CS 403X Mobile and Ubiquitous Computing

Lecture 2: Android UI Design, First Android Program

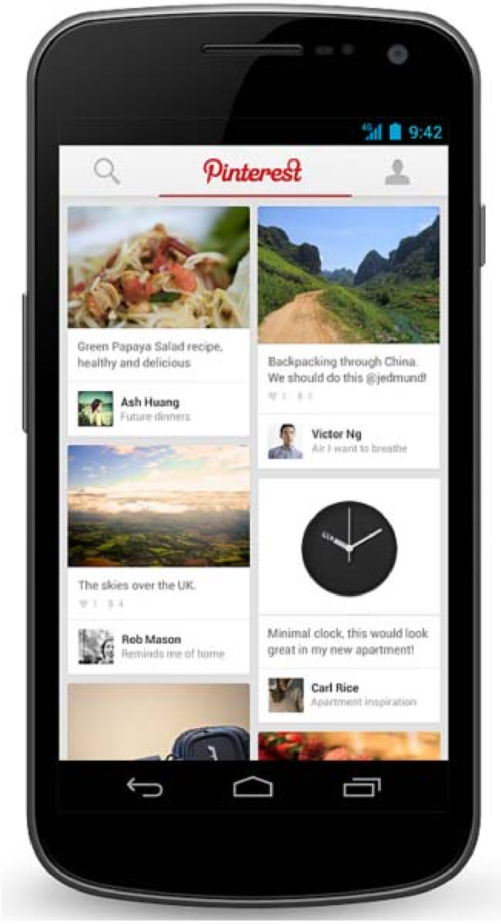
Emmanuel Agu

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”



Green Papaya Salad recipe, healthy and delicious

Ash Huang
Future dinners

Backpacking through China. We should do this @jedmund!

Victor Ng
Air I want to breathe

The skies over the UK.

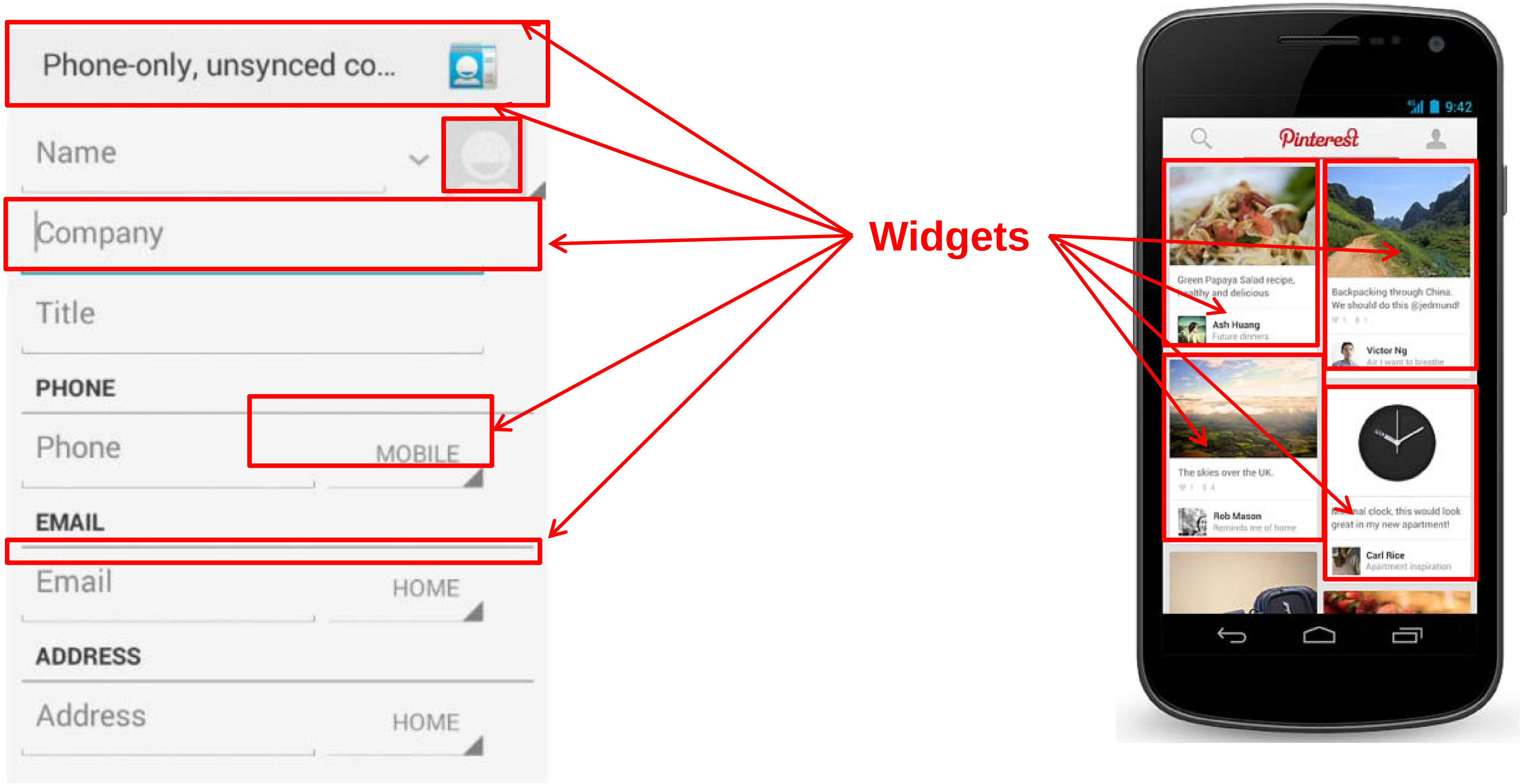
Rob Mason
Reminds me of home

Minimal clock, this would look great in my new apartment!

Carl Rice
Apartment inspiration

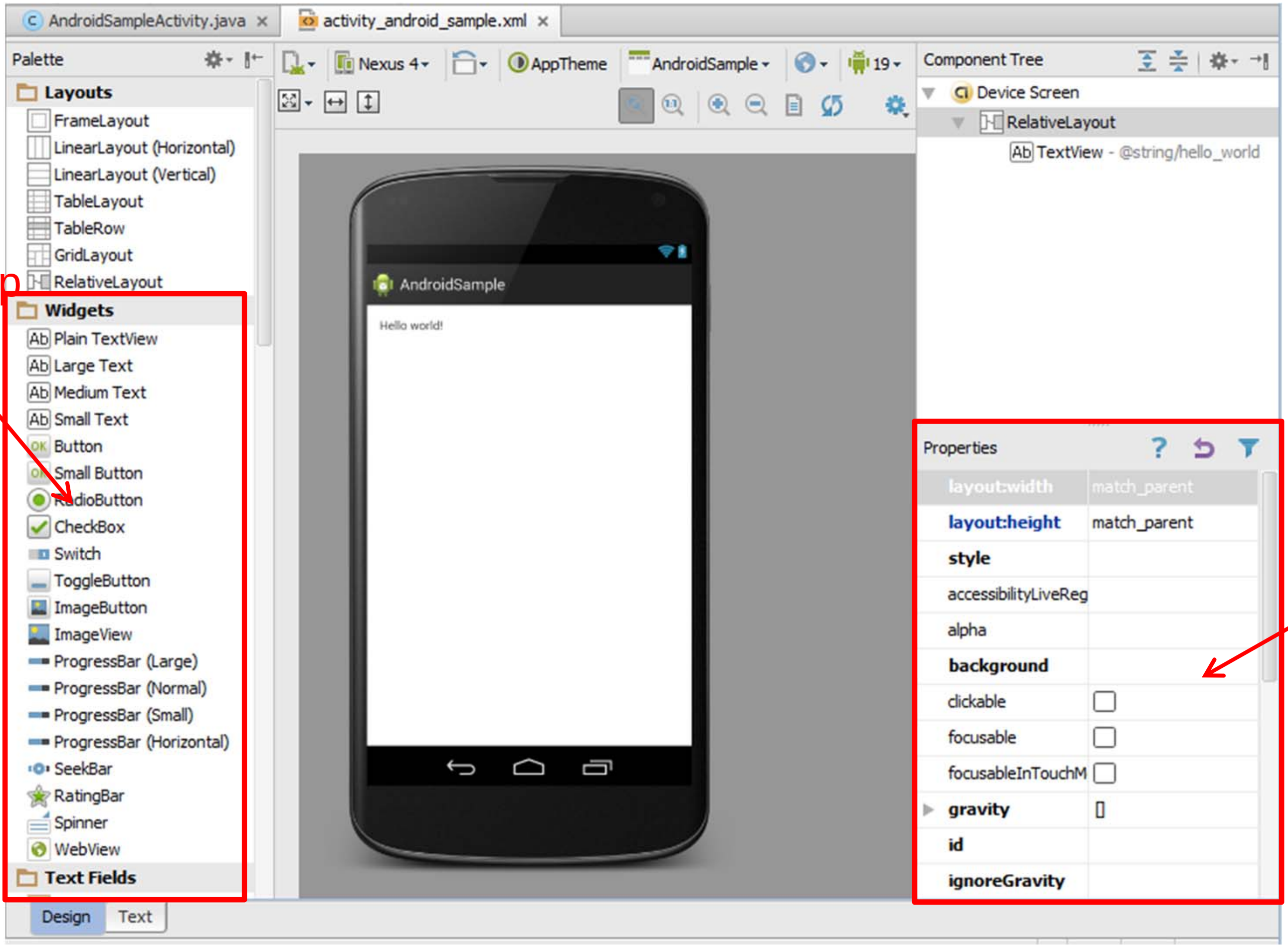
Widgets

- Widgets are visual building blocks of Android screens
- Need to specify widget attributes (dimensions, margins, padding, etc)
- ***Android UI design involves arranging widgets on a screen***



Option 1: Adding Widget in Design View

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



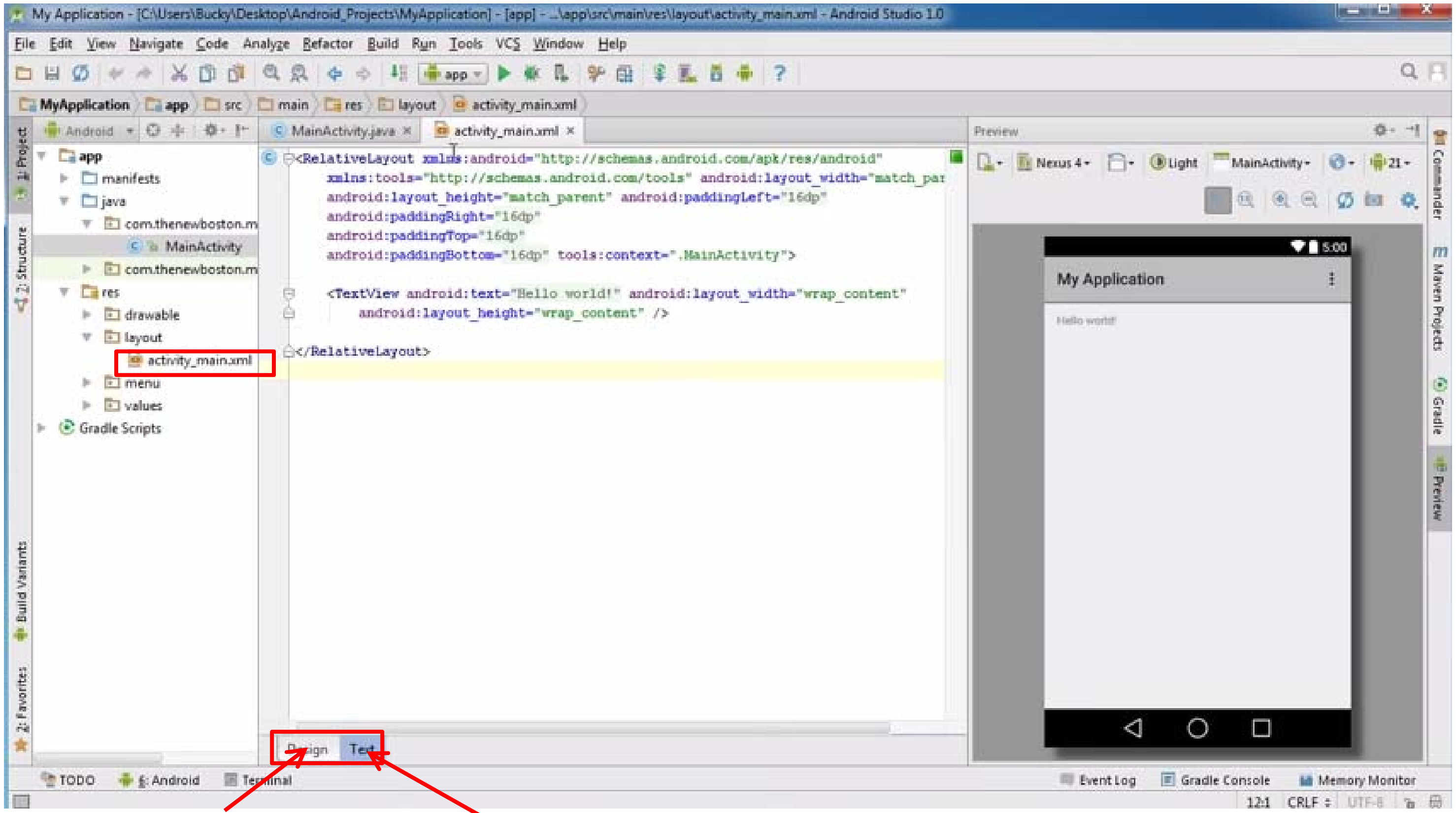
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
clickable	□
focusable	□
focusableInTouchM	□
gravity	[]
id	
ignoreGravity	

Drag and drop button or any other widget or view

Edit widget properties

Option 2: Edit XML Directly

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



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_par
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

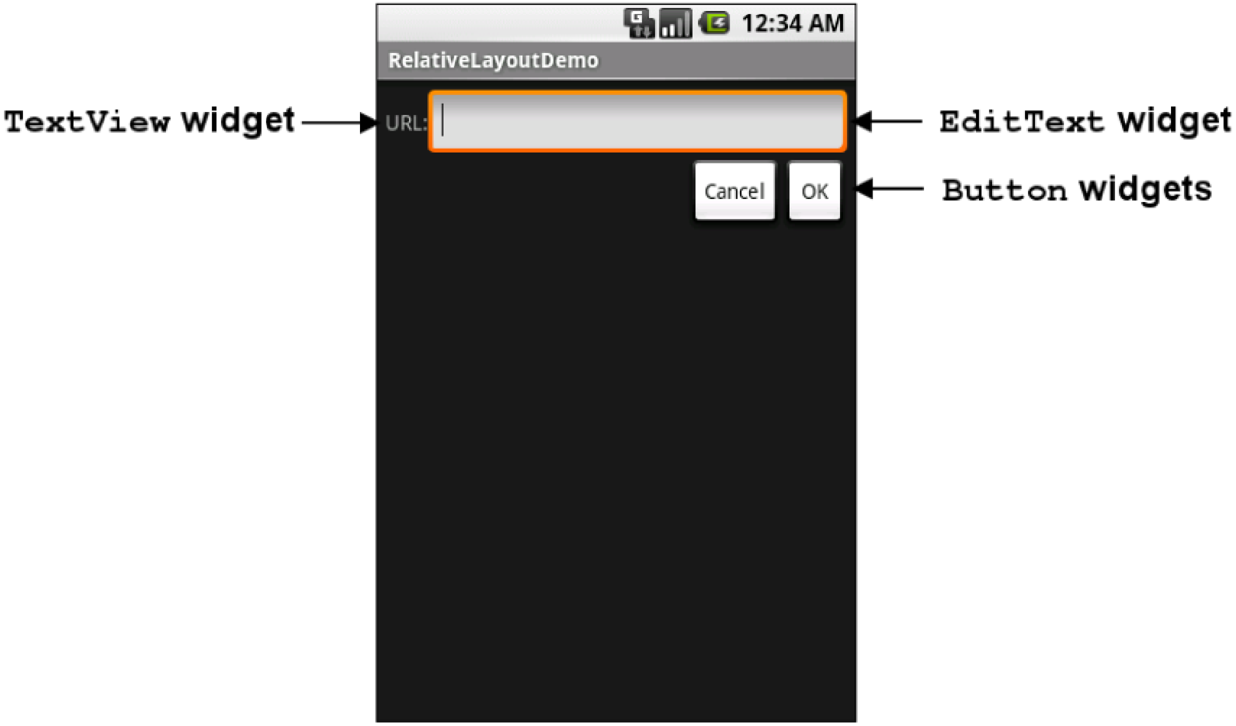
Drag and drop widget

Edit XML

Android Widgets

Example: Some Common Widgets

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



TextView widget

EditText widget

Button widgets

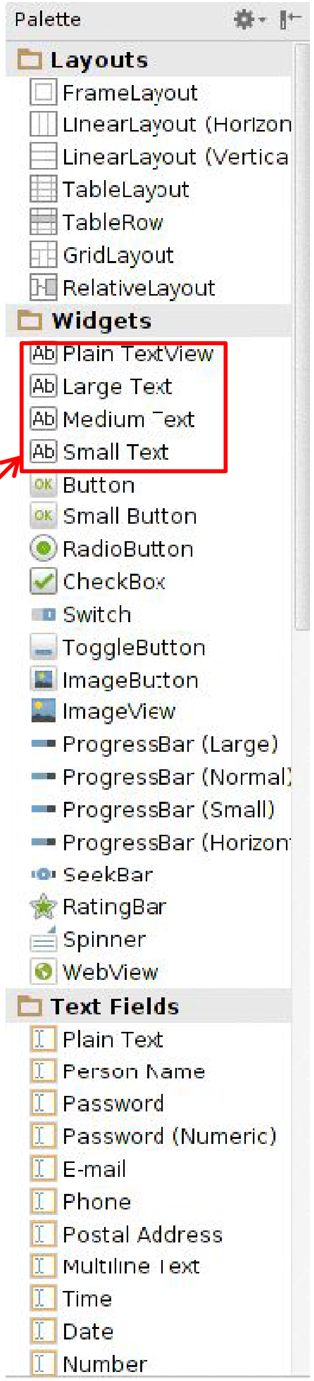
TextView

- Text in a rectangle
- Displays information, not for interaction

- 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

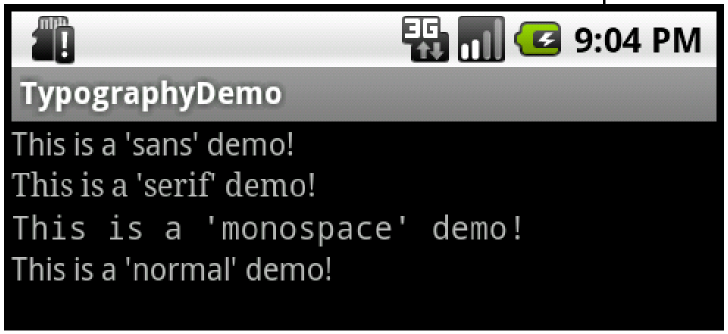
- See book demo project: **Basic/Label**



