CS 528 Mobile and Ubiquitous Computing

Lecture 2: Intro to Android Programming

Emmanuel Agu

Students: Please Introduce Yourselves!

- Name
- Status: grad/undergrad, year
- Relevant background: e.g. coal miner ☺
- Relevant courses taken:
  - *Systems:* Networks, OS,
  - *Advanced:* machine learning, advanced networks, etc
- What you would like to get out of this class? E.g
  - Understanding a hot field
  - Just a class for masters degree/PhD
  - Looking for research area, masters thesis, PhD thesis
  - Compliments your current research interests/publications
  - My spouse told me to ☺

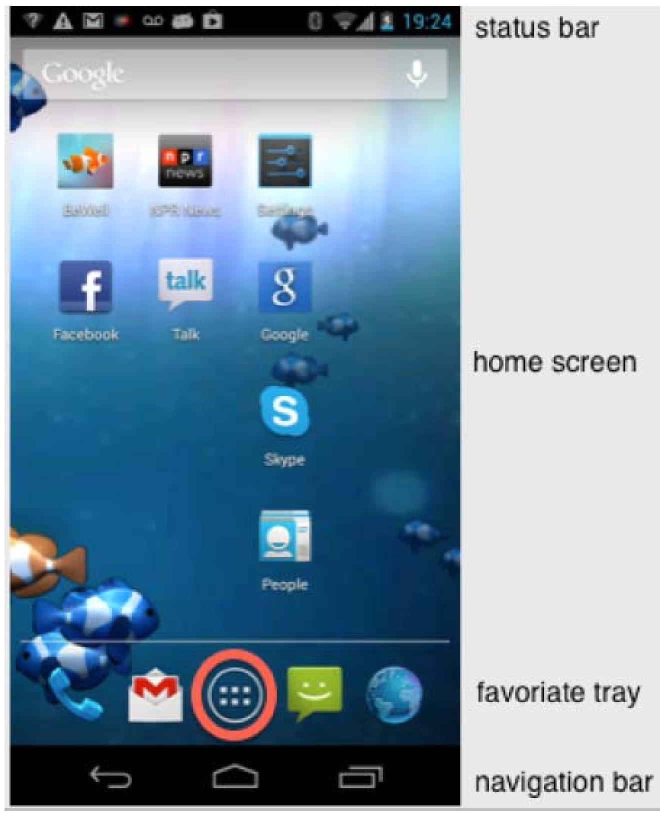
Android UI Tour

Android UI Tour

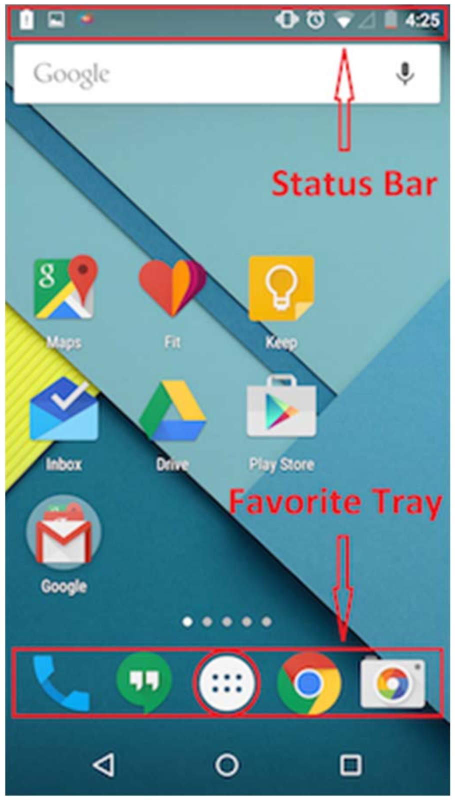
- Android UI has many standard components that developers can reuse
- Next: Review app screens and usage of UI components in popular apps
- Why? These components will also be mentioned in developing Android apps

Home Screen

- First screen after unlocking phone or hitting **home** button
- Includes **favorites** tray (e.g phone, mail, messaging, web, etc)



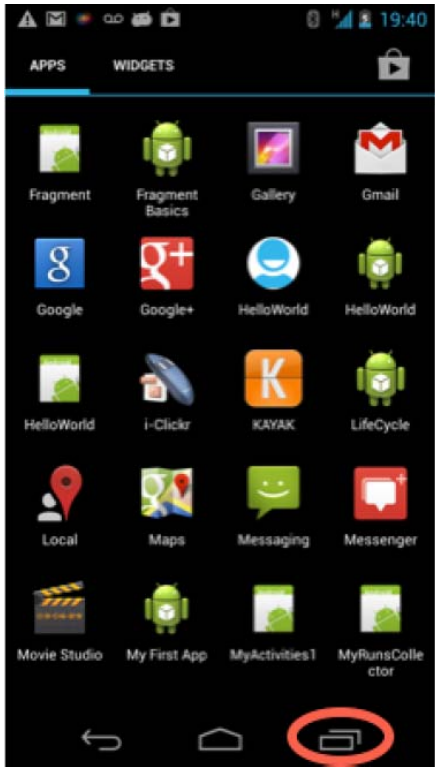
Android 4.0



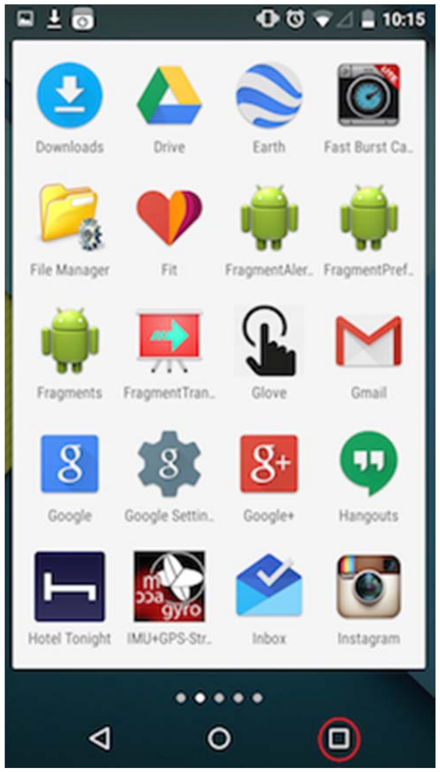
Android 5.0

All Apps Screen

- Accessed by touching **all apps button** in favorites tray
- Users can swipe through multiple app and widget screens
- Can be customized by dragging and dropping items



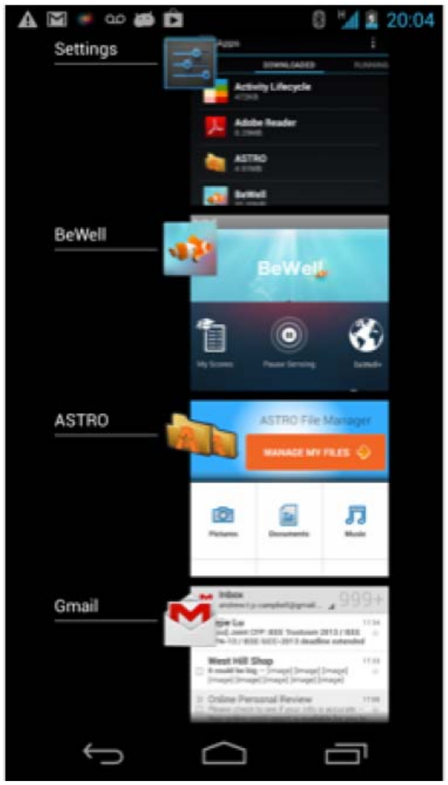
Android 4.0



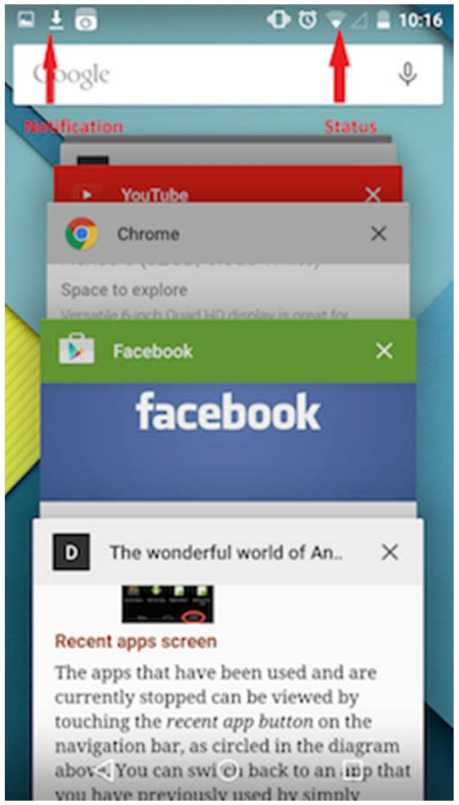
Android 5.0

Recent Apps Screen

- Accessed by touching the **recent apps button**
- Shows recently used and currently stopped apps
- Can switch to a recently used app by touching it in list



Android 4.0



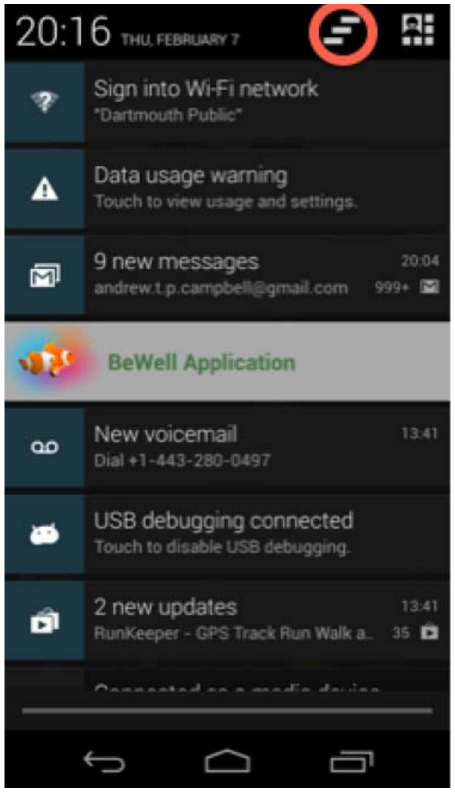
Recent apps screen

The apps that have been used and are currently stopped can be viewed by touching the *recent app button* on the navigation bar, as circled in the diagram above. You can switch back to an app that you have previously used by simply

Android 5.0

Status Bar and Notification Screen

- Displays notifications (on left) and status (on right)
- **Status:** time, battery, cell signal strength, bluetooth enabled, etc
- **Notification:** wifi, mail, bewell, voicemail, usb active, music, etc


20:16 THU, FEBRUARY 7

Sign into Wi-Fi network
"Dartmouth Public"

Data usage warning
Touch to view usage and settings.

9 new messages 20:04
andrew.t.p.campbell@gmail.com 999+

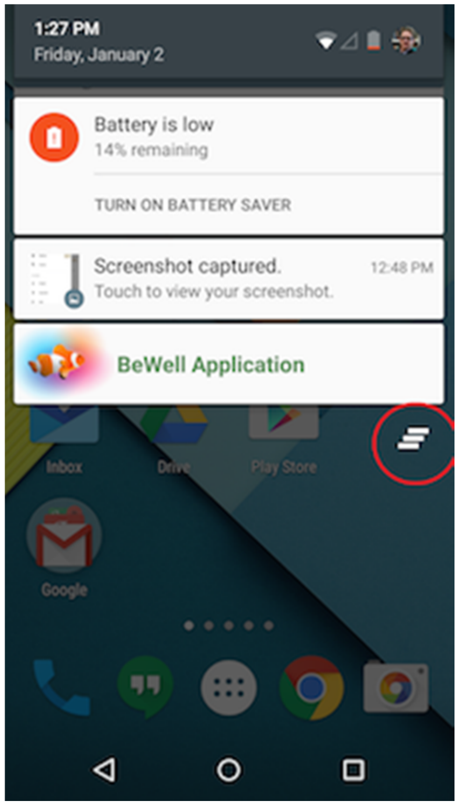
BeWell Application

New voicemail 13:41
Dial +1-443-280-0497

USB debugging connected
Touch to disable USB debugging.

2 new updates 13:41
RunKeeper - GPS Track Run Walk a... 35

Android 4.0


1:27 PM
Friday, January 2

Battery is low
14% remaining
TURN ON BATTERY SAVER

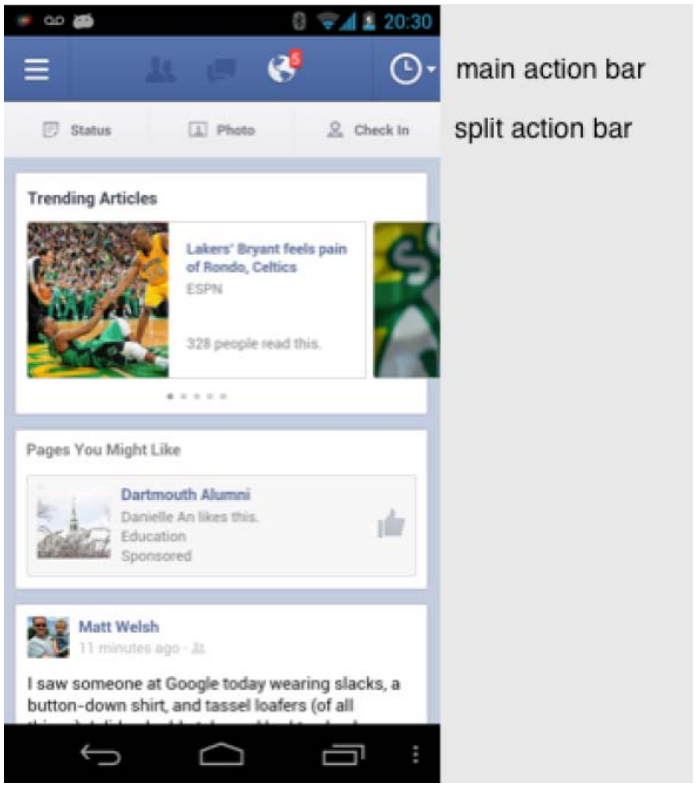
Screenshot captured. 12:48 PM
Touch to view your screenshot.

BeWell Application

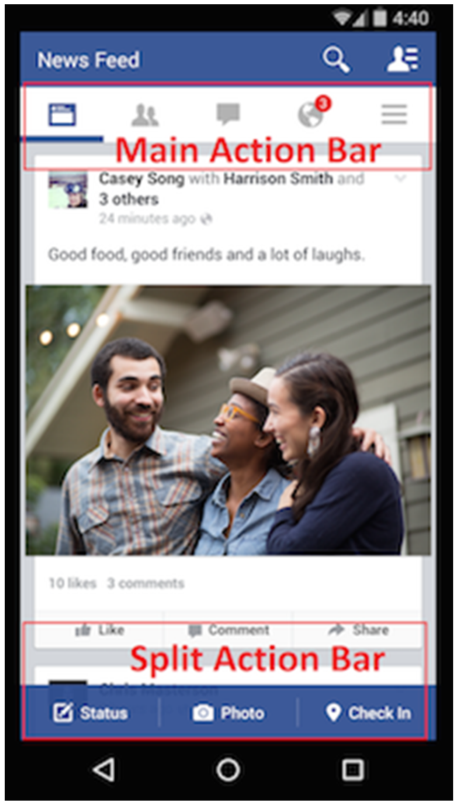
Android 5.0

Facebook UI

- Uses many standard Android UI components
- Shows main action bar and split action bar
- **Action bar:** configurable, handles user action and app navigation



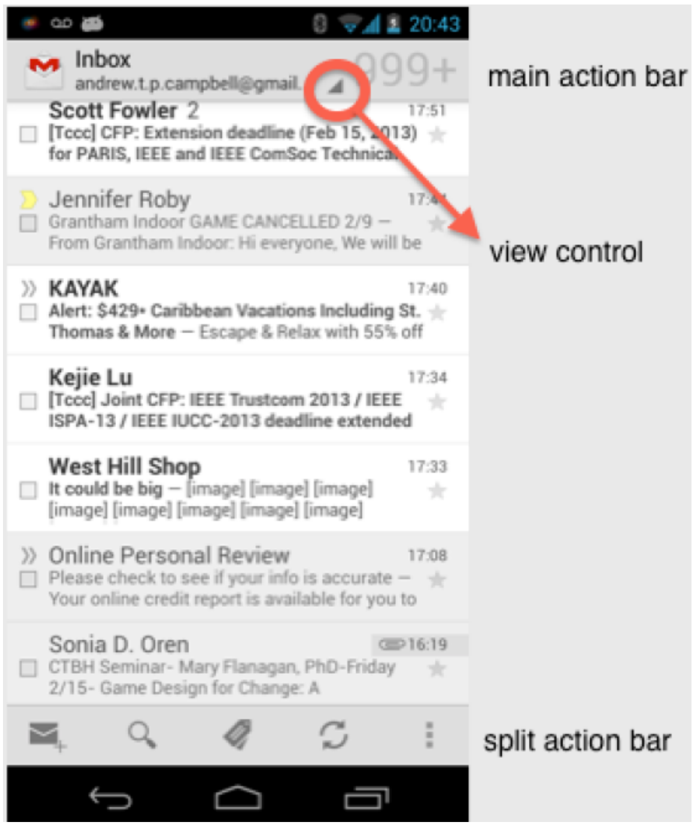
Android 4.0



Android 5.0

Gmail

- Split action bar at bottom of screen
- **View control:** allows users switch between different views (inbox, recent, drafts, sent)



20:43
Inbox
andrew.t.p.campbell@gmail. 999+

Scott Fowler 2 — 17:51
[Tccc] CFP: Extension deadline (Feb 15, 2013) for PARIS, IEEE and IEEE ComSoc Technical

Jennifer Roby — 17:4
Grantham Indoor GAME CANCELLED 2/9 — From Grantham Indoor: Hi everyone, We will be

KAYAK — 17:40
Alert: $429+ Caribbean Vacations Including St. Thomas & More — Escape & Relax with 55% off

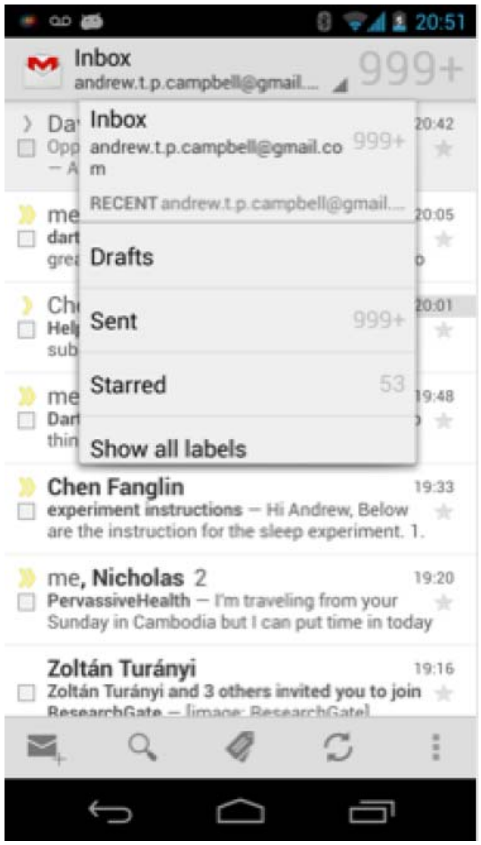
Kejie Lu — 17:34
[Tccc] Joint CFP: IEEE Trustcom 2013 / IEEE ISPA-13 / IEEE IUCC-2013 deadline extended

West Hill Shop — 17:33
It could be big — [image] [image] [image] [image] [image] [image] [image] [image]

Online Personal Review — 17:08
Please check to see if your info is accurate — Your online credit report is available for you to

Sonia D. Oren — 16:19
CTBH Seminar- Mary Flanagan, PhD-Friday 2/15- Game Design for Change: A

Android 4.0



20:51
Inbox
andrew.t.p.campbell@gmail... 999+

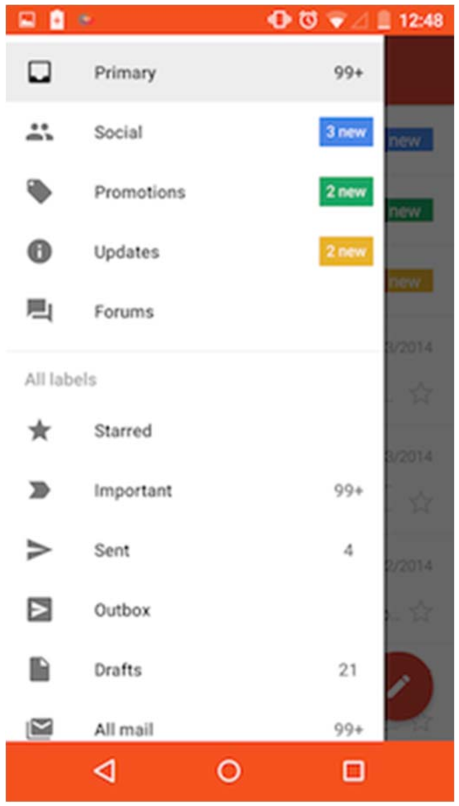
Inbox
andrew.t.p.campbell@gmail.com 999+
RECENT andrew.t.p.campbell@gmail...
Drafts
Sent 999+
Starred 53
Show all labels

Chen Fanglin — 19:33
experiment instructions — Hi Andrew, Below are the instruction for the sleep experiment. 1.

me, Nicholas 2 — 19:20
PervassiveHealth — I'm traveling from your Sunday in Cambodia but I can put time in today

Zoltán Turányi — 19:16
Zoltán Turányi and 3 others invited you to join ResearchGate — [image: ResearchGate]

Android 4.0

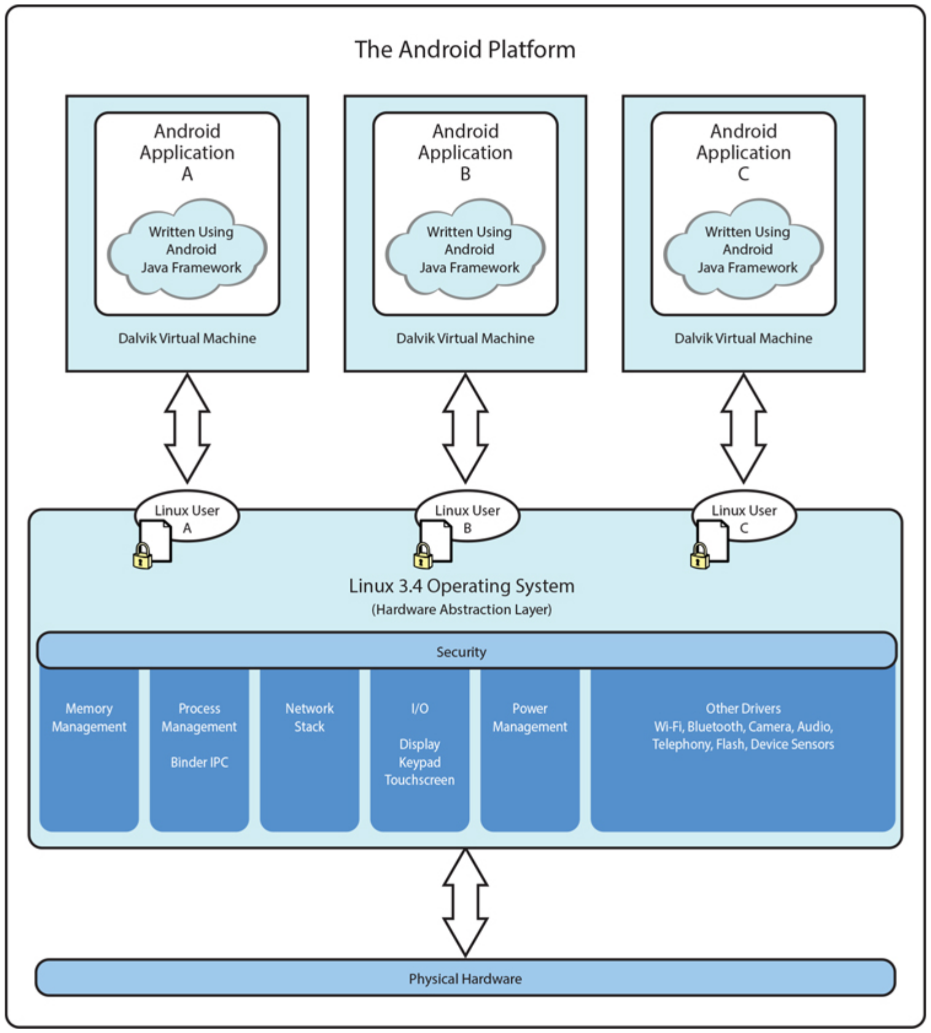


12:48

Primary	99+
Social	3 new
Promotions	2 new
Updates	2 new
Forums	
All labels	
Starred	
Important	99+
Sent	4
Outbox	
Drafts	21
All mail	99+

Android 5.0

Android Software Framework



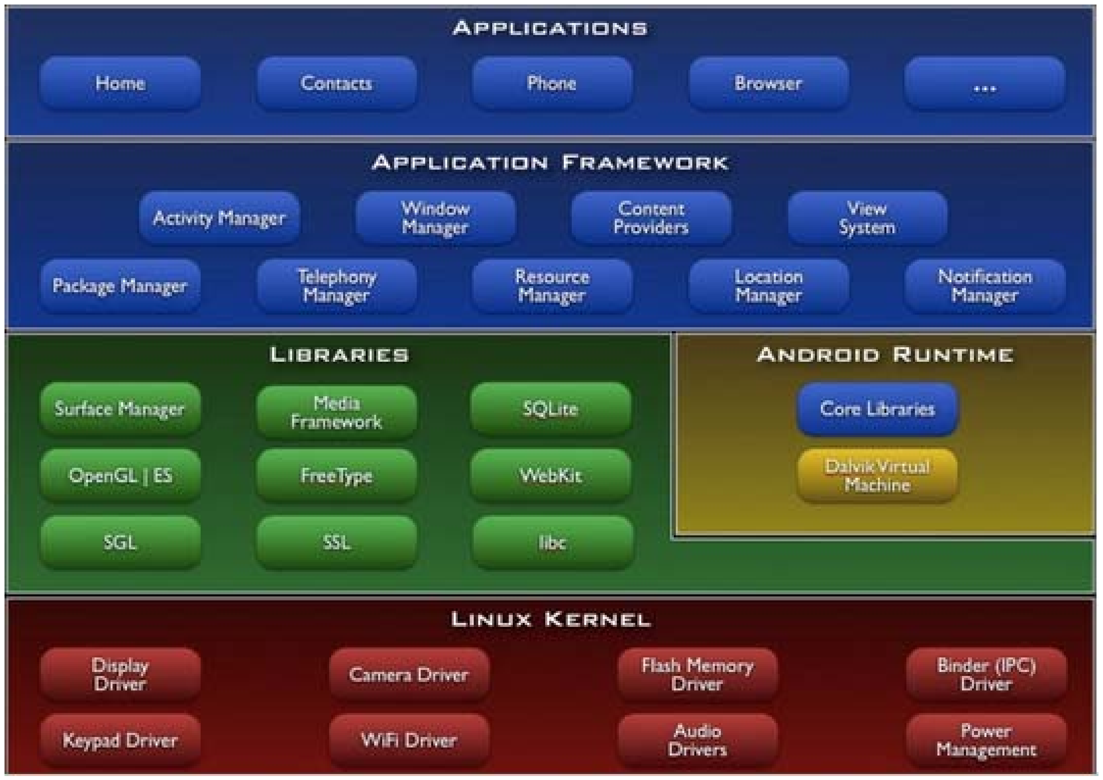
Android Software Framework

- Each Android app runs in its own security sandbox (VM, minimizes complete system crashes)
- Android OS multi-user Linux system
- Each app is a different user
- Application's files are private
- Android starts app's process when its components need to be executed, shuts down the process when no longer needed

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder

Android Software Framework

- Android system assigns each app a unique Linux user ID
  - ID is unknown to the application
- Android system sets permissions for all an app's files so that only the process with the app's user ID can access them

