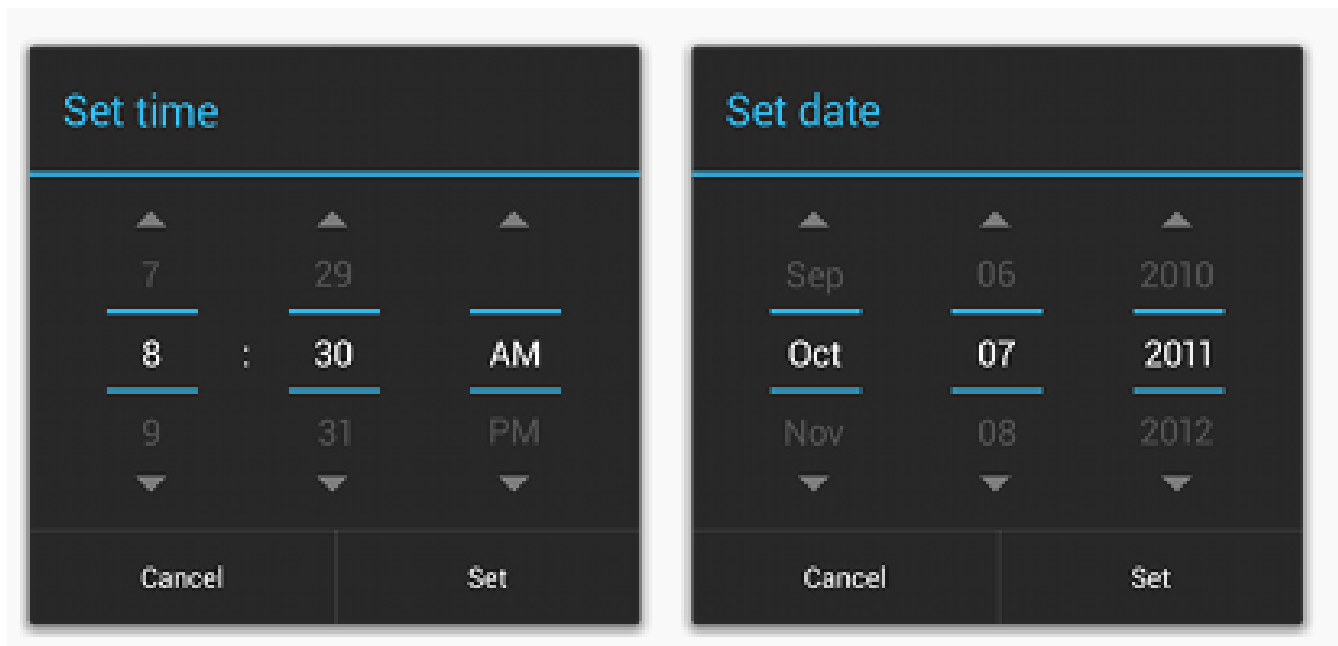
DatePicker





# Pickers

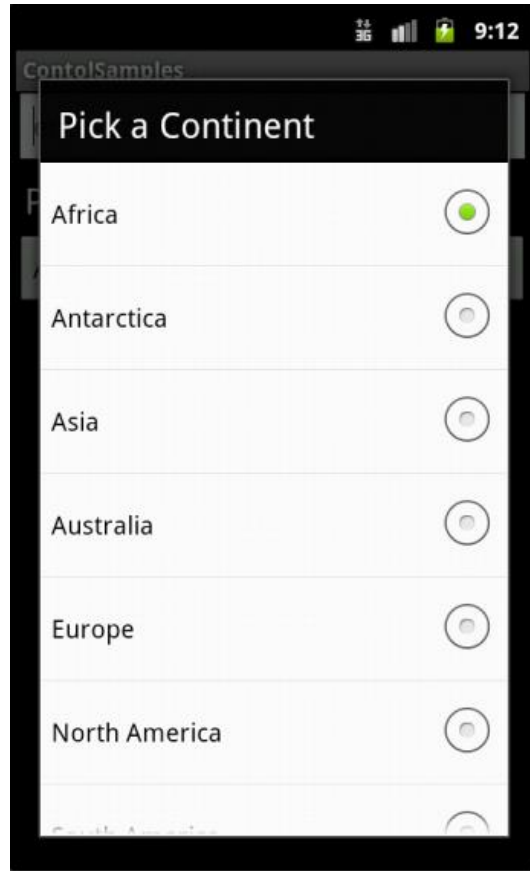
- **TimePicker:** Select a time
- **DatePicker:** Select a date
- Typically displayed in pop-up dialogs (**TimePickerDialog** or **DatePickerDialog**)

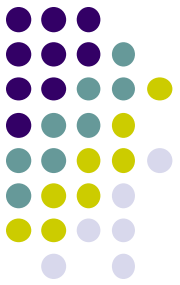


# Spinner Controls



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





# Checkbox

USB debugging

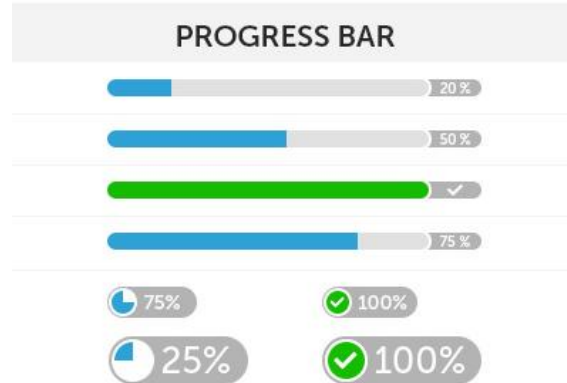
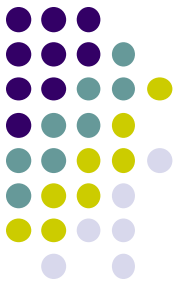
Debug mode when USB is connected