TextView

- **Declare TextView in XML (e.g. Activity_main.xml):**

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

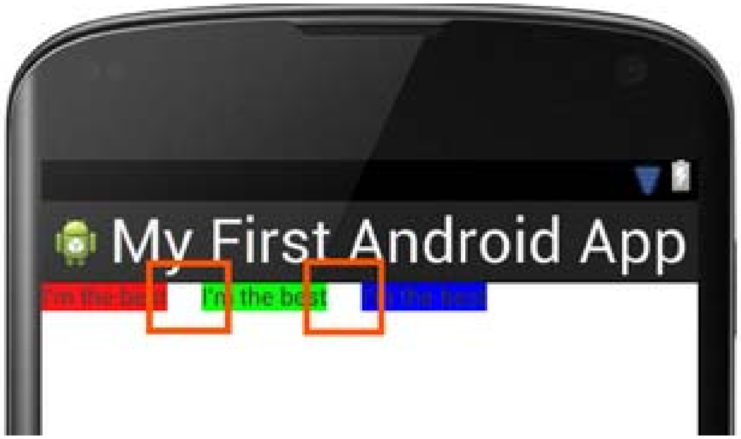


```
This is a 'sans' demo!
This is a 'serif' demo!
This is a 'monospace' demo!
This is a 'normal' demo!
```

- **match_content:** Make Textview as large as text
- **match_parent:** Make Textview as box it is declared in
- **Common attributes:**
  - **Typeface** (android:typeface e.g monospace), bold, italic, text size
  - **Text color:** (android:textColor) e.g. #FF0000 for red
  - width, height, padding, margins, visibility, background color
  - http://developer.android.com/reference/android/R.styleable.html#TextView
- **units for width / height:** px (pixels), dp or dip (density-independent pixels 160 dpi base), in (inches), mm (millimeters) (More later)
  - http://developer.android.com/guide/topics/resources/more-resources.html#Dimension

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"
/>
```

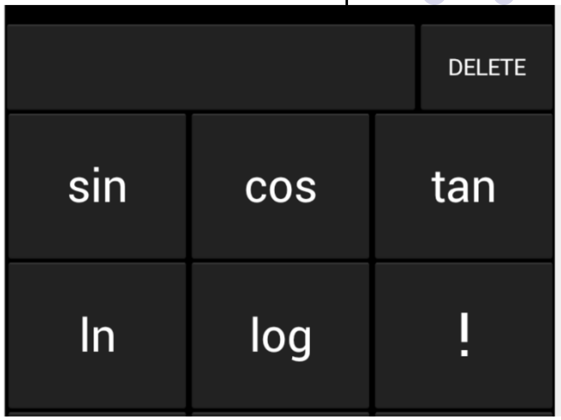


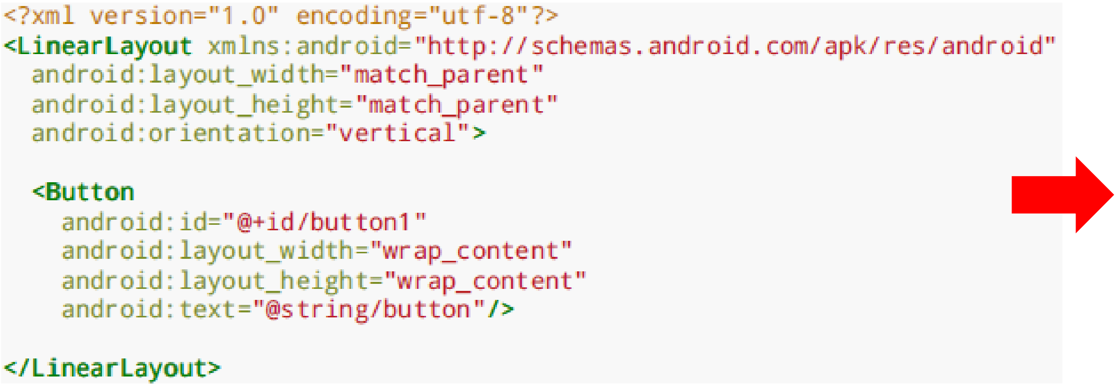
My First Android App

I'm the best I'm the best I'm the best

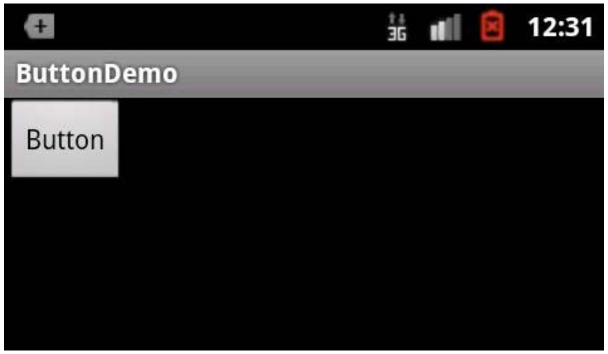
Button Widget

- 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>
```

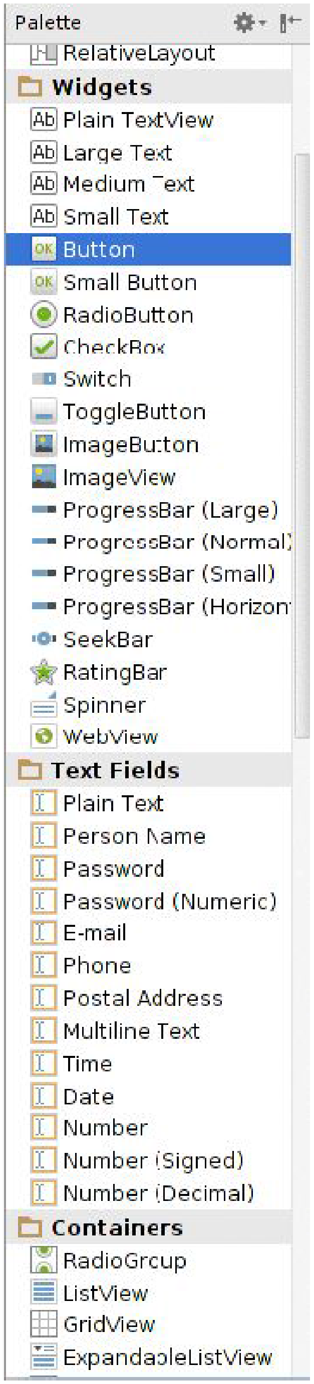


Button in Android Studio

- **Button** widget available in Android Studio graphical layout editor

- Can drag and drop button, edit attributes as with TextView

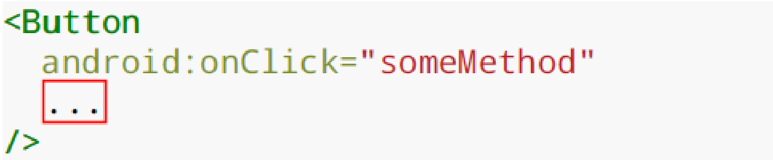
- See book demo project: Basic/Button



Responding to Button Clicks

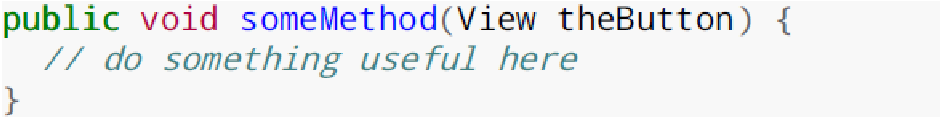
- May want Button press to trigger some action
- How?

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



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

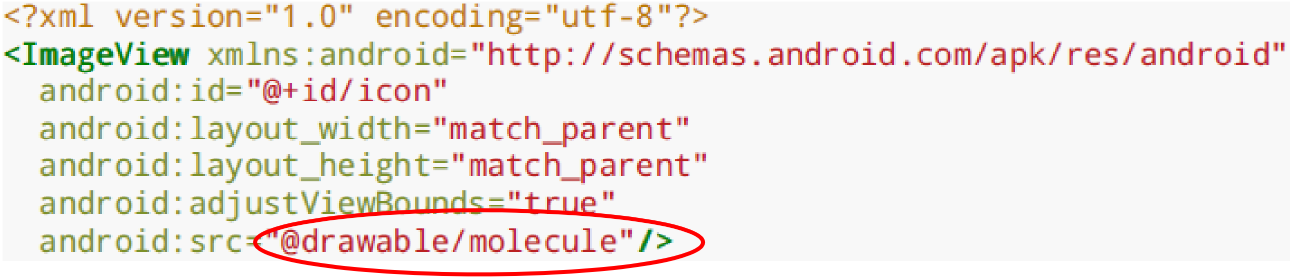
2. **In Java file (e.g. MainActivity.java) declare method/handler to take desired action**



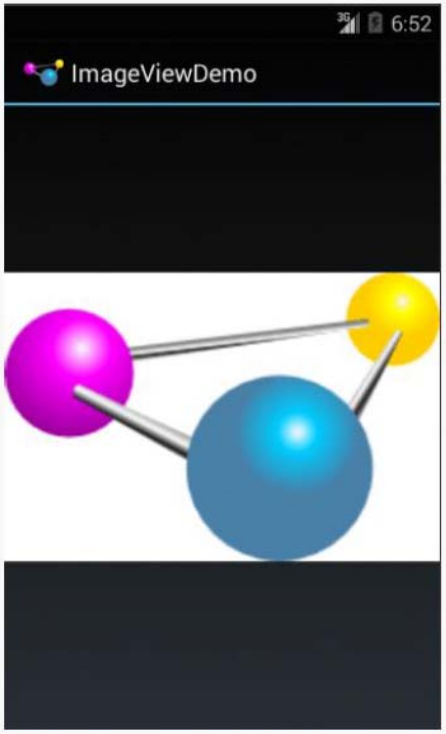
```
public void someMethod(View theButton) {
  // do something useful here
}
```

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 book demo project: Basic/ImageView

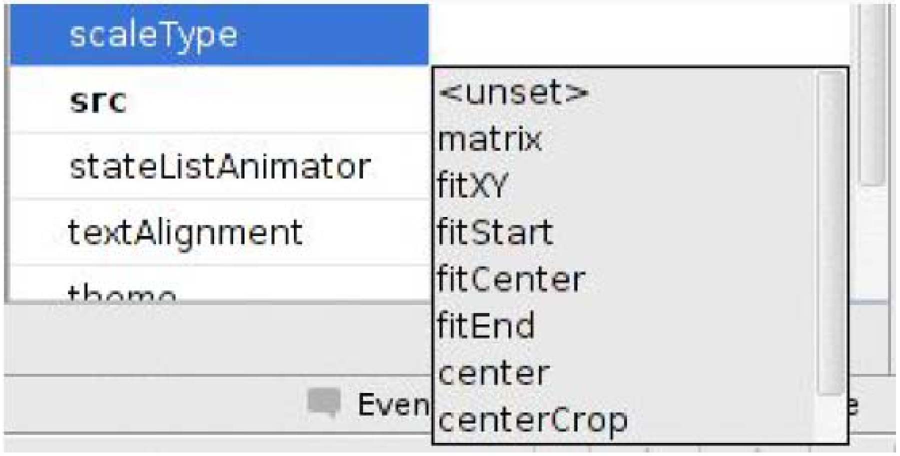


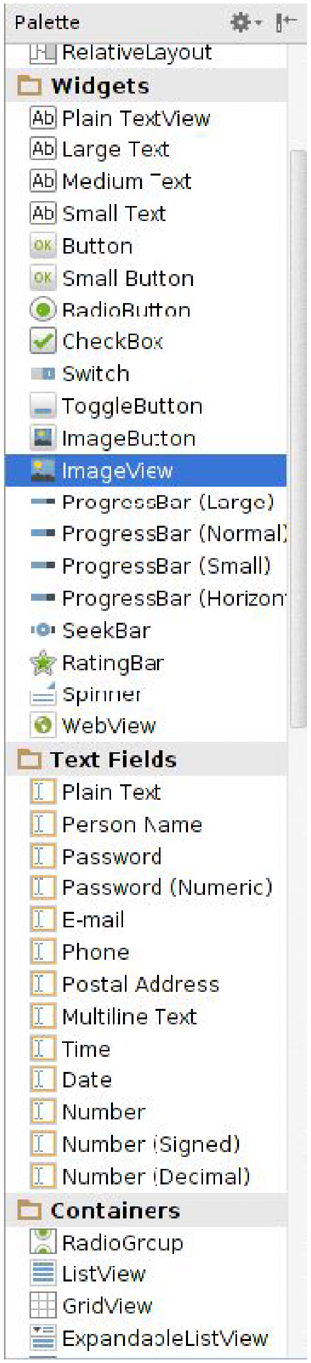
```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/icon"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:adjustViewBounds="true"
  android:src="@drawable/molecule"/>
```



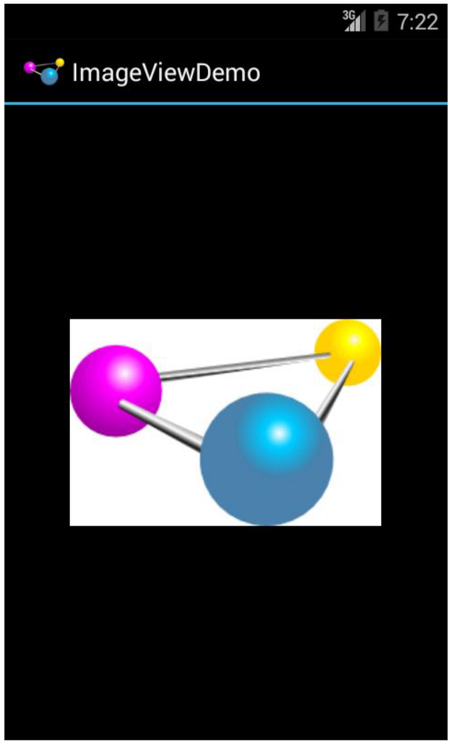
ImageView in Widgets Palette

- 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

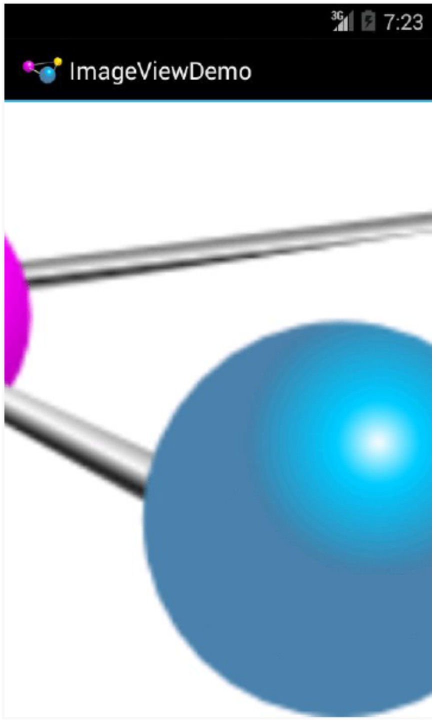




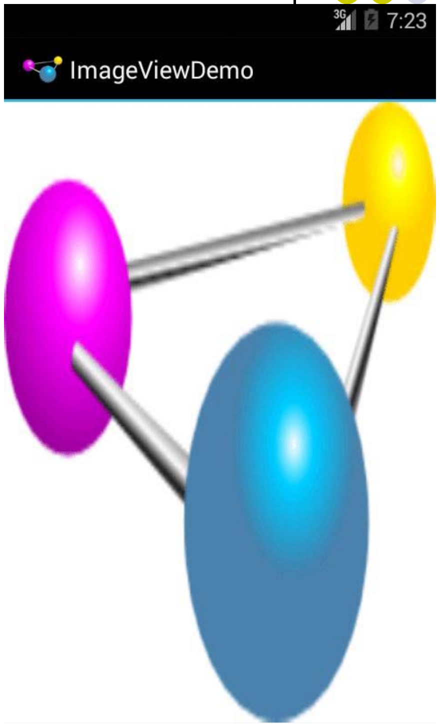
Options for Scaling Images (scaleType)



“center” centers image (no scaling)



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



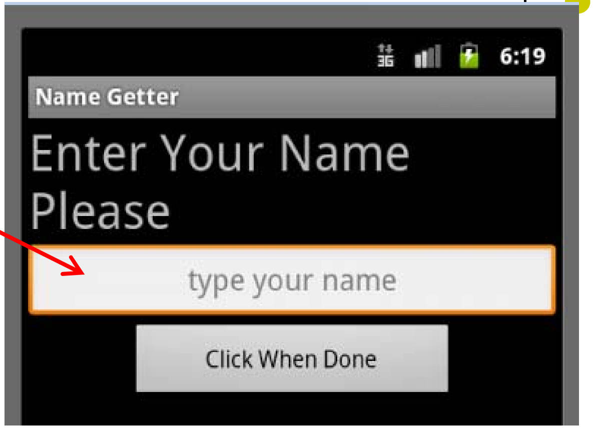
“fitXY” scales image to fit ImageView, ignoring aspect ratio

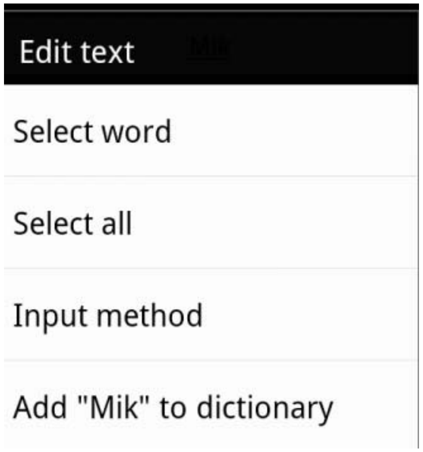
EditText Widget

- UI Component for user to enter information
- long press brings up context menu
- Example XML declaration:

```
<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" />
```