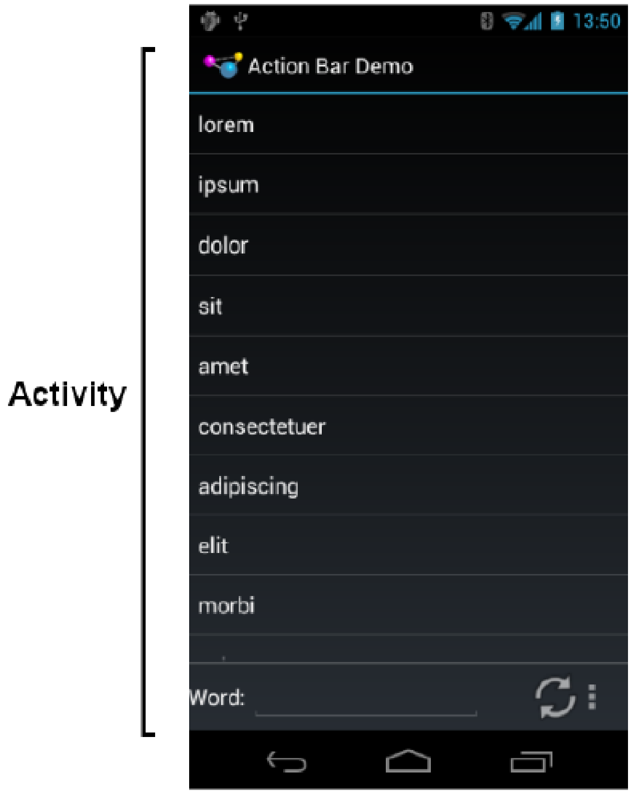


Our First Android App

Activities

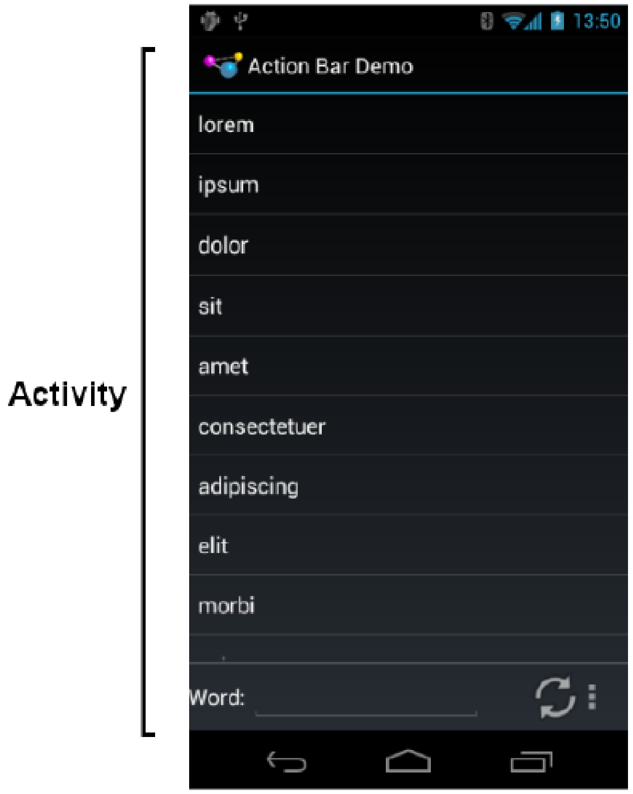
- Activity: main building block of Android UI
- Analogous to a window or dialog box in a desktop application
- Apps
  - have at least 1 activity that deals with UI
  - Entry point of app similar to **main( )** in C
- Typically have multiple activities

- Example: A camera app
  - **Activity 1:** to focus, take photo, start activity 2
  - **Activity 2:** to present photo for viewing, save it



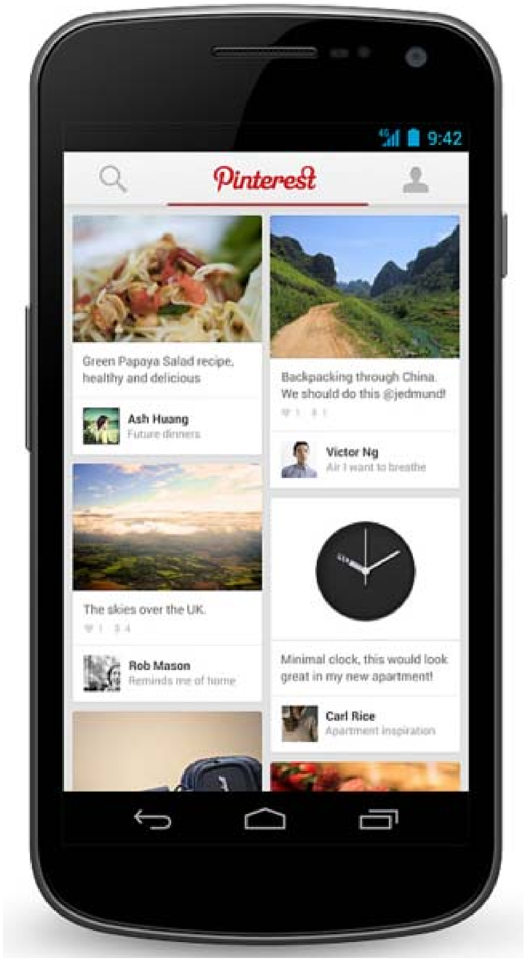
Activities

- Each activity controls 1 or more screens
- Activities independent of each other
- Can be coupled by control or data
- App Activities are sub-class of **Activity** class



Recall: Files Hello World Android Project

- 3 Files:
  - **Activity_my.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all screens, components of app
    - How these components attach themselves to overall Android system
    - Analogous to 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), launches activity with a tag “LAUNCHER”


9:42
Pinterest
Green Papaya Salad recipe, healthy and delicious
Ash Huang
Backpacking through China. We should do this @jedmund!
Victor Ng
The skies over the UK.
Rob Mason
Minimal clock, this would look great in my new apartment!
Carl Rice

Execution Order

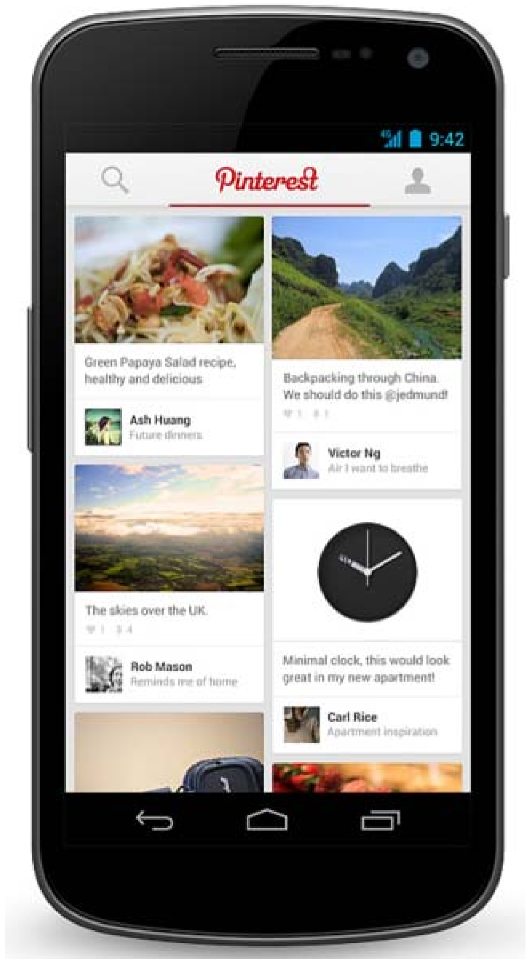
Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity
tagged Launcher

↓

Create/execute activities
(declared in java files)
E.g. MainActivity.Java

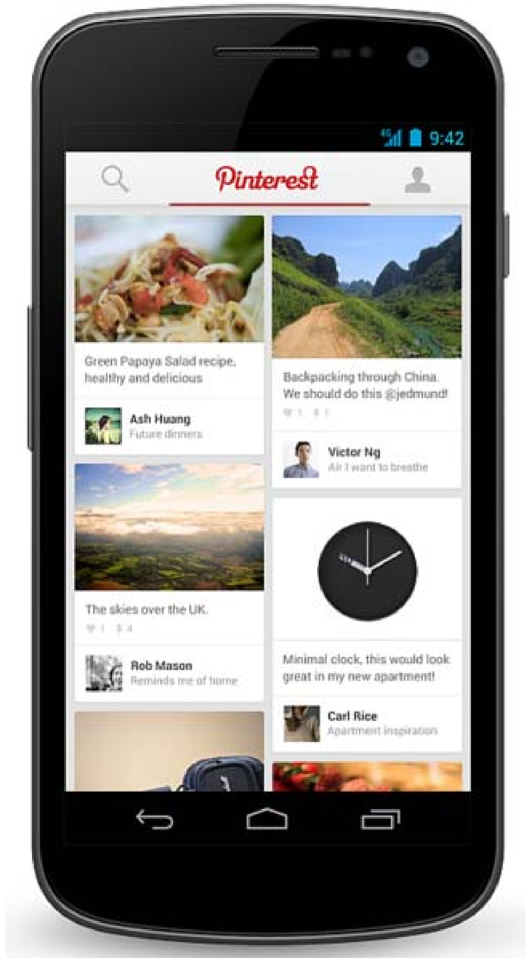
↓

Format each activity using layout
In XML file (e.g. Activity_my.xml)


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

Recall: Files Hello World Android Project

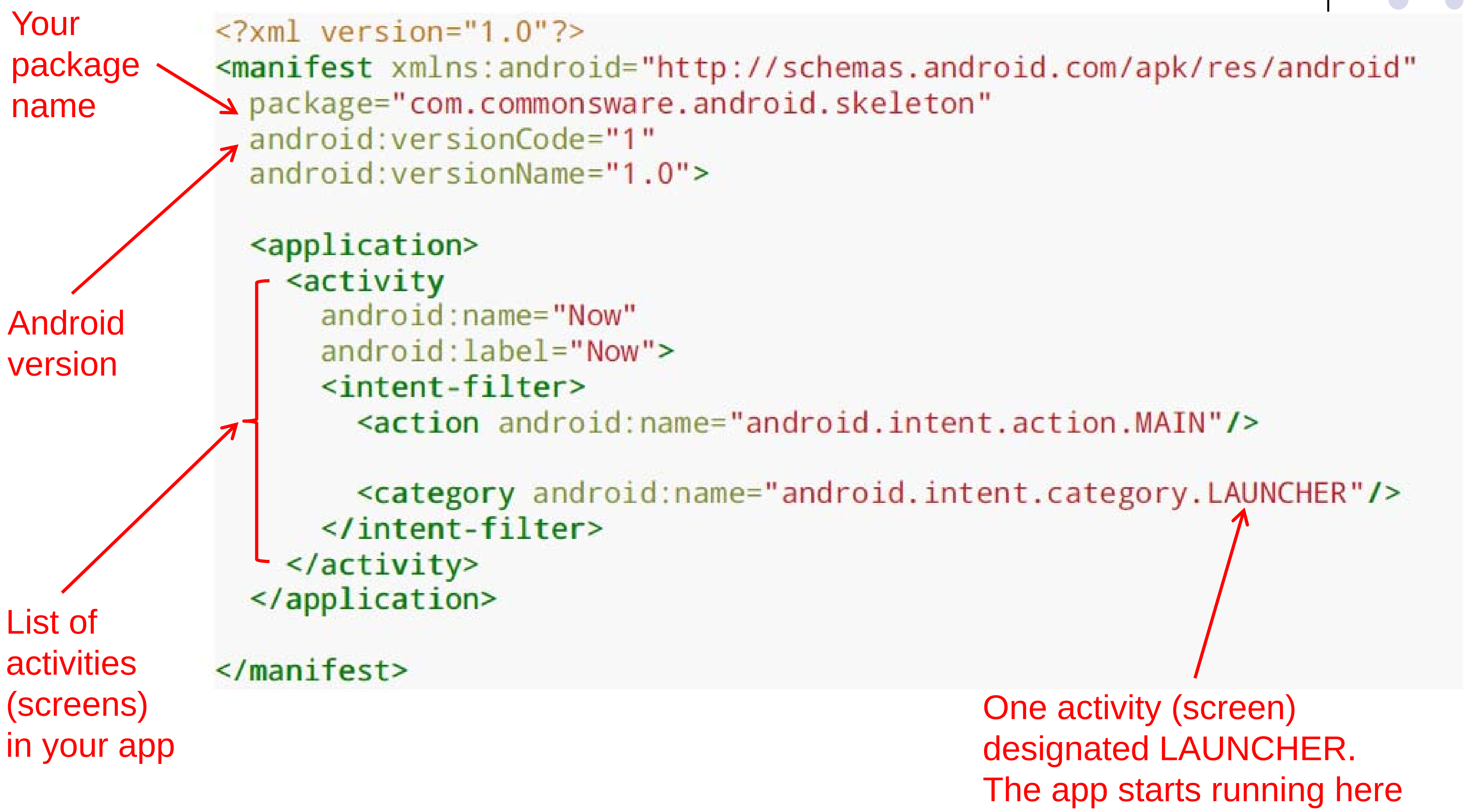
- 3 Files:
  - **Activity_my.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all screens, components of app
    - How these components attach themselves to overall Android system
    - Analogous to 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

Next: Let’s look at AndroidManifest.XML

Recall: Inside "Hello World" AndroidManifest.xml



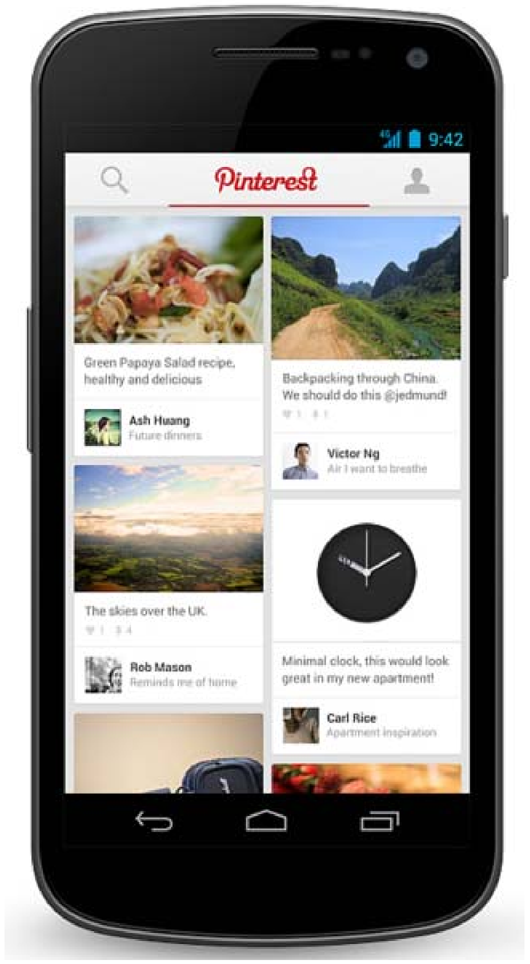
```
<?xml version="1.0"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.commonsware.android.skeleton"
  android:versionCode="1"
  android:versionName="1.0">

  <application>
    <activity
      android:name="Now"
      android:label="Now">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>

</manifest>
```

Your package name
Android version
List of activities (screens) in your app
One activity (screen) designated LAUNCHER. The app starts running here

Recall: Files Hello World Android Project

- 3 Files:
  - **Activity_my.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all screens, components of app
    - How these components attach themselves to overall Android system
    - Analogous to 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"

Next: Let's look at Simple java file



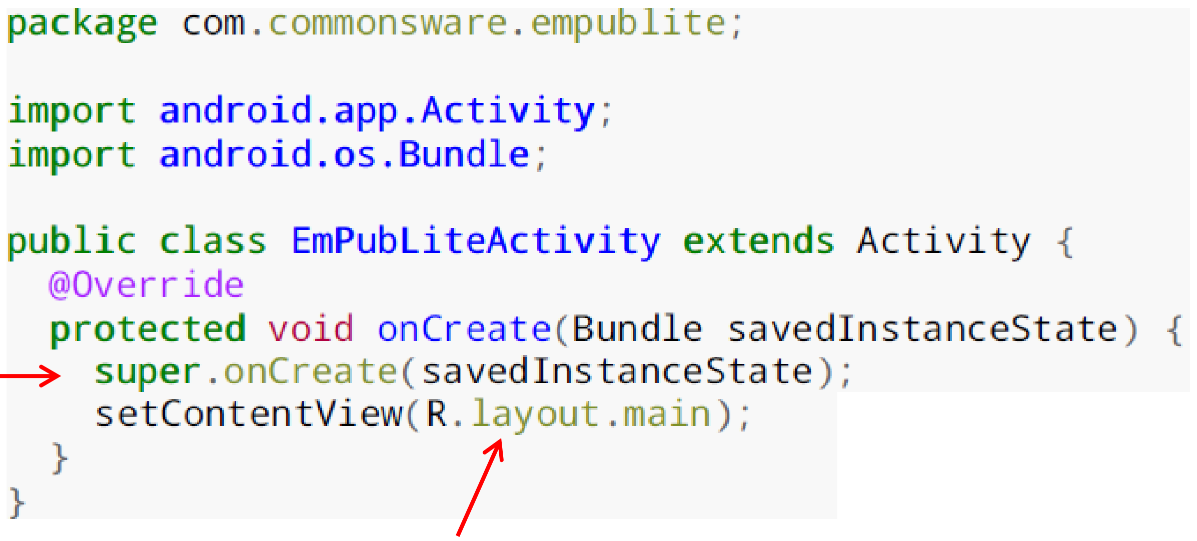
Example Activity Java file (E.g. MainActivity.java)

Package declaration (Same as chosen initially)

Import needed classes

My class inherits from Android activity class

Initialize by calling onCreate( ) method of base Activity class



```java
package com.commonsware.empublite;

import android.app.Activity;
import android.os.Bundle;

public class EmPubLiteActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

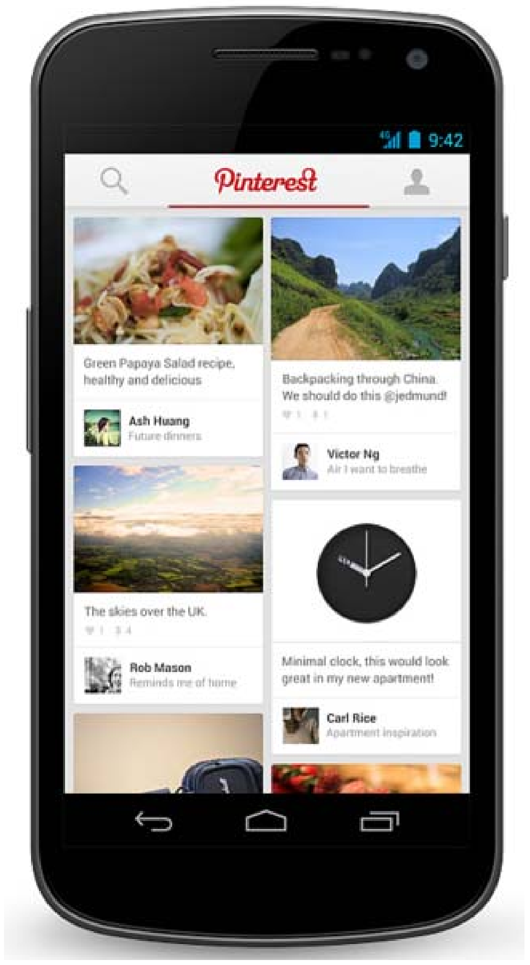
Use screen layout (design) declared in file main.xml stored in folder res/layout

Note: onCreate Method is called once your Activity is created

Recall: Files Hello World Android Project

XML file used to design Android UI

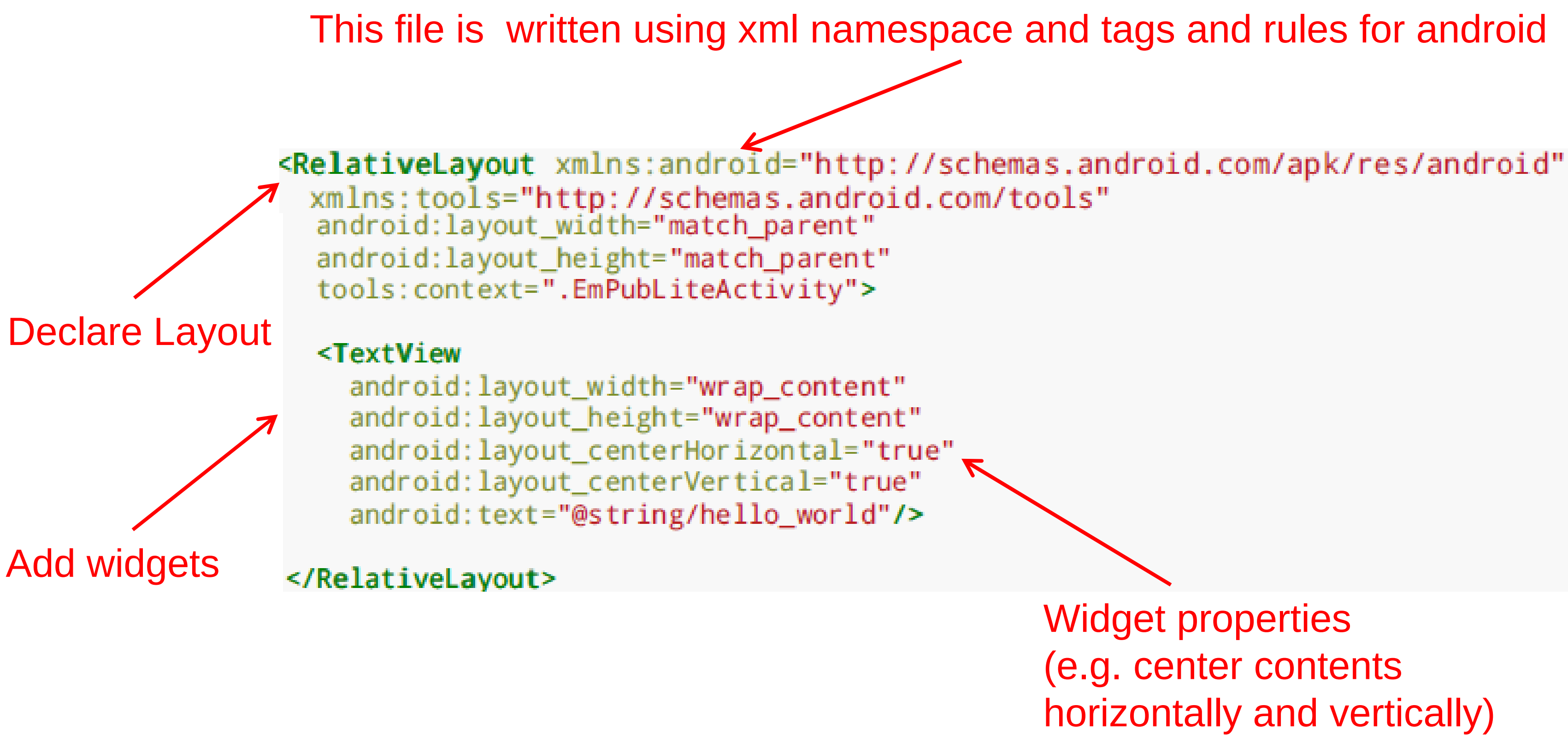
- 3 Files:
  - **Activity_my.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all screens, components of app
    - How these components attach themselves to overall Android system
    - Analogous to 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

Simple XML file Designing UI

- After choosing the layout, then widgets added to design UI

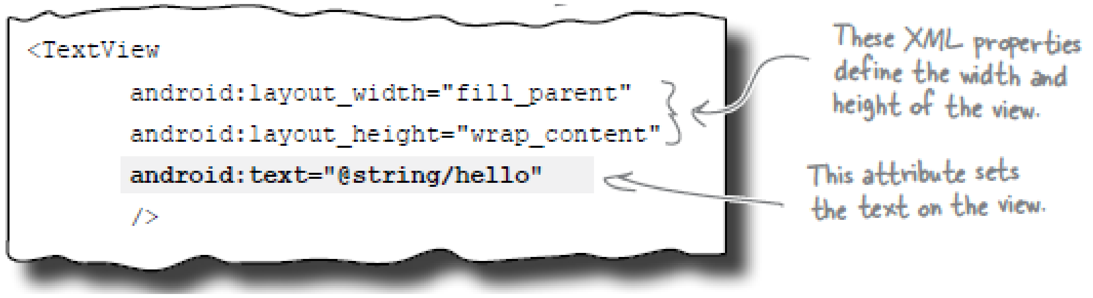

This file is written using xml namespace and tags and rules for android
Declare Layout
Add widgets
Widget properties
(e.g. center contents
horizontally and vertically)

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".EmPubLiteActivity">

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

</RelativeLayout>
```

Resources

View Properties and String Resources

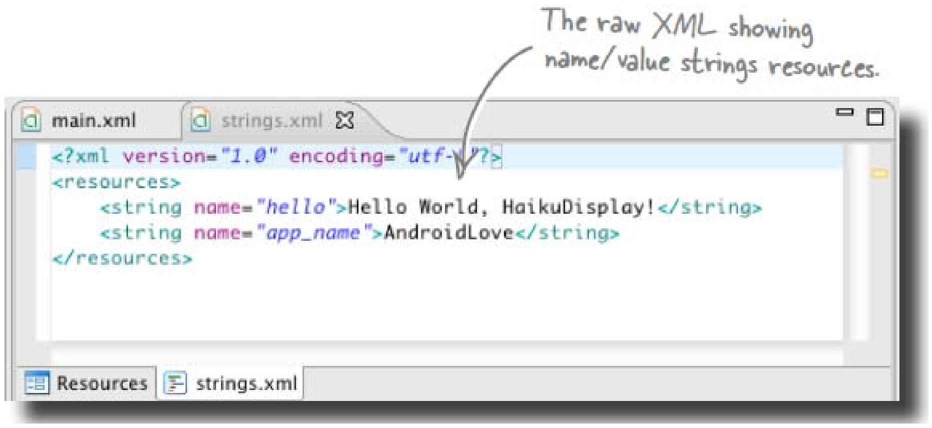
- Views are declared with attributes for configuring them
- Consider the following TextView example



```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
```

These XML properties define the width and height of the view.
This attribute sets the text on the view.

- **@string/hello** is a variable declared in another file, **strings.xml**

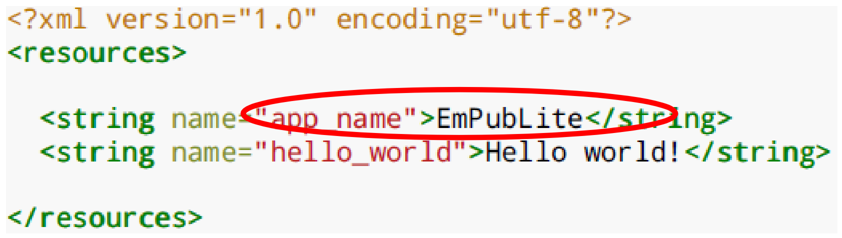


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HaikuDisplay!</string>
    <string name="app_name">AndroidLove</string>
</resources>
```

Strings in AndroidManifest.xml

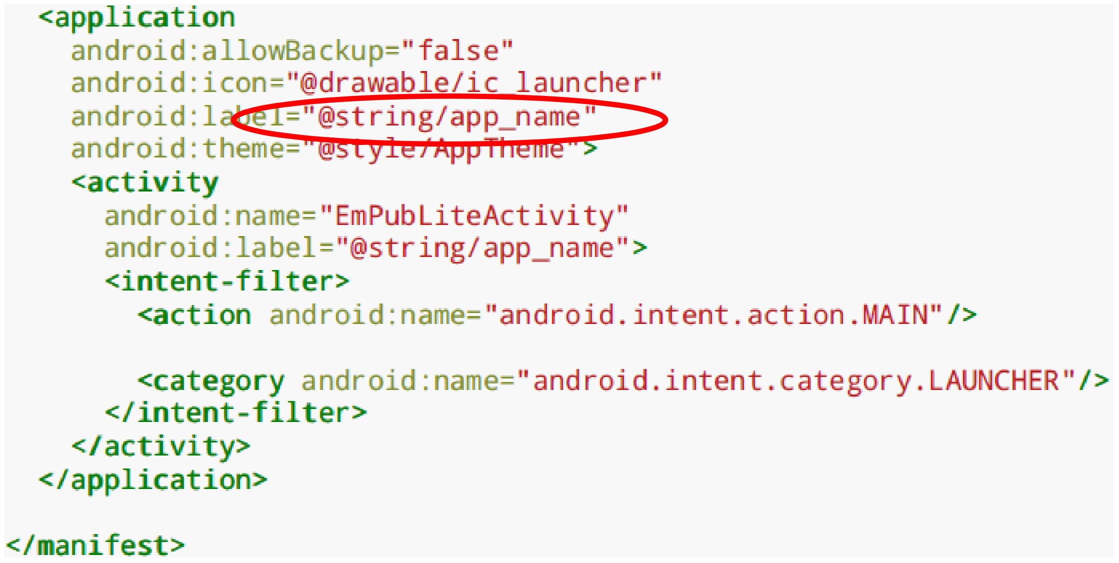
- Strings declared in strings.xml can be referenced by all other XML files (activity_my.xml, AndroidManifest.xml)

String declaration in strings.xml



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

  <string name="app_name">EmPubLite</string>
  <string name="hello_world">Hello world!</string>

</resources>
```

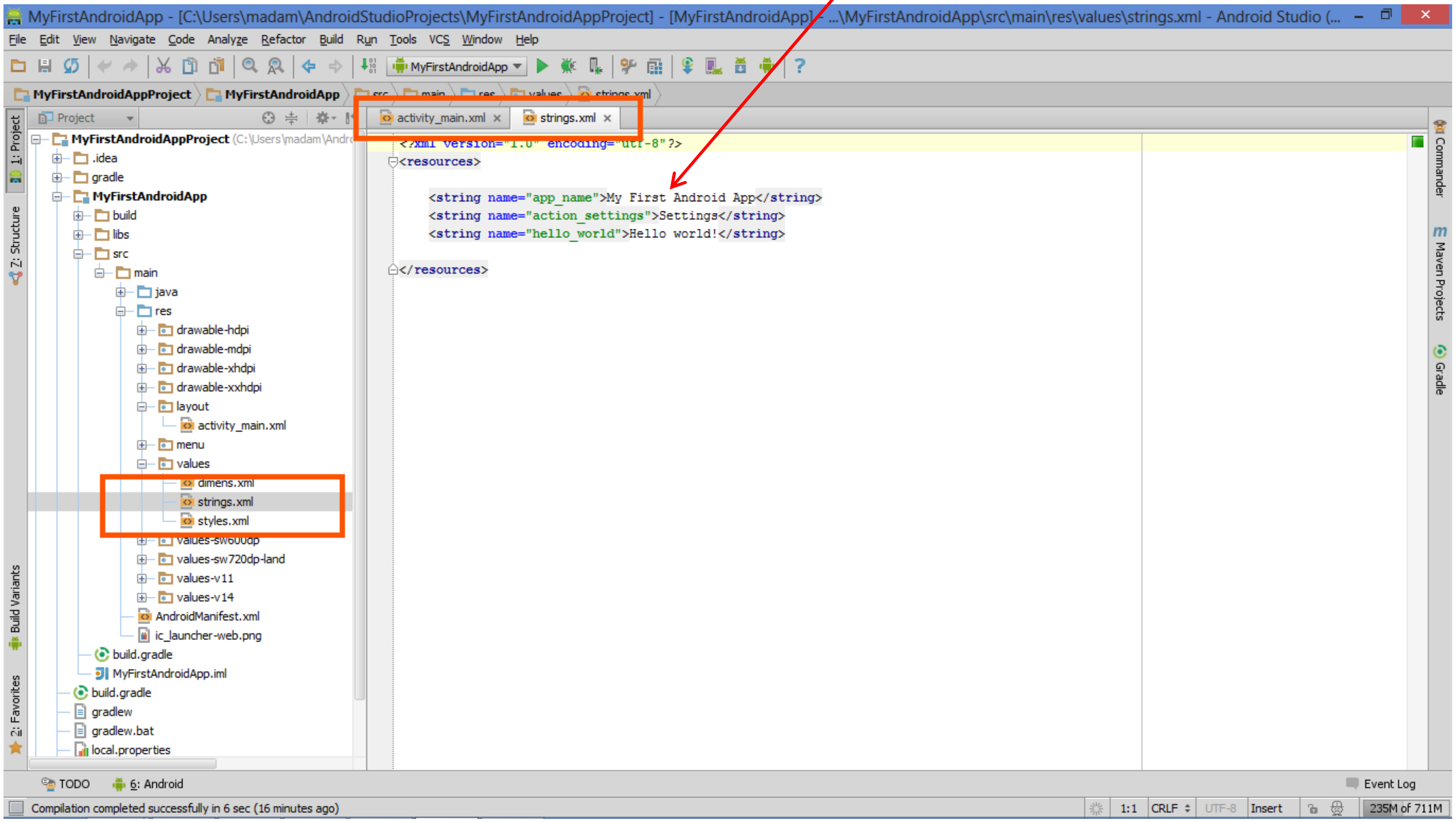
String usage in AndroidManifest.xml



```xml
  <application
    android:allowBackup="false"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    <activity
      android:name="EmPubLiteActivity"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
  </application>

</manifest>
```

Where is strings.xml in Android Studio?