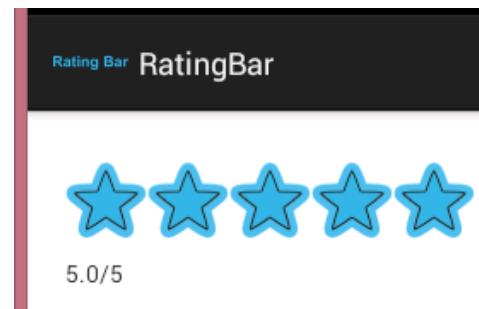
- Checkbox has 2 states: checked and unchecked
- XML code to create Checkbox

```
<?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

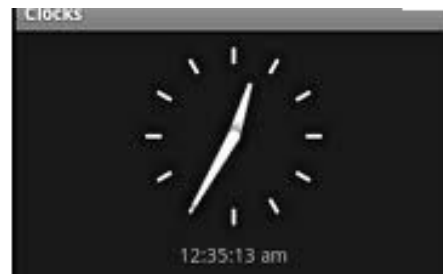


- ProgressBar

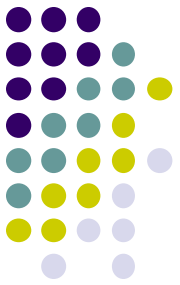


- RatingBar

- Chronometer
- DigitalClock
- AnalogClock







# Android Layouts in XML

# Android UI using XML Layouts

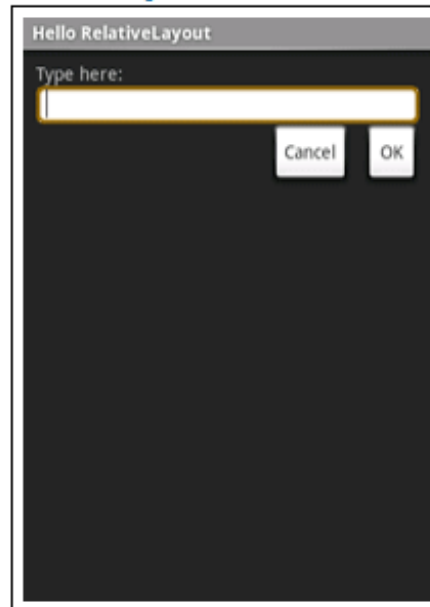


- Layout? Pattern in which multiple widgets are arranged
- Layouts contain widgets
- Layouts (XML files) stored in **res/layout**

LinearLayout



RelativeLayout

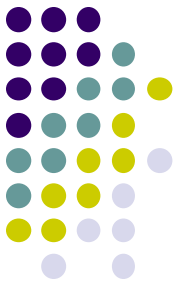


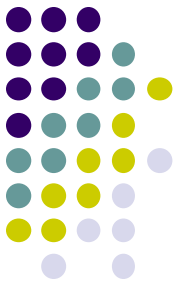
TableLayout



# Some Layouts

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





# LinearLayout

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

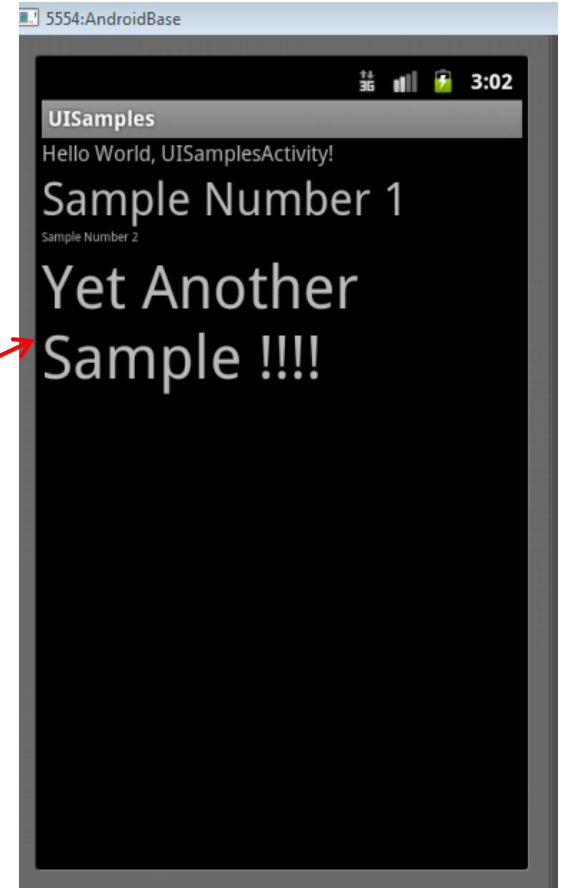
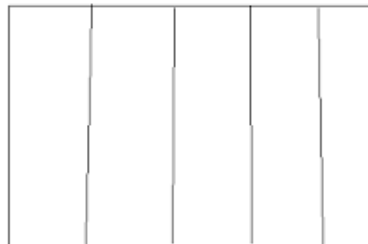
- E.g. `android:orientation="vertical"`

Linear Layout

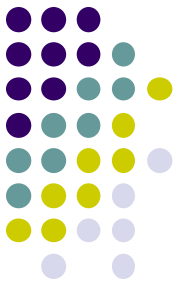
Orientation: vertical



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**

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

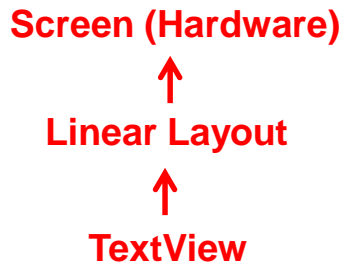
Text widget height should be as wide as the content (text)

```
<?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>
```

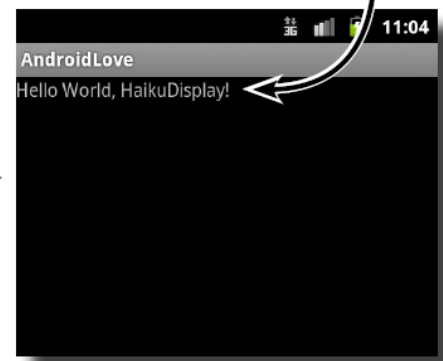
The View inside the layout is a TextView, a View specifically made to display text.



main.xml



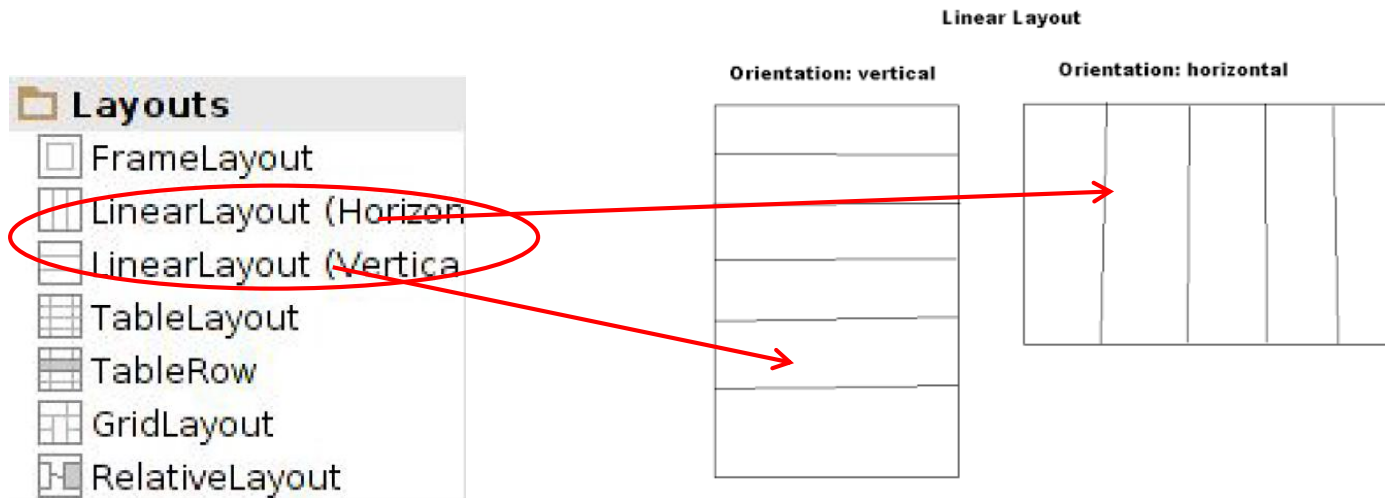
The ViewGroup, in this case a LinearLayout fills the screen.





# LinearLayout in Android Studio

- LinearLayout in Android Studio Graphical Layout Editor



- After selecting LinearLayout, toolbars buttons to set parameters



**Toggle width, height between  
match\_parent and wrap\_content**

**Change gravity of  
LinearLayout**

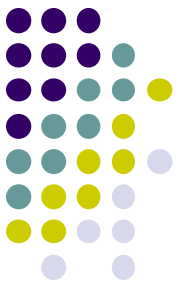
# LinearLayout Attributes



## 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.html>



# 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