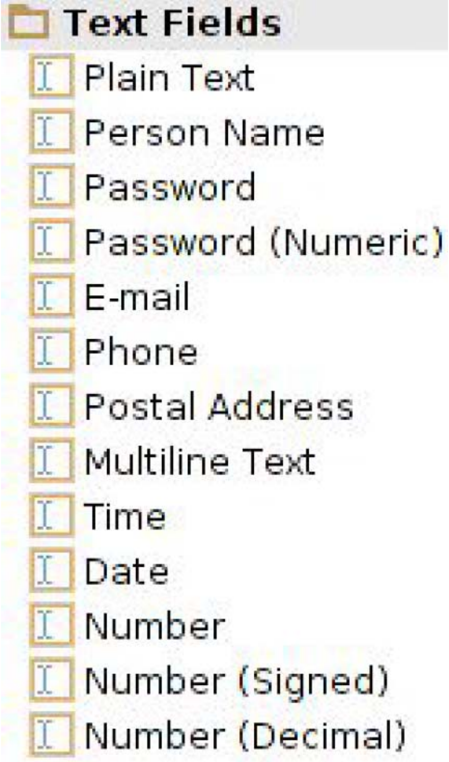
- **android:inputType:** defines input type (number, date, password, or email address)





EditText Widget in Android Studio Palette

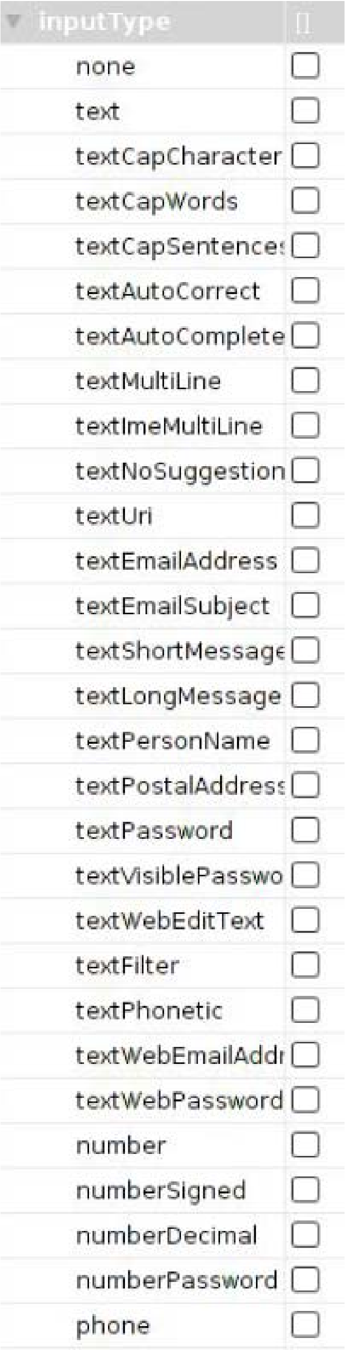
- A whole section of Android Studio palette dedicated to EditText widgets (or text fields)



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



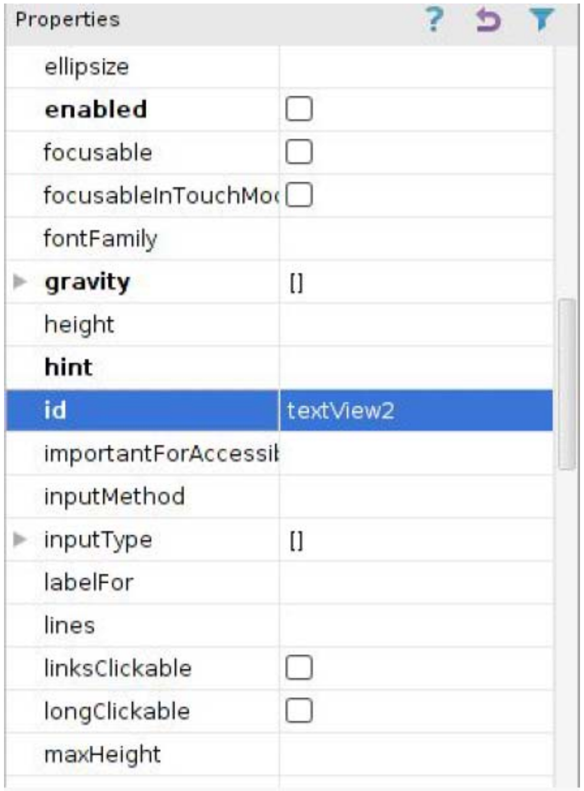
inputType []

- none □
- text □
- textCapCharacter □
- textCapSentence □
- textAutoCorrect □
- textAutoComplete □
- textMultiLine □
- textImeMultiLine □
- textNoSuggestion □
- textUri □
- textEmailAddress □
- textEmailSubject □
- textShortMessage □
- textLongMessage □
- textPersonName □
- textPostalAddres □
- textPassword □
- textVisiblePasswo □
- textWebEditText □
- textFilter □
- textPhonetic □
- textWebEmailAddr □
- textWebPassword □
- number □
- numberSigned □
- numberDecimal □
- numberPassword □
- phone □

EditText inputType menu

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

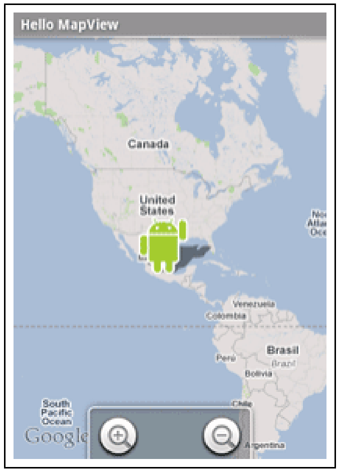


Properties

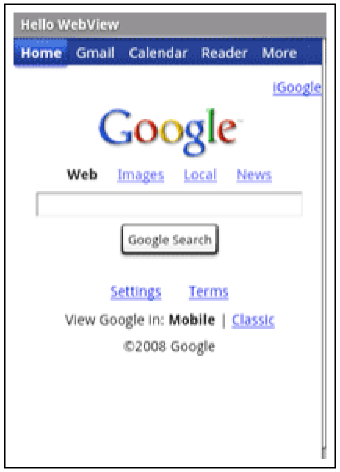
ellipsize	
enabled	□
focusable	□
focusableInTouchMo	□
fontFamily	
▸ **gravity**	[]
height	
hint	
id	textView2
importantForAccessil	
inputMethod	
▸ inputType	[]
labelFor	
lines	
linksClickable	□
longClickable	□
maxHeight	

Other Available Widgets

MapView



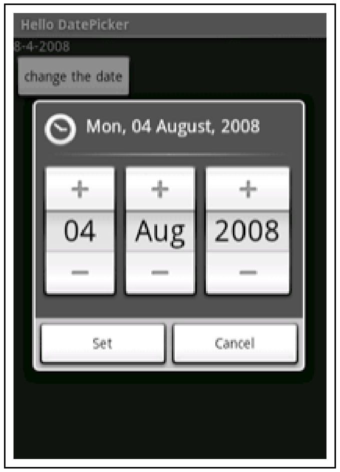
WebView



View Google in: **Mobile** | Classic

©2008 Google

DatePicker

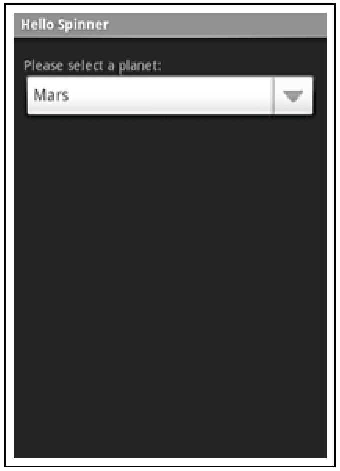


8-4-2008

Mon, 04 August, 2008

04 Aug 2008

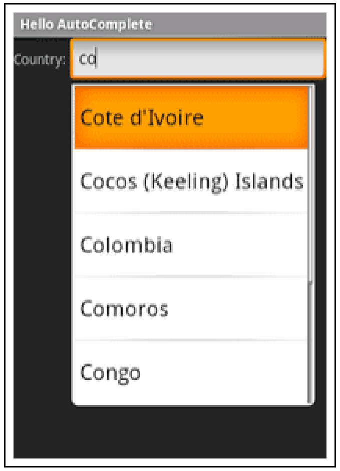
Spinner



Please select a planet:

Mars

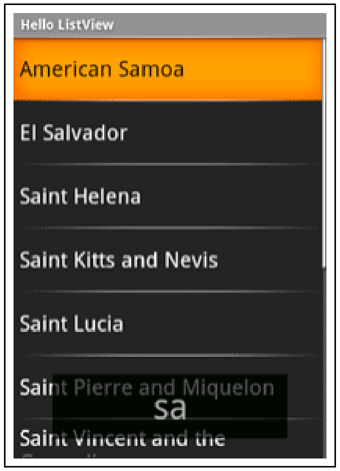
AutoComplete



Country: co

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

ListView



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

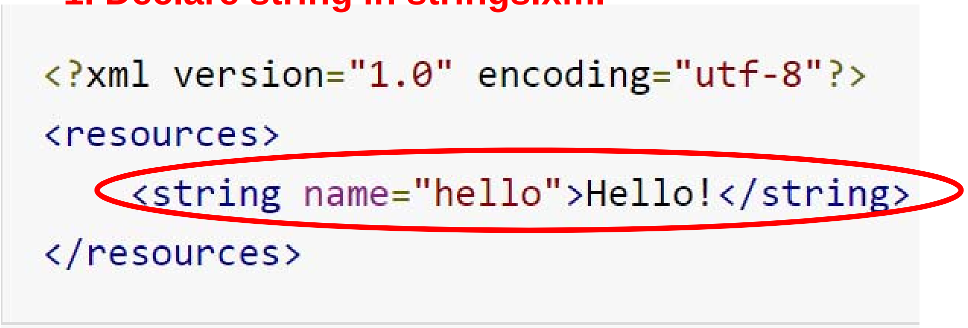
http://developer.android.com/resources/tutorials/views/index.html

Strings

Declaring Strings in Strings.xml

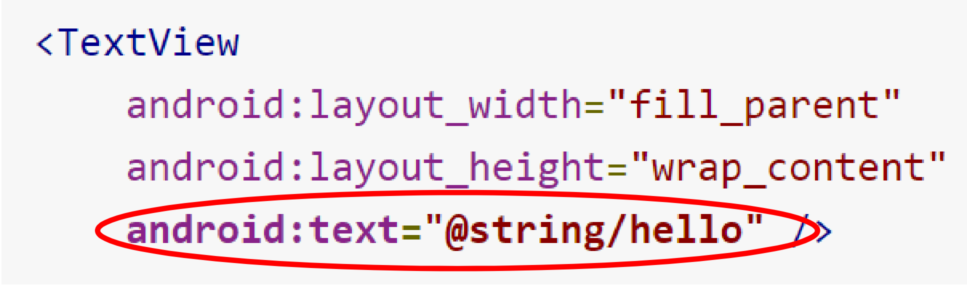
- Declare all strings in a single file
- Strings declared in strings.xml can be referenced by all other XML files (activity_my.xml, AndroidManifest.xml)
- **Example:**

1. Declare string in strings.xml



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

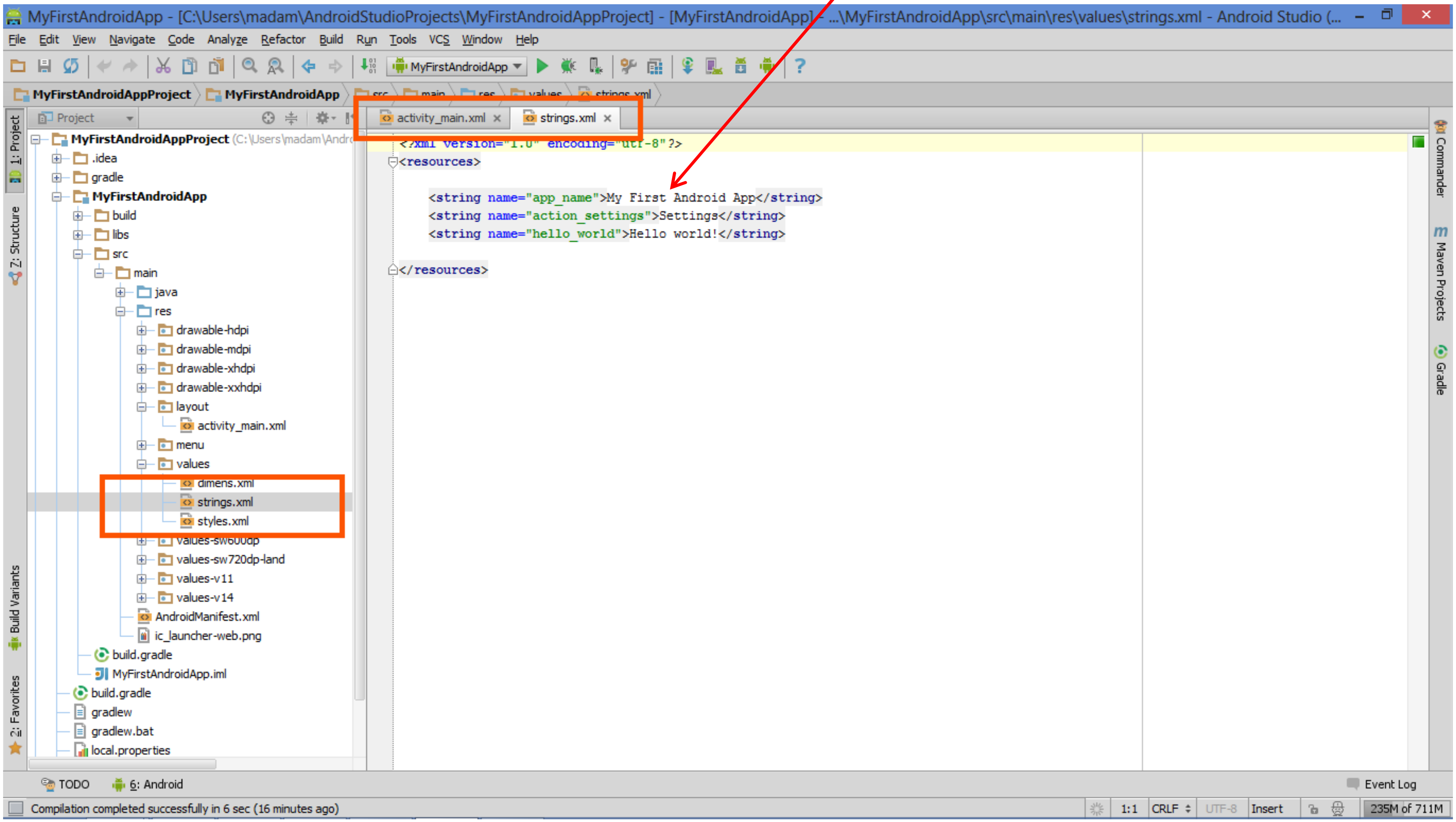
2. Use string in Activity_main.xml



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

Where is strings.xml in Android Studio?

Editting any string here changes it wherever it is displayed



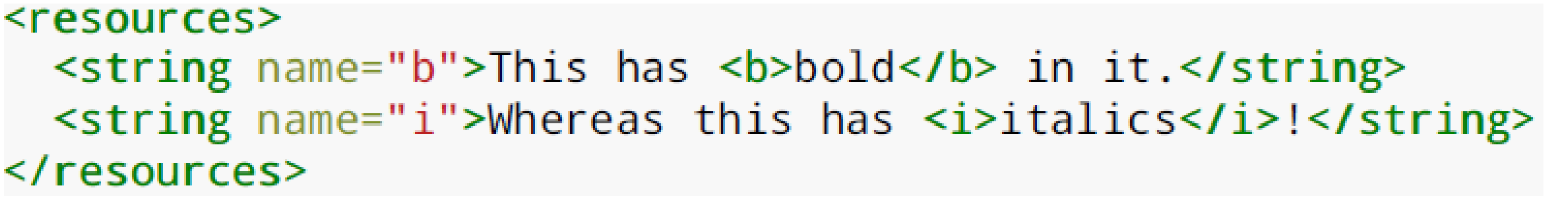
MyFirstAndroidApp - [C:\Users\madam\AndroidStudioProjects\MyFirstAndroidAppProject] - [MyFirstAndroidApp] - ...\MyFirstAndroidApp\src\main\res\values\strings.xml - Android Studio (...

```
<?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>
```

Compilation completed successfully in 6 sec (16 minutes ago)

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

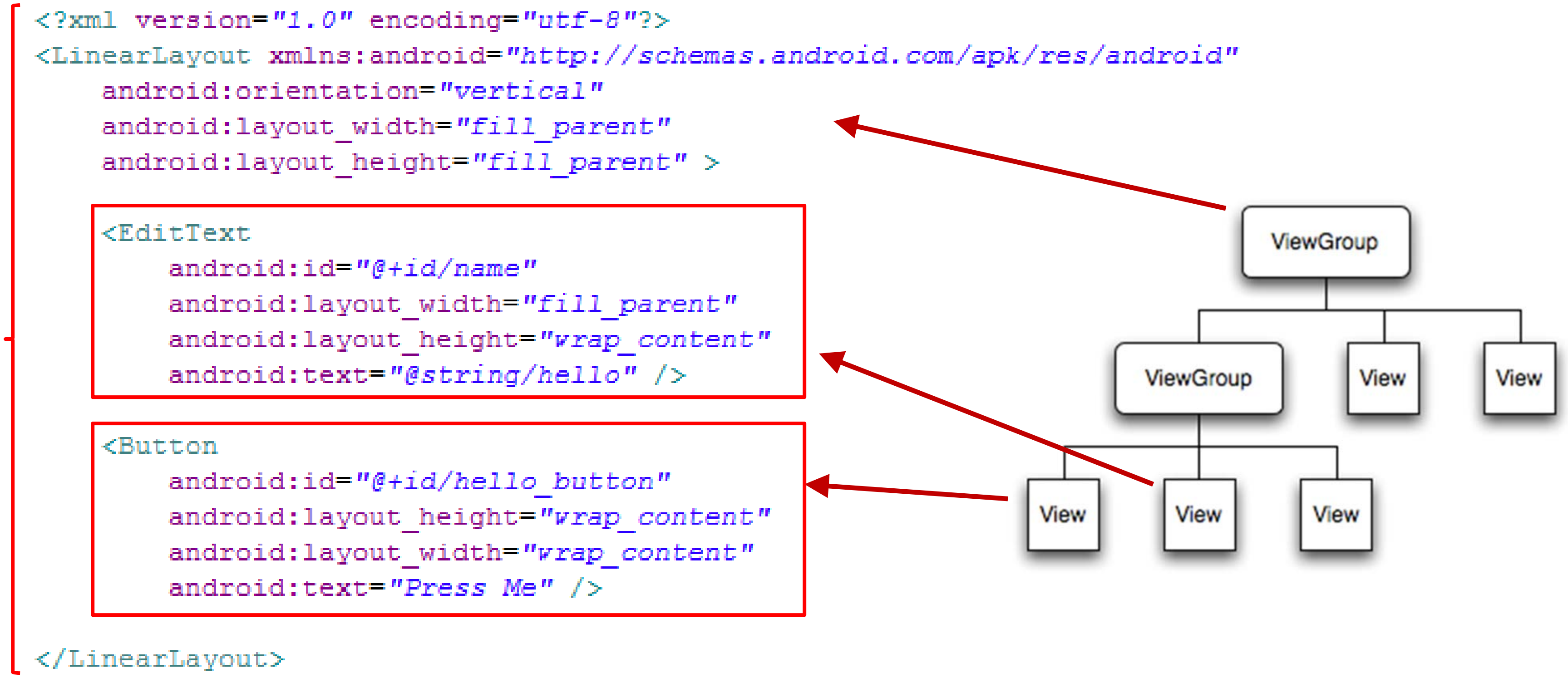


```
<resources>
  <string name="b">This has <b>bold</b> in it.</string>
  <string name="i">Whereas this has <i>italics</i>!</string>
</resources>
```