Editting any string here changes it wherever it is displayed



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">My First Android App</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

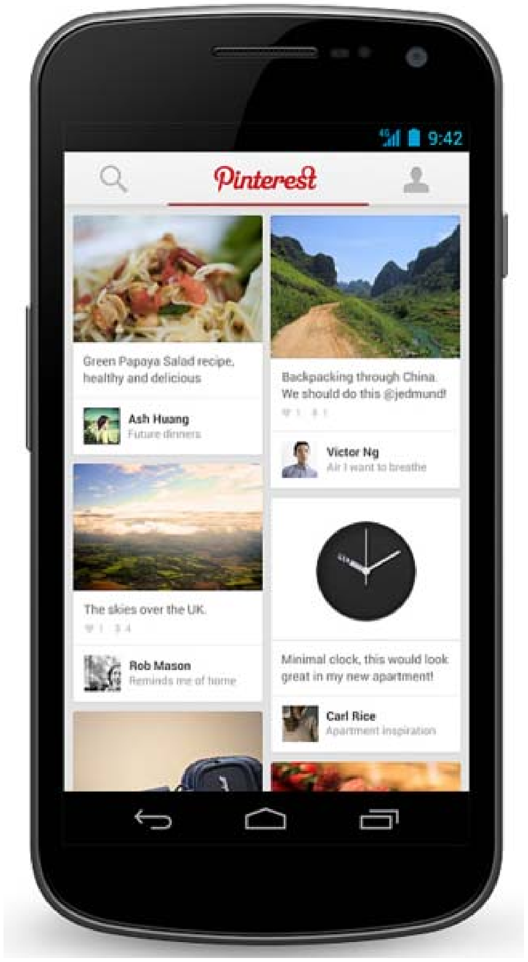
Styled Text

- In HTML, tags can be used for italics, bold, etc
- E.g. <i> Hello </i> makes text *Hello*
- <b> Hello <b> makes text **Hello**
- Can use the same HTML tags to add style (italics, bold, etc) to your Android strings

```
<resources>
  <string name="b">This has <b>bold</b> in it.</string>
  <string name="i">Whereas this has <i>italics</i>!</string>
</resources>
```

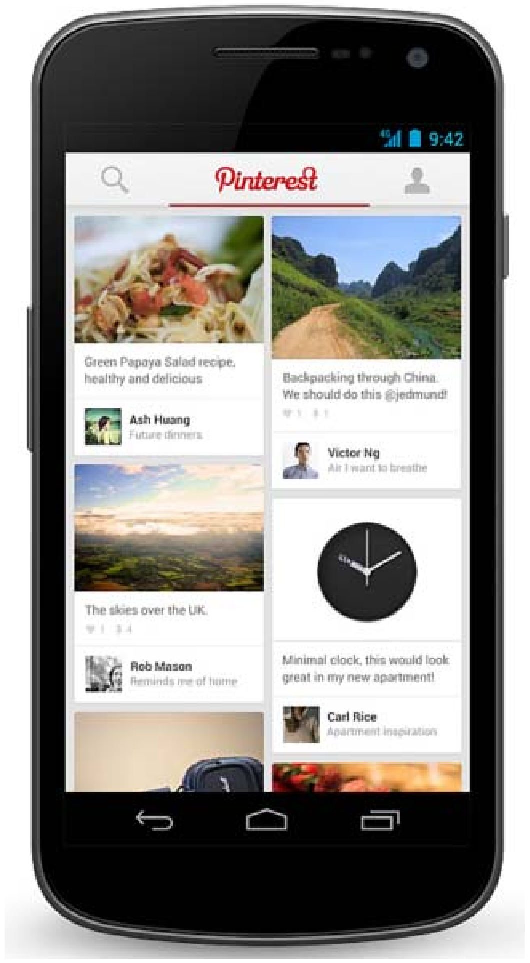
Recall: Example: Files in an Android Project

- **res/layout:** The width, height, layout of screen cells are specified in XML file here
- **res/drawable-xyz/:** The images stored in jpg or other format here
- **java/:** App's behavior when user clicks on screen (e.g. button) specified in java file here
- **AndroidManifext.XML:** Contains app name (Pinterest), list of app screens, etc


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

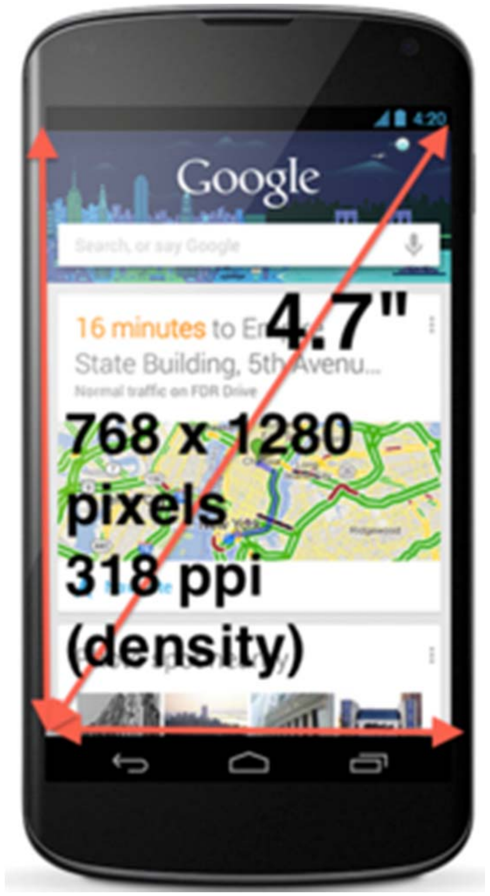
Resource Files in an Android Project

- Resources (stored in **/res** folder) are static bits of information outside java code (e.g. layout, images, etc). E.g.
  - **res/drawable-xyz/**
  - **res/layout:**
- Can have multiple resource definitions, used under different conditions. E.g internalization (text in different languages)
- In Android Studio, the **res/** folder is **app/src/main/**


9:42
Pinterest
Green Papaya Salad recipe, healthy and delicious
Ash Huang
Backpacking through China. We should do this @jedmund!
Victor Ng
The skies over the UK.
Rob Mason
Minimal clock, this would look great in my new apartment!
Carl Rice

Phone Dimensions Used in Android UI

- Physical dimensions measured diagonally
  - E.g. Nexus 4 is 4.7 inches diagonally
- Resolution in pixels
  - E.g. Nexus 4 resolution 768 x 1280 pixels
- Pixels per inch (PPI) =
  - Sqrt[(768 x 768) + (1280 x 1280)] = 318
- Dots per inch (DPI) is number of pixels in a physical area
  - Low density (ldpi) = 120 dpi
  - Medium density (mdpi) = 160 dpi
  - High density (hdpi) = 240 dpi
  - Extra High Density (xhdpi) = 320 dpi

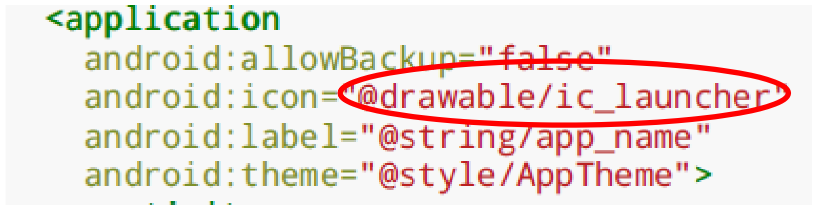


Adding Pictures

- Android supports images in PNG, JPEG and GIF formats
- GIF officially discouraged, PNG preferred format
- Default directory for images (drawables) is **res/drawable-xyz**
- Images in **res/drawable-xyz** can be referenced by XML and java files
  - **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)
- Images in these directories are same size, different resolutions

Adding Pictures

- Just the generic picture name is used
  - No format e.g. .png,
  - No specification of what resolution to use
  - 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">
```

- Android chooses which directory (e.g. –mdpi) based on actual device
- Android studio tools for generating icons
  - **Icon wizard or Android asset studio:** generates icons in various densities from starter image
  - Cannot edit images (e.g. dimensions) with these tools

Editting Pictures

- **Image dimensions:**
  - **px:** hardware pixels, varies from device to device
  - **in** and **mm:** inches or millimeters based on actual screen size
  - **pt:** 1/72nd of an inch
  - **dip** (or **dp**)**:** density-independent pixels
    - 1 dip = 1 hardware pixel on ~160 dpi screen
    - 1 dip = 2 hardware pixels on ~ 320 dpi screen
  - **sp** (or scaled pixels)**:** scaled pixels
- Dimensions declared in **dimens.xml**

```
<resources>
  <dimen name="thin">10dip</dimen>
  <dimen name="fat">1in</dimen>
</resources>
```

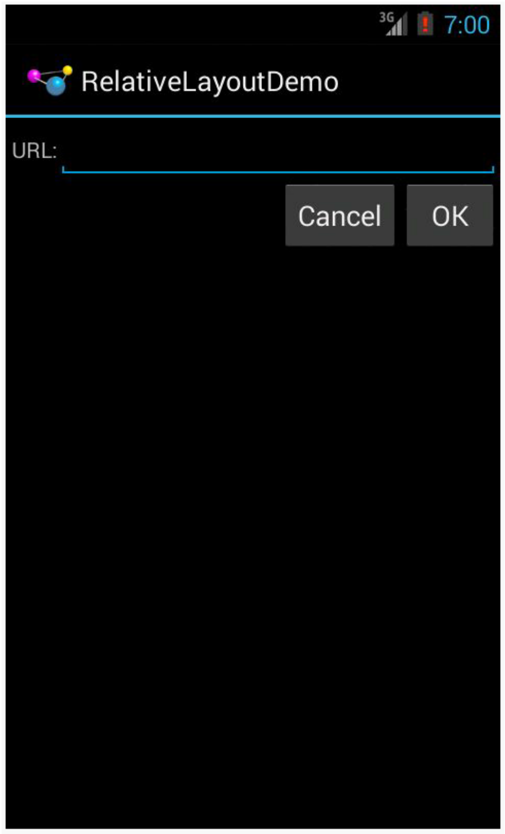
- Can reference “thin” declared above
  - In XML layout files as **@dimen/thin**
  - In Java using **Resources.getDimension(R.dimen.thin)**

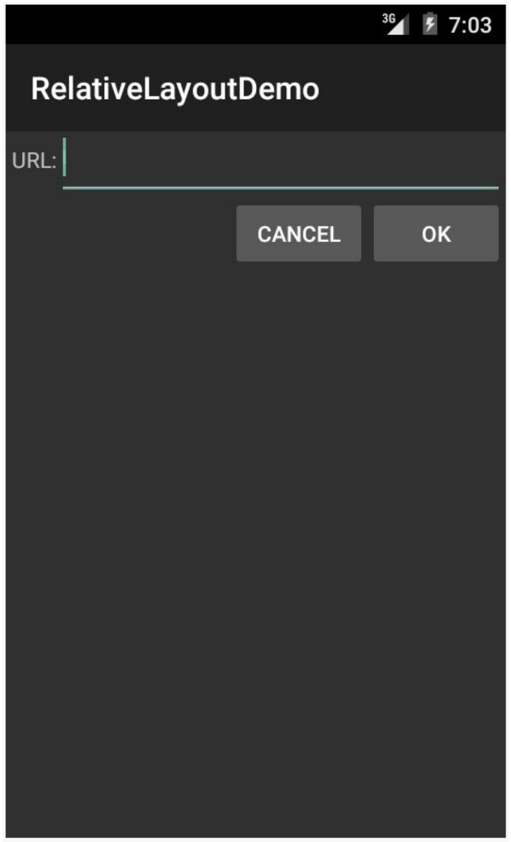
Styles

- Styles specify rules for look of Android screen
- Similar to Cascaded Style Sheets (CSS) in HTML
- E.g CSS enables setting look of certain types of tags.
  - E.g. font and size of all <h1> and <h2> elements
- Android widgets have properties
  - E.g. Foreground color = red
- **Styles in Android:** collection of values for properties
- Styles can be specified one by one or themes (e.g. Theme, Theme.holo and Theme.material) can be used

Default Themes

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

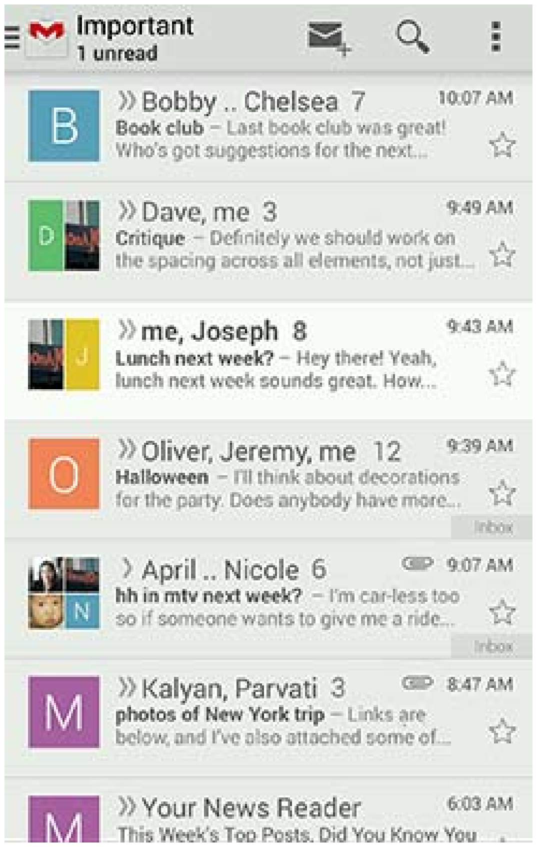




Theme.Holo: default theme in Android 3.0

Theme.Material: default theme in Android 5.0

Examples of Themes in Use



Important
1 unread

Bobby .. Chelsea 7 — 10:07 AM
Book club – Last book club was great! Who's got suggestions for the next...

Dave, me 3 — 9:49 AM
Critique – Definitely we should work on the spacing across all elements, not just...

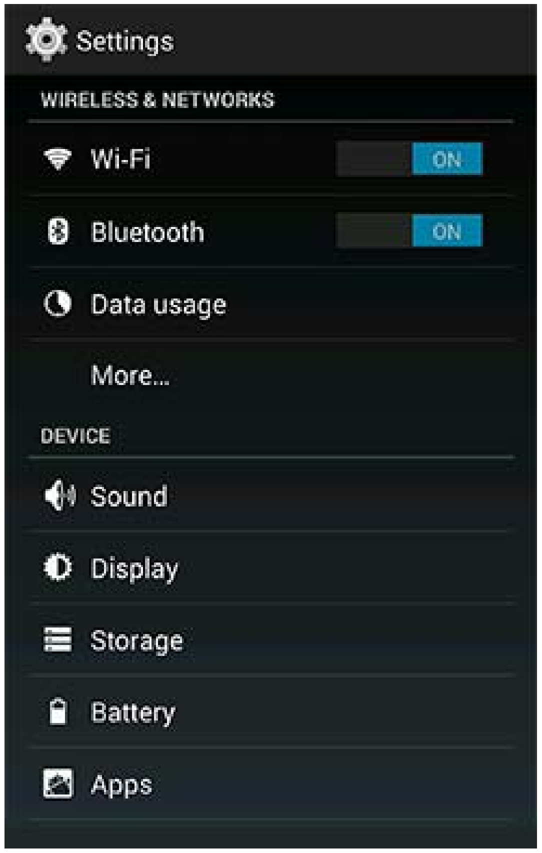
me, Joseph 8 — 9:43 AM
Lunch next week? – Hey there! Yeah, lunch next week sounds great. How...

Oliver, Jeremy, me 12 — 9:39 AM
Halloween – I'll think about decorations for the party. Does anybody have more...
Inbox

April .. Nicole 6 — 9:07 AM
hh in mtv next week? – I'm car-less too so if someone wants to give me a ride...
Inbox

Kalyan, Parvati 3 — 8:47 AM
photos of New York trip – Links are below, and I've also attached some of...

Your News Reader — 6:03 AM
This Week's Top Posts. Did You Know You



Settings

WIRELESS & NETWORKS
Wi-Fi ON
Bluetooth ON
Data usage
More...

DEVICE
Sound
Display
Storage
Battery
Apps

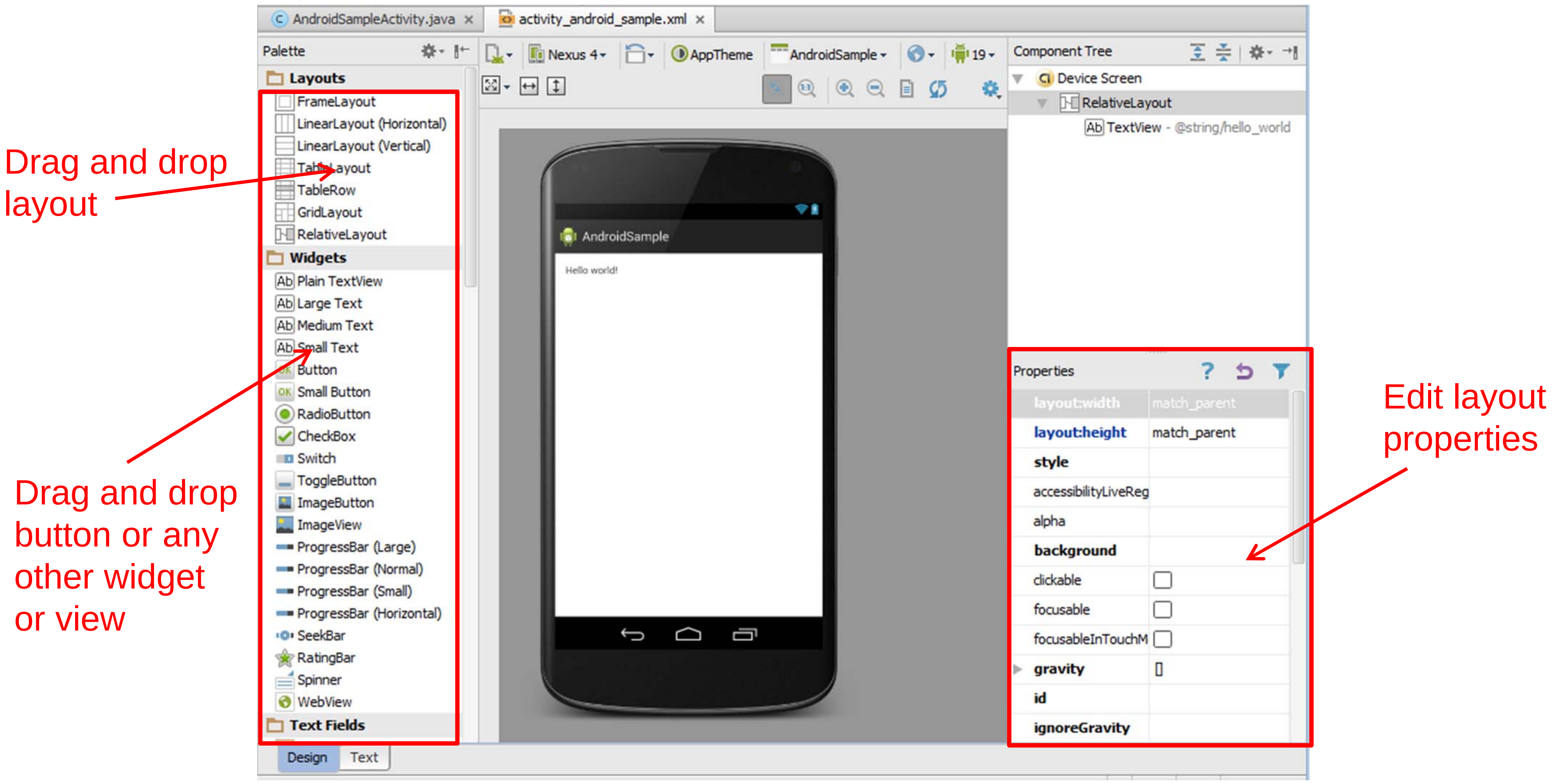
GMAIL in Holo Light

Settings in Holo Dark

Android UI Design in XML

Recall: Edit XML Layouts using Graphical IDE

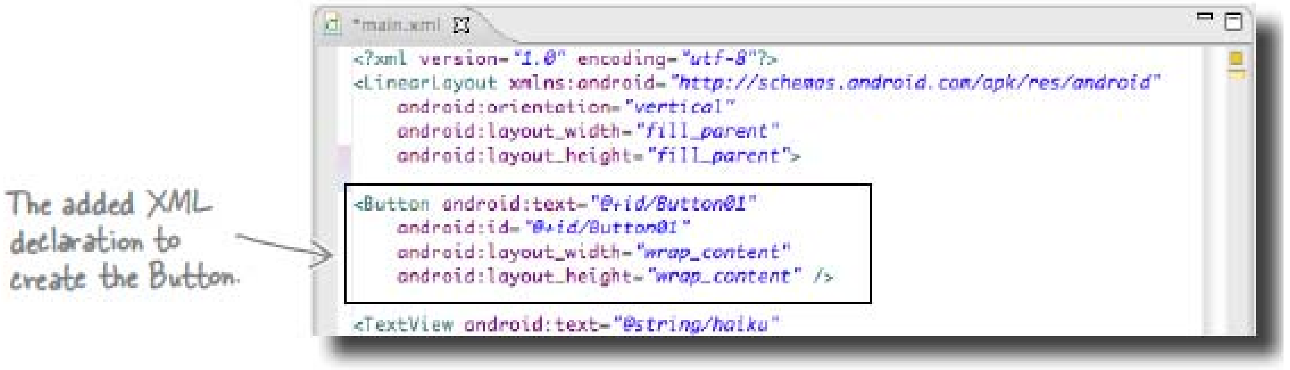
- Can drag and drop widgets, layouts in Android Studio
- Can also edit their properties (e.g. height, width, color, etc)



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

XML Generated

- Clicking and dragging button or widget adds corresponding XML to appropriate XML file (e.g. main.xml)

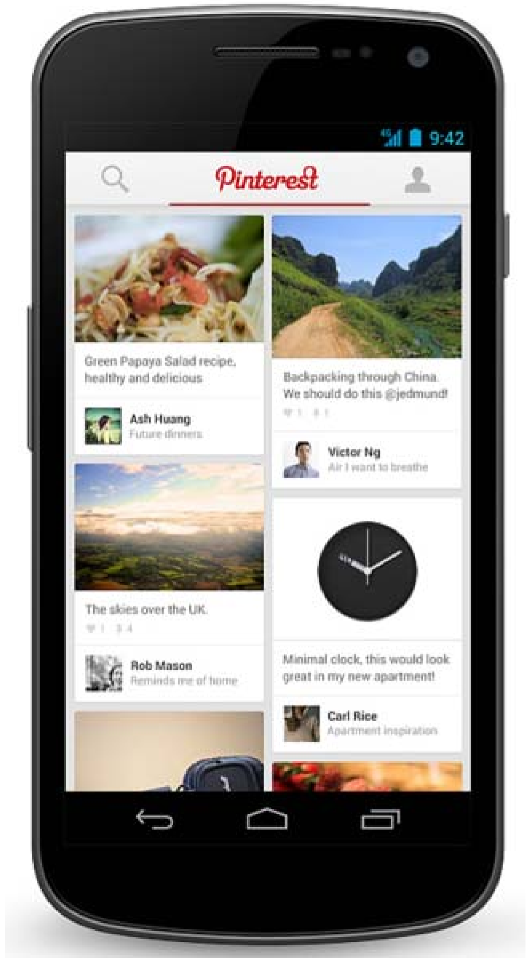


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<Button android:text="@+id/Button01"
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView android:text="@string/haiku"
```

Recall: Files Hello World Android Project

XML file used to design Android UI

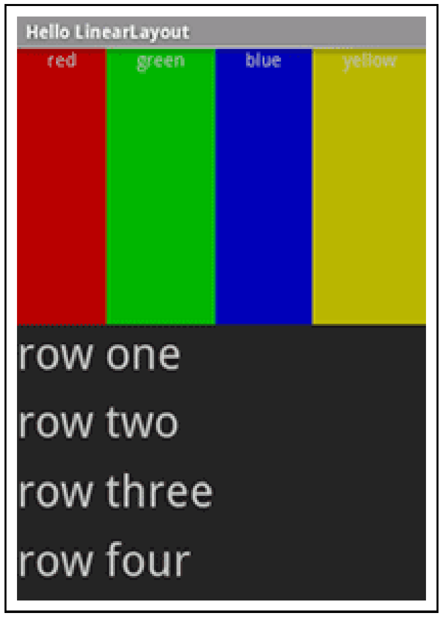
- 3 Files:
  - **Activity_my.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all screens, components of app
    - How these components attach themselves to overall Android system
    - Analogous to 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

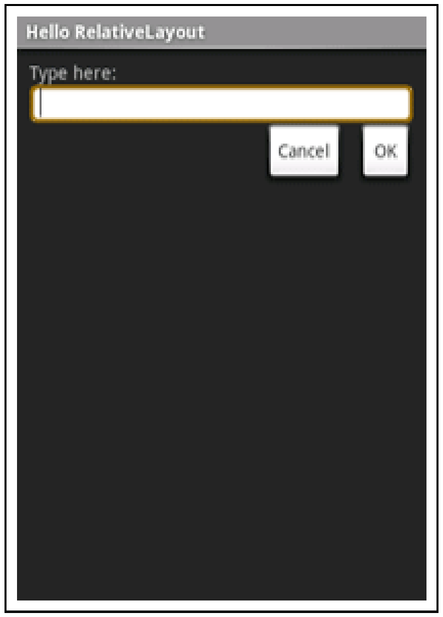
Android UI using XML Layouts

- Android UI are usually designed in XML
- **Note:** Android UI can also be designed in Java (more later)
- In the XML file, we have to choose a layout to use