```
<RelativeLayout ...
```

```
    android:paddingBottom="16dp"
```

```
    android:paddingLeft="16dp"
```

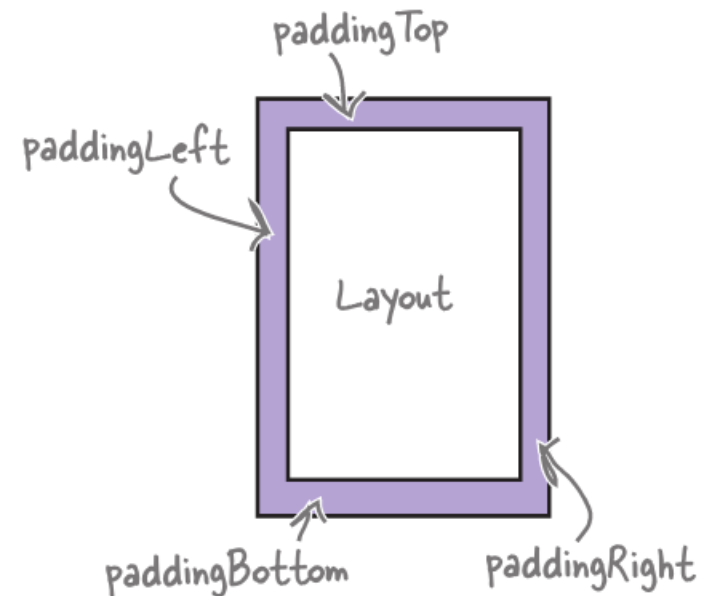
```
    android:paddingRight="16dp"
```

```
    android:paddingTop="16dp">
```

```
    ...
```

```
</RelativeLayout>
```

Add padding of 16dp.



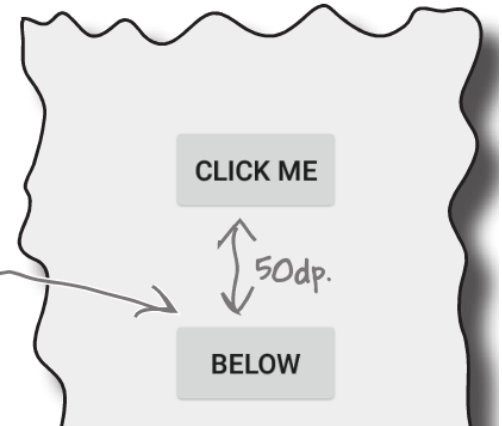
# Setting Margins



- 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

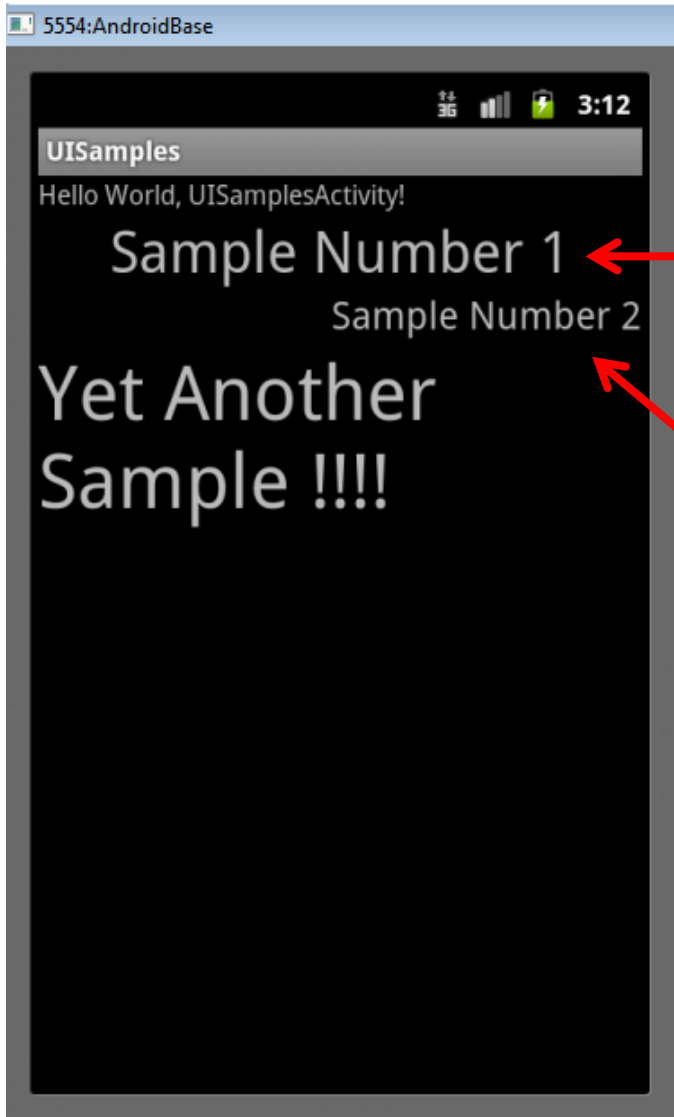
**android:layout\_marginLeft**



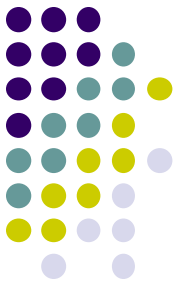
**android:layout\_marginRight**



# Gravity Attribute



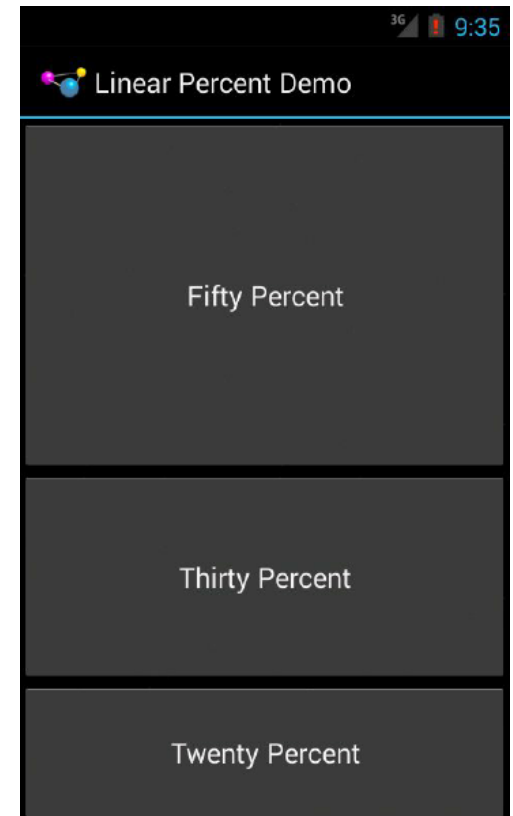
- By default, linearlayout left- and top-aligned
- Gravity attribute changes alignment :
  - e.g. `android:gravity = "right"`



# 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

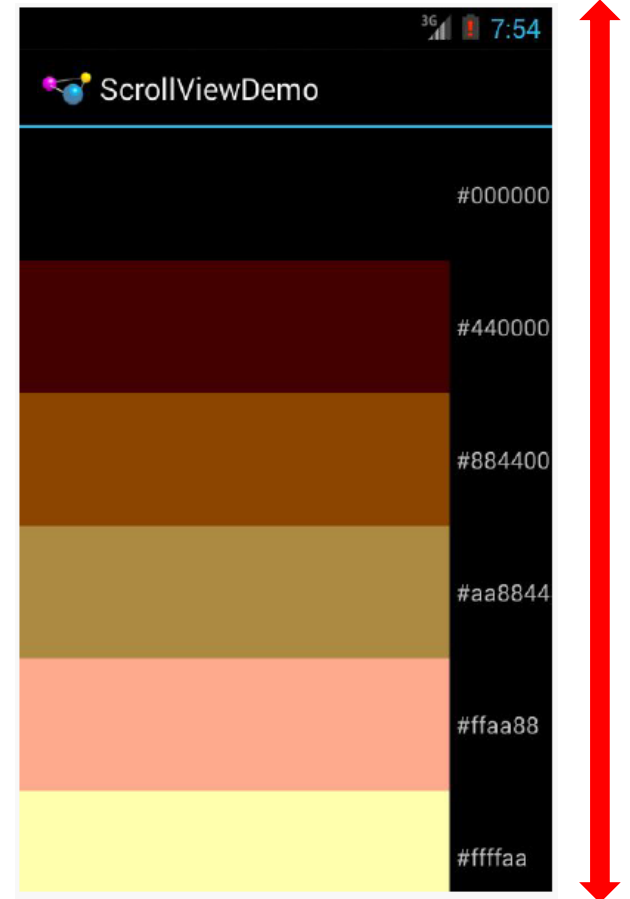
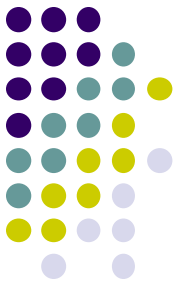
```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">  
  