Android Layouts in XML

Views and ViewGroups

- Widgets are declared as views in Android
- ViewGroup (e.g. a layout) contains multiple Views
- **Hierarchical arrangement:** Widgets are children of parent viewgroup, etc



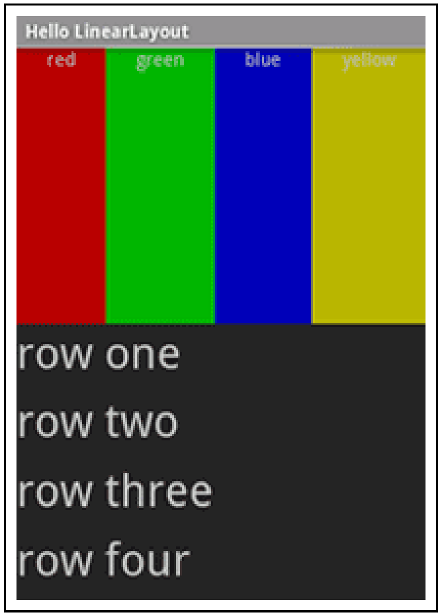
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <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

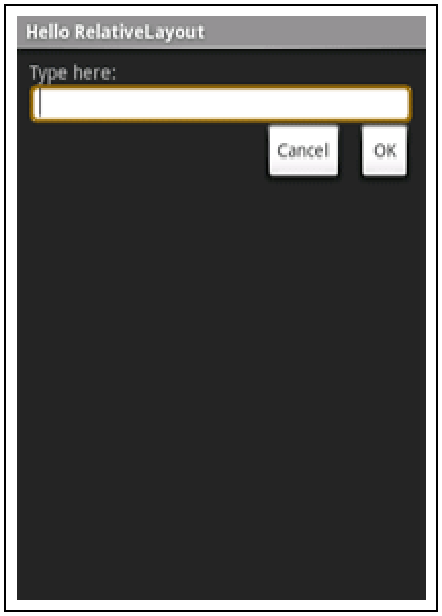
Android UI using XML Layouts

- In the XML file, we have to choose a layout (viewgroup) to use
- Examples:

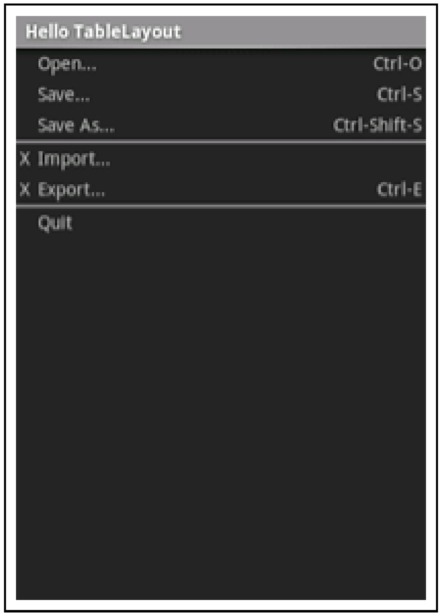
LinearLayout



RelativeLayout



TableLayout



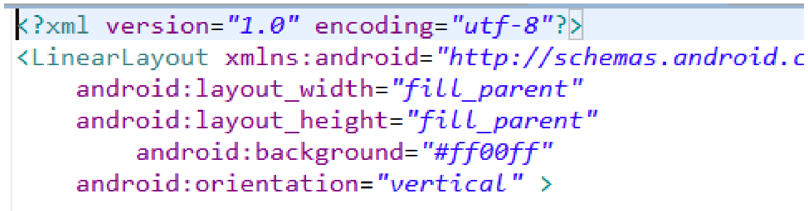
http://developer.android.com/resources/tutorials/views/index.html

Layouts

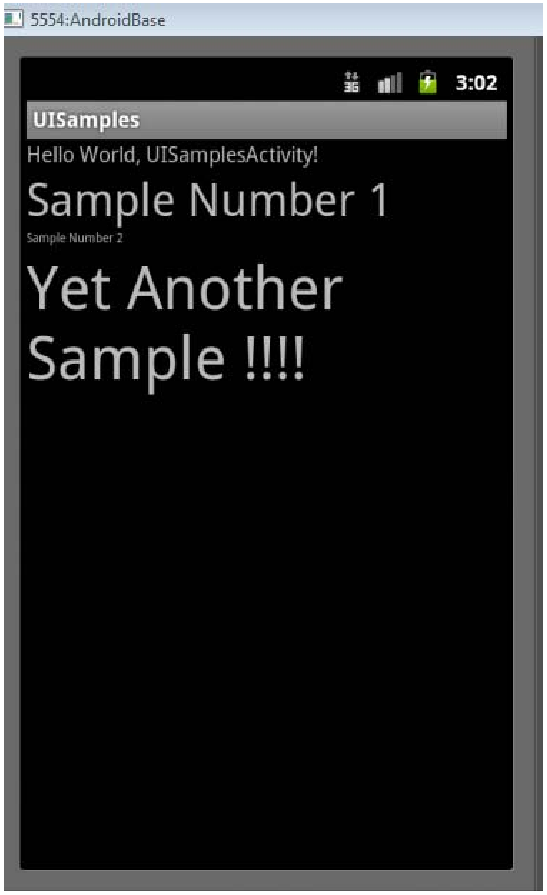
- Layouts are stored in **res/layout**
- Some Android Layouts:
  - FrameLayout,
  - LinearLayout,
  - TableLayout,
  - GridLayout,
  - RelativeLayout,
  - ListView,
  - GridView,
  - ScrollView,
  - DrawerLayout,
  - ViewPager
- More on layouts next

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" >
```



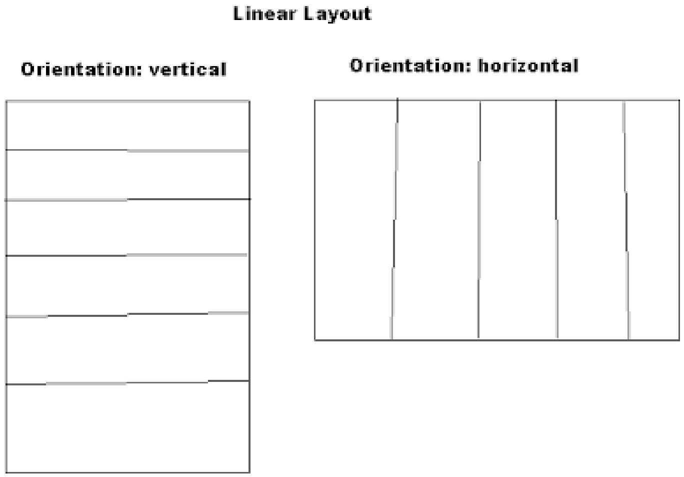
Hello World, UISamplesActivity!

Sample Number 1

Sample Number 2

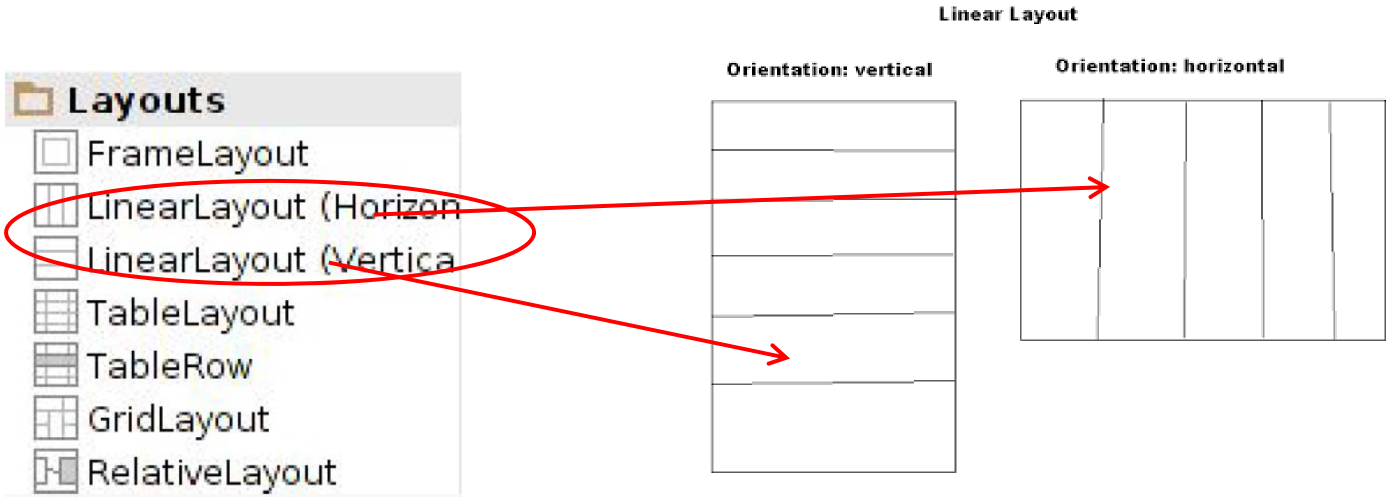
Yet Another Sample !!!!

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

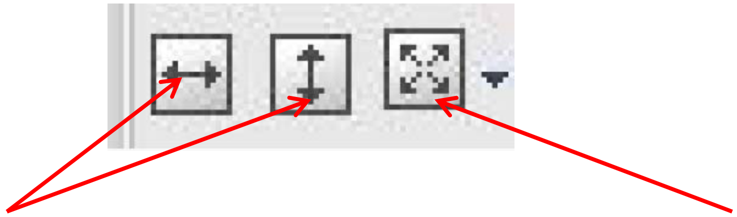


LinearLayout in Android Studio

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



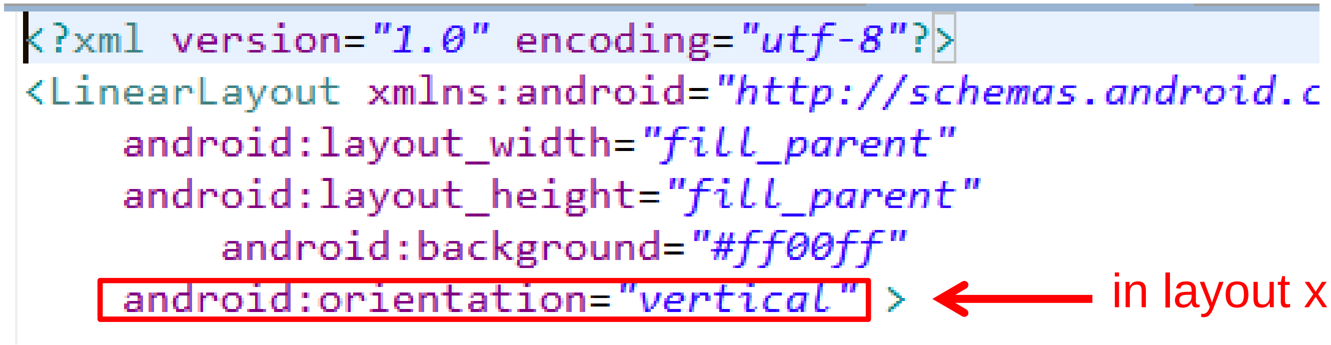
- After selecting LinearLayout, toolbars buttons to set parameters



Toggle width, height between match_parent and wrap_content

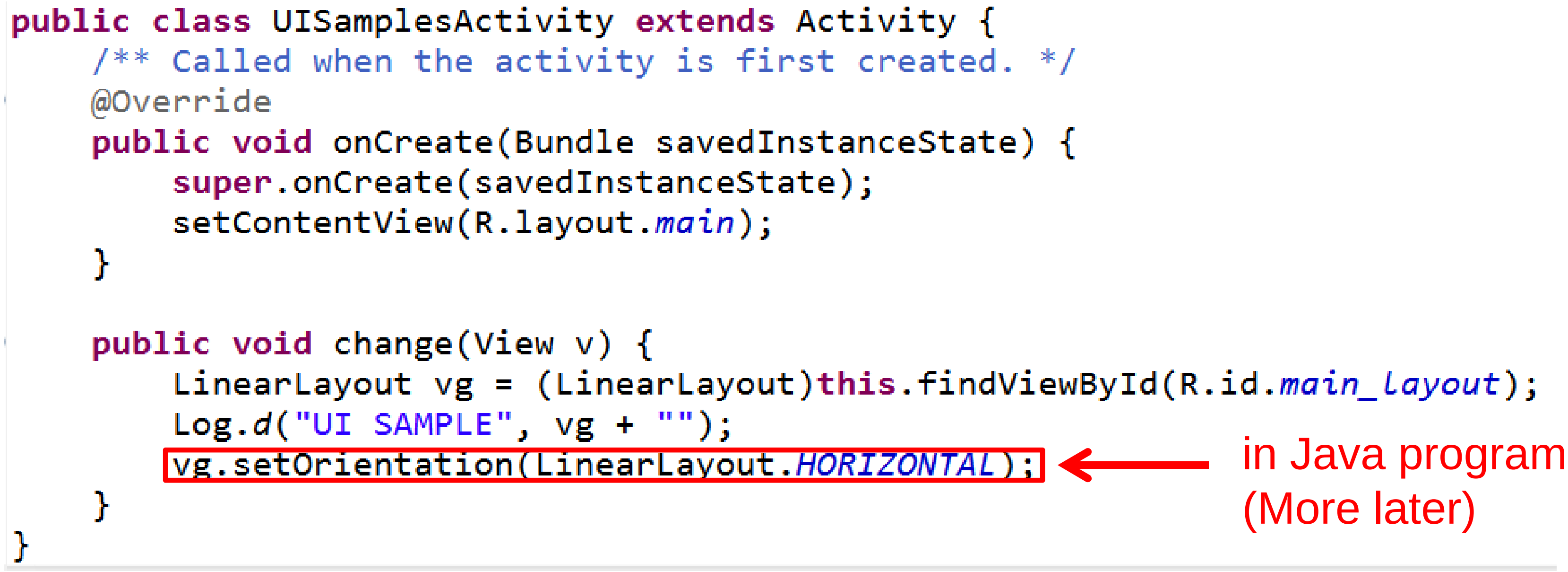
Change gravity of LinearLayout

Setting Layout &Widget 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" >
```