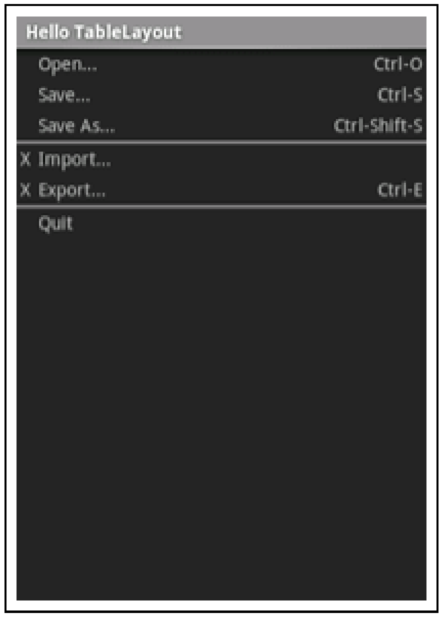
LinearLayout



RelativeLayout



TableLayout



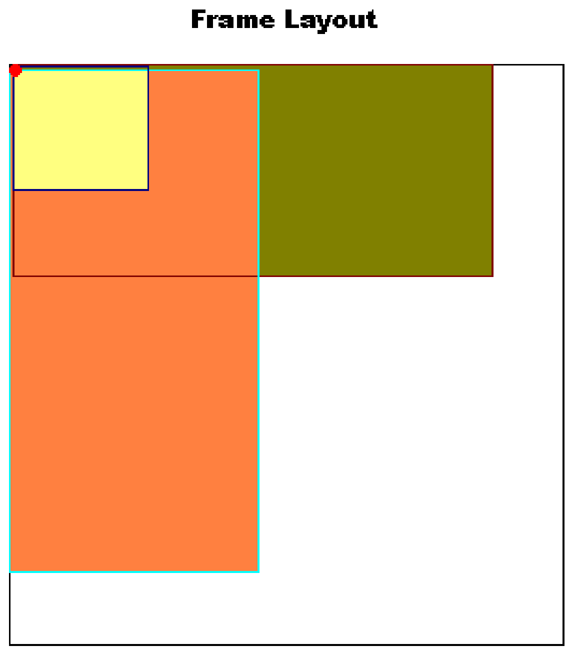
http://developer.android.com/resources/tutorials/views/index.html

Layouts

- Layouts can contain UI elements (provided and custom)
- Stored in **res/layout**
- Useful Layouts:
  - FrameLayout,
  - LinearLayout,
  - TableLayout,
  - GridLayout,
  - RelativeLayout,
  - ListView,
  - GridView,
  - ScrollView,
  - DrawerLayout,
  - ViewPager
- More on layouts next

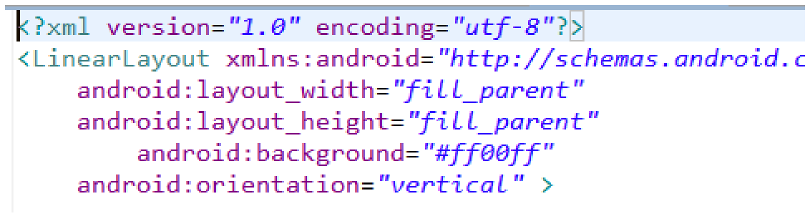
FrameLayout

- 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

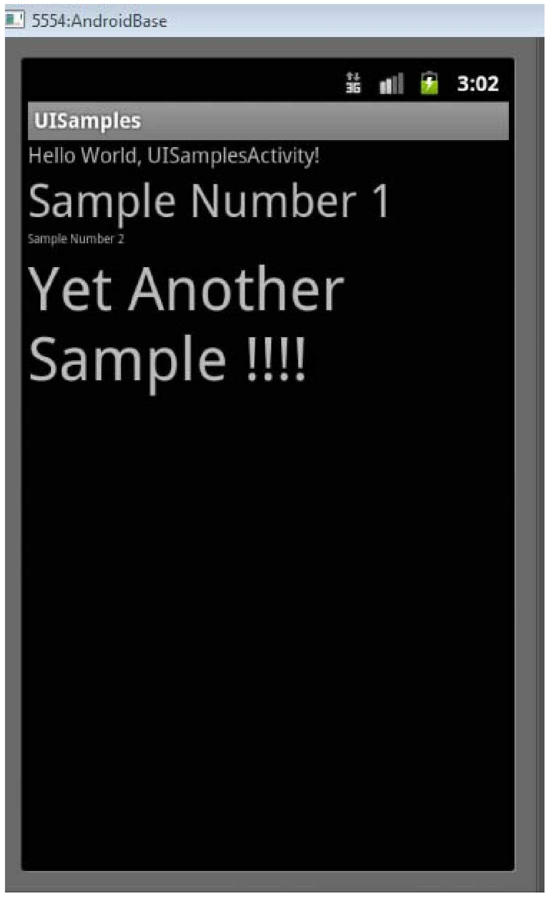


LinearLayout

- 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" >
```



5554:AndroidBase

3:02

UISamples

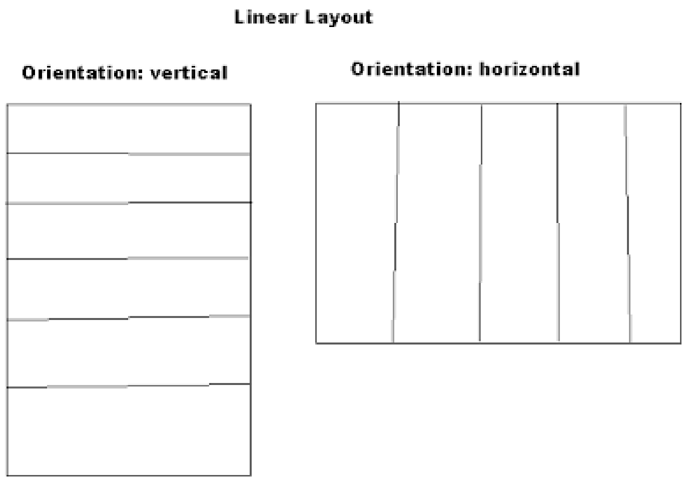
Hello World, UISamplesActivity!

Sample Number 1

Sample Number 2

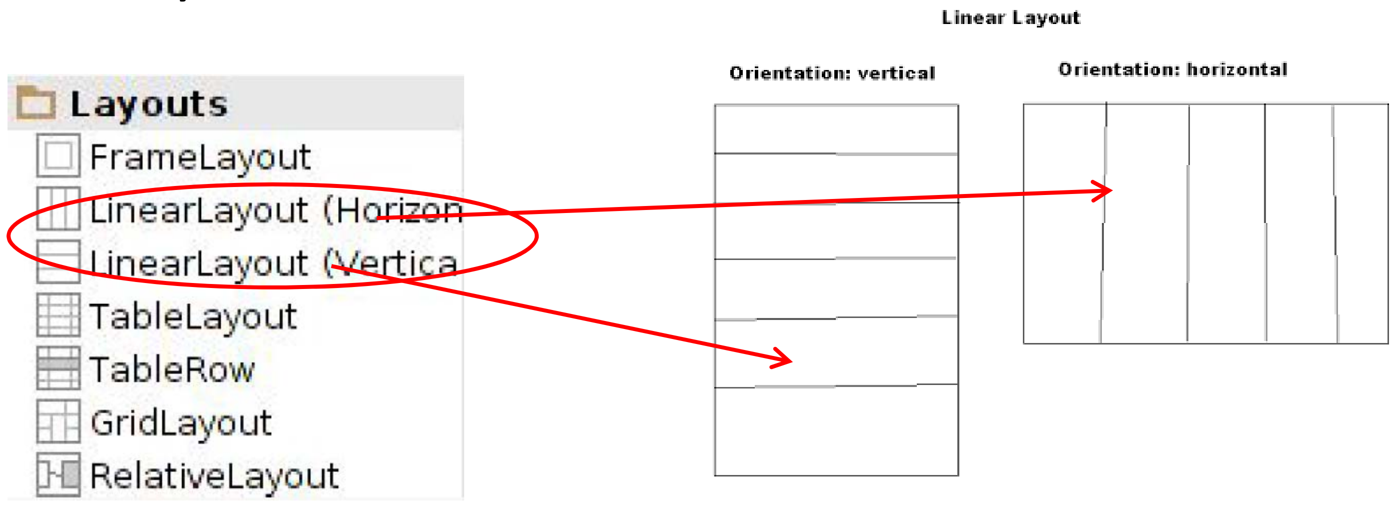
Yet Another Sample !!!!

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

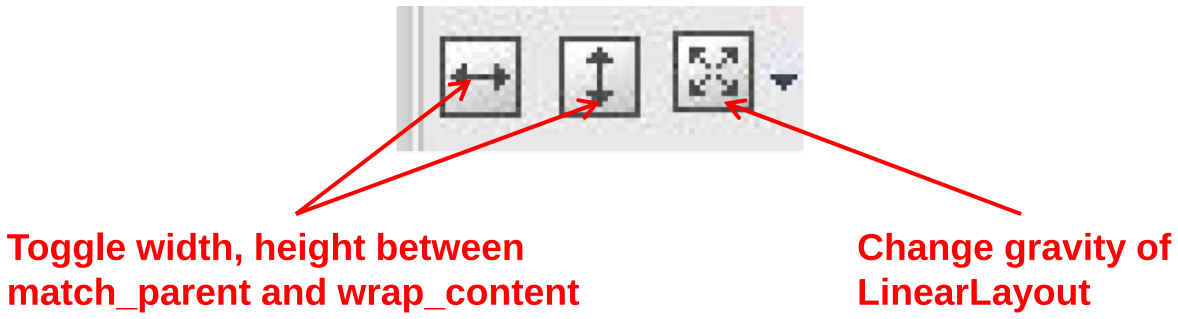


LinearLayout in Android Studio

- LinearLayout can be found in palette of Android Studio Graphical Layout Editor



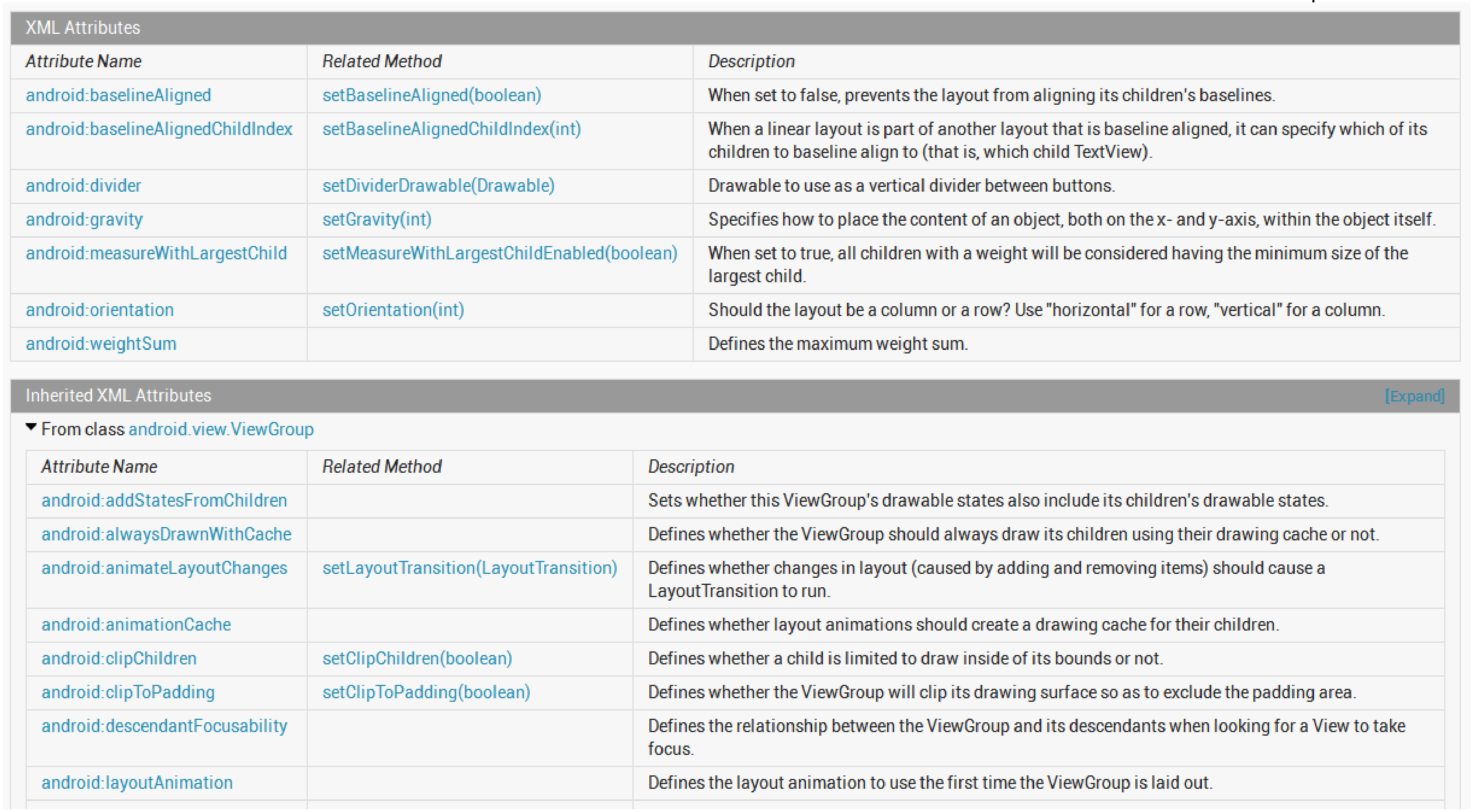
- After selecting LinearLayout, toolbars buttons to set parameters


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

Attributes

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

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.

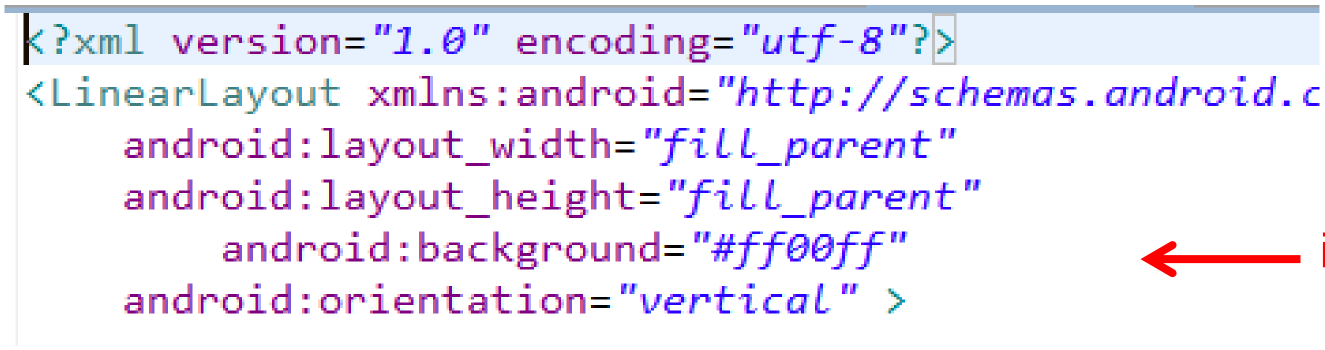
Inherited XML Attributes [Expand]

▼ 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.

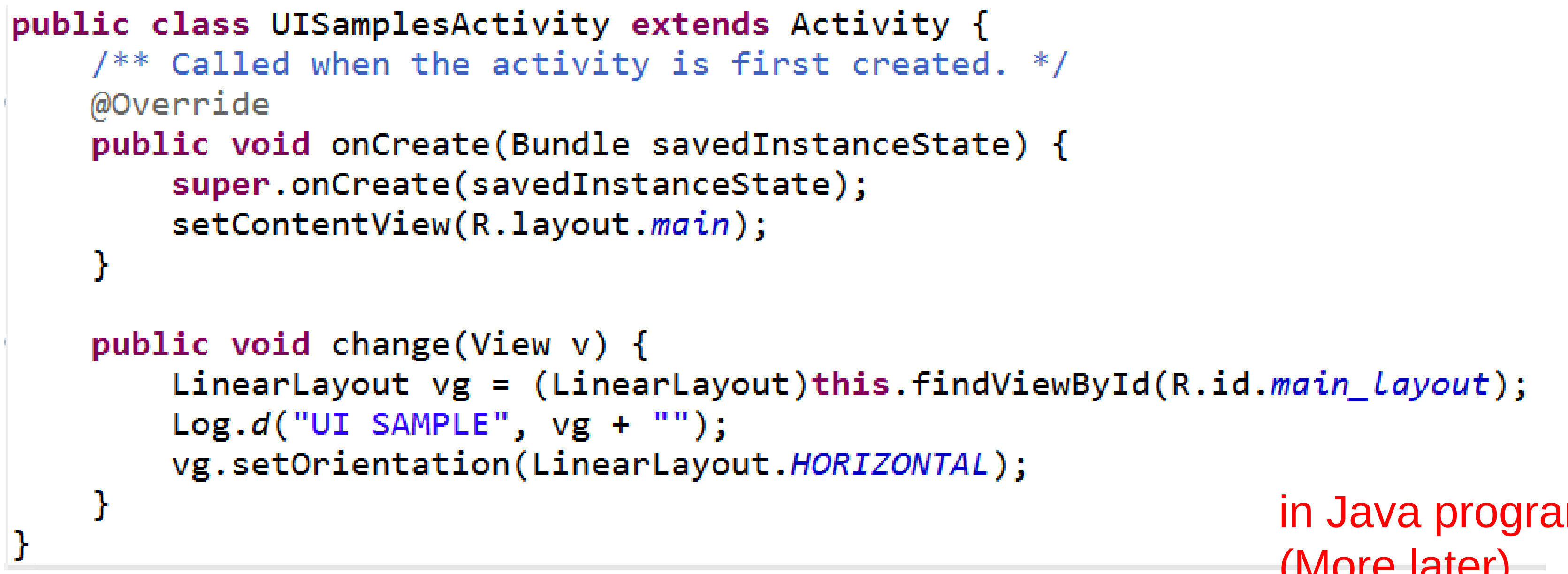
Can find complete list of attributes, possible values on Android Developer website

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);
    }
}
```

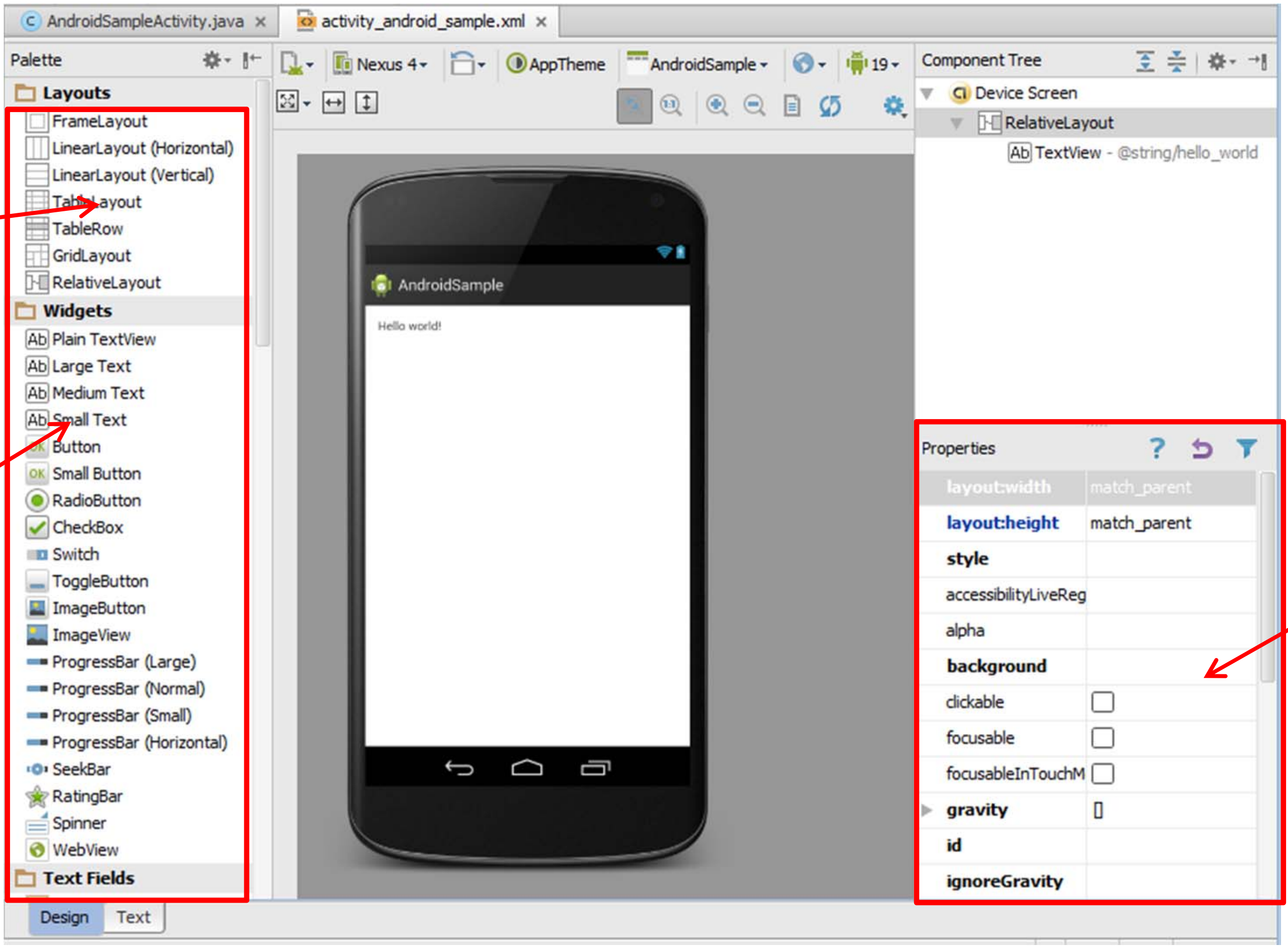
in Java program
(More later)

Recall: Edit XML Layouts using Graphical IDE

- Can drag and drop widgets, layouts in Android Studio
- Can also edit their properties (e.g. height, width, color, etc)

Drag and drop layout

Drag and drop button or any other widget or view

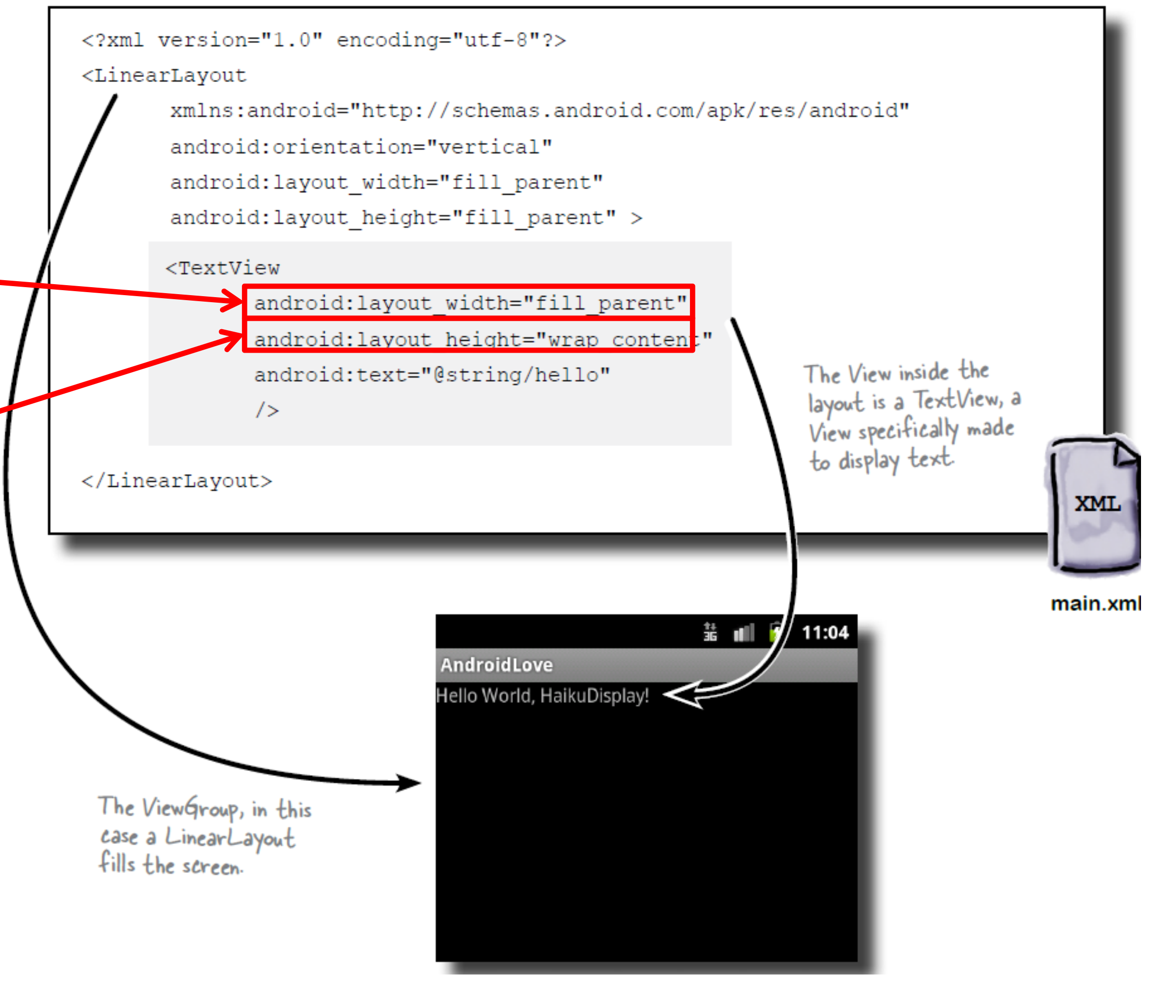


layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
clickable	☐
focusable	☐
focusableInTouchM	☐
gravity	[]
id	
ignoreGravity	

Edit layout attributes

Layout Width and Height Attributes

- **match_parent:** widget as wide/high as its parent
- **wrap_content:** widget as wide/high as its content (e.g. text)
- **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)

LinearLayout - Horizontal Orientation

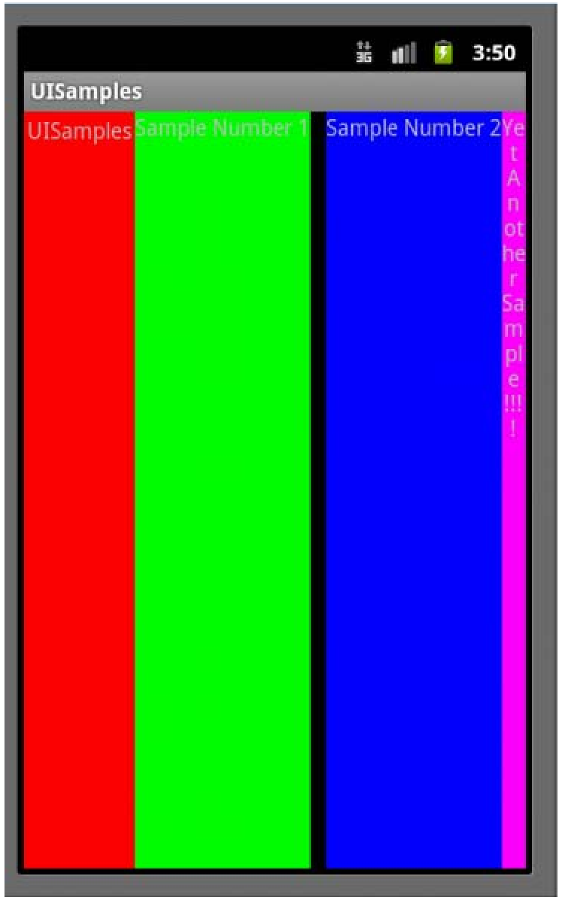
- Set
  - Padding

    E.g. android:layout_paddingTop = “20dp”

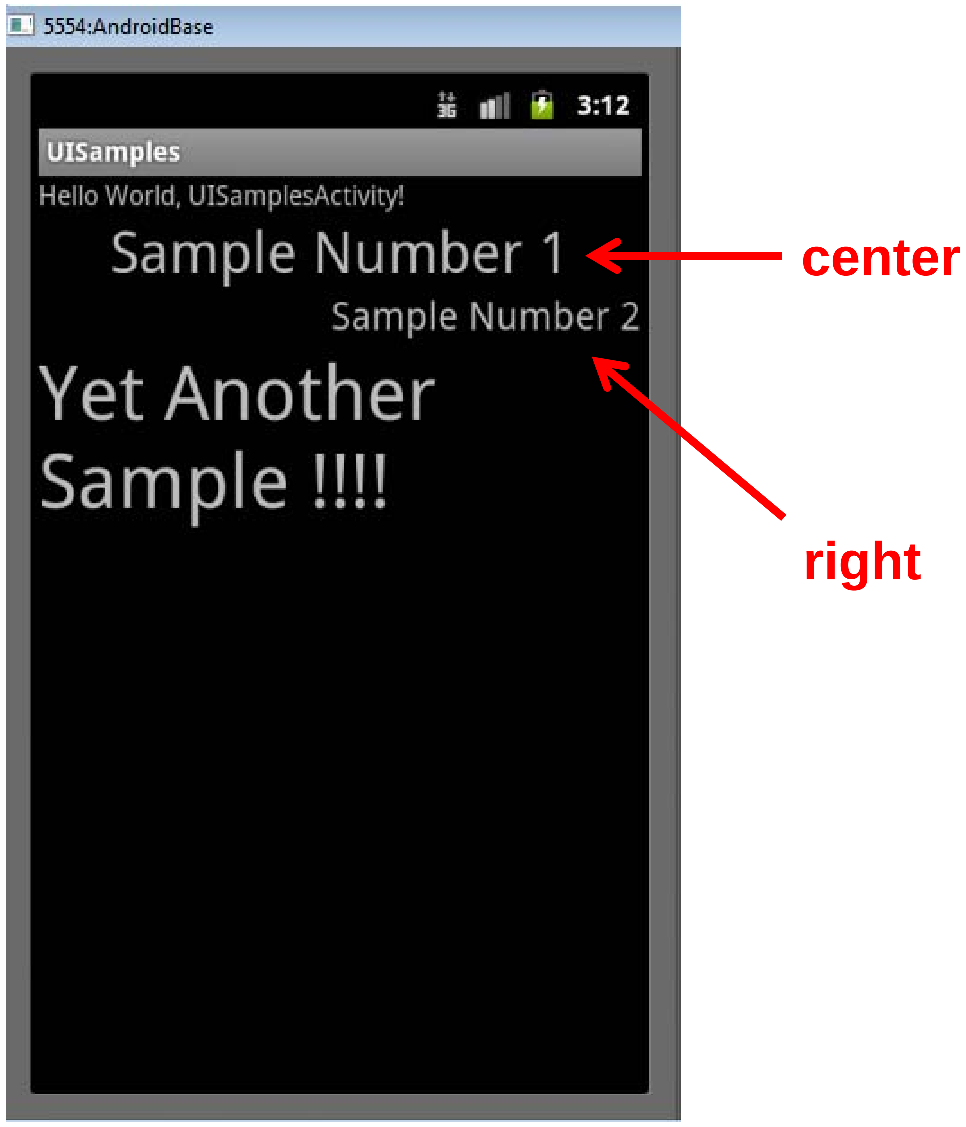
  - background color

    E.g. android:background = “00FF00”

  - Margins
  - E.g. “android:layout_marginLeft = “10dp”



Gravity Attribute



Hello World, UISamplesActivity!