    <Button  
        android:layout_width="match_parent"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

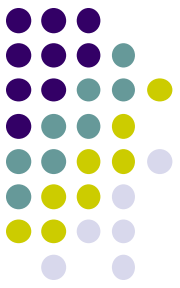
- Phone screens are small, scrolling content helps
- Examples: Scroll through
  - large image
  - Linear Layout with lots of elements
- Views for Scrolling:
  - **ScrollView** for vertical scrolling
  - **HorizontalScrollView**
- Rules:
  - Only one direct child View
  - Child could have many children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

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

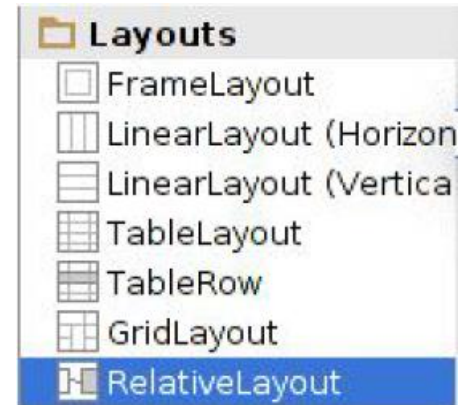
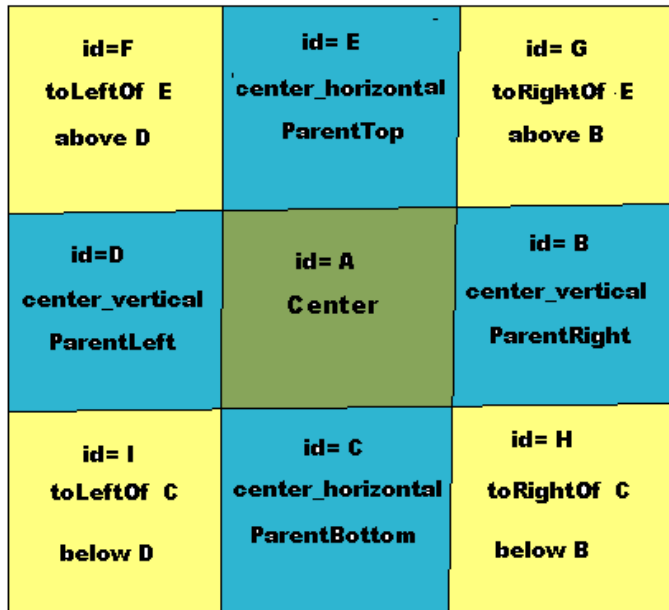


# RelativeLayout

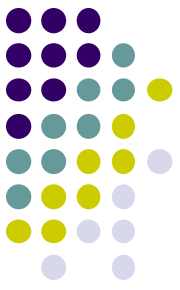
- First element listed is placed in "center"
- Positions of children specified relative to parent or to each other.