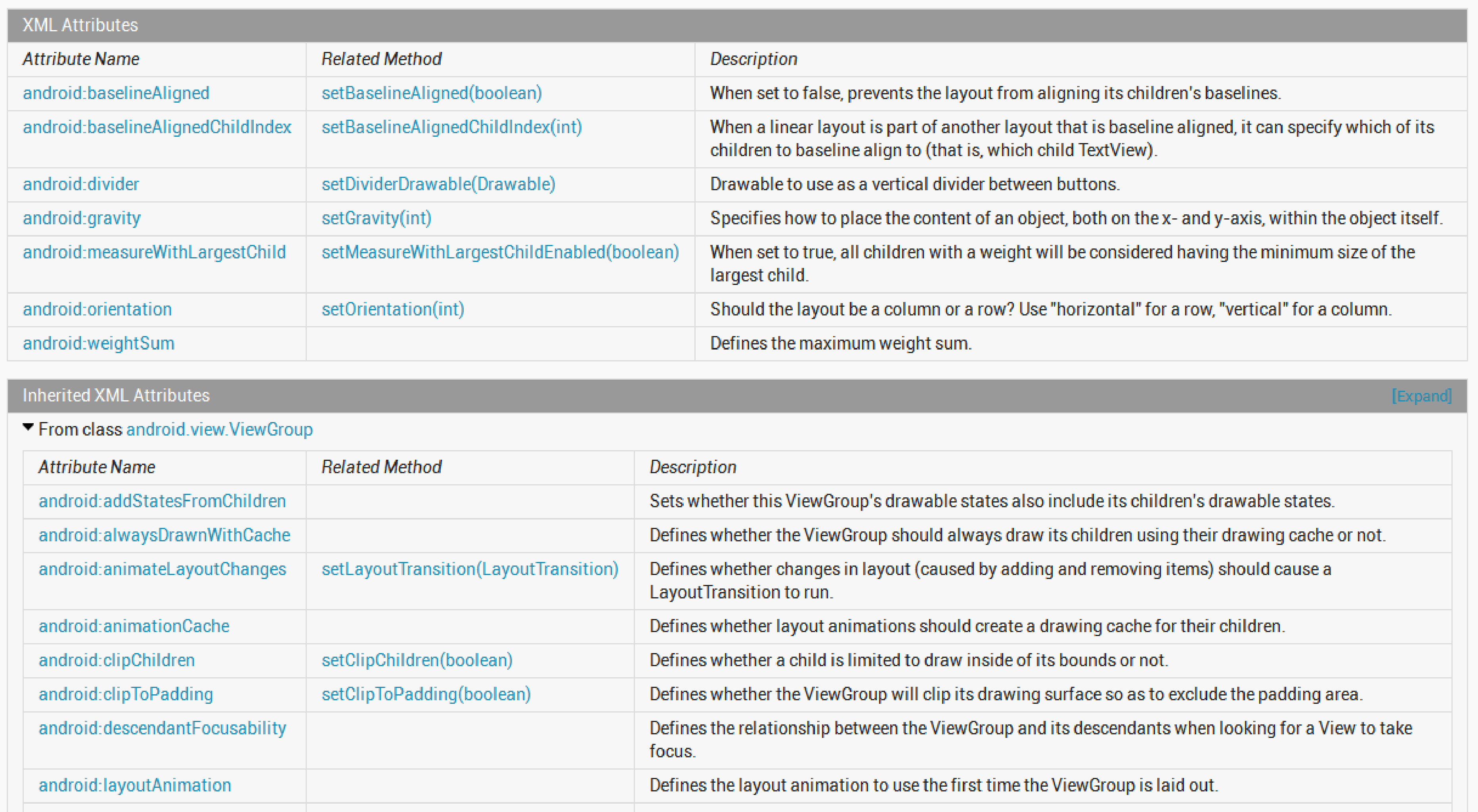
in layout xml file



```
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)

Some LinearLayout Attributes



XML Attributes

Attribute Name	*Related Method*	*Description*
android:baselineAligned	setBaselineAligned(boolean)	When set to false, prevents the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	setBaselineAlignedChildIndex(int)	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider	setDividerDrawable(Drawable)	Drawable to use as a vertical divider between buttons.
android:gravity	setGravity(int)	Specifies how to place the content of an object, both on the x- and y-axis, within the object itself.
android:measureWithLargestChild	setMeasureWithLargestChildEnabled(boolean)	When set to true, all children with a weight will be considered having the minimum size of the largest child.
android:orientation	setOrientation(int)	Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum		Defines the maximum weight sum.

Inherited XML Attributes [Expand]

▼ From class android.view.ViewGroup

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

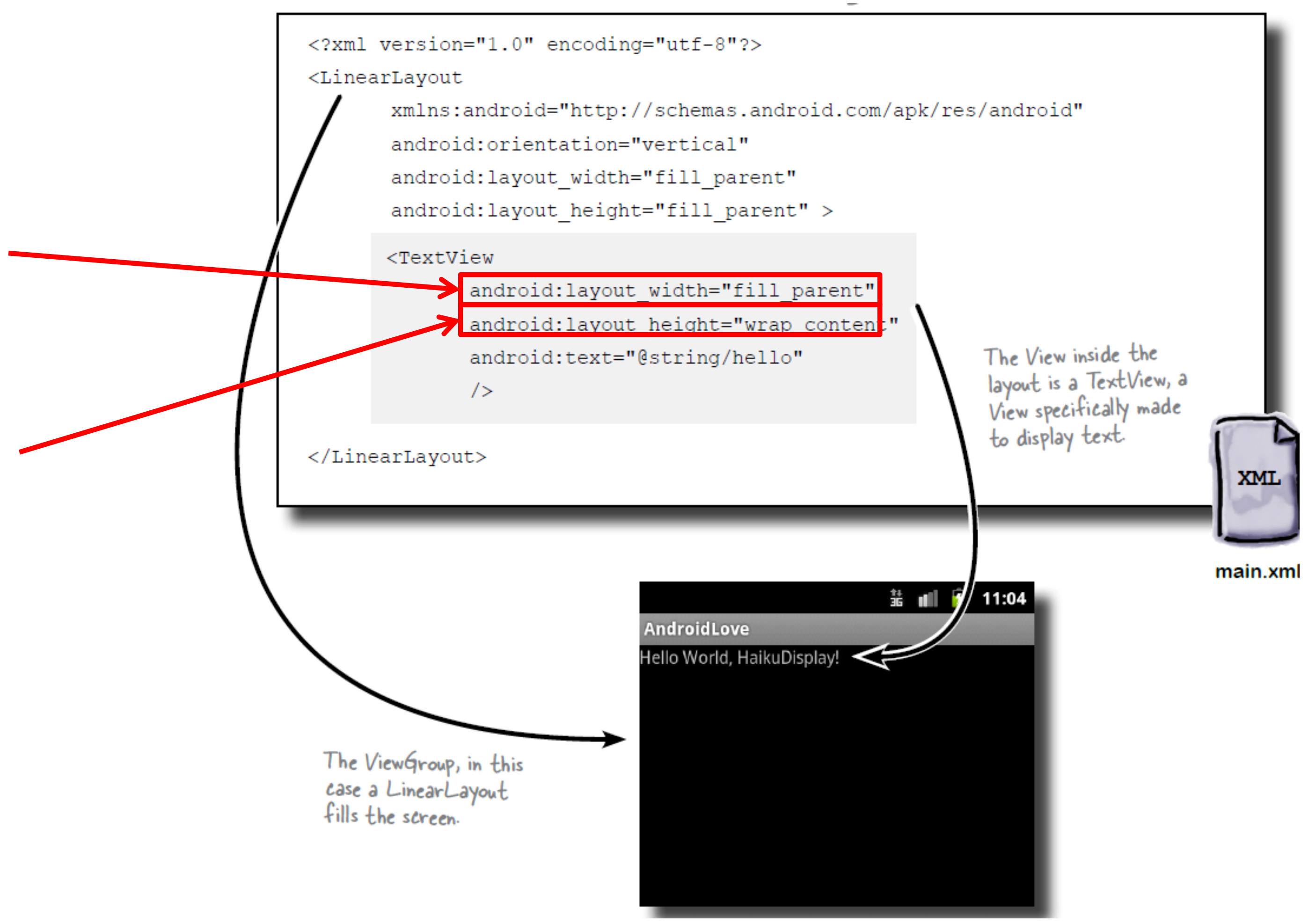
Can find complete list of attributes, possible values on Android Developer website

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**

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.

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

LinearLayout - Horizontal Orientation

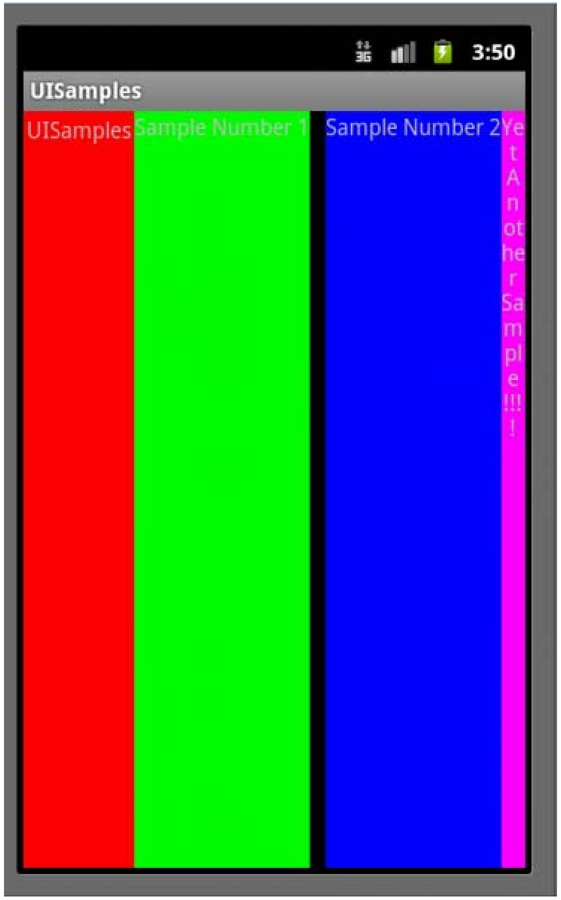
- Set
  - Padding

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

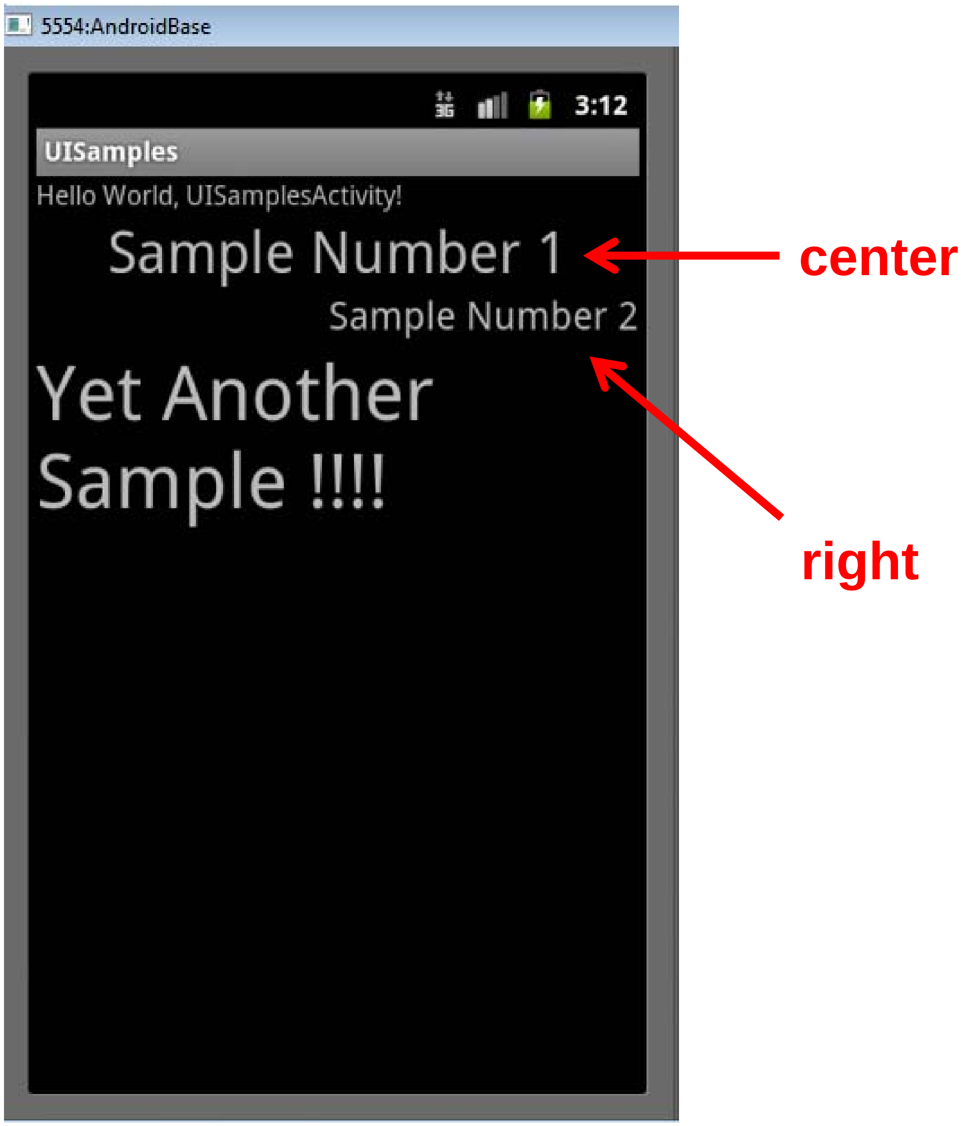
  - background color

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

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



Gravity Attribute



5554:AndroidBase

3:12

UISamples

Hello World, UISamplesActivity!

Sample Number 1

Sample Number 2

Yet Another Sample !!!!

center

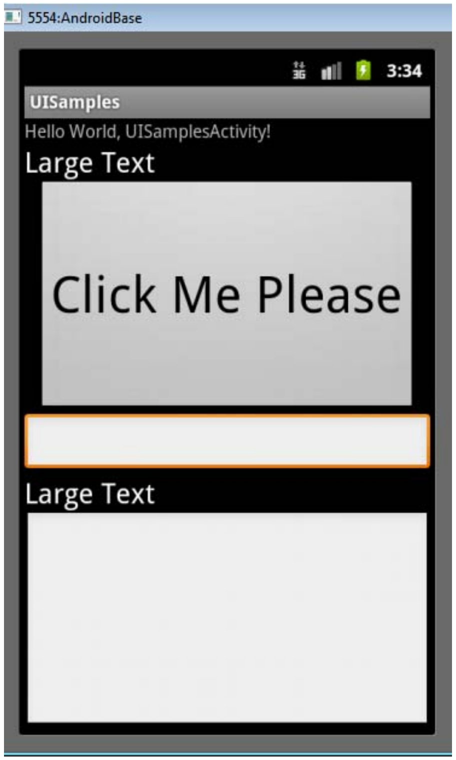
right

- 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



5554:AndroidBase

3:34

UISamples

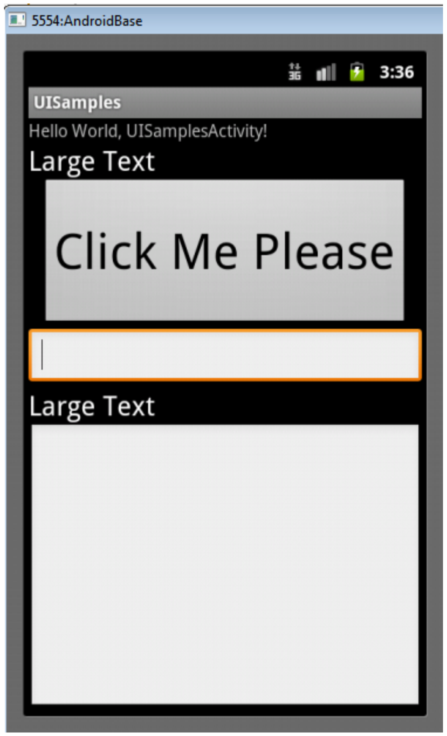
Hello World, UISamplesActivity!

Large Text

Click Me Please

Large Text

button and bottom edit text weight of 2



5554:AndroidBase

3:36

UISamples

Hello World, UISamplesActivity!

Large Text

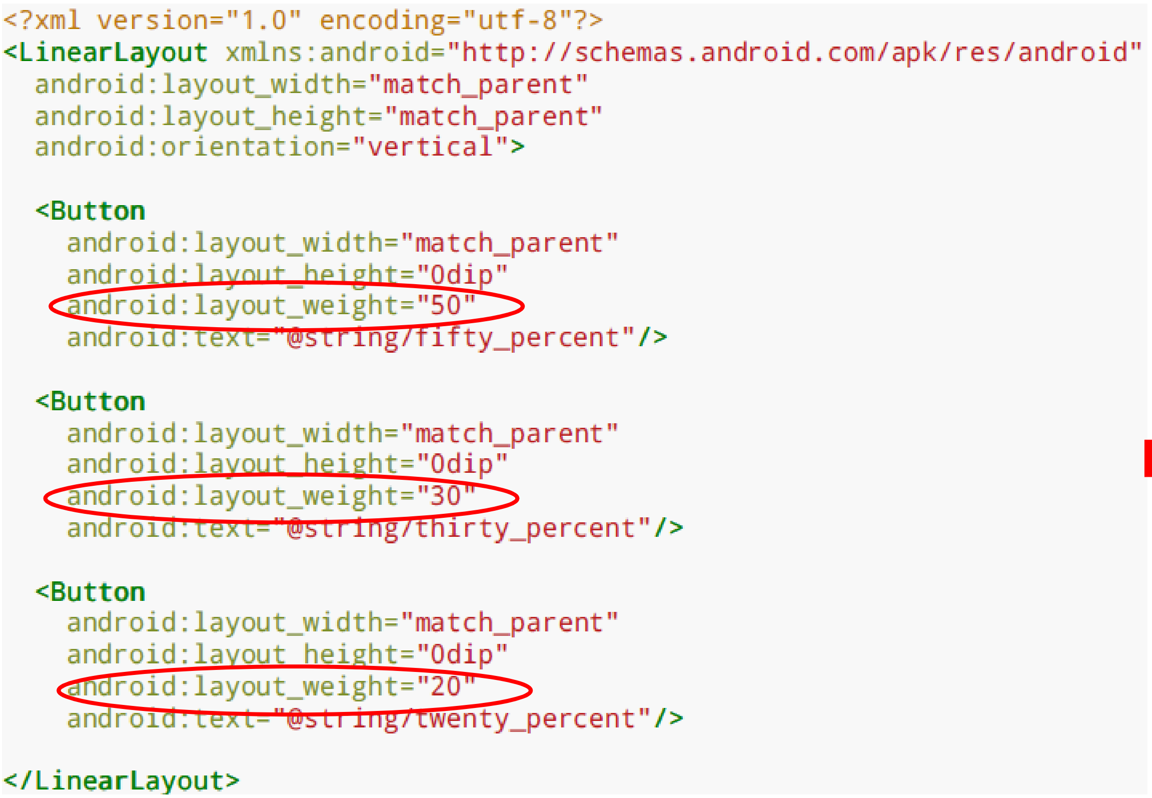
Click Me Please

Large Text

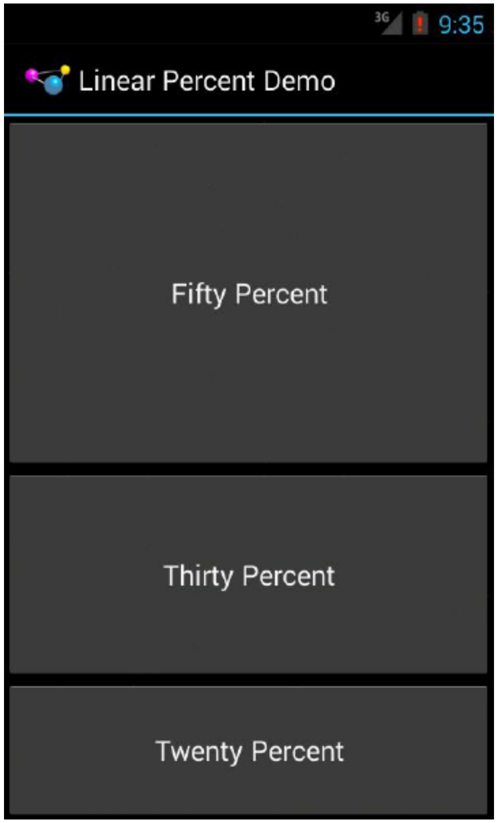
button weight 1 and
bottom edit text weight of 2

Linear Layout

- Alternate way to control widget size
  - width, height = 0 then
  - weight = percent of height/width you want element to cover



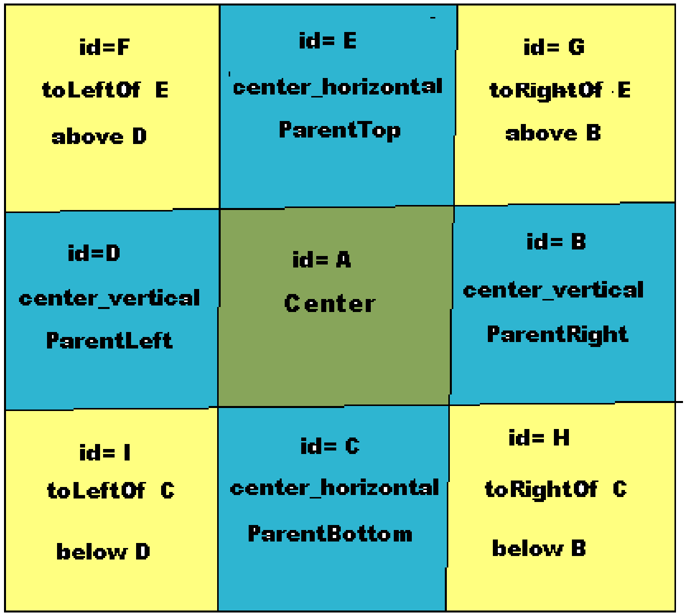
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <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>
```