Sample Number 1

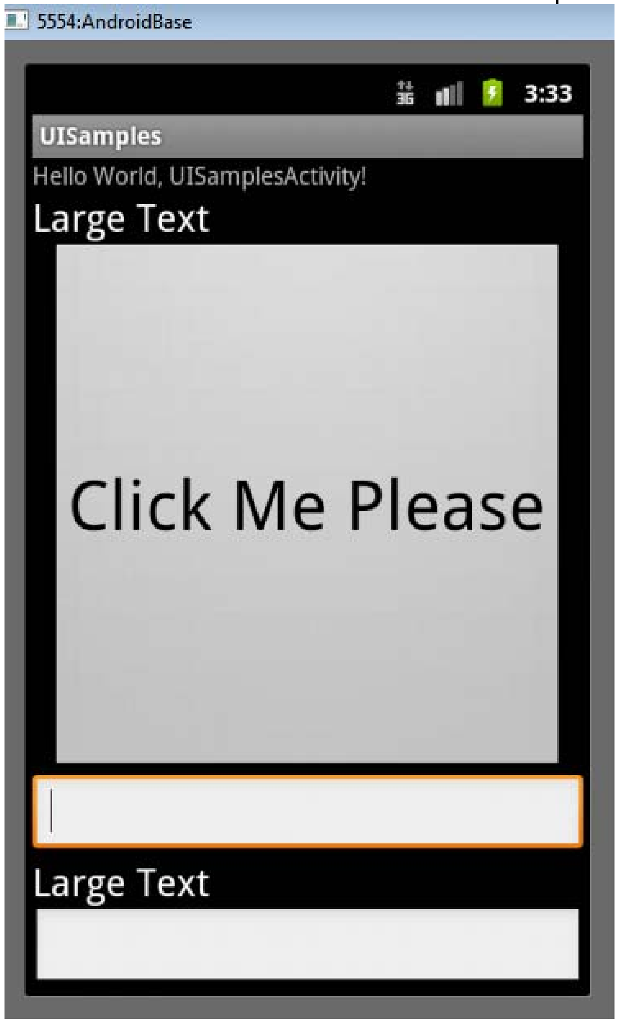
Sample Number 2

Yet Another Sample !!!!

- By default, linearlayout left- and top-aligned
- Gravity attribute can change position of :
  - Widget within Linearlayout
  - Contents of widgets (e.g. android:gravity = "right")

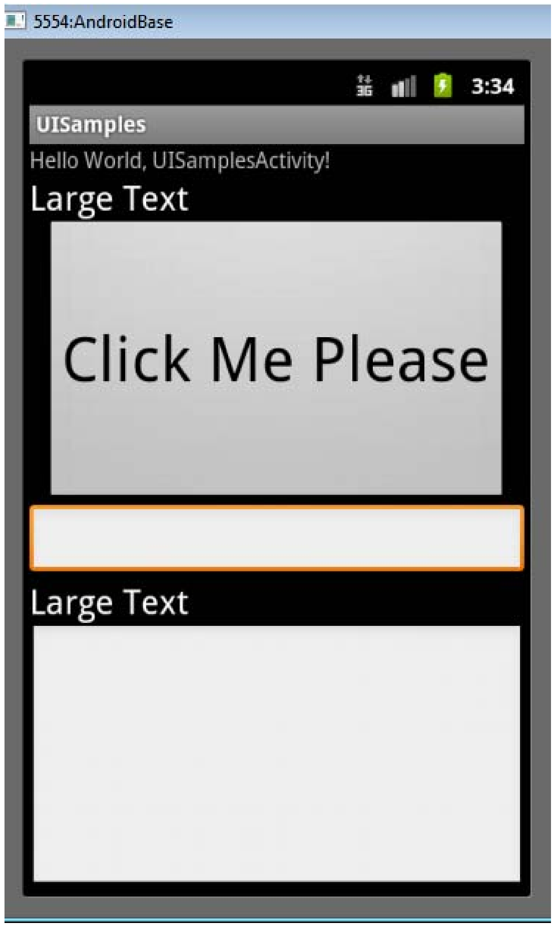
Weight

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

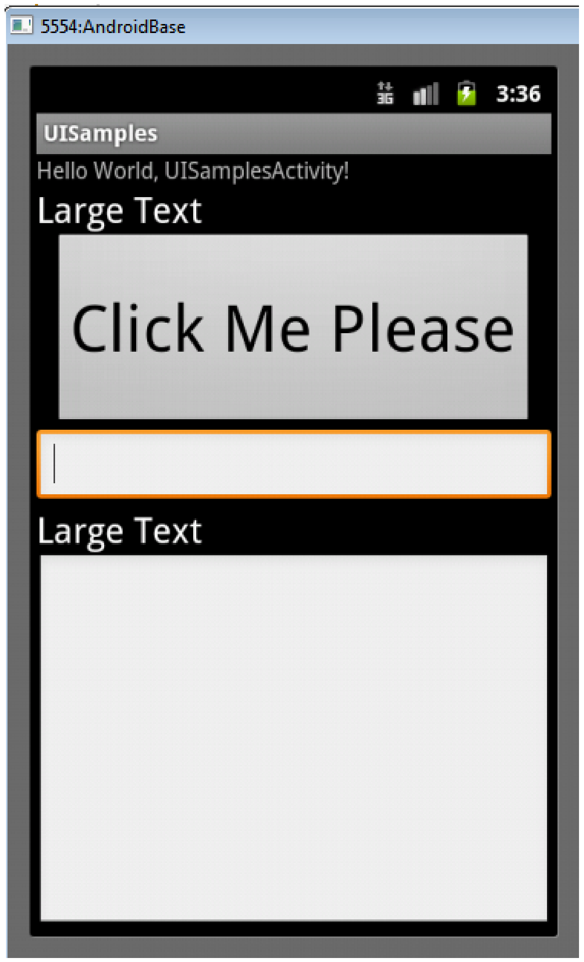


Another Weight Example

button and bottom edit text weight of 2

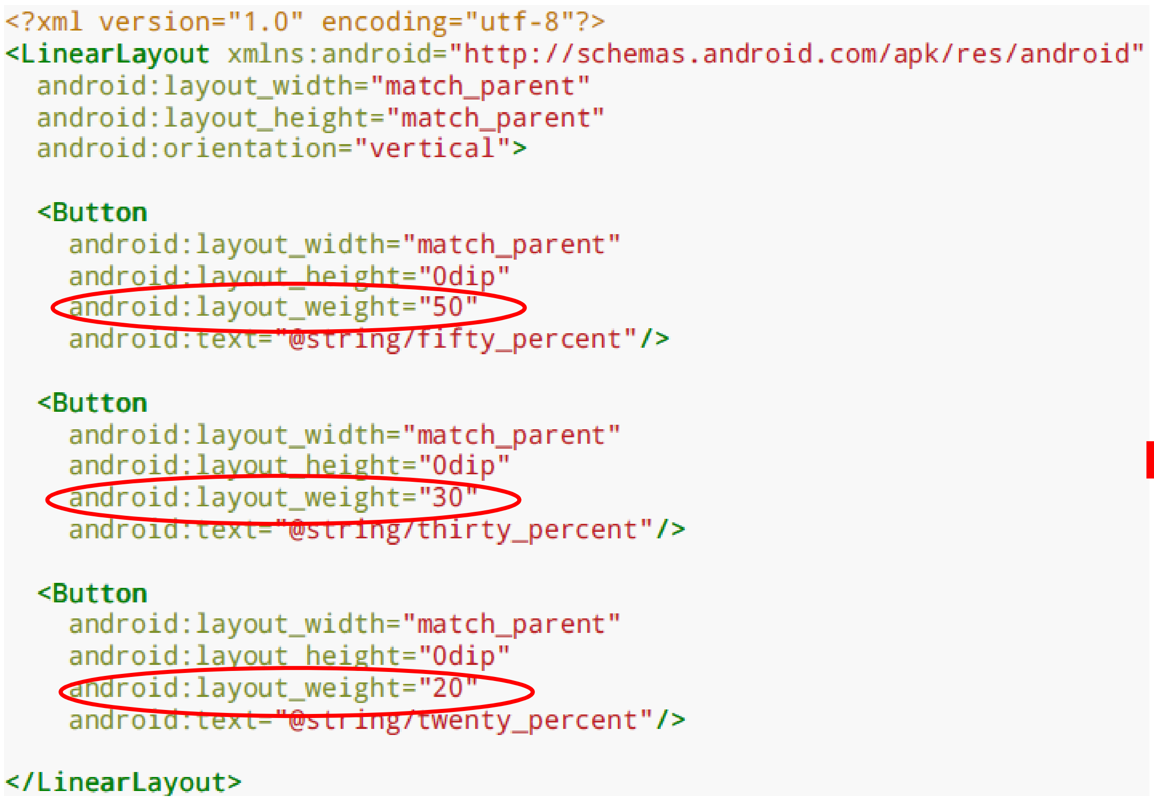


button weight 1 and bottom edit text weight of 2

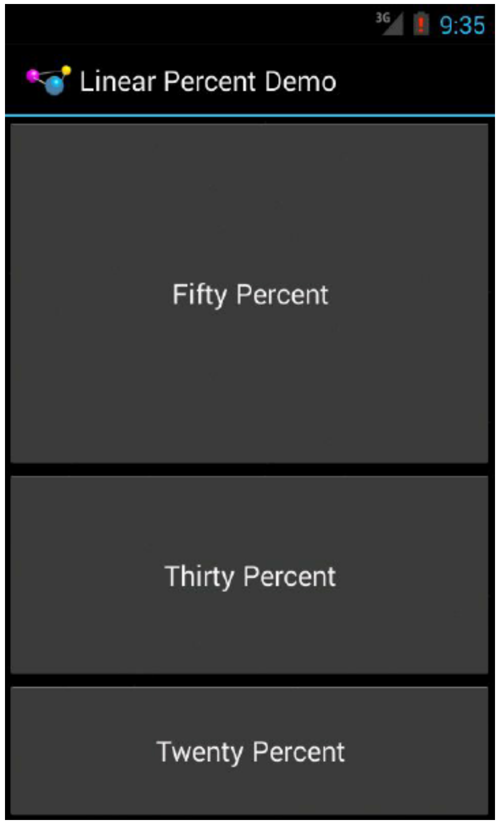


Linear Layout

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



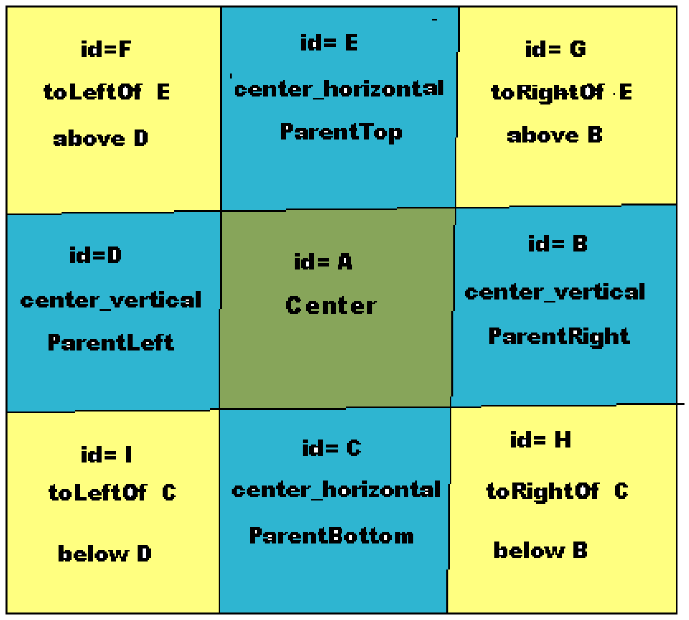
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <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>
```

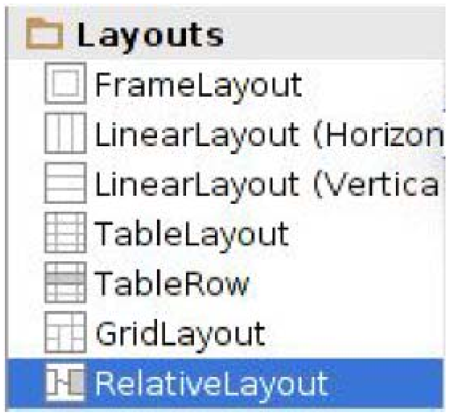


RelativeLayout

- First element listed is placed in "center"
- Positions of children specified relative to parent or to each other.
  - E.g. **android:layout_toRightOf = “true”:** widget should be placed to the right of widget referenced in the property
  - **android:layout_alignParentBottom = “true”:** align widget’s bottom with container’s bottom

Relative Layout





RelativeLayout available In Android Studio palette

RelativeLayout XML Example

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

  <TextView
    android:id="@+id/label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/entry"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="4dip"
    android:text="@string/url"/>

  <EditText
    android:id="@id/entry"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@id/label"
    android:inputType="text"/>

  <Button
    android:id="@+id/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/entry"
    android:layout_below="@id/entry"
    android:text="@string/ok"/>

  <Button
    android:id="@+id/cancel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/ok"
    android:layout_toLeftOf="@id/ok"
    android:text="@string/cancel"/>

</RelativeLayout>
```

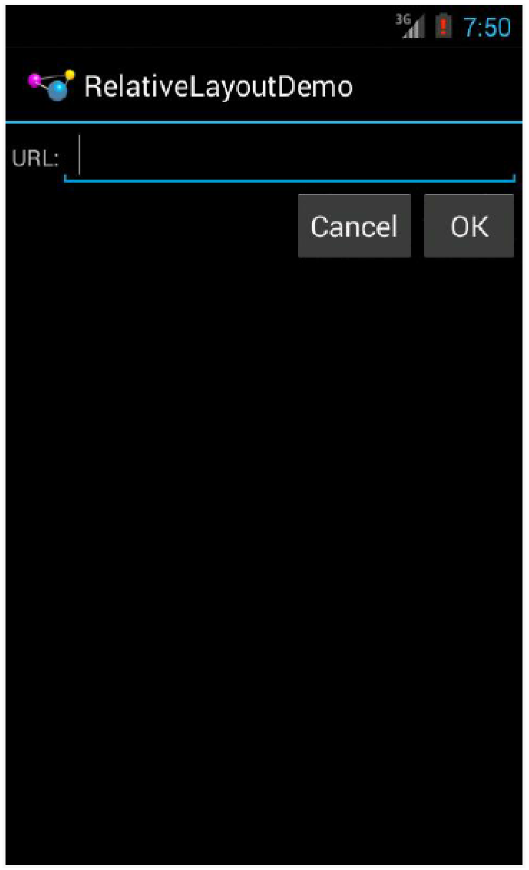
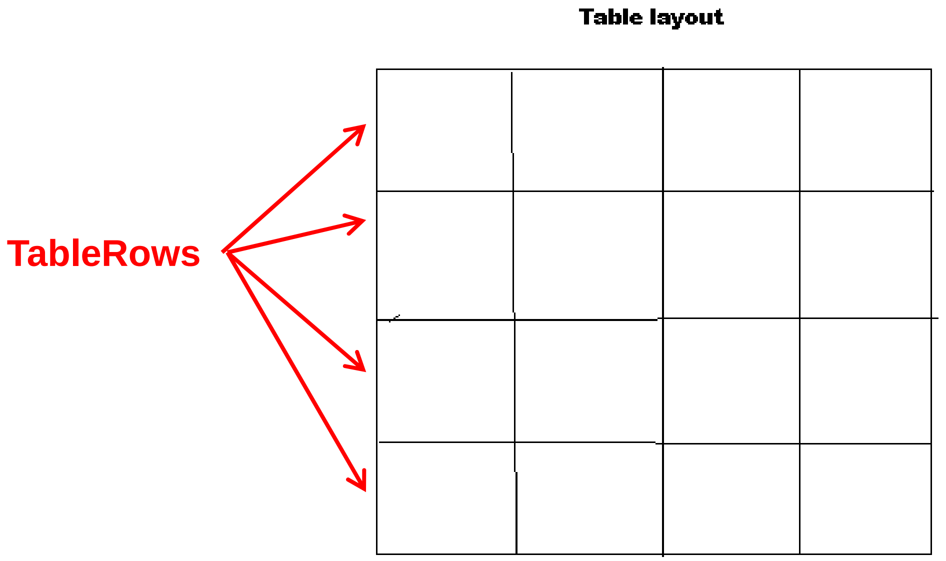
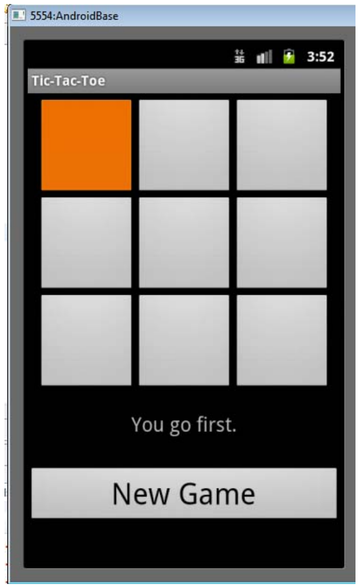
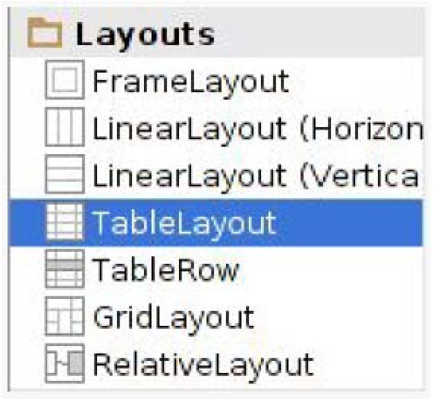


Table Layout

- Specify number of rows and columns
- Rows specified using **TableRows** (subclass of LinearLayout)
- **TableRows** contain other elements such as buttons, text, etc.
- Available in Android Studio palette

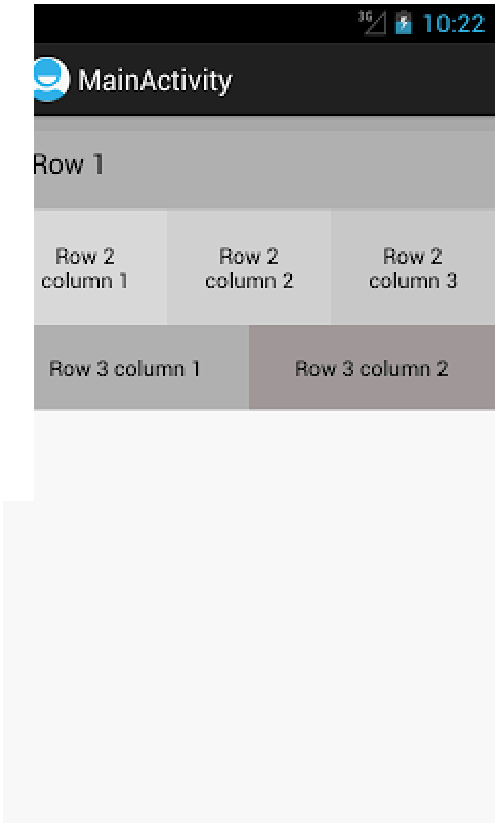






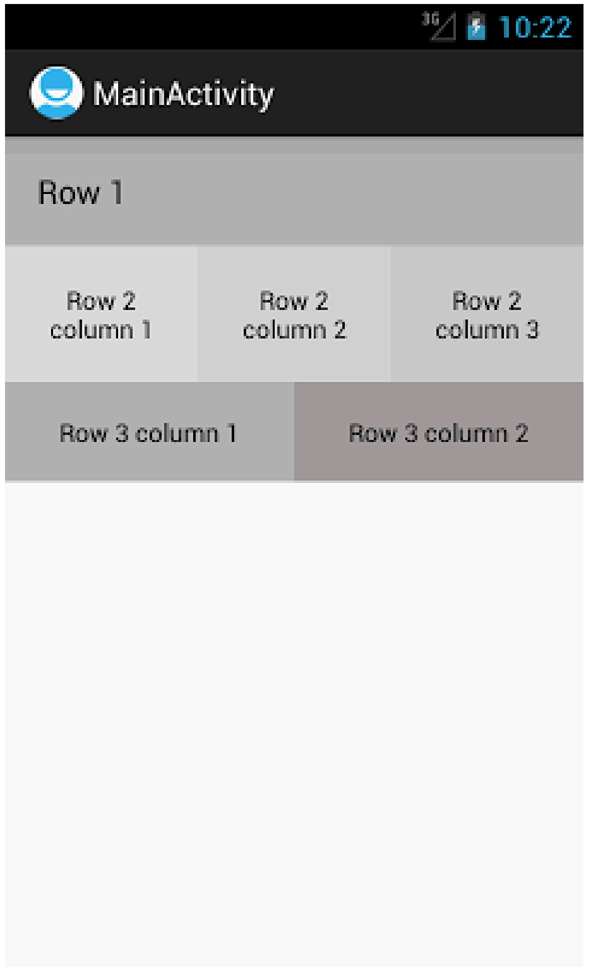
TableLayout Example

```
<TableLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:shrinkColumns="*"
        android:stretchColumns="*"
        android:background="#ffffff">

            <!-- Row 1 with single column -->

            <TableRow
                android:layout_height="wrap_content"
                android:layout_width="fill_parent"
                android:gravity="center_horizontal">

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:textSize="18dp"
                    android:text="Row 1"
                    android:layout_span="3"
                    android:padding="18dip"
                    android:background="#b0b0b0"
                    android:textColor="#000"/>

            </TableRow>
```



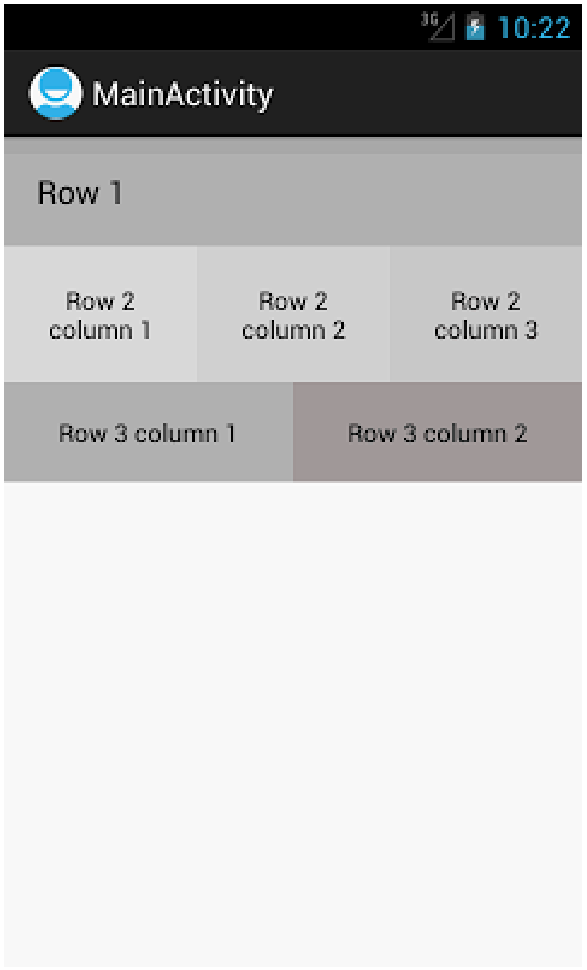
TableLayout Example

```
<!-- Row 2 with 3 columns -->

<TableRow
   android:id="@+id/tableRow1"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">

   <TextView
      android:id="@+id/TextView04"
      android:text="Row 2 column 1"
      android:layout_weight="1"
      android:background="#dcdcdc"
      android:textColor="#000000"
      android:padding="20dip"
      android:gravity="center"/>
   <TextView
      android:id="@+id/TextView04"
      android:text="Row 2 column 2"
      android:layout_weight="1"
      android:background="#d3d3d3"
      android:textColor="#000000"
      android:padding="20dip"
      android:gravity="center"/>
   <TextView
      android:id="@+id/TextView04"
      android:text="Row 2 column 3"
      android:layout_weight="1"
      android:background="#cac9c9"
      android:textColor="#000000"
      android:padding="20dip"
      android:gravity="center"/>
</TableRow>
```



TableLayout Example

```
    <!-- Row 3 with 2 columns -->

    <TableRow
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:gravity="center_horizontal">

        <TextView
            android:id="@+id/TextView04"
            android:text="Row 3 column 1"
            android:layout_weight="1"
            android:background="#b0b0b0"
            android:textColor="#000000"
            android:padding="18dip"
            android:gravity="center"/>

        <TextView
            android:id="@+id/TextView04"
            android:text="Row 3 column 2"
            android:layout_weight="1"
            android:background="#a09f9f"
            android:textColor="#000000"
            android:padding="18dip"
            android:gravity="center"/>
    </TableRow>

</TableLayout>
```

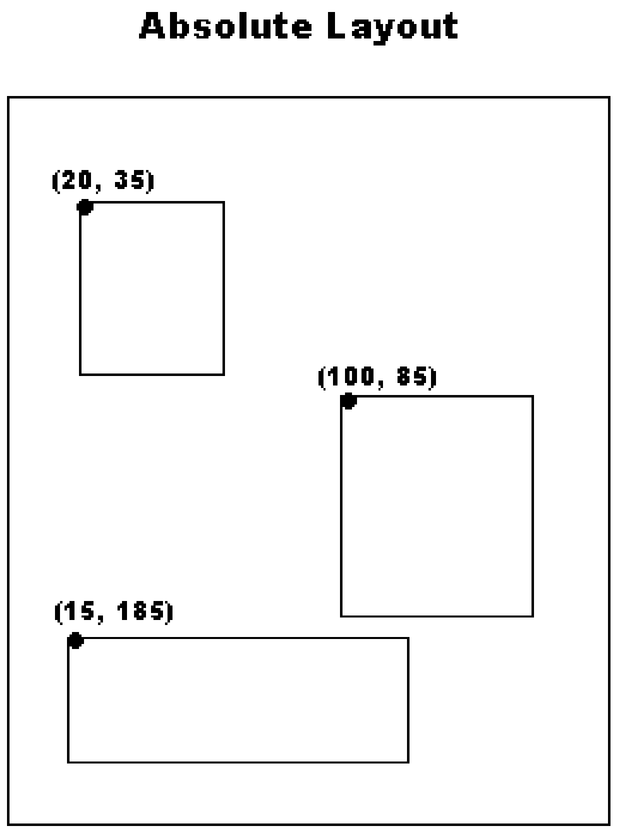


GridLayout

- 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
- child views specify row and column they are in or what rows and columns they span
- Gives greater design flexibility
- For more details see section “Introducing GridLayout” in Busy Coders (pg 1021)