Relative Layout



**RelativeLayout available  
In Android Studio palette**

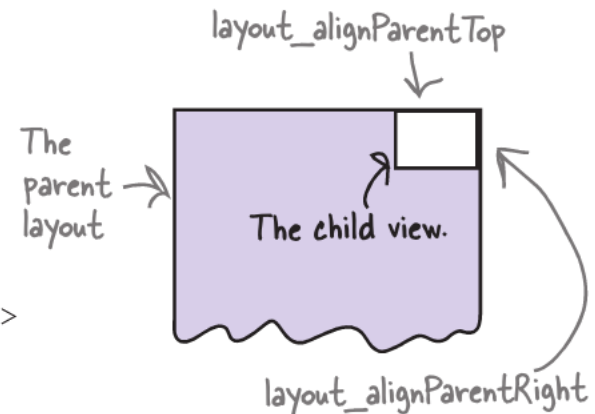


# Positioning Views Relative to Parent Layout

- 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>
```

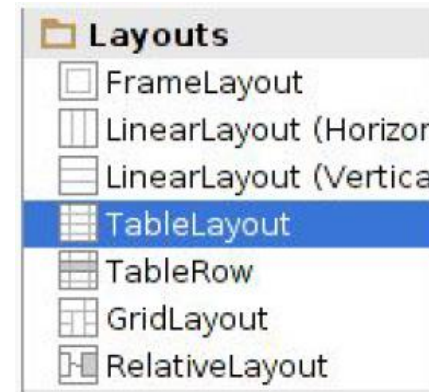
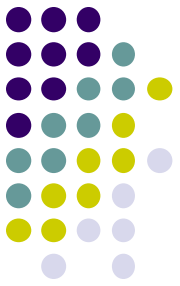
The layout contains the button, so the layout is the button's parent.



See [Head First Android Development page 169](#) for more examples

# Table Layout

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

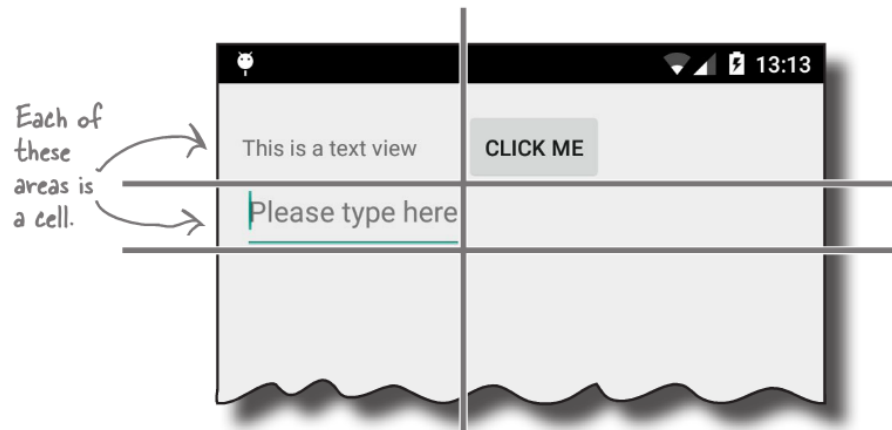






# GridLayout

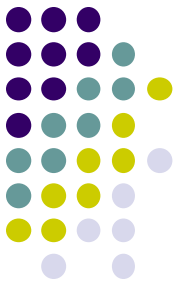
- In `TableLayout`, Rows can span multiple columns only
- In `GridLayout`, child views/controls can span multiple rows **AND** columns



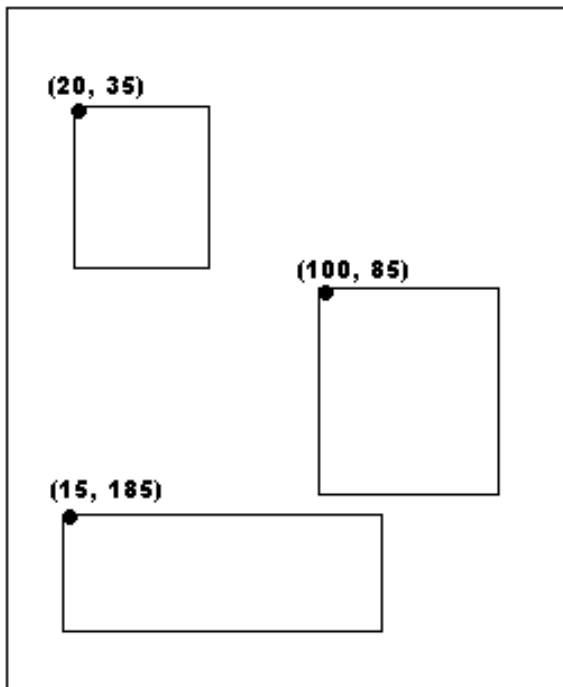
- See section “`GridLayout` Displays Views in a Grid” in *Head First Android Development* (pg 189)

# Absolute Layout

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



## Absolute Layout

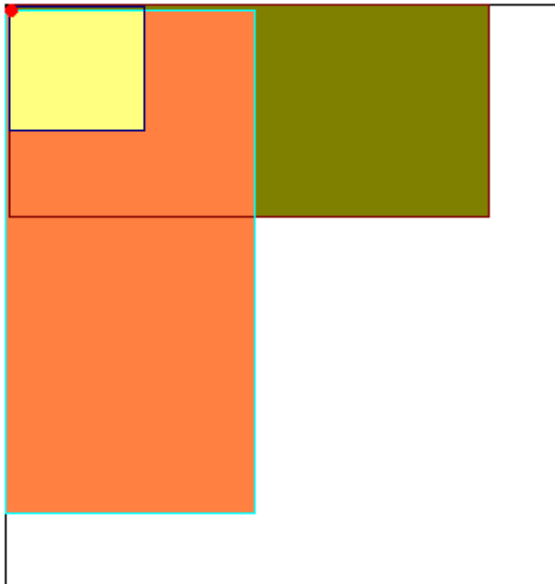




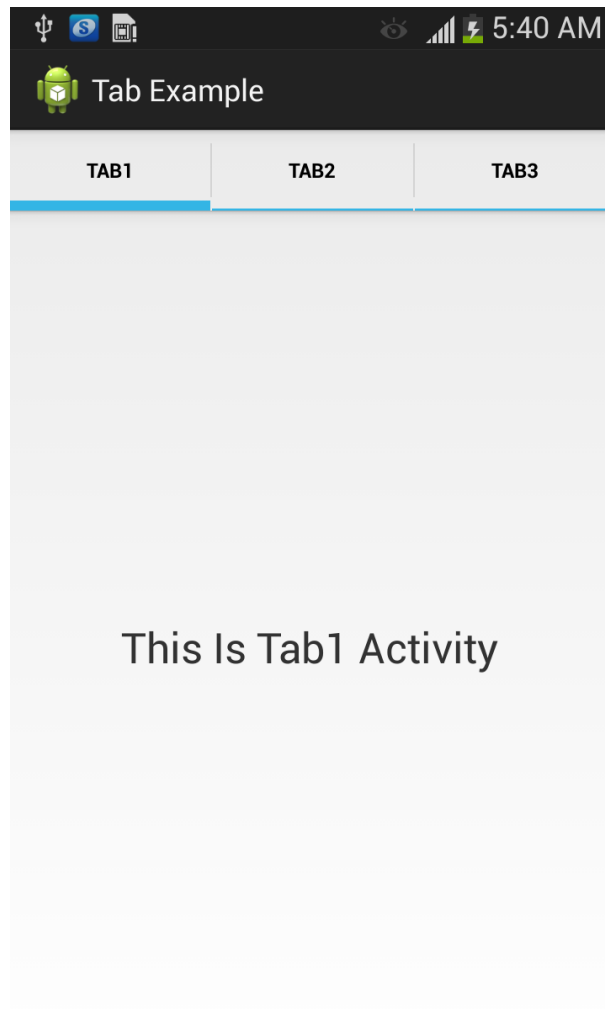
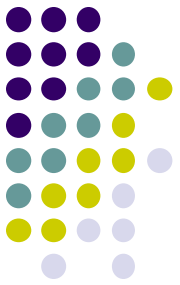
# FrameLayout

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

Frame Layout



# Other Layouts: Tabbed Layouts



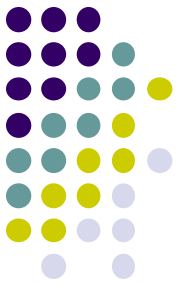
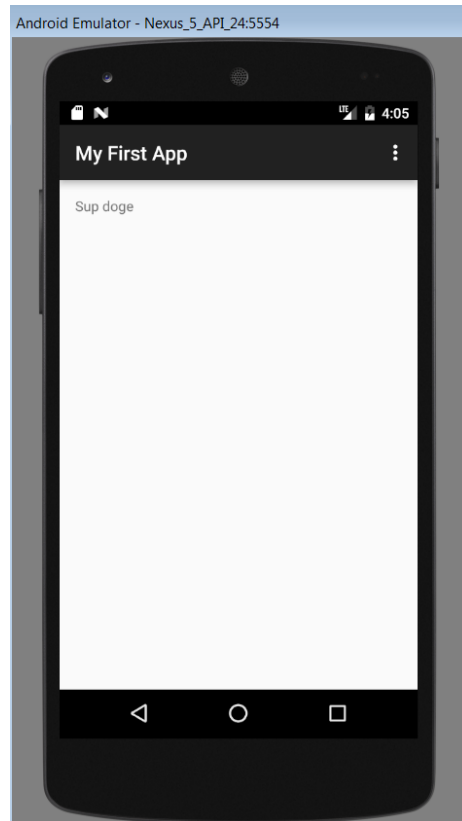


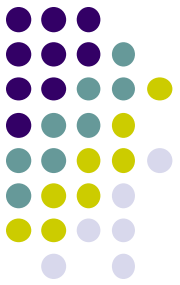