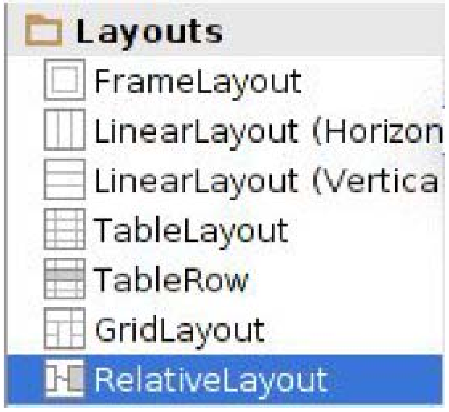
RelativeLayout

- 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



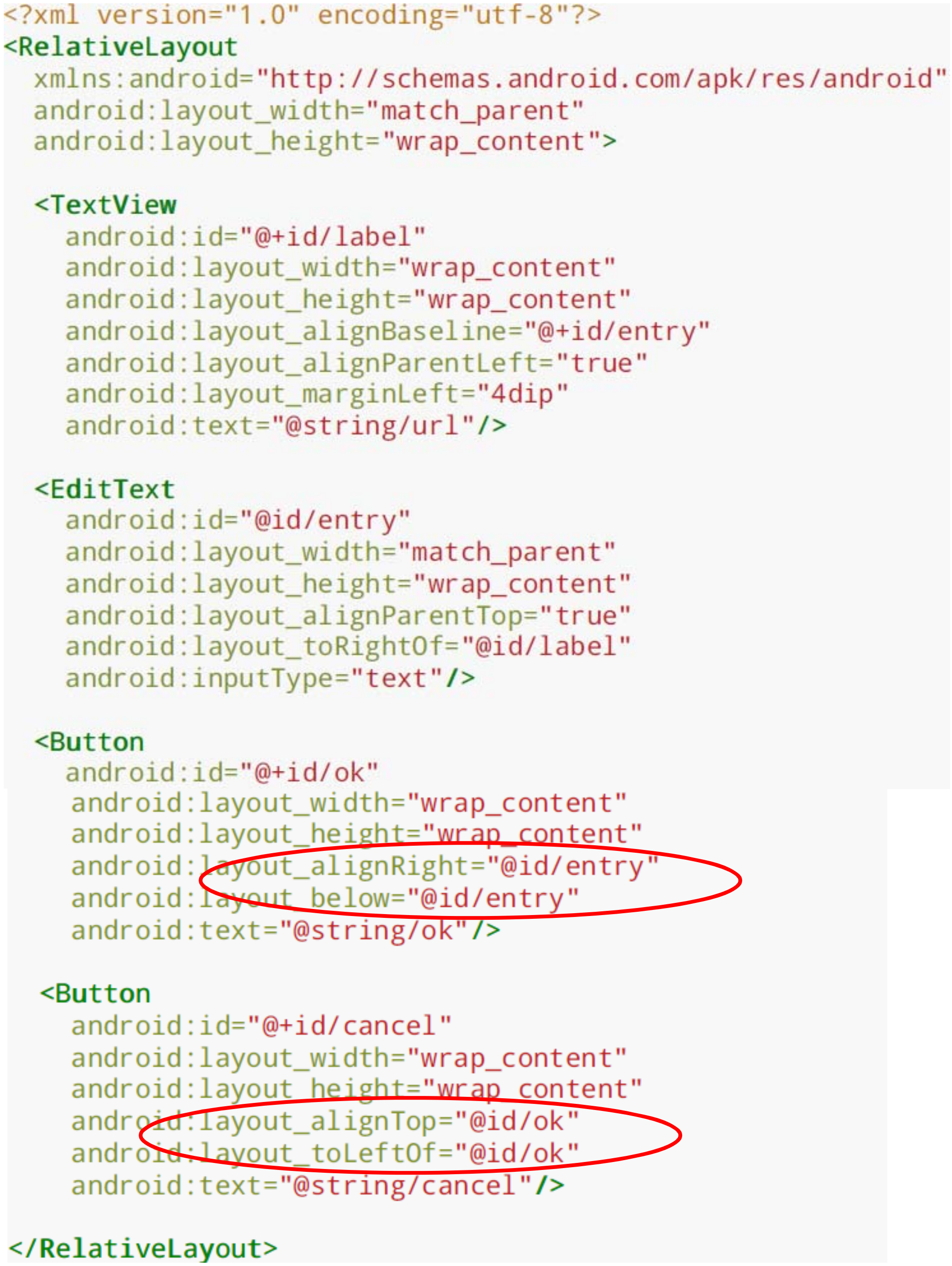
id=F toLeftOf E above D	id= E center_horizontal ParentTop	id= G toRightOf E above B
id=D center_vertical ParentLeft	id= A Center	id= B center_vertical ParentRight
id= I toLeftOf C below D	id= C center_horizontal ParentBottom	id= H toRightOf C below B



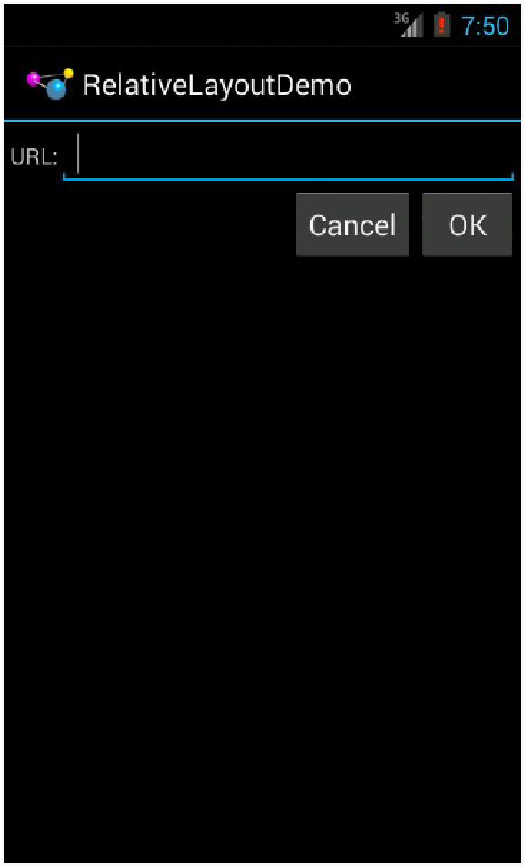
Layouts
FrameLayout
LinearLayout (Horizon
LinearLayout (Vertica
TableLayout
TableRow
GridLayout
RelativeLayout

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>
```



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

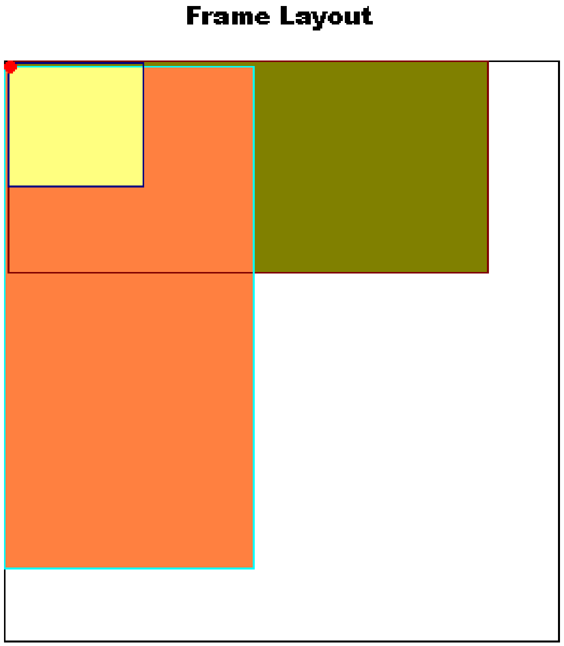
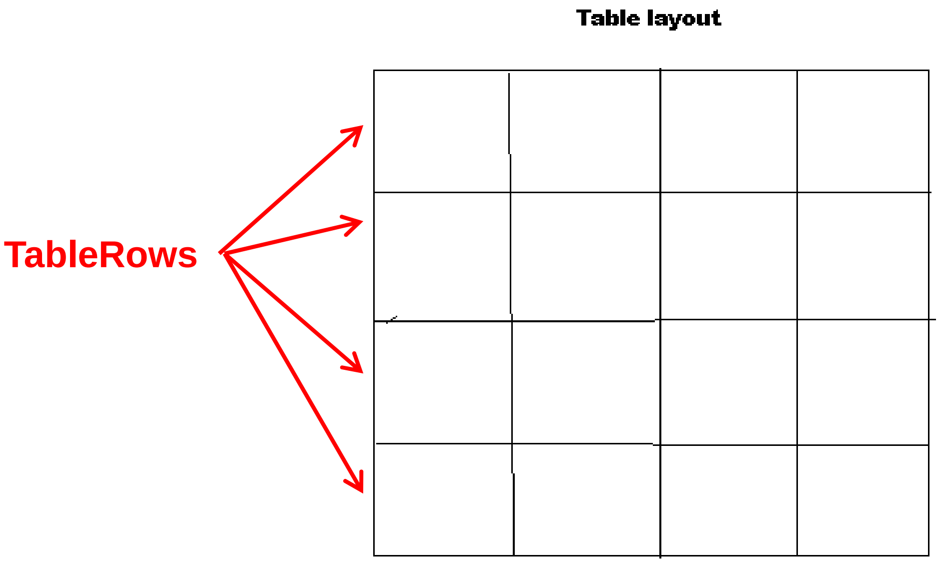
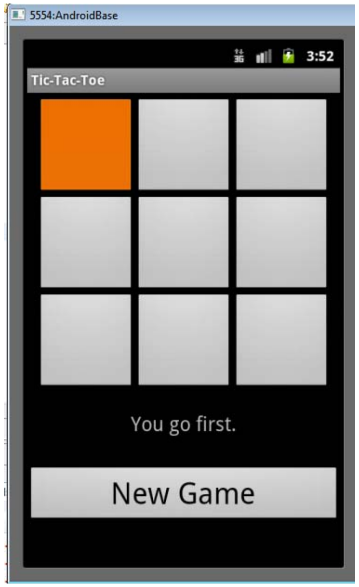


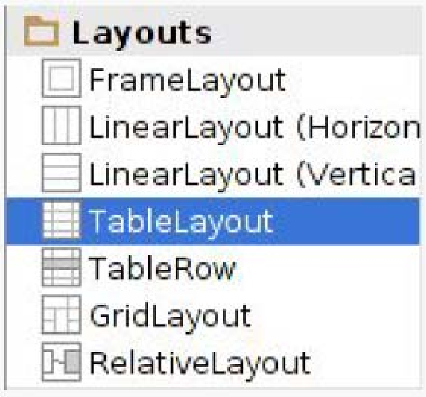
Table Layout

- 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



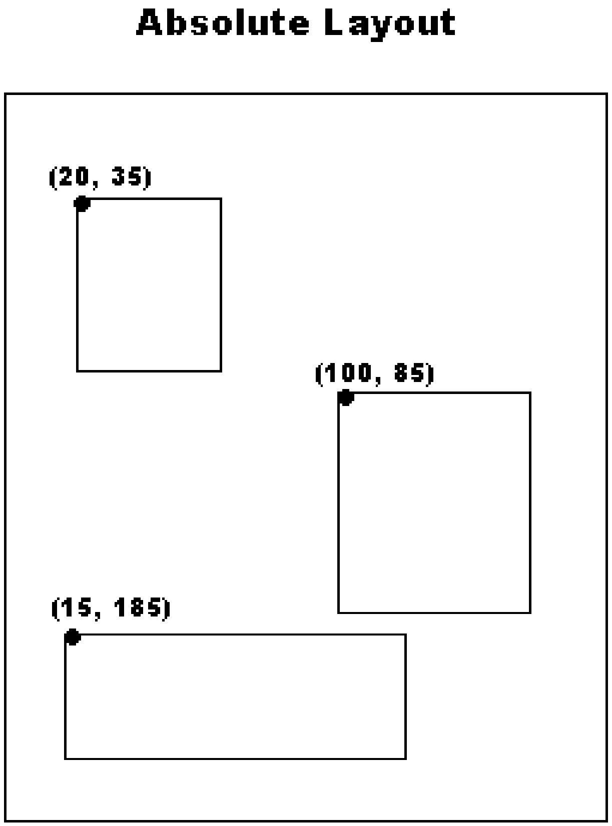


You go first.



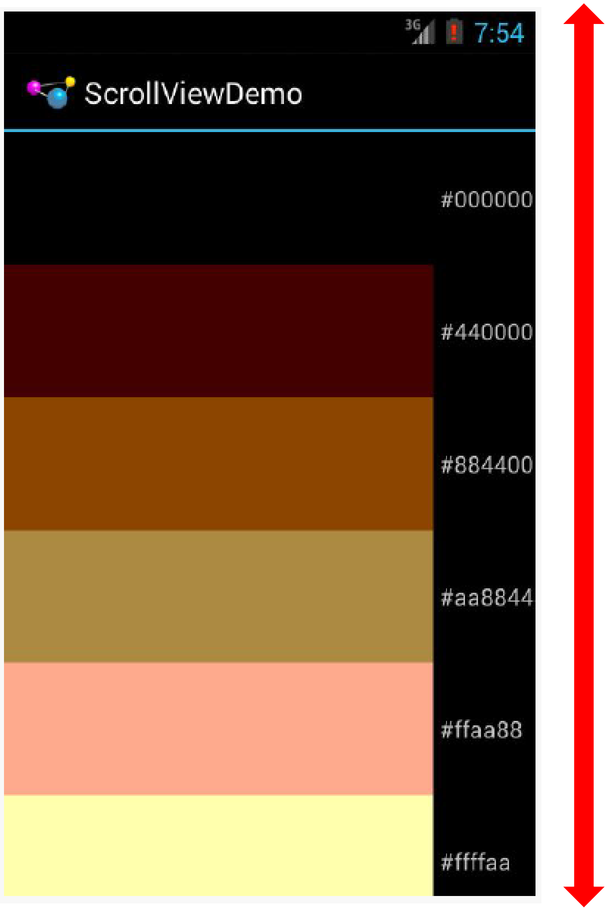
Absolute Layout

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



Scrolling

- Phone screens are small, scrolling content helps
- ListView supports vertical scrolling
- Other views for Scrolling:
  - **ScrollView** for vertical scrolling
  - **HorizontalScrollView**
- examples:
  - scroll through large image
  - Linear Layout with lots of elements



Android UI Youtube Tutorials

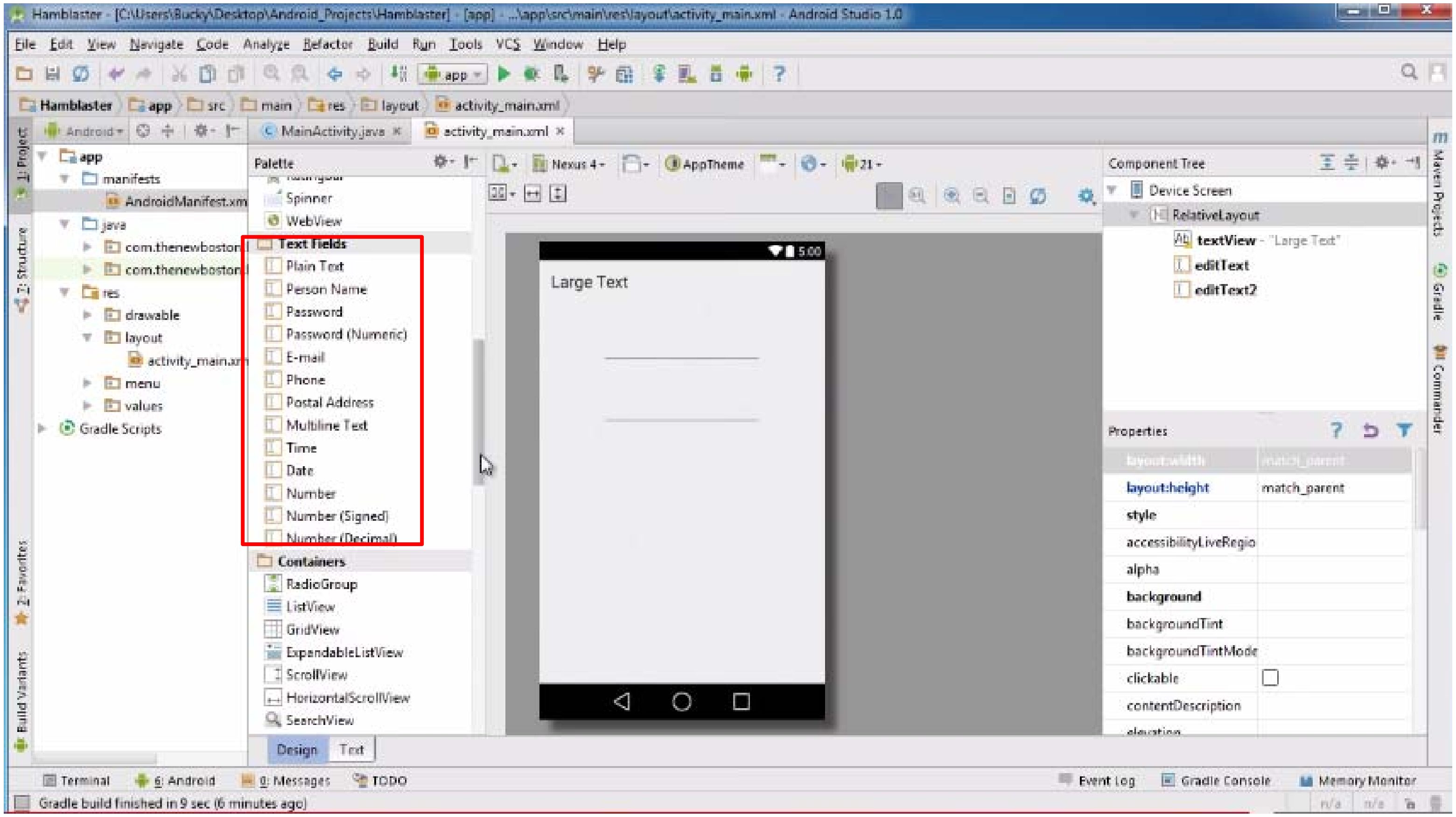
Tutorial 11: Designing the User Interface

- 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)



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

layout:width match_parent

layout:height match_parent

Gradle build finished in 9 sec (6 minutes ago)

Tutorial 12: More on User Interface

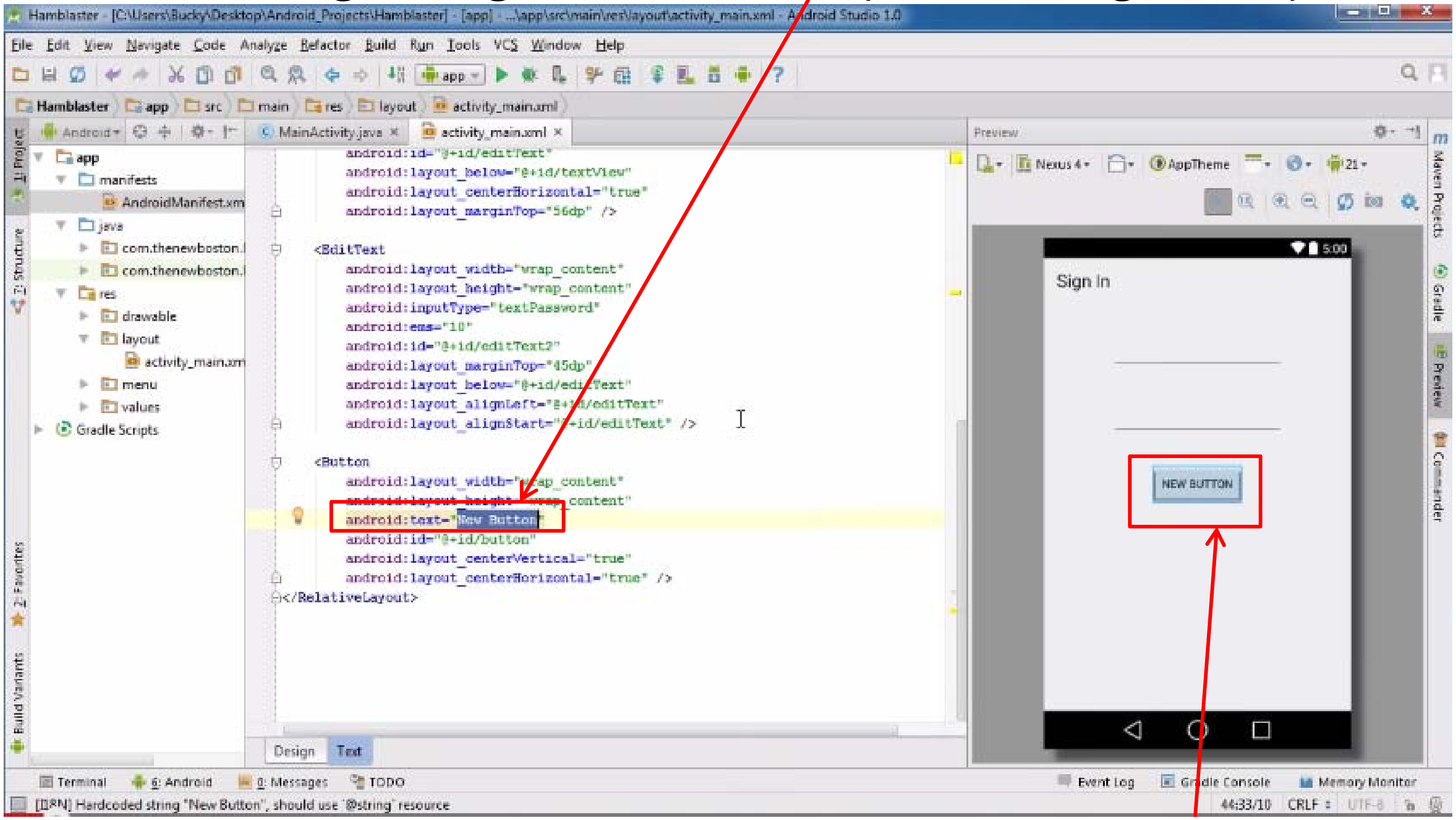
- Tutorial 12: More on User Interface [10:24 mins]
  - https://www.youtube.com/watch?v=72mf0rmjNAA

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

Changing Widget text in Text View

Change text “New Button” in XML file,

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