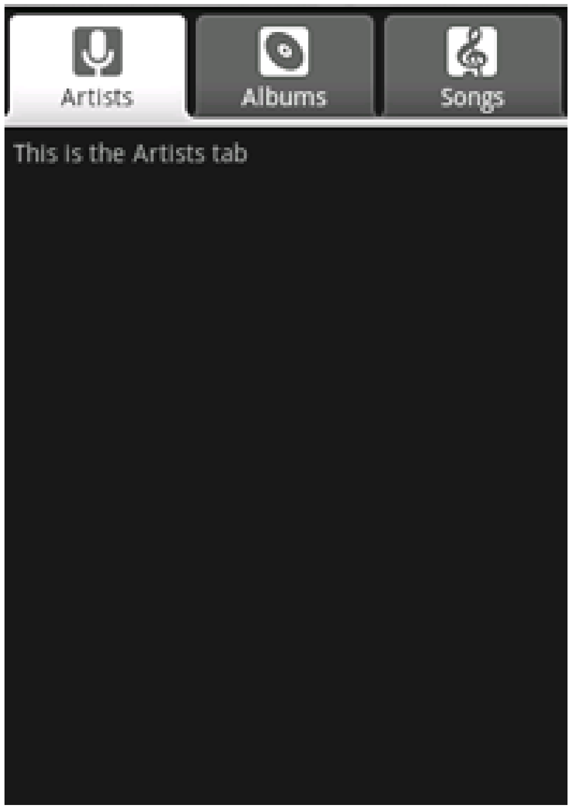
Absolute Layout

- Allows specificification of exact locations (x/y coordinates) of its children.
- Less flexible and harder to maintain than other types of layouts



Other Layouts - Tabbed Layouts

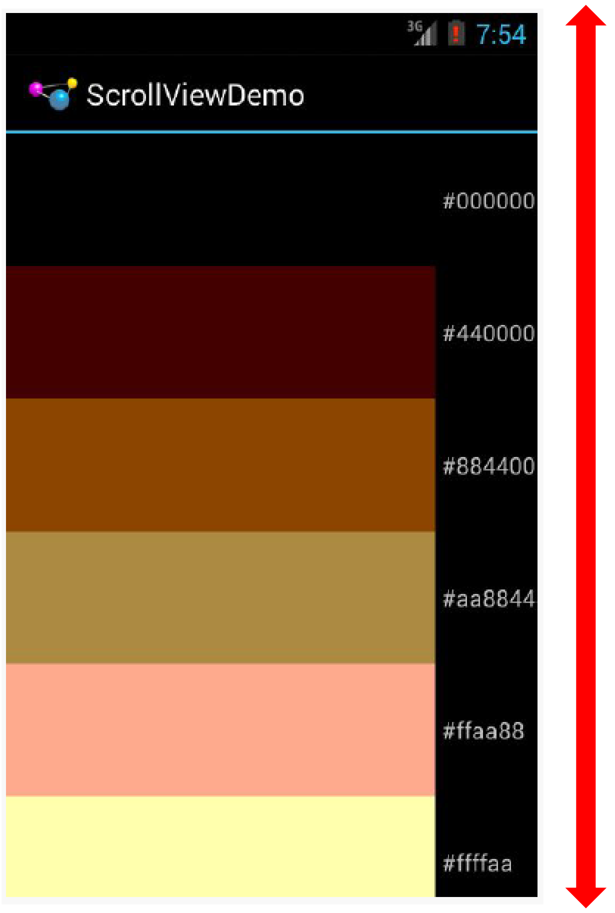
- Uses a TabHost and TabWidget
- TabHost consists of TabSpecs
- Can use a TabActivity to simplify some operations
- Tabs can be
  - predefined View
  - Activity launched via Intent
  - generated View from TabContentFactory



This is the Artists tab

Scrolling

- Phone screens are small, scrolling content helps
- ListView supports vertical scrolling
- Other views for Scrolling:
  - **ScrollView** for vertical scrolling
  - **HorizontalScrollView**
- Only one child View
  - but could have children of its own
- examples:
  - scroll through large image
  - Linear Layout with lots of elements
- Cannot contain scrollable items

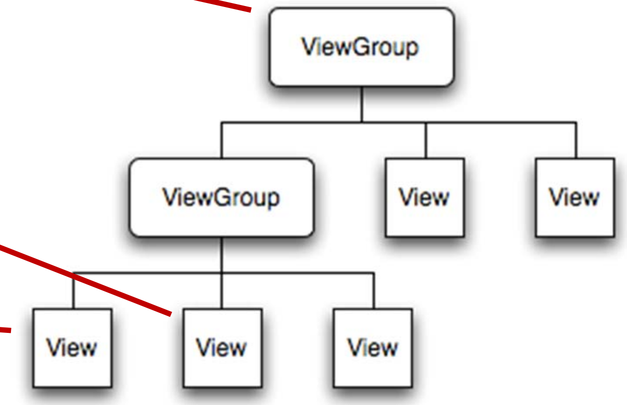


Android Views, Widgets and ViewGroups

Views and ViewGroups

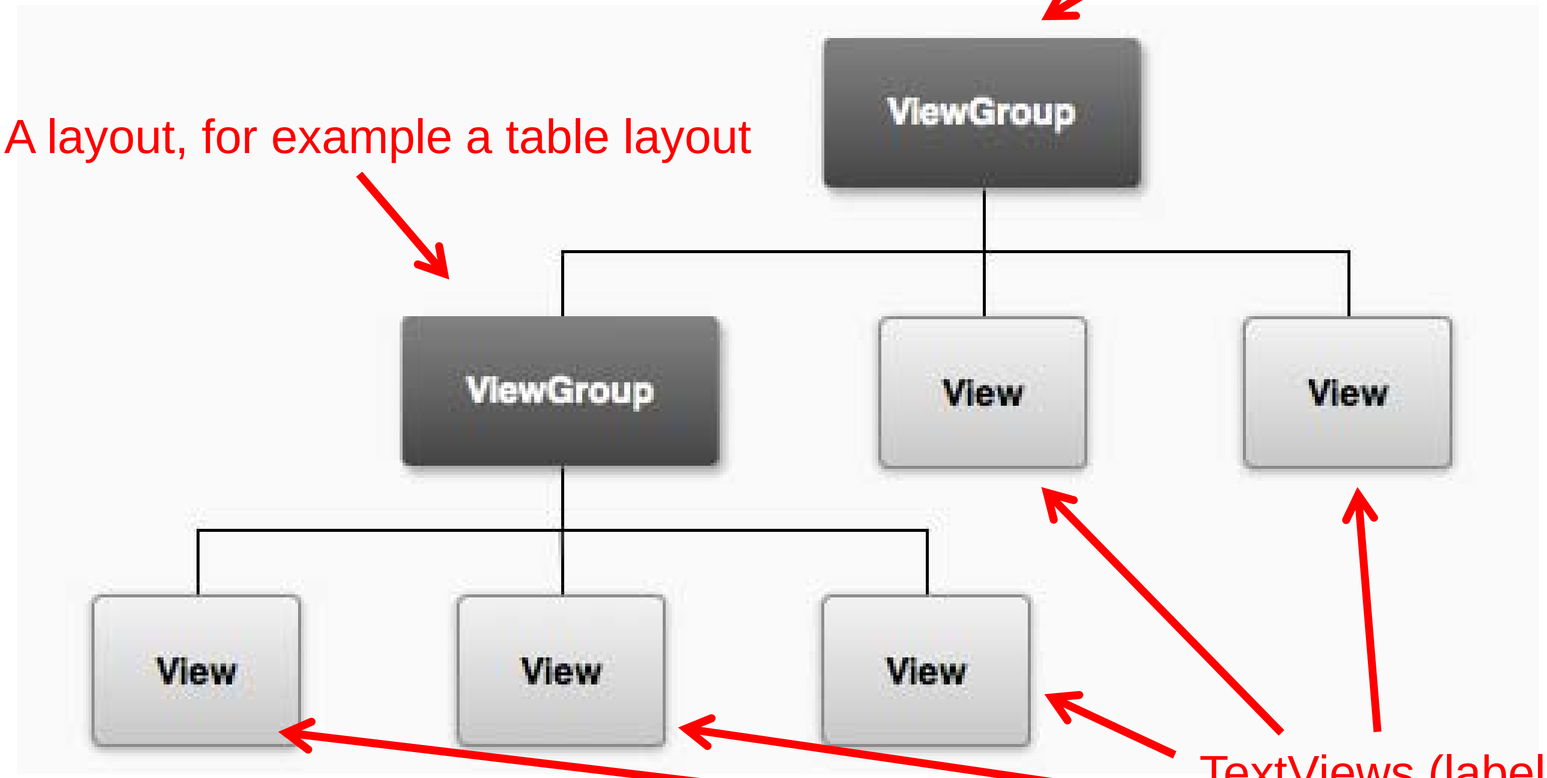
- A view (e.g. buttons, text fieds) is basic UI building block
- View occupies rectangular area on screen
- ViewGroup (e.g. a layout) contains multiple Views

```
<?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" >

    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <Button
        android:id="@+id/hello_button"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Press Me" />

</LinearLayout>
```



Tree from: http://developer.android.com/guide/topics/ui/index.html

Views and ViewGroups



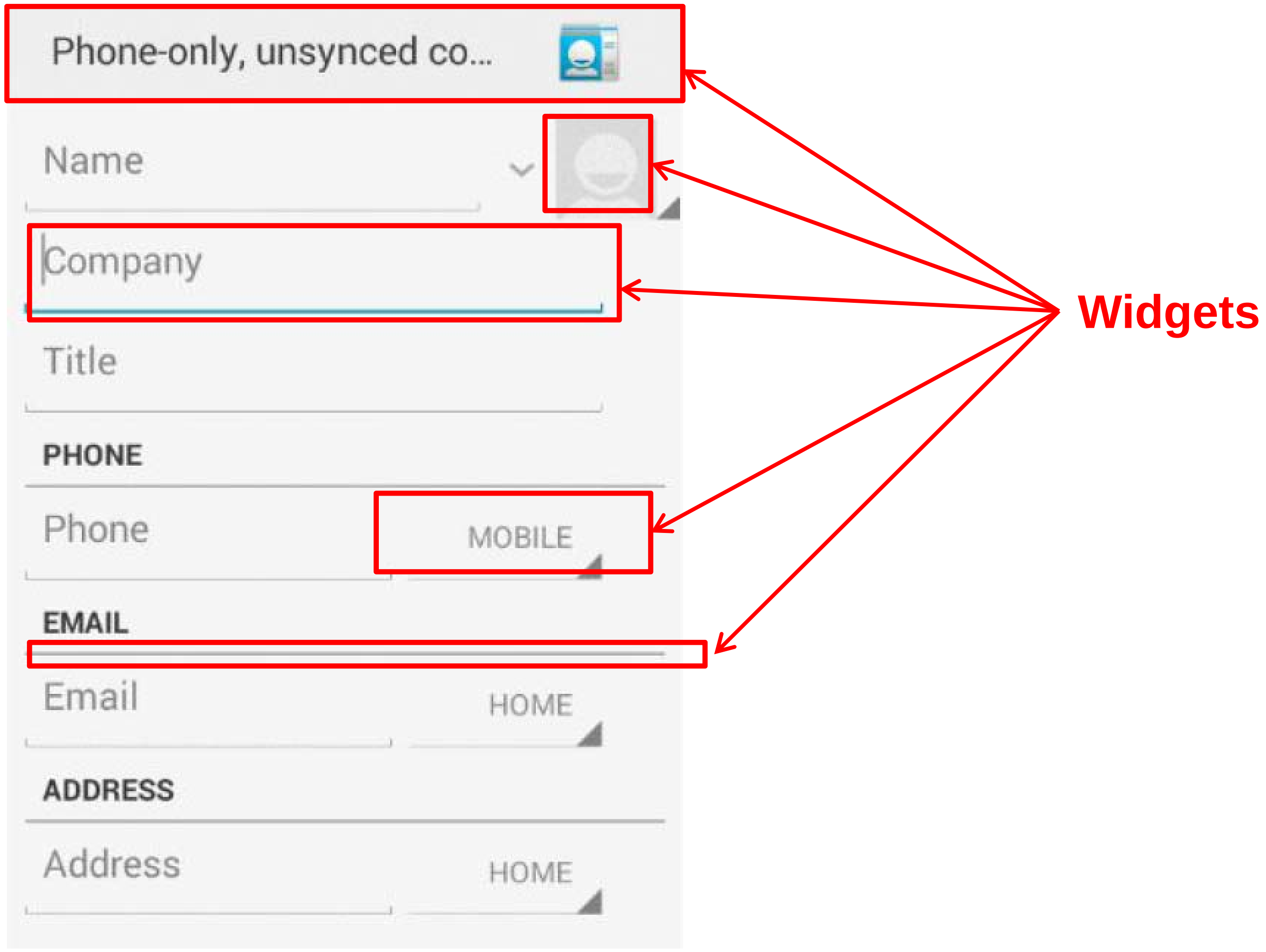
A layout, for example a linear layout

A layout, for example a table layout

TextViews (labels), ImageViews,
Controls such as buttons, etc.

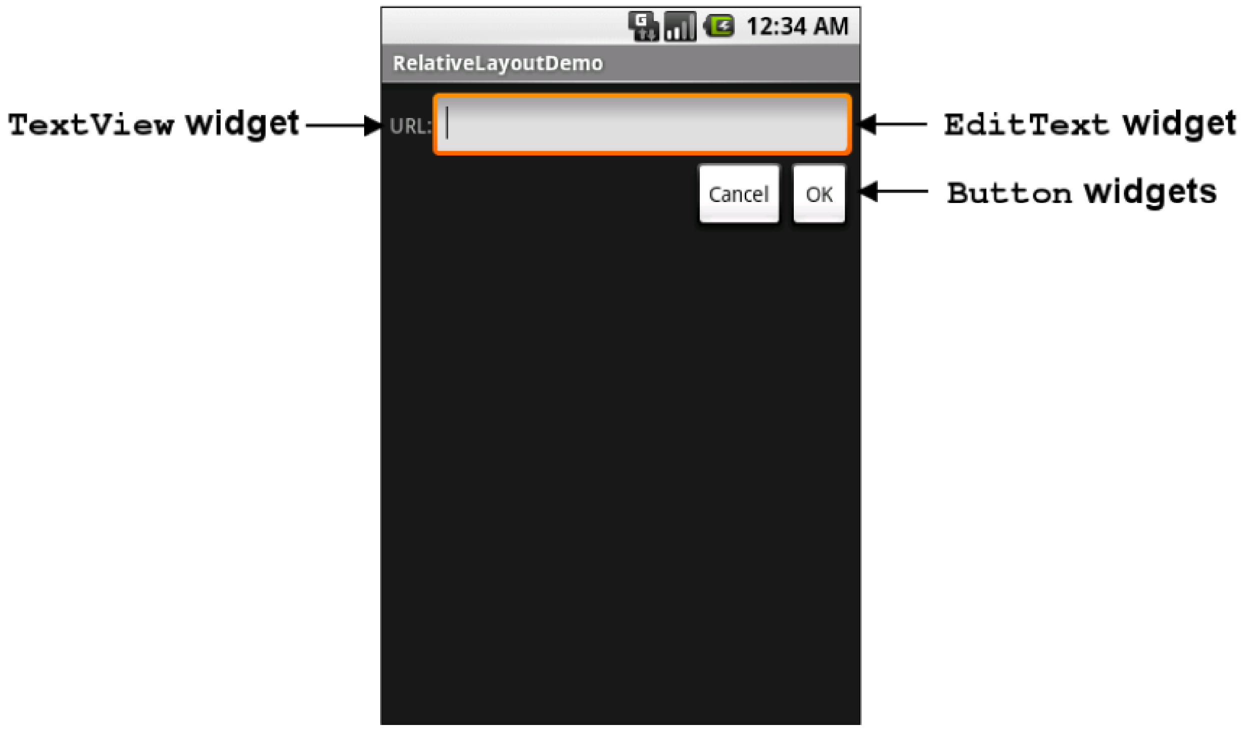
Widgets

- Widgets are visual building blocks used to compose Android screens (Activities)
- Need to specify size, margins and padding of widgets



Widgets

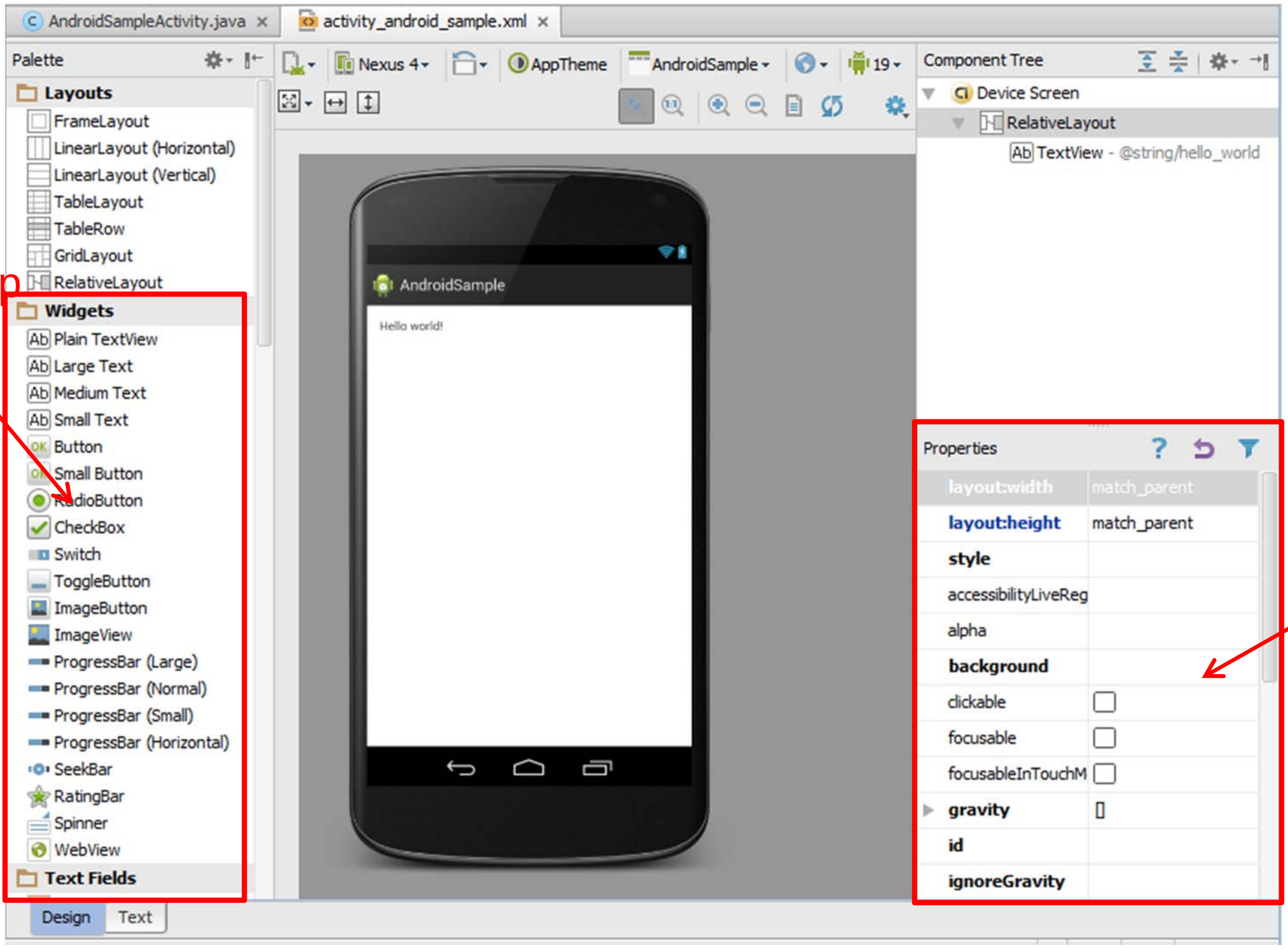
- Most Android UI developed using widgets (fields, lists, text boxes, buttons, etc)
- Can also render using OpenGL2D or OpenGL3D



Adding Button using Widget

- Can drag and drop widgets, layouts in Android Studio
- Can also edit their properties (e.g. height, width, color, etc)

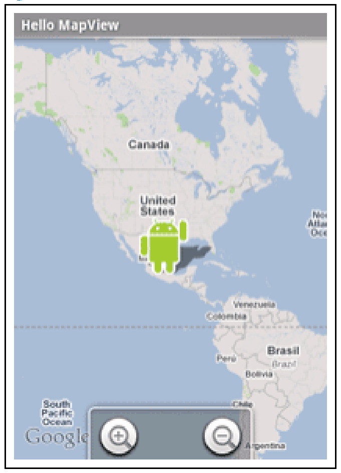
Drag and drop button or any other widget or view



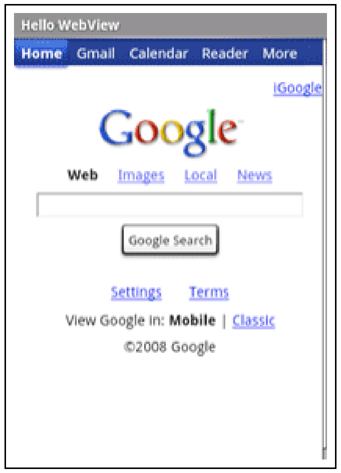
Edit widget properties

Other Available Widgets

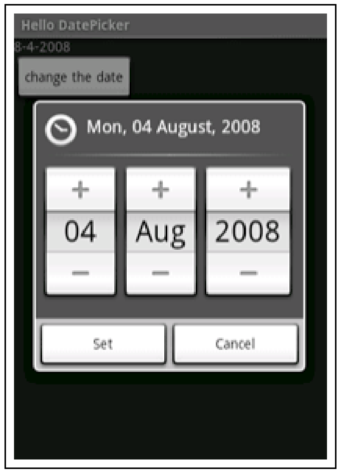
MapView



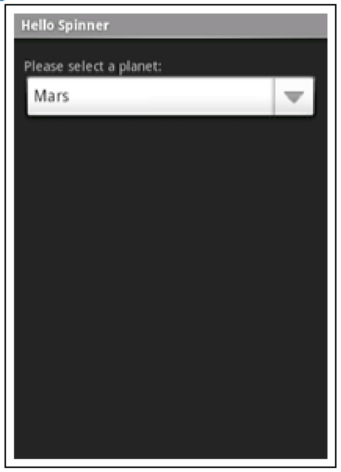
WebView



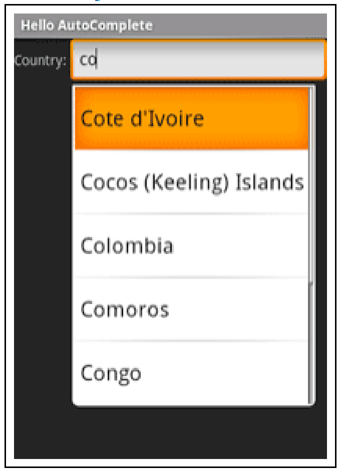
DatePicker



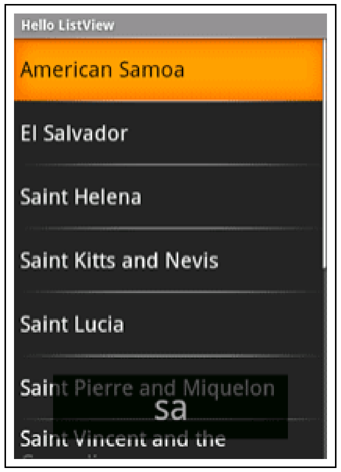
Spinner



AutoComplete



ListView



http://developer.android.com/resources/tutorials/views/index.html

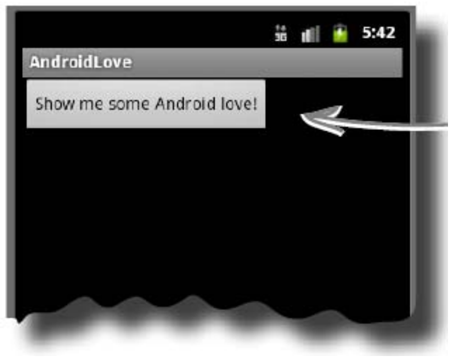
Containers

- Containers provide structured way of organizing multiple widgets
- Containers have children (widgets or other containers)
- Rules used to specify how children of containers are laid out. E.g:
  - Put all children in a row
  - Put all children in a column
  - Arrange children into a table or grid with X rows and Y columns
- Containers have size, margins and padding

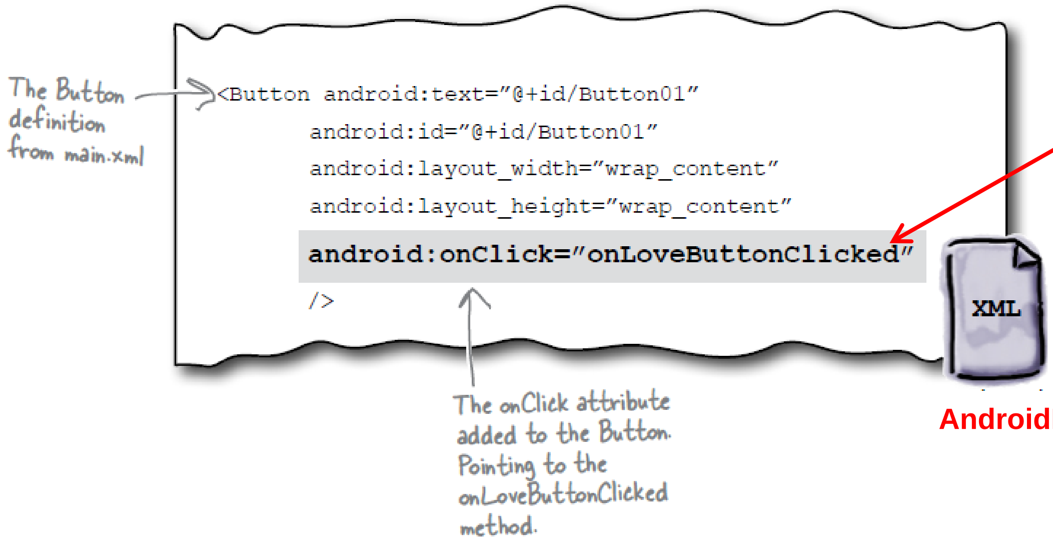
Android UI Components: Controls

Example: Make Button Responding to Clicks

- **Task:** Display some text when user clicks a button



- In declaration of the button, add property “onClick”, give name of method to call onClick


The Button definition from main.xml
The onClick attribute added to the Button. Pointing to the onLoveButtonClicked method.
XML

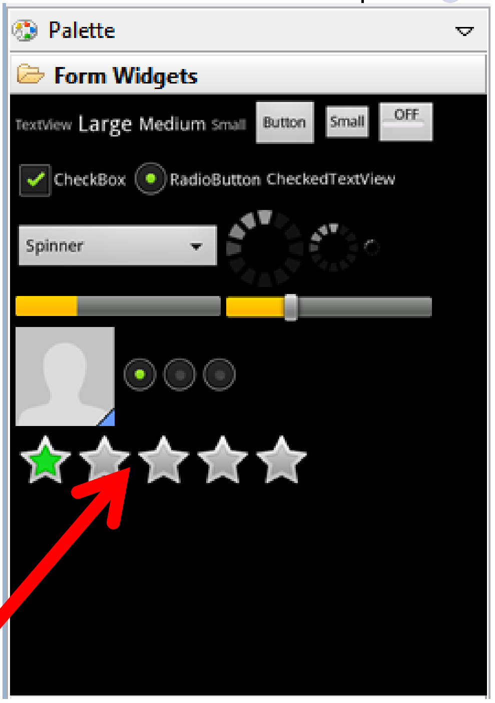
```
<Button android:text="@+id/Button01"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onLoveButtonClicked"
        />
```

This method has to be implemented in java file

AndroidMain.XML

Adding Controls

- Controls can be added in XML layout or in Java code
- Can drag and drop to add component in visual editor
  - XML code automatically generated
  - tweak XML code as desired