# **Android Example: My First App**

## **(Ref: Head First Android)**

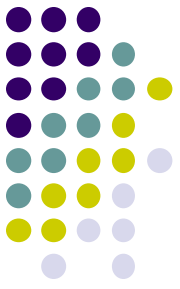
# My First App

- Hello World program in Head First Android Development (Chapter 1)
- Creates app, types “Sup doge” in a TextView





# Android UI Youtube Tutorials



# YouTube Tutorial 11 & 12 from thenewBoston

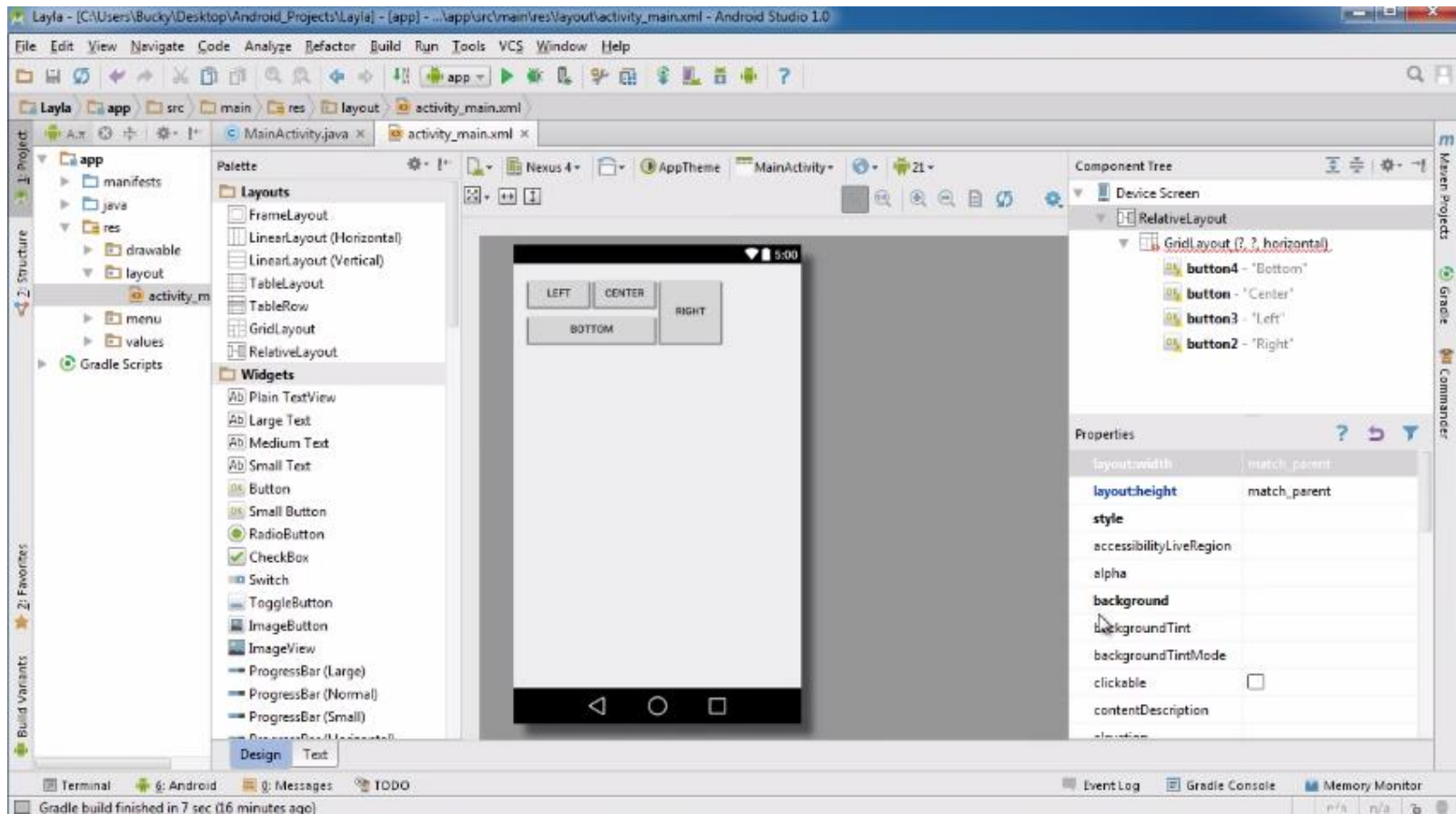
- Tutorial 11: Designing the User Interface [6:19 mins]
  - <https://www.youtube.com/watch?v=72mf0rmjNAA>
  - Designing the UI
  - Adding activity (screen)
  - Dragging in widgets
  - Changing the text in widgets
  
- Tutorial 12: More on User Interface [10:24 mins]
  - <https://www.youtube.com/watch?v=72mf0rmjNAA>
  - Changing text in widgets
  - Changing strings from hardcoded to resources (variables)

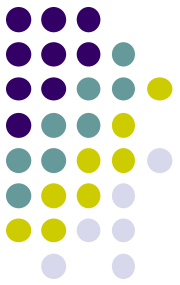


# Tutorial 17: GridLayout



- Tutorial 17: GridLayout [9:40 mins]  
(<https://www.youtube.com/watch?v=4bXOr5Rk1dk>)
  - Creating GridLayout: Layout that places its children in a grid
  - Add widgets (buttons) to GridLayout
  - Format width, height, position of widgets





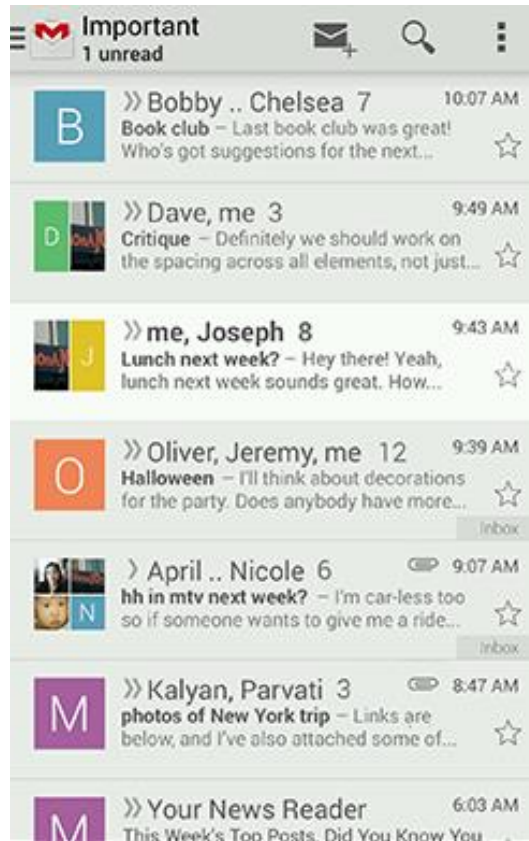
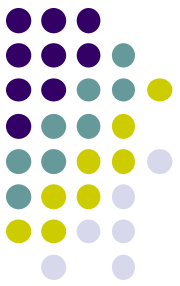
# Android Themes

# Styles

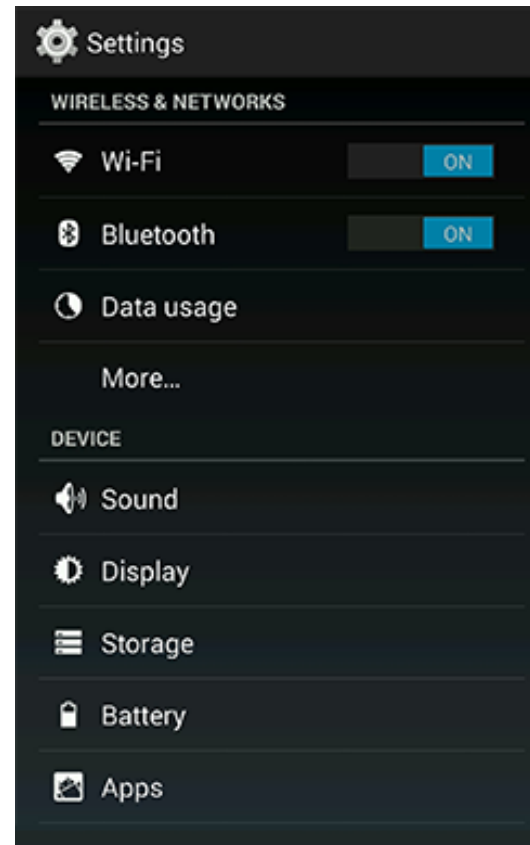


- Android widgets have properties
  - E.g. Foreground color = red
- **Styles in Android:** specifies properties for multiple attributes of **1 widget**