```
        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”

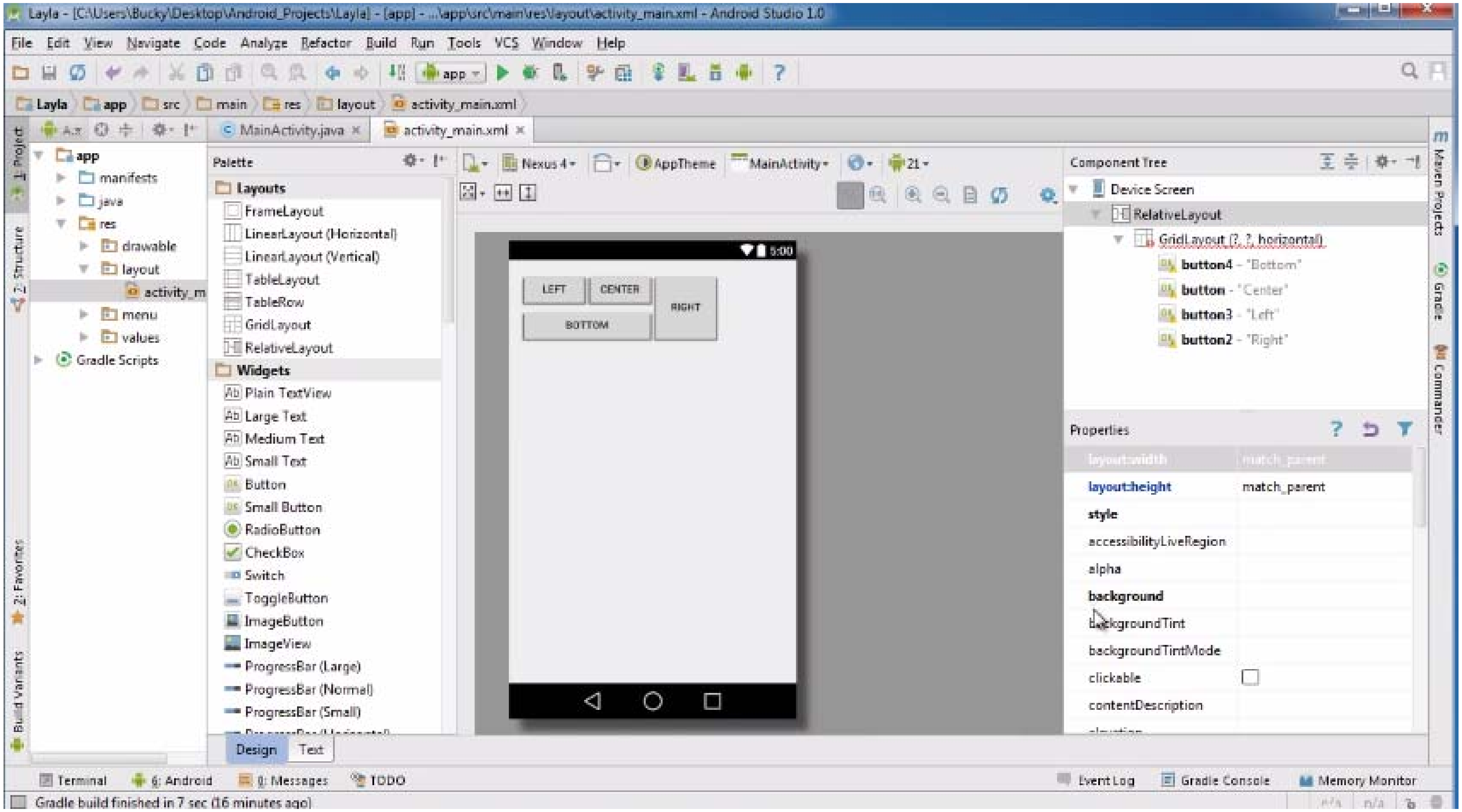
Tutorial 17: GridLayout

- 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



Component Tree

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

Properties

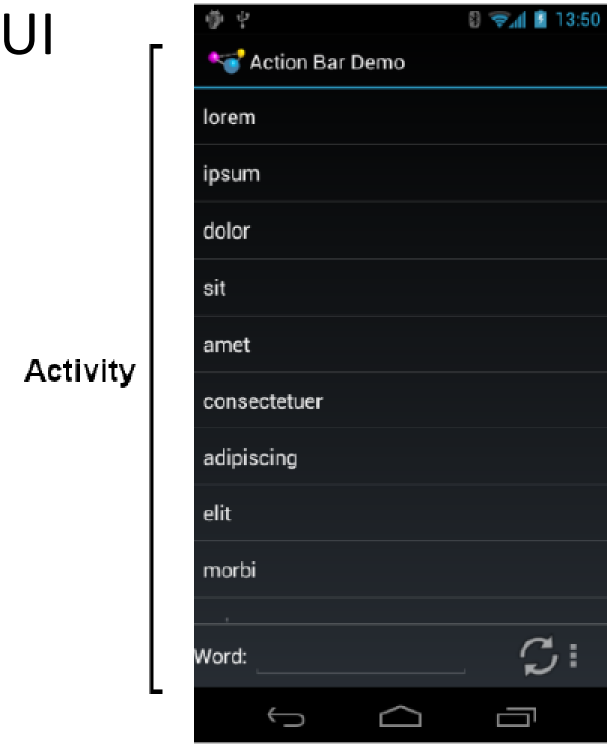
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□
contentDescription	

Gradle build finished in 7 sec (16 minutes ago)

Our First Android App

Activities

- Single Android window or dialog box
- Apps have at least 1 activity that deals with UI
  - An entry point of app similar to **main( )** in C
- Many apps have multiple activities screens
- Example: A camera app
  - **Activity 1:** to focus, snap photo, start activity 2
  - **Activity 2:** to preview picture, save it



13:50

Action Bar Demo

lorem

ipsum

dolor

sit

amet

consectetuer

adipiscing

elit

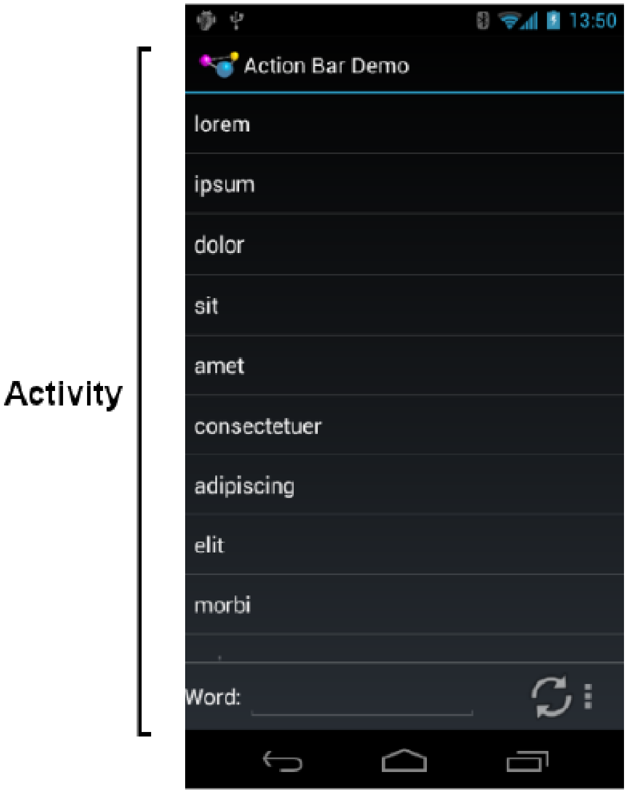
morbi

Word:

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
- E.g. to declare activity

```
Public class EmPubLiteActivity extends Activity{

// …..write code to control activity


}
```

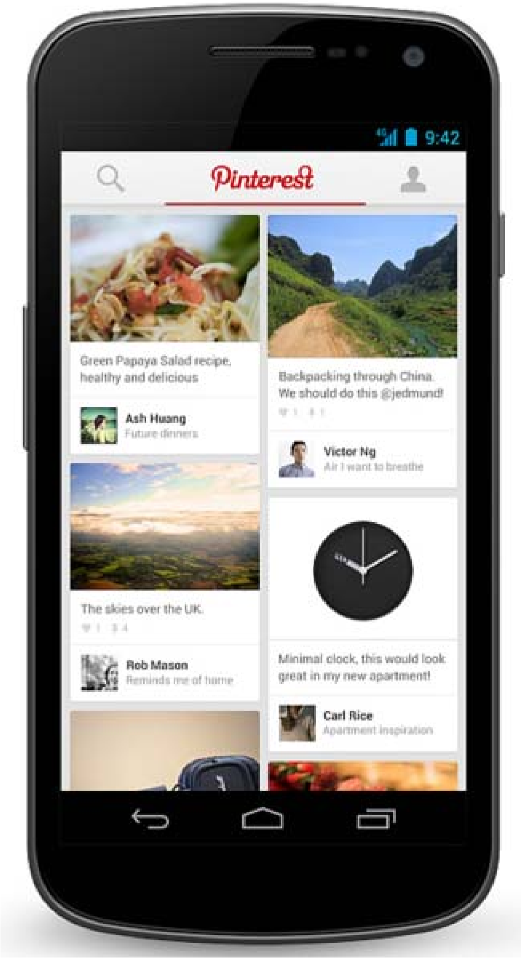


lorem
ipsum
dolor
sit
amet
consectetuer
adipiscing
elit
morbi

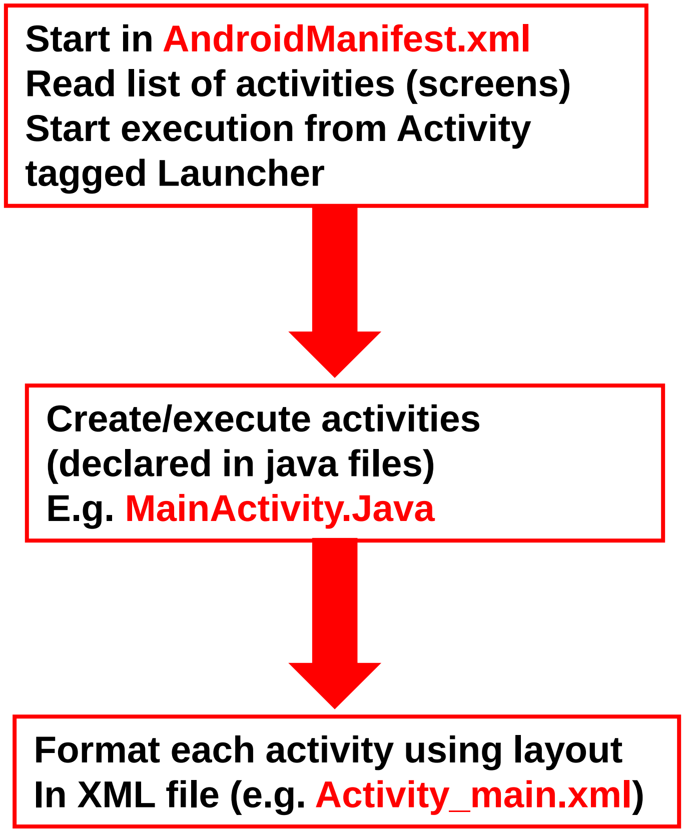
Word:

Recall: Files Hello World Android Project

- 3 Files:
  - **Activity_main.xml:** XML file, specifies screen layout
  - **MainActivity.Java:** Java code to define app behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all app components, activities (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), launches 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

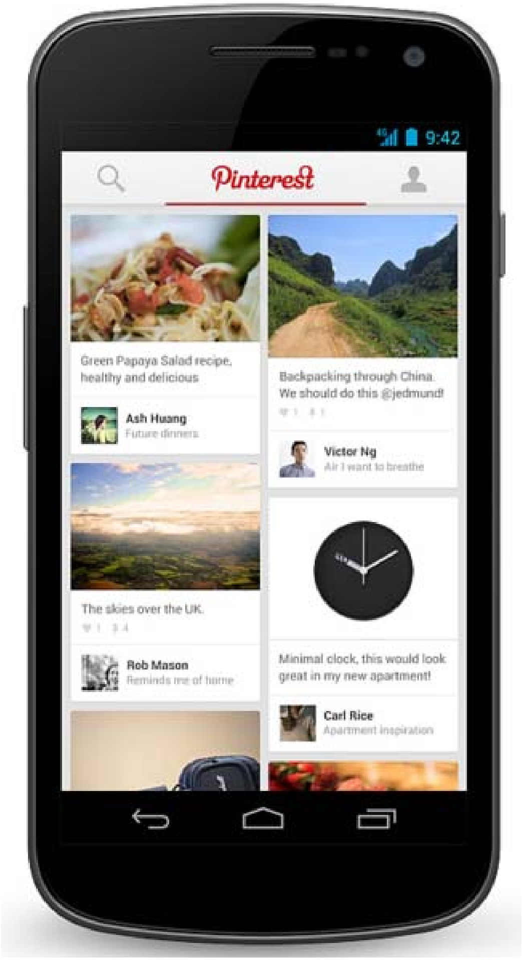
Execution Order



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

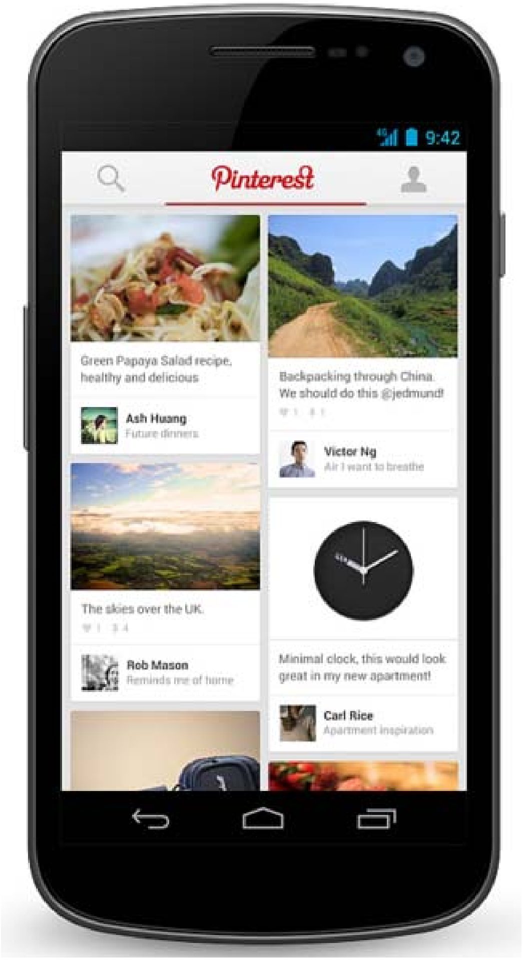
Create/execute activities
(declared in java files)
E.g. MainActivity.Java

Format each activity using layout
In XML file (e.g. Activity_main.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_main.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
    - 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”



Green Papaya Salad recipe, healthy and delicious

Backpacking through China. We should do this @jedmund!