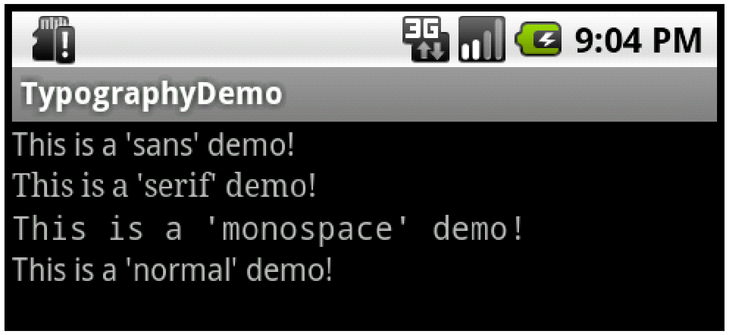


```
<RatingBar
    android:id="@+id/ratingBar1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

TextView

- Text in a rectangle, a simple label
- display information, not for interaction
- **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
  - units for width / height: px (pixels), dp or dip (density-independent pixels 160 dpi base), sp (scaled pixels based on preferred font size), in (inches), mm (millimeters)
  - recommended units: sp for font sizes, and dp for everything else
  - http://developer.android.com/guide/topics/resources/more-resources.html#Dimension

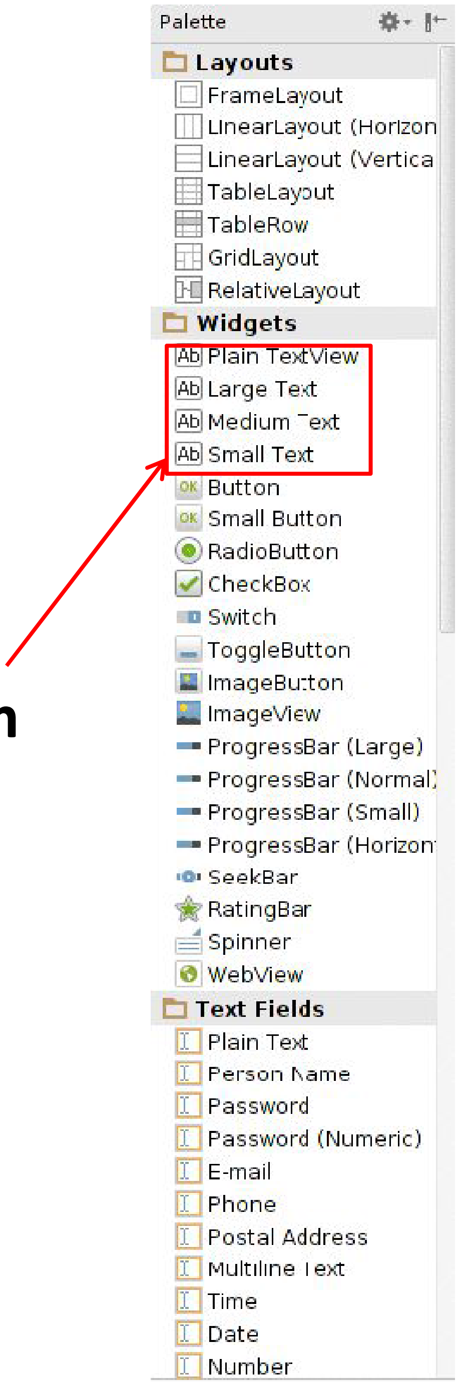
```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is a 'sans' demo!"
    android:typeface="sans"
/>
```



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

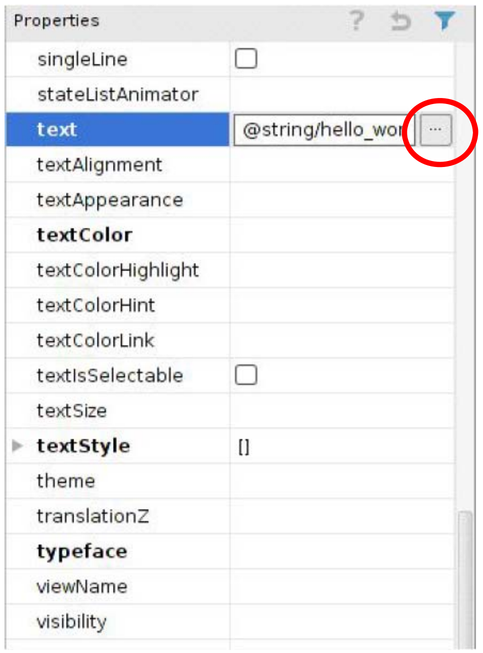
TextView

- 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
- See demo project: Basic/Label

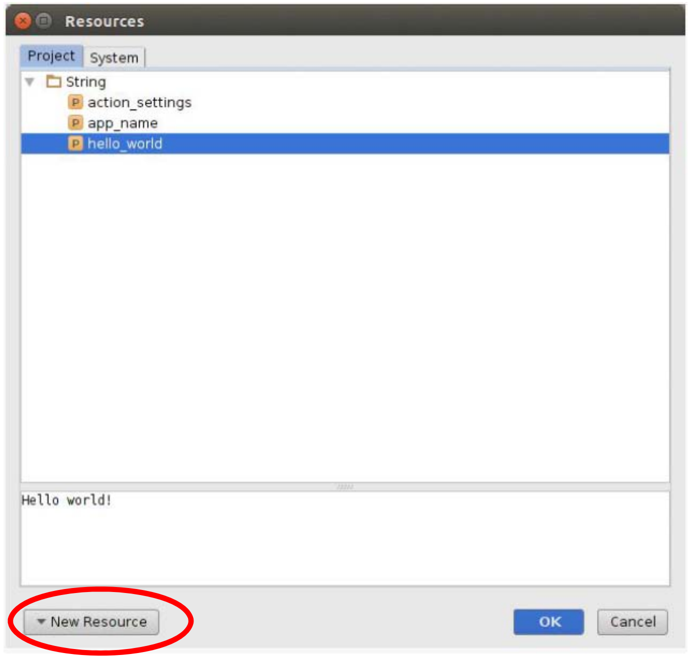


Setting Text Properties

- Can edit text properties
- Can either type in literal string or click … button to pick a string you have previously declared (e.g. in strings.xml)
- You can also declare new string by clicking on "New Resource"



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



Resources

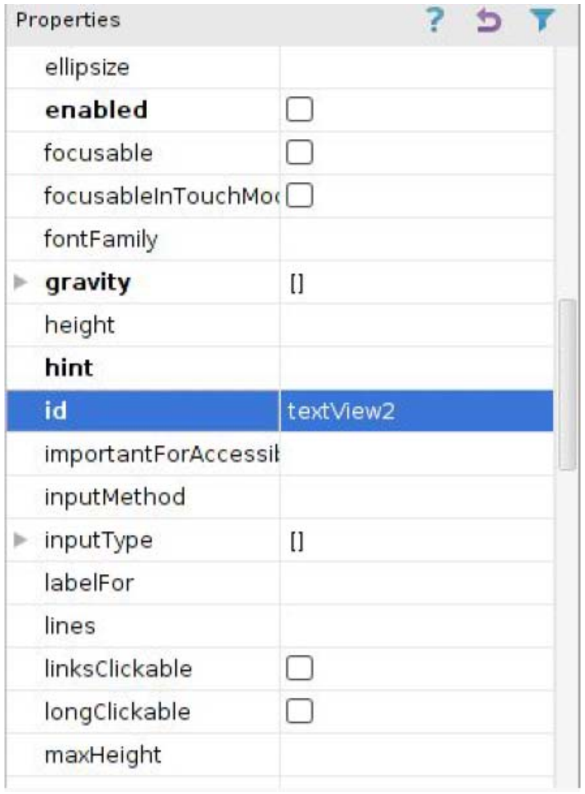
Project | System

String
- action_settings
- app_name
- hello_world

```
Hello world!
```

Widget ID

- Every widget has ID whose value is stored in **android:id** attribute
- To manipulate this widget or set its attributes in Java code, need to reference it using its ID
- More on this 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	

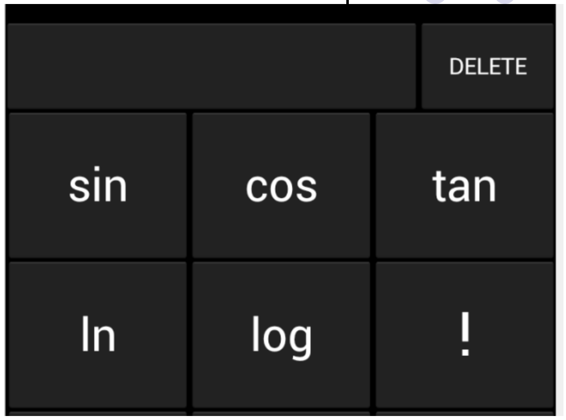
Other TextView Attributes

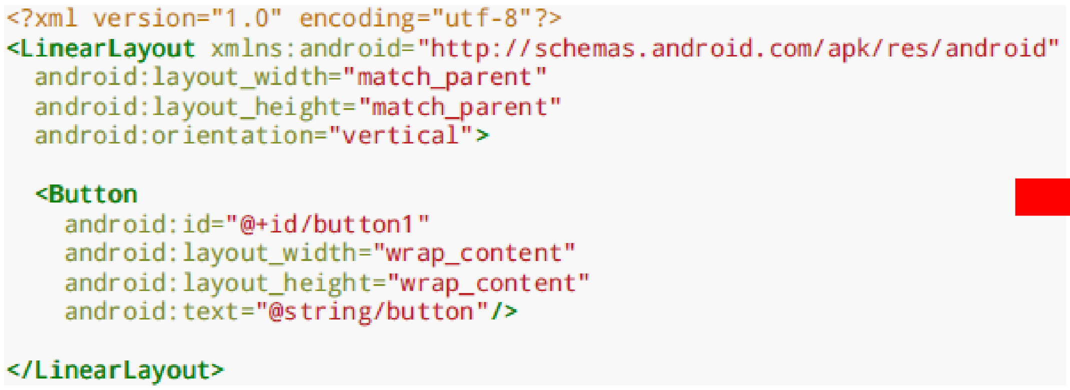
- set number of lines of text that are visible
  - android:lines="2"

- contextual links to email address, url, phone number,
  - autolink attribute set to none, web, email, phone, map, or all

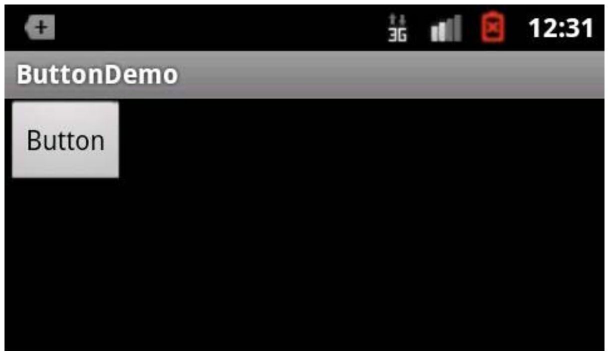
Button Widget

- Text or icon or both on View (Button)
- E.g. “Click Here”
- Declared as subclass of TextView so similar attributes
- Appearance of buttons can be customized
  http://developer.android.com/guide/topics/ui/controls/button.html#CustomBackground



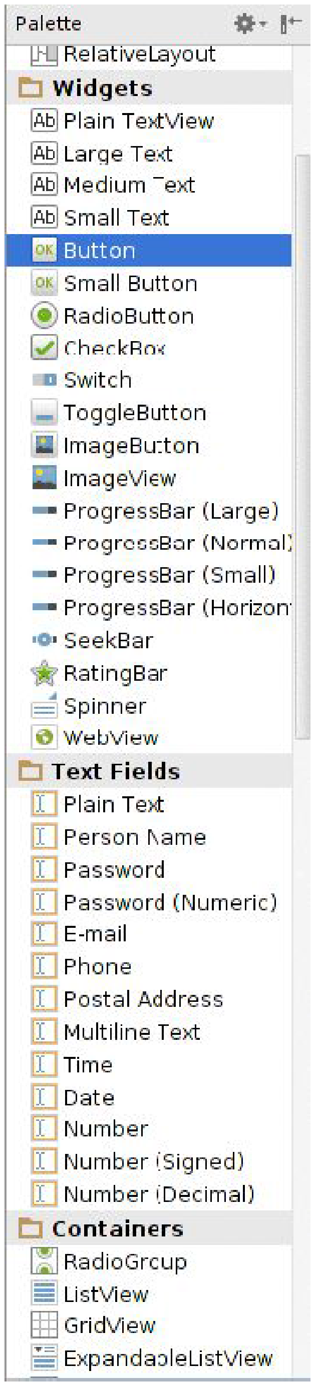


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

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

</LinearLayout>
```



Button in Android Studio

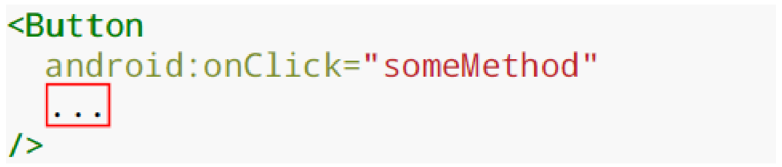
- **Button** widget available in palette of Android Studio graphical layout editor
- Can drag and drop button, edit attributes as with TextView
- See demo project: Basic/Button



Responding to Button Clicks

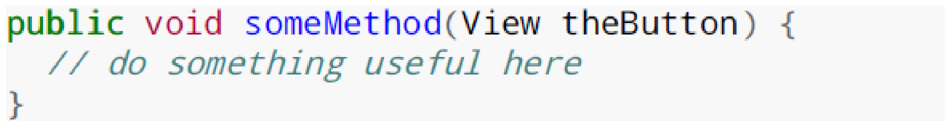
- May want Button press to trigger some action
- How?

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



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

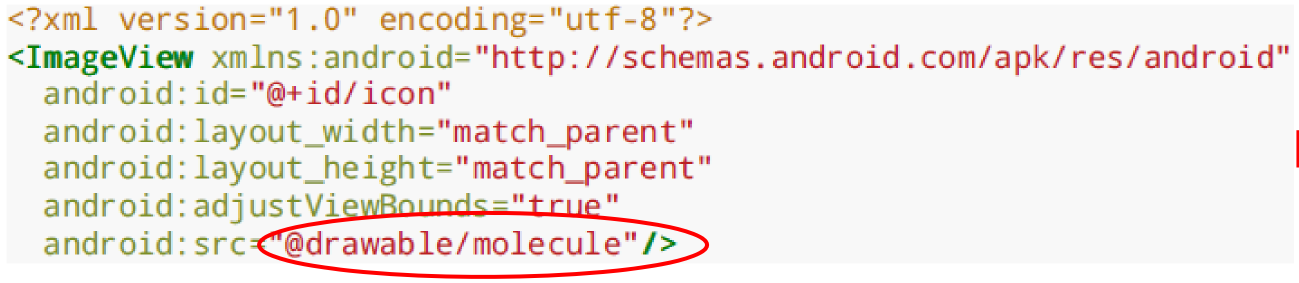
2. **In Java file (e.g. MainActivity.java) declare method/handler to take desired action**



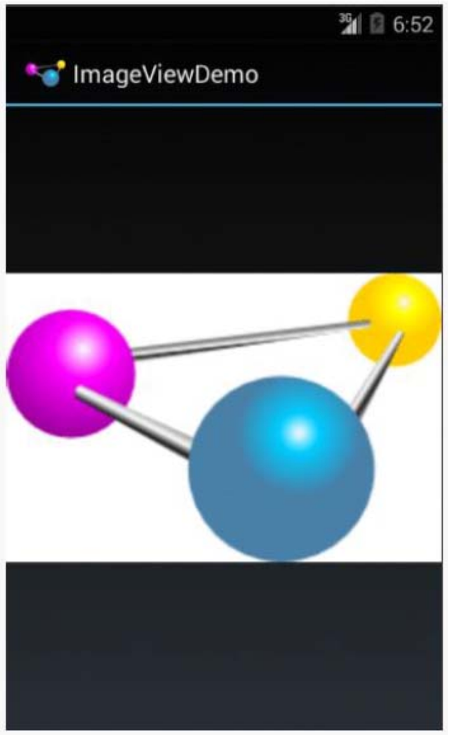
```
public void someMethod(View theButton) {
  // do something useful here
}
```

Embedding Images: ImageView and ImageButton

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

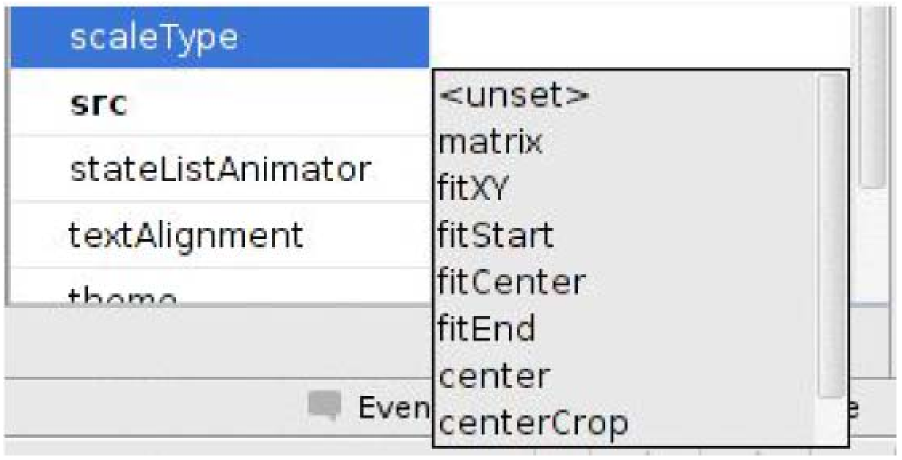


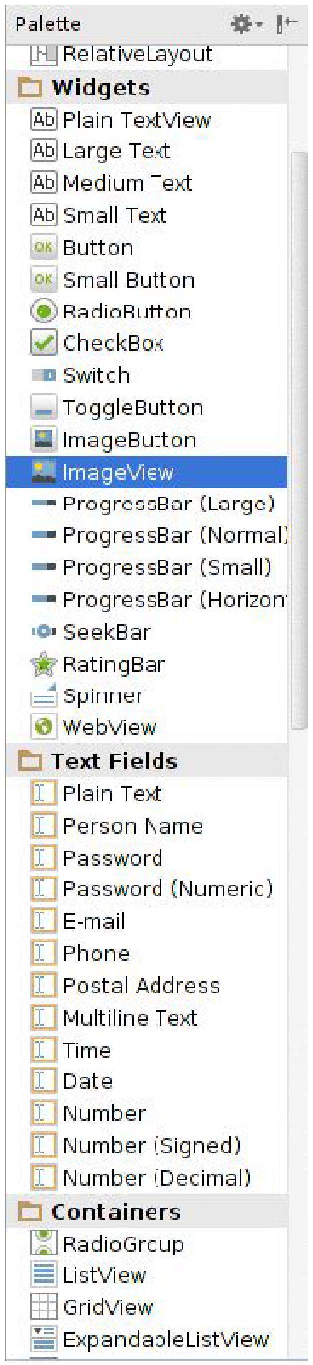
```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/icon"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:adjustViewBounds="true"
  android:src="@drawable/molecule"/>
```



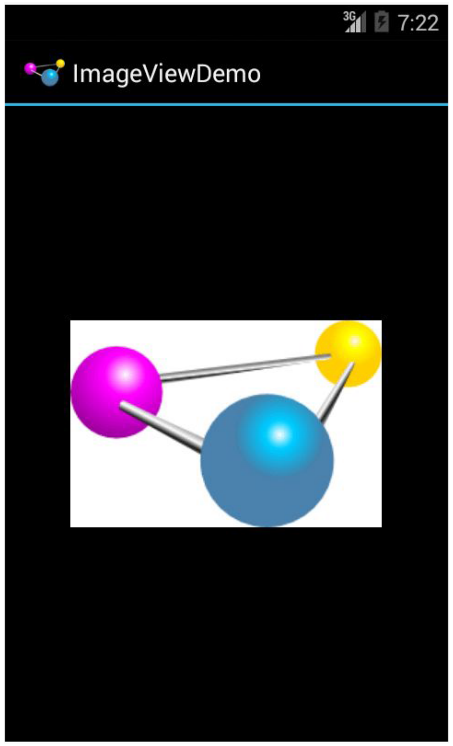
ImageView in Widgets Palette

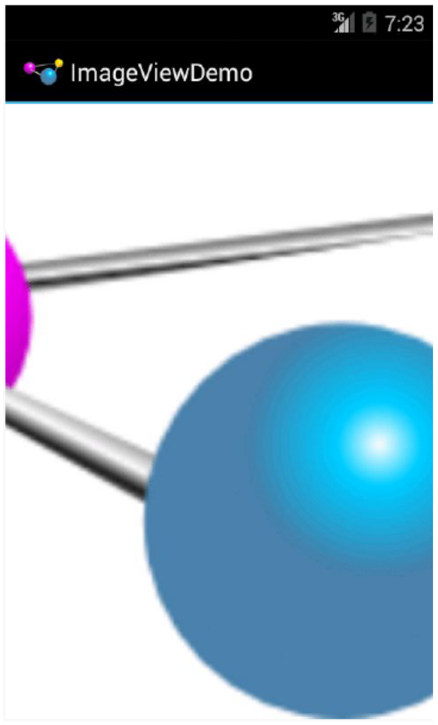
- Can drag and drop ImageView from Widgets Palette
- Can also use menus to specify:
  - **src:** to choose image to be displayed
  - **scaleType:** to choose how image should be scaled

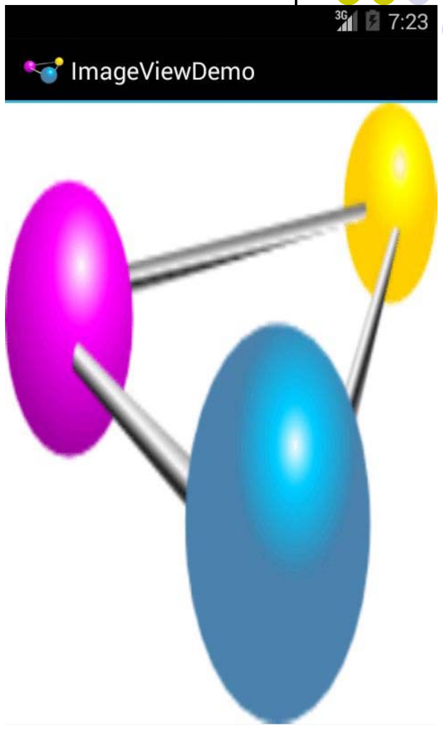




Options for Scaling Images (scaleType)







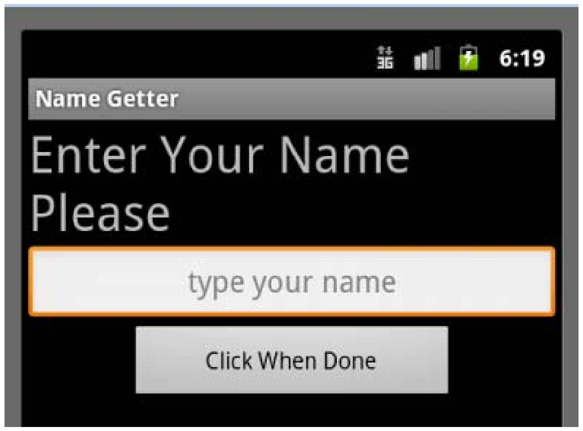
"center" centers image but does not scale it

"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

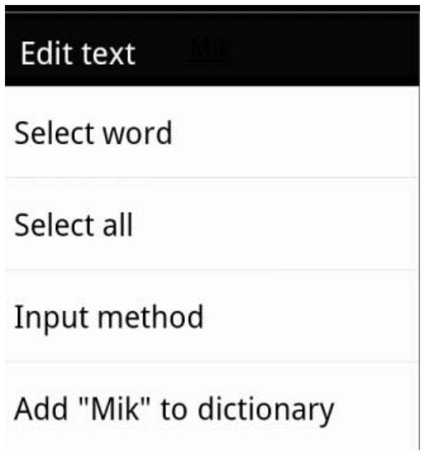
EditText Widget

- UI Component used to get information from user



- long press brings up context menu
- 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" />
```

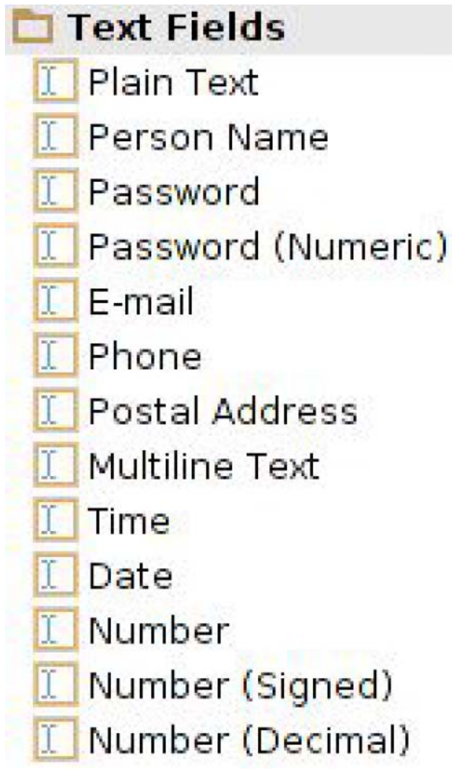


EditText

- can span multiple lines via android:lines attribute
- Text fields can have different input types such as number, date, password, or email address
- **android:inputType** attribute sets input type, affects
  - What type of keyboard pops up for user
  - Behaviors such as is every word capitalized

EditText Widget in Android Studio Palette

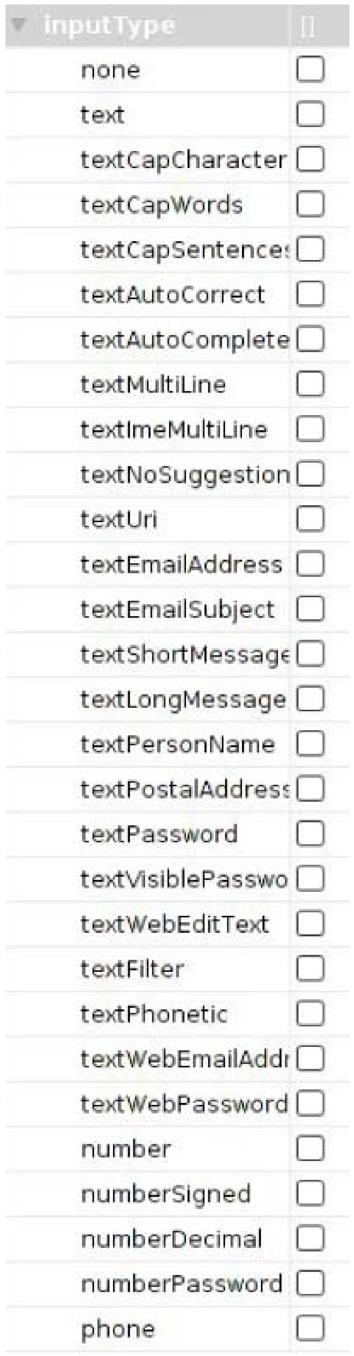
- A whole section of the Android Studio palette dedicated to EditText widgets (or text fields)



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	☐

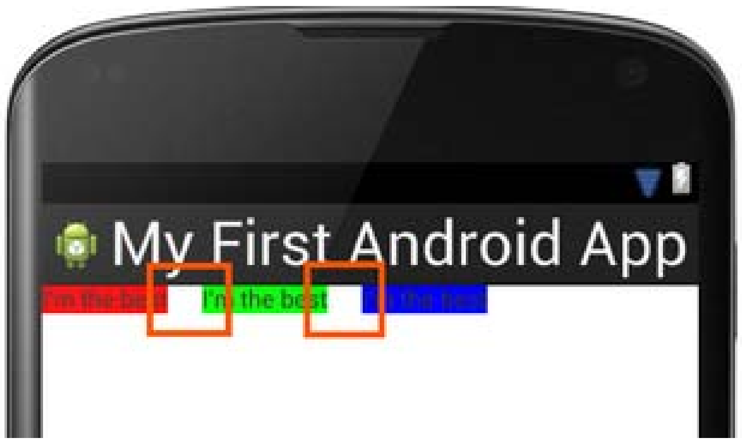
EditText inputType menu

Widget Attributes

- Some attributes apply to most types of widgets
- **Padding:** Can either set all sides (android:padding) or per-side (e.g. android:paddingLeft)
  - Units either in DIP or millimeters
- **Margins:** Can be set for all sides (android:layout_margin) or per-side (e.g. android:layout_marginTop)
  - Units either in dp or DIP
- **Colors:**
  - Some colors attributes take single color (e.g. android:background)
  - Other attributes take ColorStateList (different colors under different conditions)

Margin Example

```
<TextView
android:id="text1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginRight="20dp"
android:text="@string/my_best_text"
android:background="#FF0000"
/>