  - E.g. height, padding, font color, font size, background color
- Similar to Cascaded Style Sheets (CSS) in HTML
- Themes apply styles to **all widgets in an Activity (screen)**
  - E.g. all widgets on a screen can adopt the same font
- Example Android themes: Theme, Theme.holo and Theme.material)

# Examples of Themes in Use



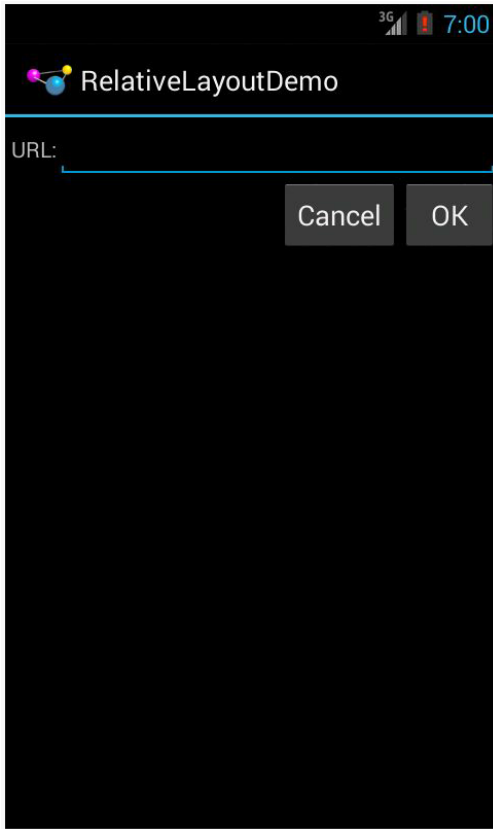
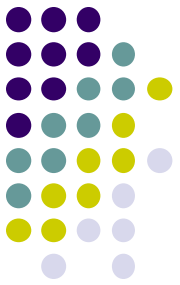
**GMAIL in Holo Light**



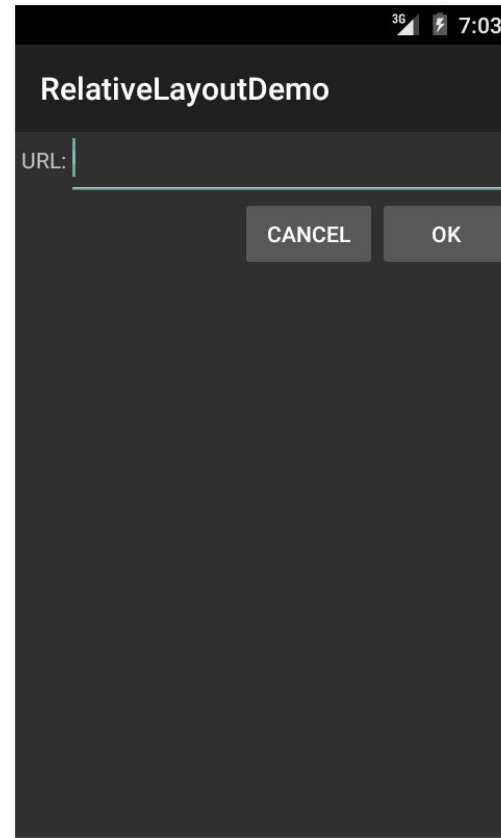
**Settings screen in Holo Dark**

# Default Themes

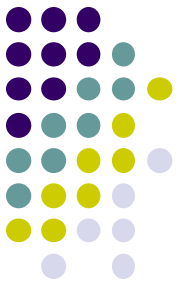
- Many stock themes to choose from
- Android chooses a default theme if you specify none



**Theme.Holo:** default theme in Android 3.0

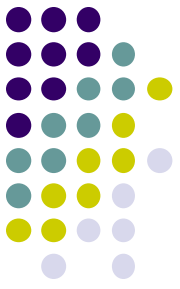


**Theme.Material:** default theme in Android 5.0

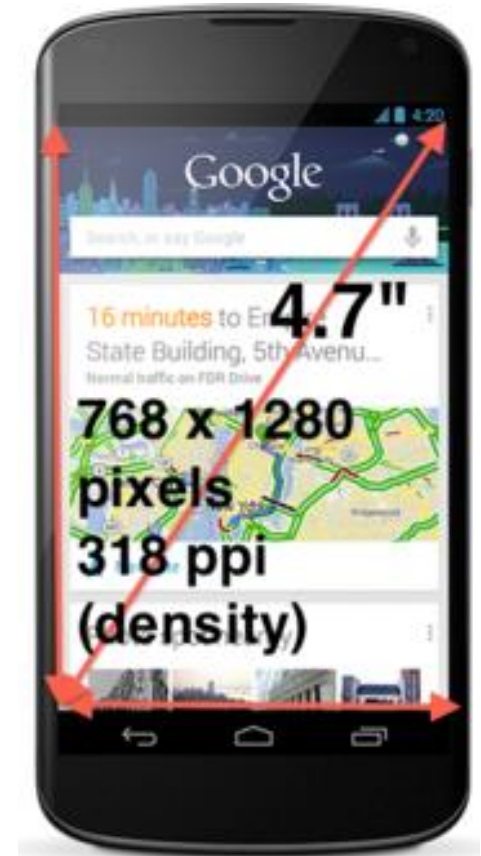


# Adding Pictures in Android

# Phone Dimensions Used in Android UI



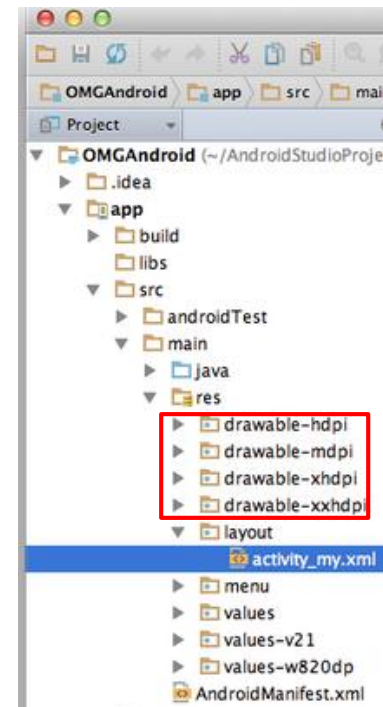
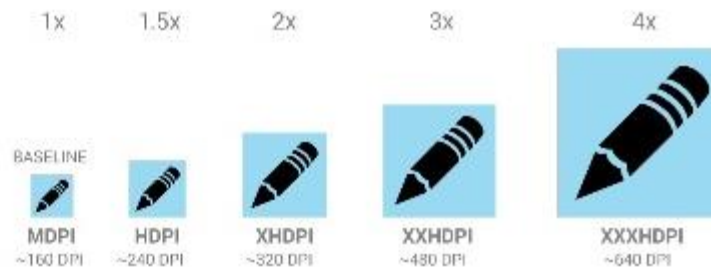
- Physical dimensions (inches) diagonally
  - E.g. Nexus 4 is 4.7 inches diagonally
- Resolution in pixels
  - E.g. Nexus 4 resolution 768 x 1280 pixels
  - Pixels diagonally:  $\text{Sqrt}[(768 \times 768) + (1280 \times 1280)]$
- Pixels per inch (PPI) =
  - $\text{Sqrt}[(768 \times 768) + (1280 \times 1280)] / 4.7 = 318$



# Adding Pictures

- Android supports images in PNG, JPEG and GIF formats
- Put different resolutions of **same image** into different directories
  - **res/drawable-ldpi**: low dpi images (~ 120 dpi of dots per inch)
  - **res/drawable-mdpi**: medium dpi images (~ 160 dpi)
  - **res/drawable-hdpi**: high dpi images (~ 240 dpi)
  - **res/drawable-xhdpi**: extra high dpi images (~ 320 dpi)
  - **res/drawable-xxhdpi**: extra extra high dpi images (~ 480 dpi)
  - **res/drawable-xxxhdpi**: high dpi images (~ 640 dpi)

**res/drawable-mdpi**  
**res/drawable-tvdpi**  
**res/drawable-hdpi**  
**res/drawable-xhdpi**  
**res/drawable-xxhdpi**  
**res/drawable-xxxhdpi**





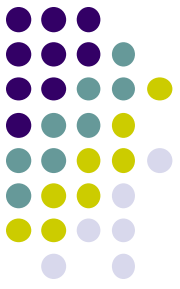
# Adding Pictures



- Use generic picture name in code (no .png, .jpg, etc)
  - E.g. to reference an image **ic\_launcher.png**

```
<application
  android:allowBackup="false"
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme">
```

- At run-time, Android chooses which resolution/directory (e.g. –mdpi) based on phone resolution
- **Image Asset Studio:** generates icons in various densities from original image  
Ref: <https://developer.android.com/studio/write/image-asset-studio.html>



# References

- Android App Development for Beginners videos by Bucky Roberts (thenewboston)
- 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](https://www.youtube.com/watch?v=n5Yjzn3b_aQ)
- 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