<TextView
android:id="text2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginRight="20dp"
android:text="@string/my_best_text"
android:background="#00FF00"
/>
```

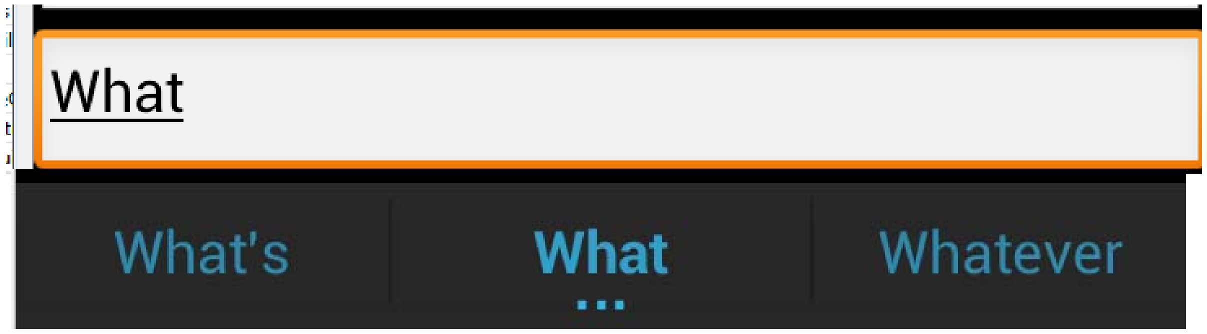


Widget Attributes: Other Attributes

- **android:visibility:** Controls whether the widget is visible
- **android:contentDescription:**
  - similar to alt attribute on an HTML <img>
  - Defines content that briefly defines the content of the widget
  - Very important for widgets like **ImageView**

Auto Complete Options

- Depending on EditText inputType suggestions can be displayed
  - works on actual devices

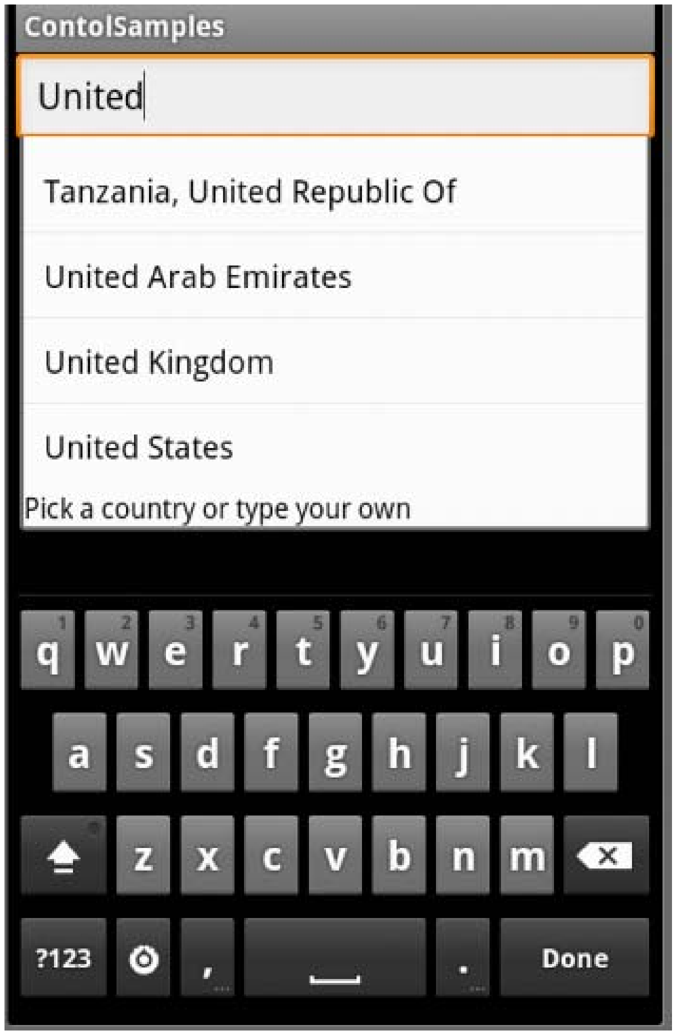


- Other options for exist for auto complete from list
  - AutoCompleteTextView
    - choose one option
  - MultiAutoCompleteTextView
    - choose multiple options (examples tags, colors)

AutoCompleteTextView

- Two types
  - we provide list of choices
  - user provides list
- Developer list
  - use ArrayAdapter connected to array
  - best practice: put array in array.xml file

AutoComplete Using Array


ContolSamples

United

Tanzania, United Republic Of

United Arab Emirates

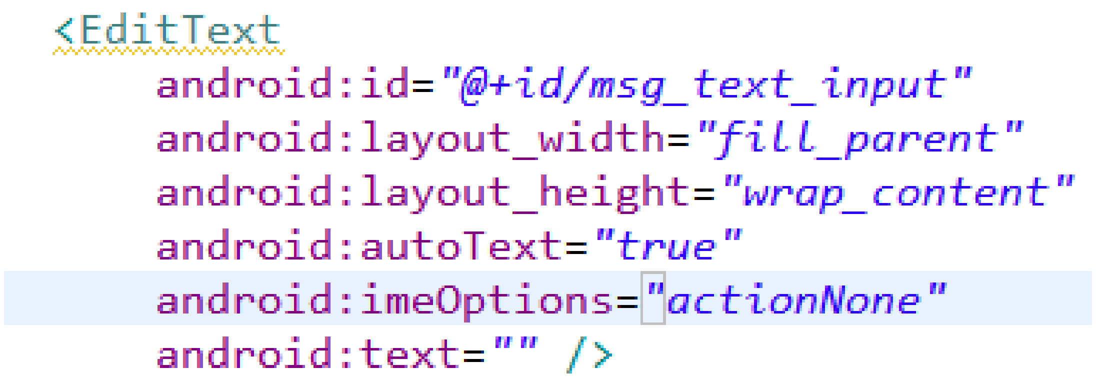
United Kingdom

United States

Pick a country or type your own

EditText

- Auto complete option using device dictionary:

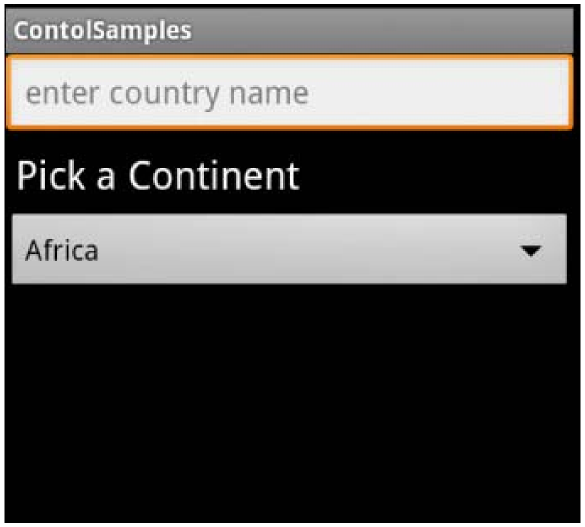


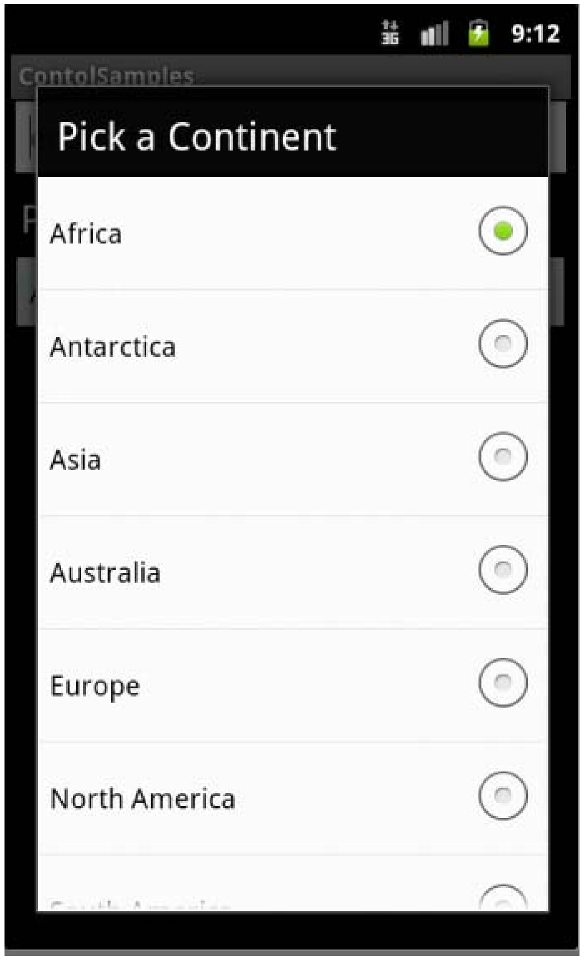
```
<EditText
    android:id="@+id/msg_text_input"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:autoText="true"
    android:imeOptions="actionNone"
    android:text="" />
```

- No dictionary on emulator!

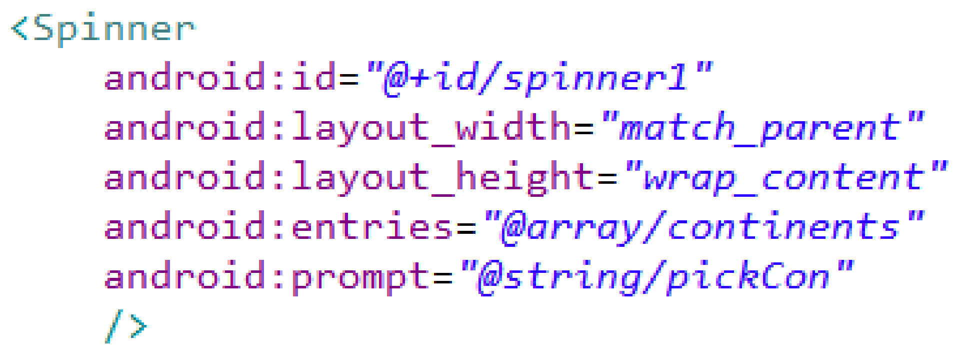
Spinner Controls

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



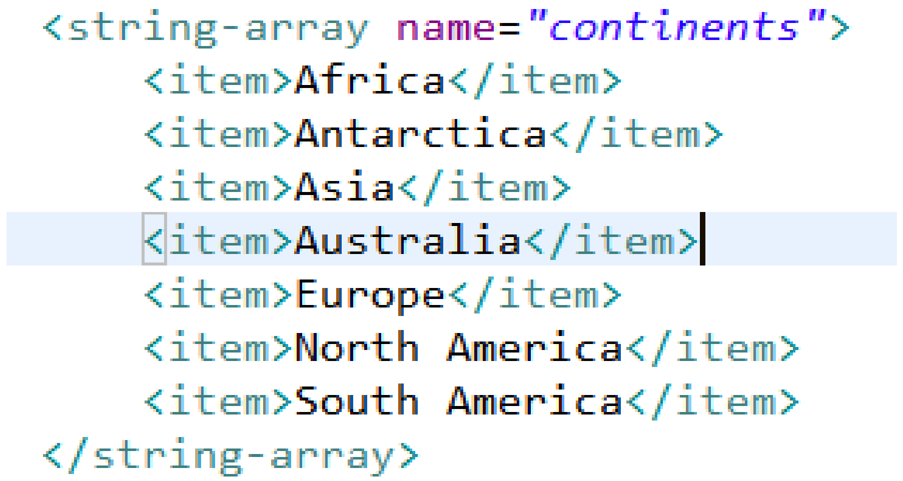


Spinner Control



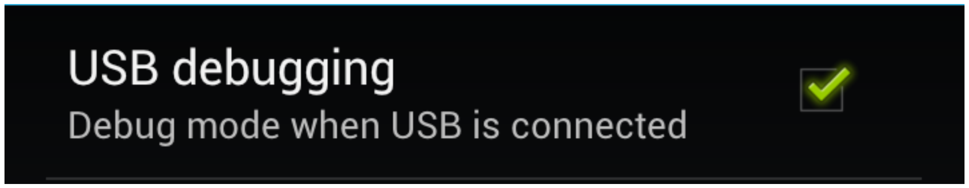
```
<Spinner
    android:id="@+id/spinner1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/continents"
    android:prompt="@string/pickCon"
    />
```

arrays.xml in res/values



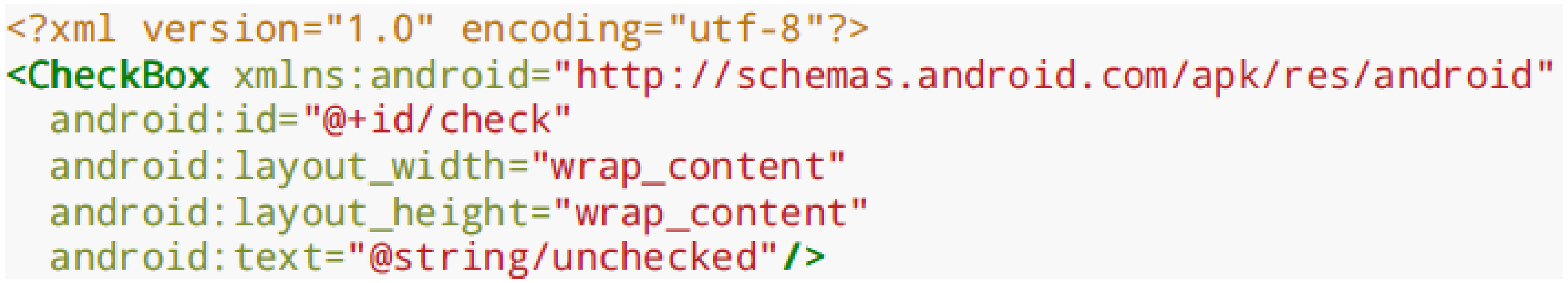
```
<string-array name="continents">
    <item>Africa</item>
    <item>Antarctica</item>
    <item>Asia</item>
    <item>Australia</item>
    <item>Europe</item>
    <item>North America</item>
    <item>South America</item>
</string-array>
```

Checkbox


USB debugging

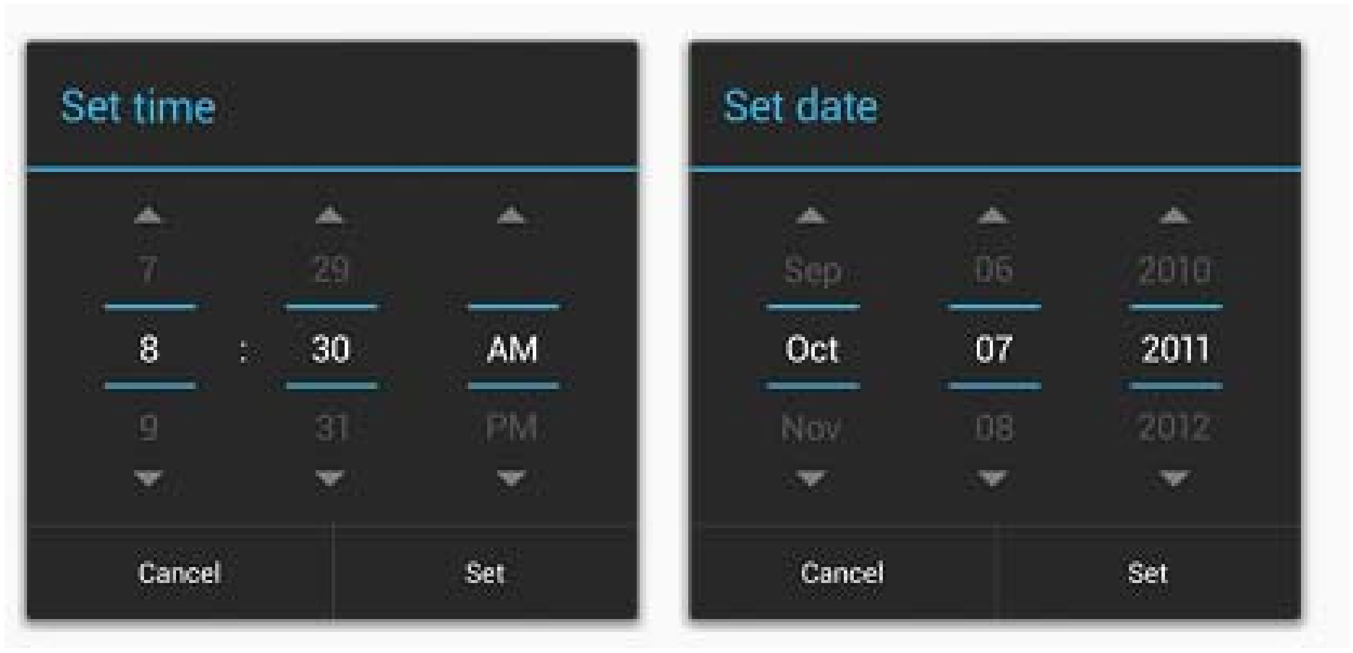
Debug mode when USB is connected

- Checkbox has 2 states: checked and unchecked
- Clicking on checkbox toggles between these 2 states
- Used to indicate a choice (e.g. Add rush delivery)
- 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"/>
```

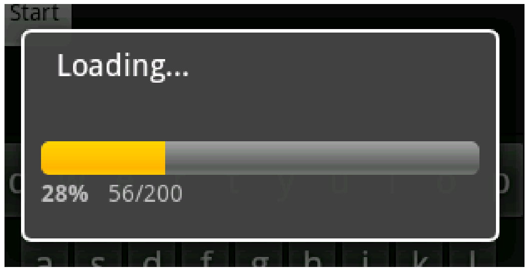
Pickers

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

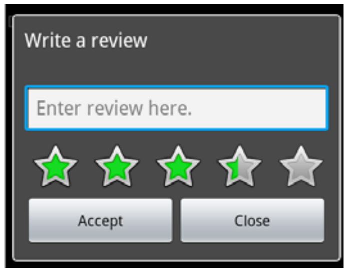


Indicators

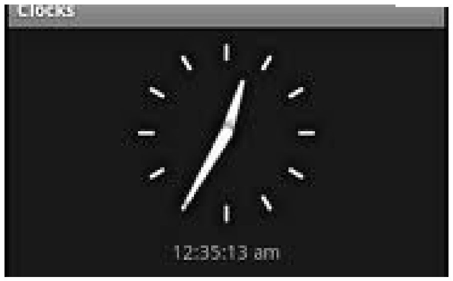
- Variety of built in indicators in addition to TextView
- ProgressBar



- RatingBar



- Chronometer
- DigitalClock
- AnalogClock



Android UI Youtube Tutorials

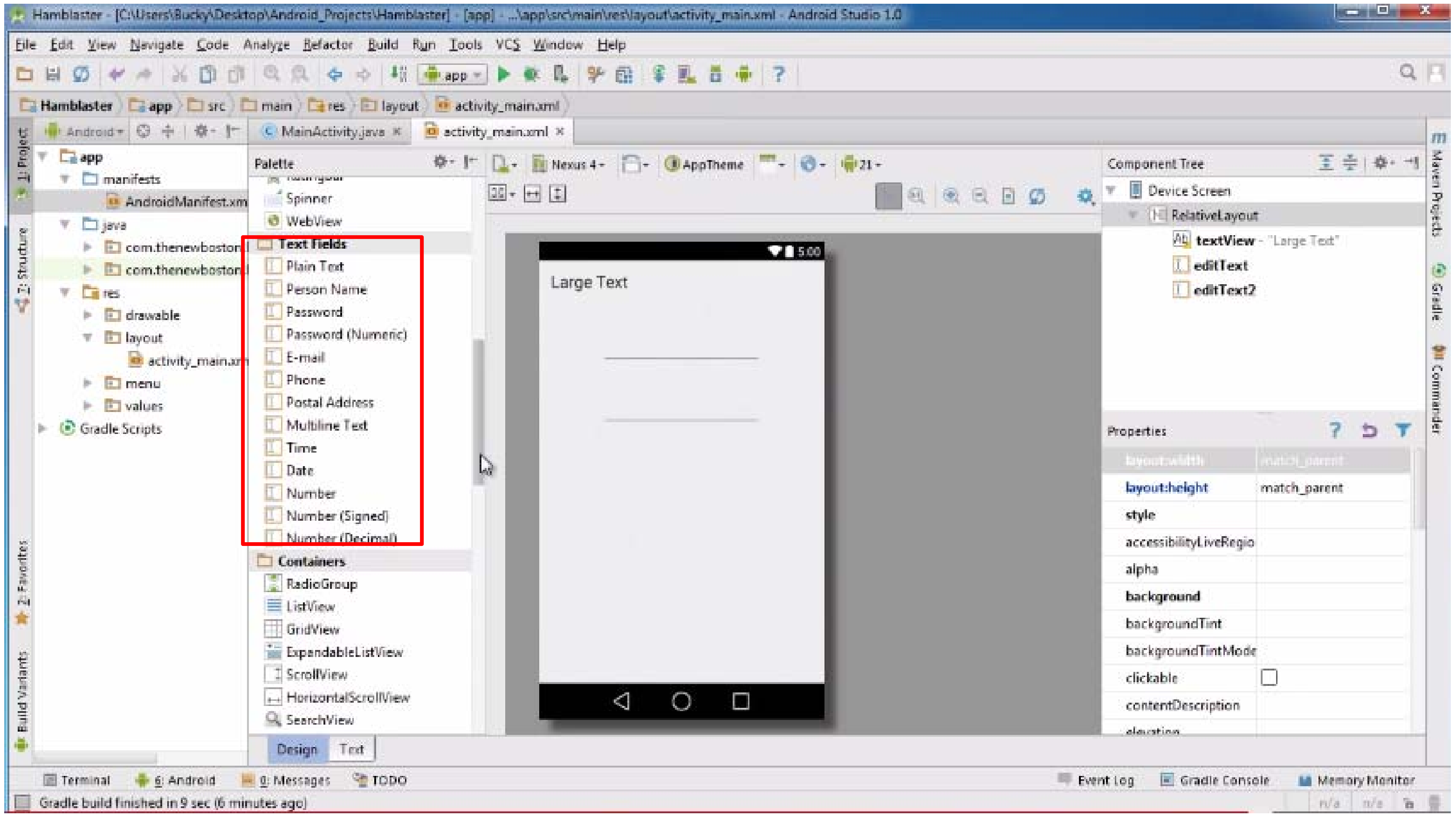
Tutorial 11: Designing the User Interface

- 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

Drag and Drop in Widgets

- Android Studio creates 2 files as usual (MainActivity.java, activity_main.xml)
- Drag and drop in widgets (e.g. Large text, Text boxes)



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

Gradle build finished in 9 sec (6 minutes ago)

Tutorial 12: More on User Interface

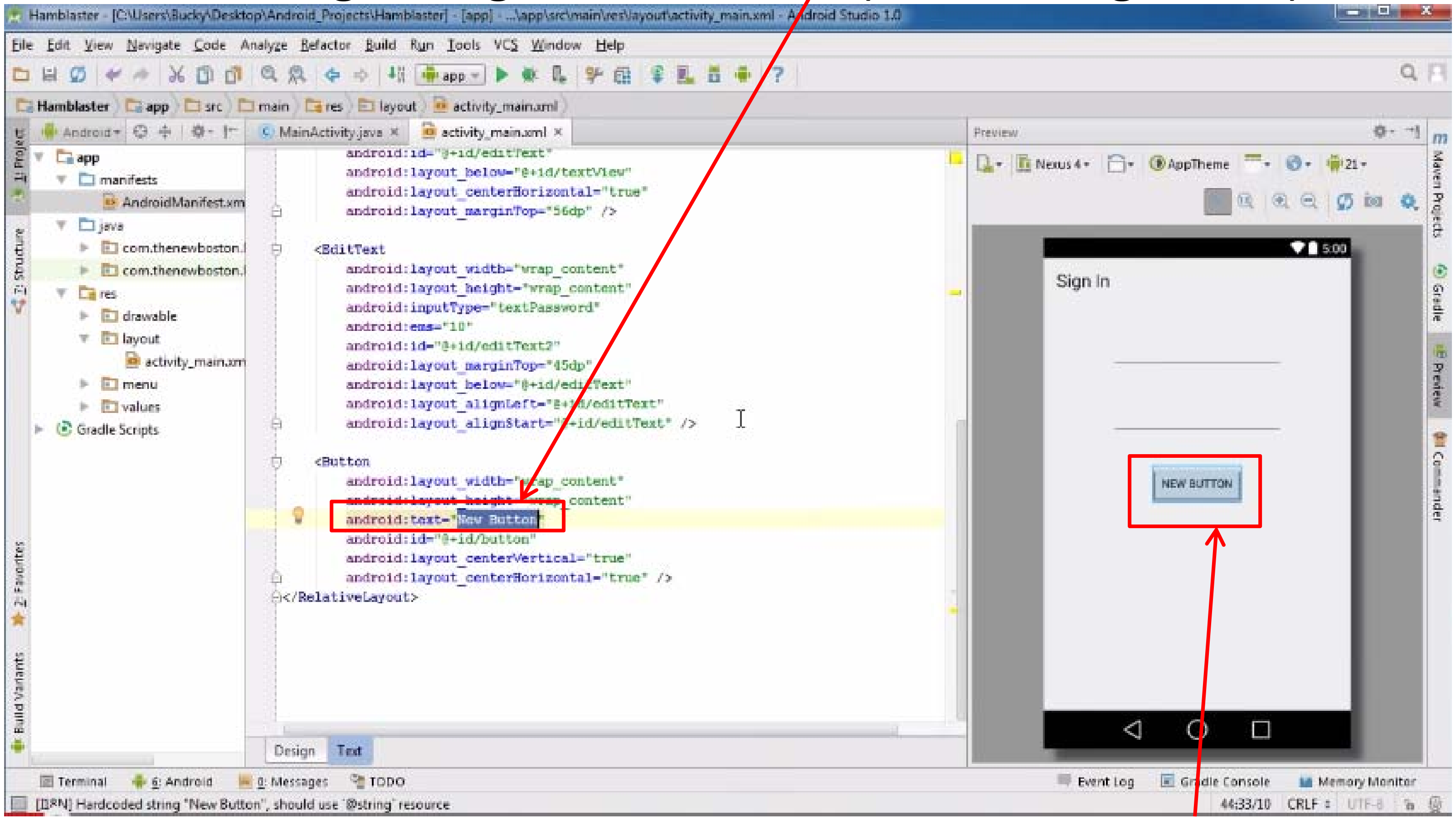
- Tutorial 12: More on User Interface [10:24 mins]
  - https://www.youtube.com/watch?v=72mf0rmjNAA

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

Changing Widget text in Text View

- E.g. Change text on New Button in activity_main.xml
- Can also change widget dimensions (width, height, etc)

Change text “New Button” in XML file,



Hamblaster - [C:\Users\Bucky\Desktop\Android_Projects\Hamblaster] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 1.0

```
        android:id="@+id/editText"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="56dp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/editText2"
        android:layout_marginTop="45dp"
        android:layout_below="@+id/editText"
        android:layout_alignLeft="@+id/editText"
        android:layout_alignStart="@+id/editText" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

[I18N] Hardcoded string "New Button", should use '@string' resource

We want to change Text “New Button”

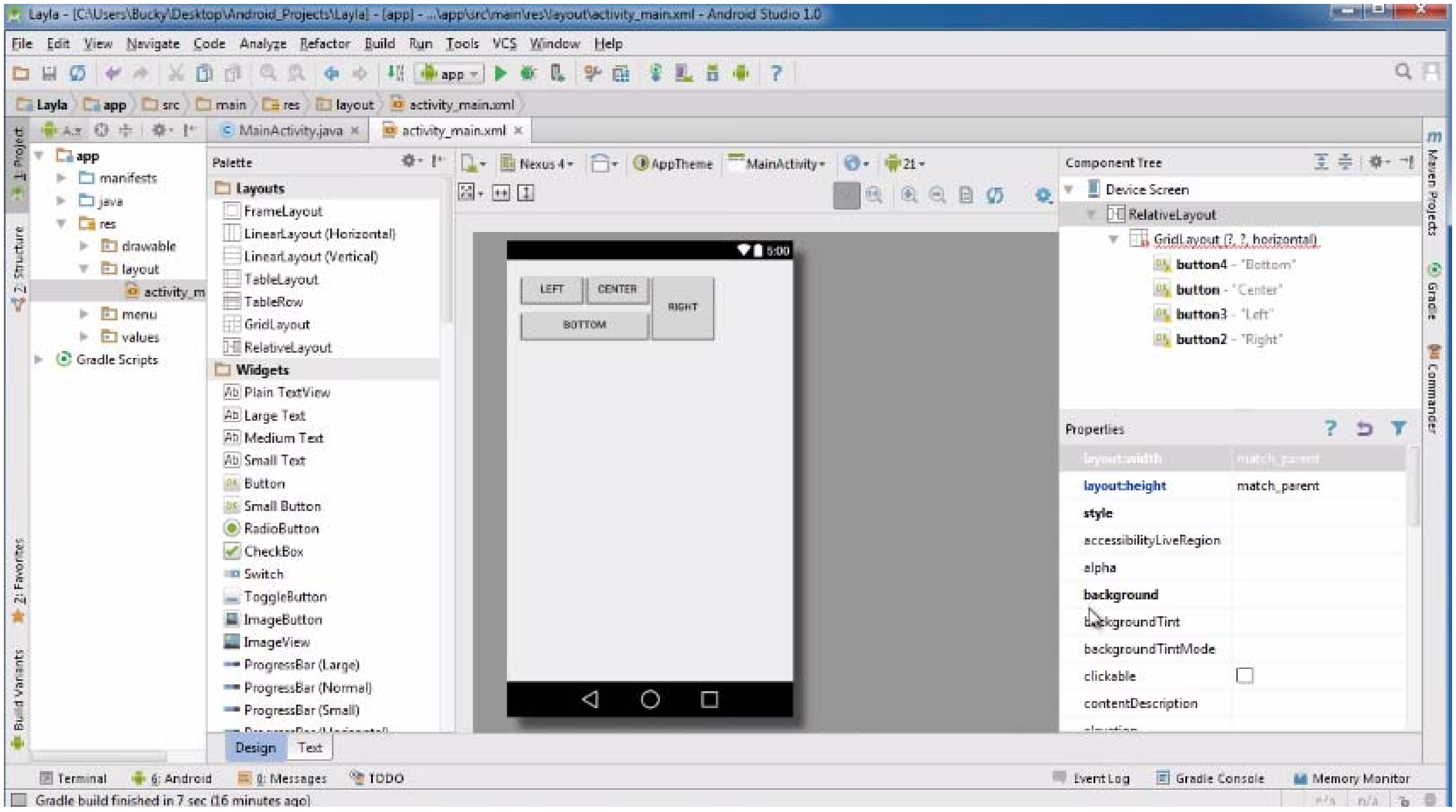
Tutorial 17: GridLayout

- 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