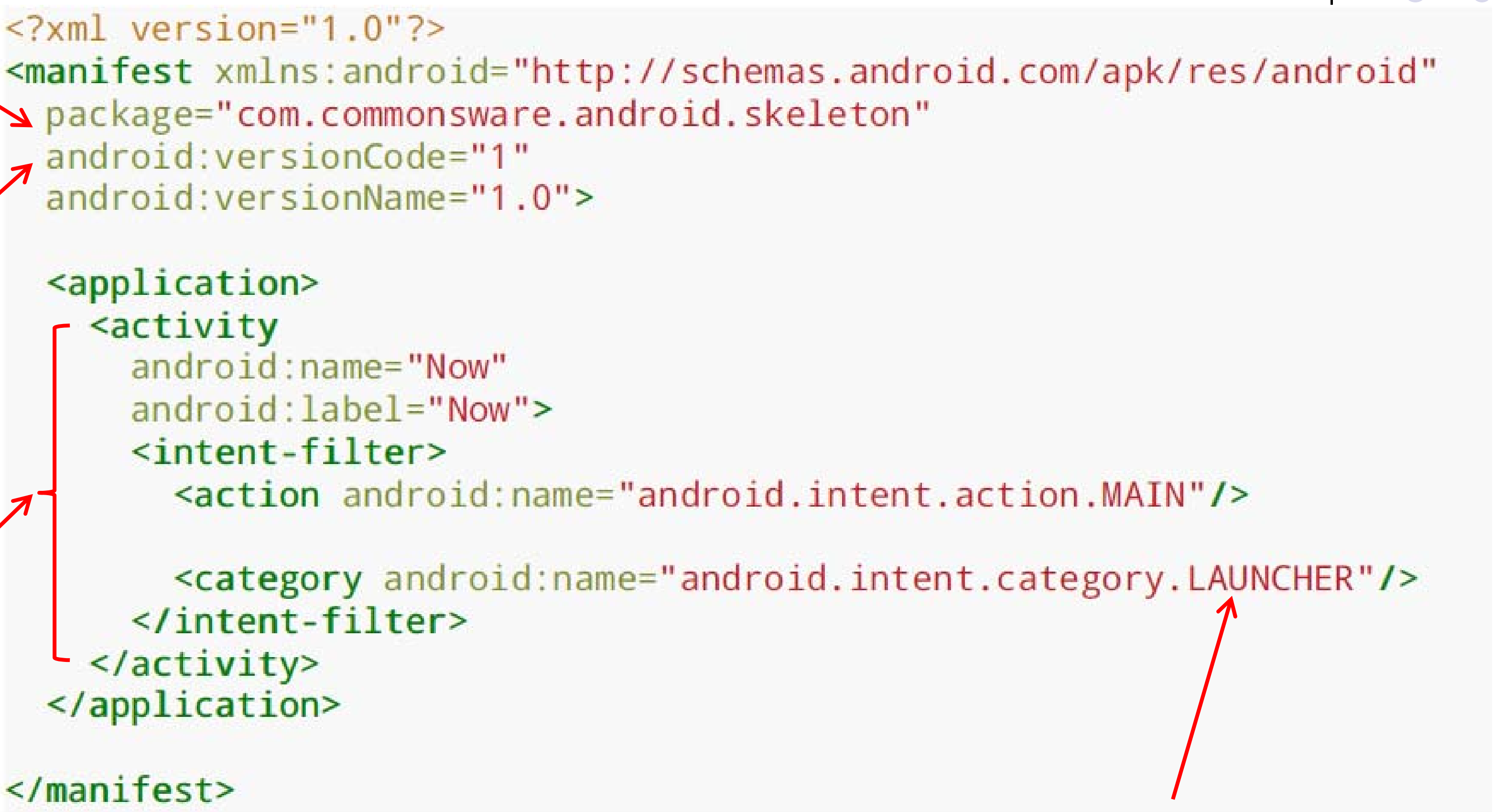
The skies over the UK.

Minimal clock, this would look great in my new apartment!

Next: Let’s look at AndroidManifest.XML

Recall: Inside “Hello World” AndroidManifest.xml

Your package name



```
<?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>
```

Android version

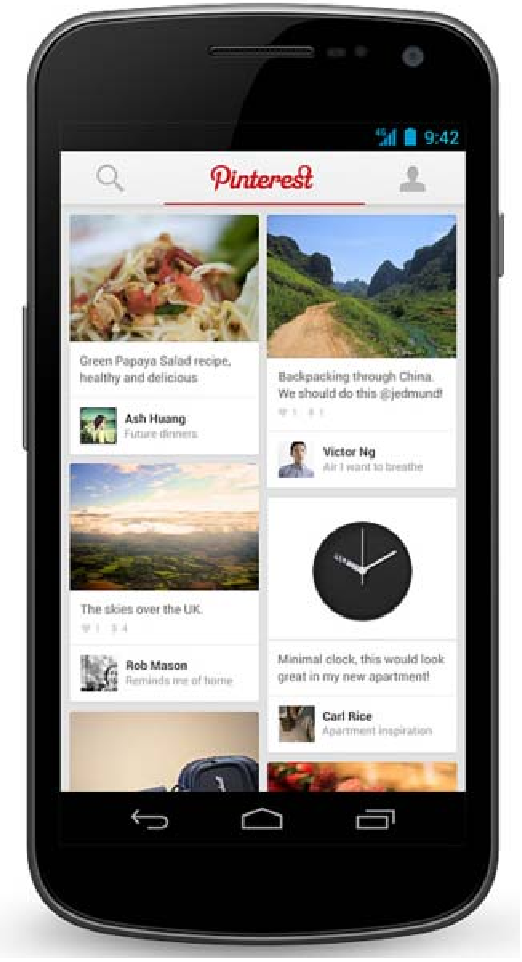
1 activity (screen) listed for this app

One activity (screen) designated LAUNCHER. The app starts running here

Recall: Files Hello World Android Project

Next: Let's look at Simple java file

- 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 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"



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

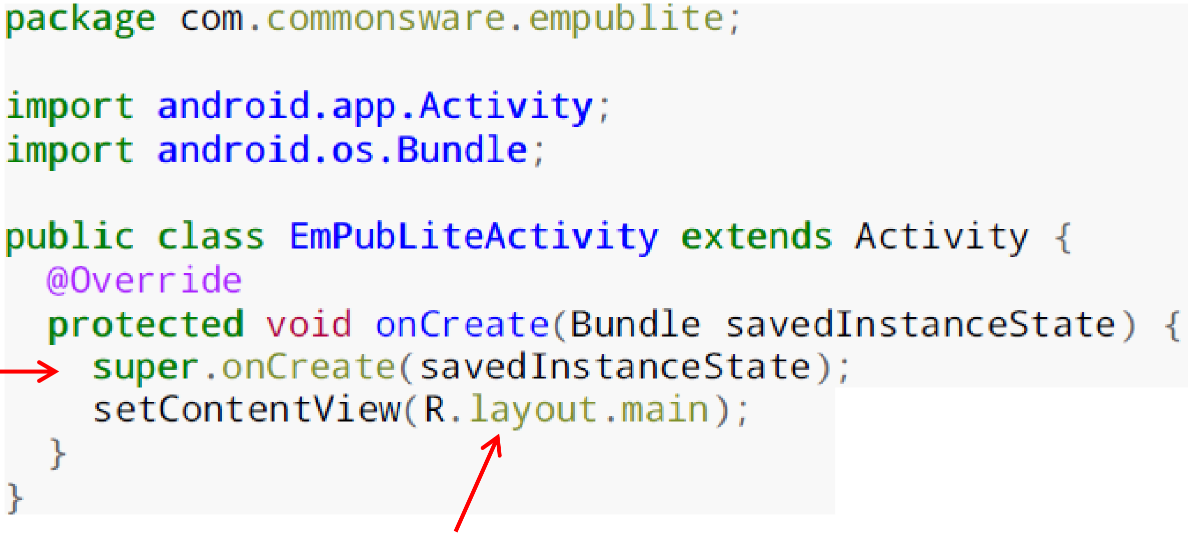
Example Activity Java file (E.g. MainActivity.java)

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



```
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);
  }
}
```

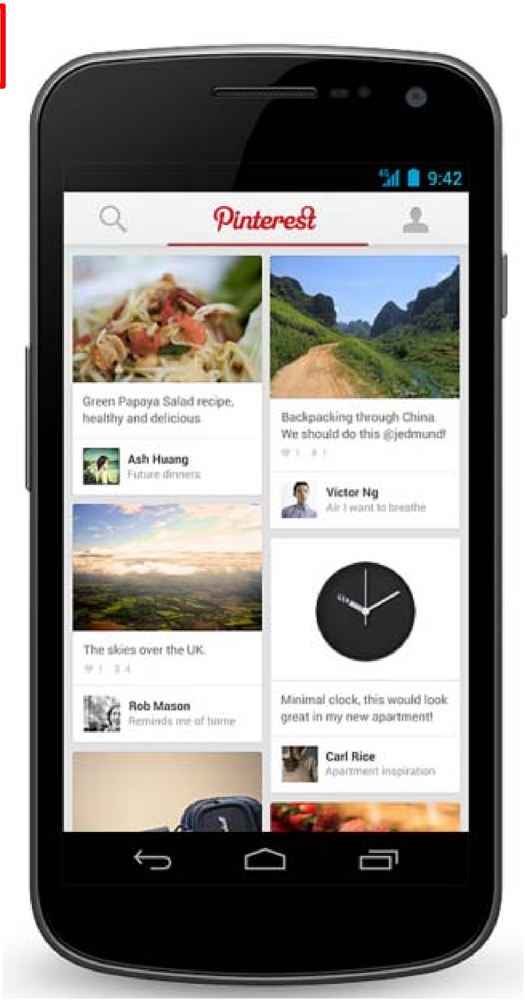
Use screen layout (design)
declared in file main.xml stored
in folder res/layout

Note: Android OS calls your 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_main.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

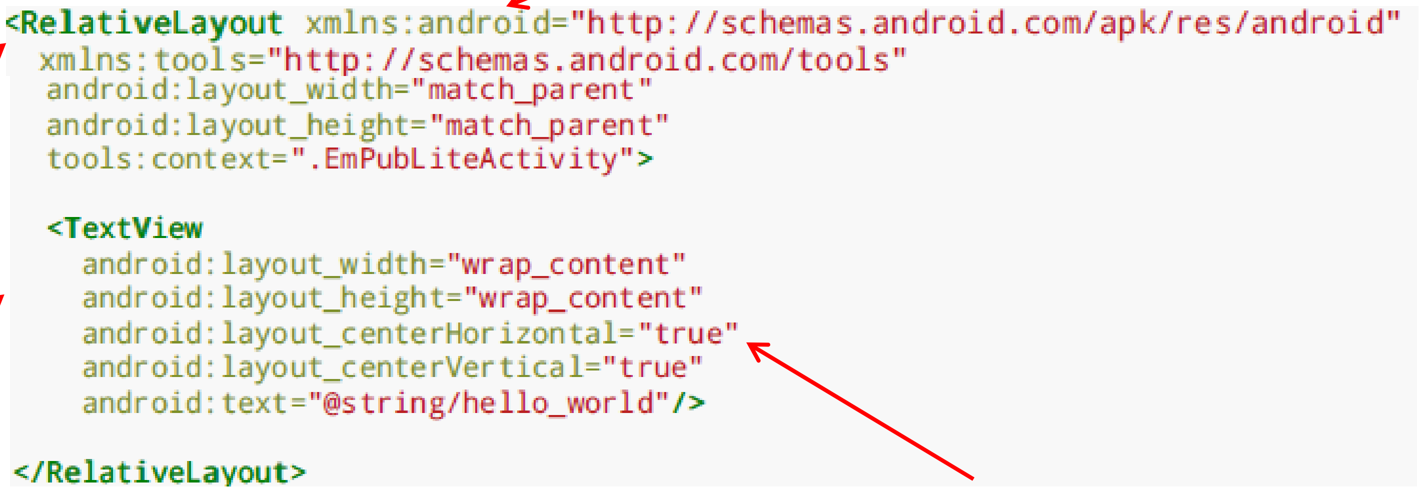
Simple XML file Designing UI

- 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



```
<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>
```

Widget properties
(e.g. center contents
horizontally and vertically)

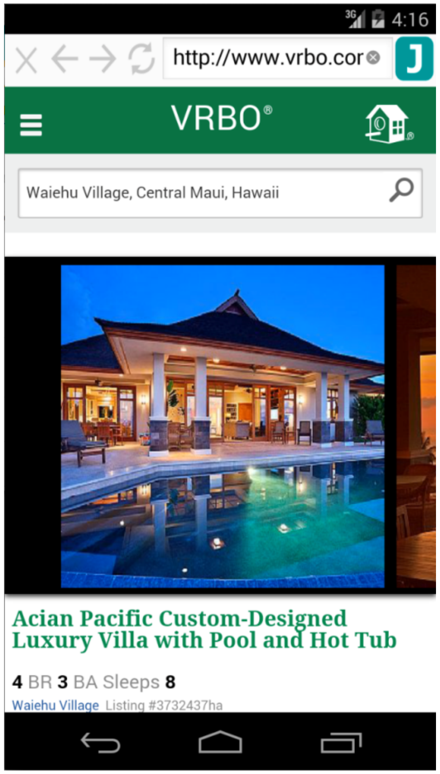
WebView Widget

WebView Widget

- A View that display web pages
  - Can be used for creating your own web browser
  - OR just display some online content inside your app
- Uses WebKit rendering engine (lots of memory)
  - http://www.webkit.org/
- Webkit used in many web browsers including Safari

The WebKit Open Source Project

- Web pages in WebView same look same as Safari



Waiehu Village, Central Maui, Hawaii

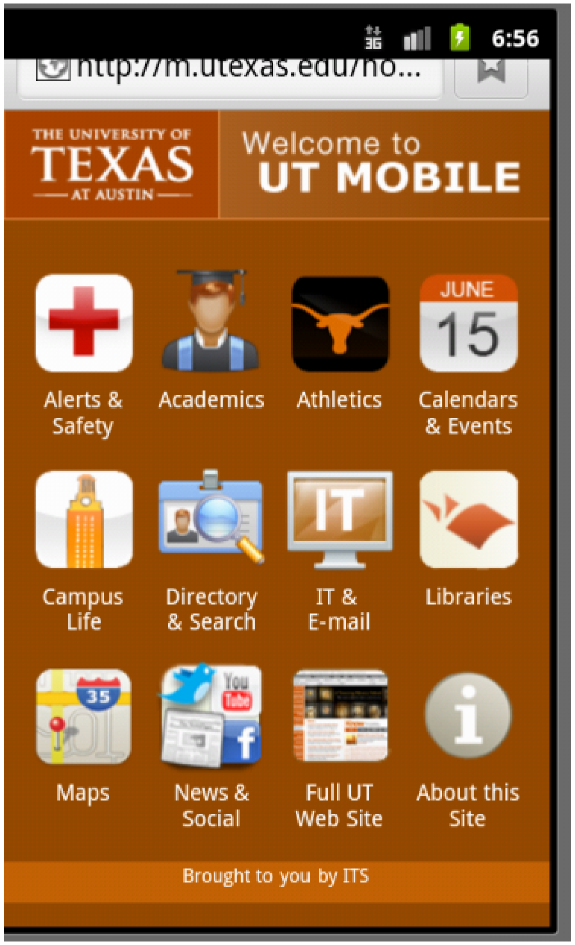
Acian Pacific Custom-Designed Luxury Villa with Pool and Hot Tub

4 BR 3 BA Sleeps 8

Waiehu Village Listing #3732437ha

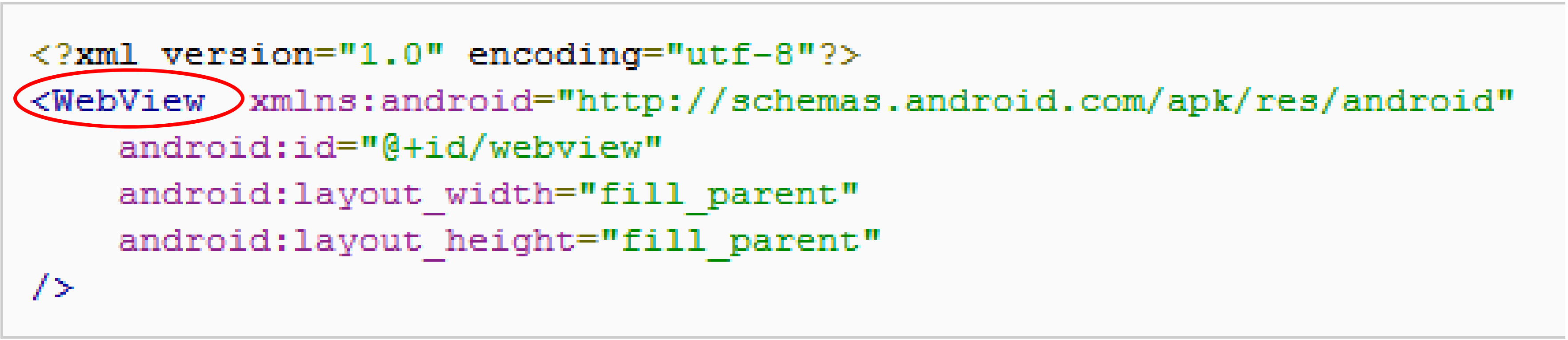
WebView Widget Functionality

- **Display Web page** containing HTML, CSS, Javascript
- **Navigation history** of URLs to support forward and backwards
- **Zoom in and out**
- perform **searches**
- Additional functionality:
  - capture images of page
  - Search page for string
  - Deal with cookies on a per application basis



WebView Example

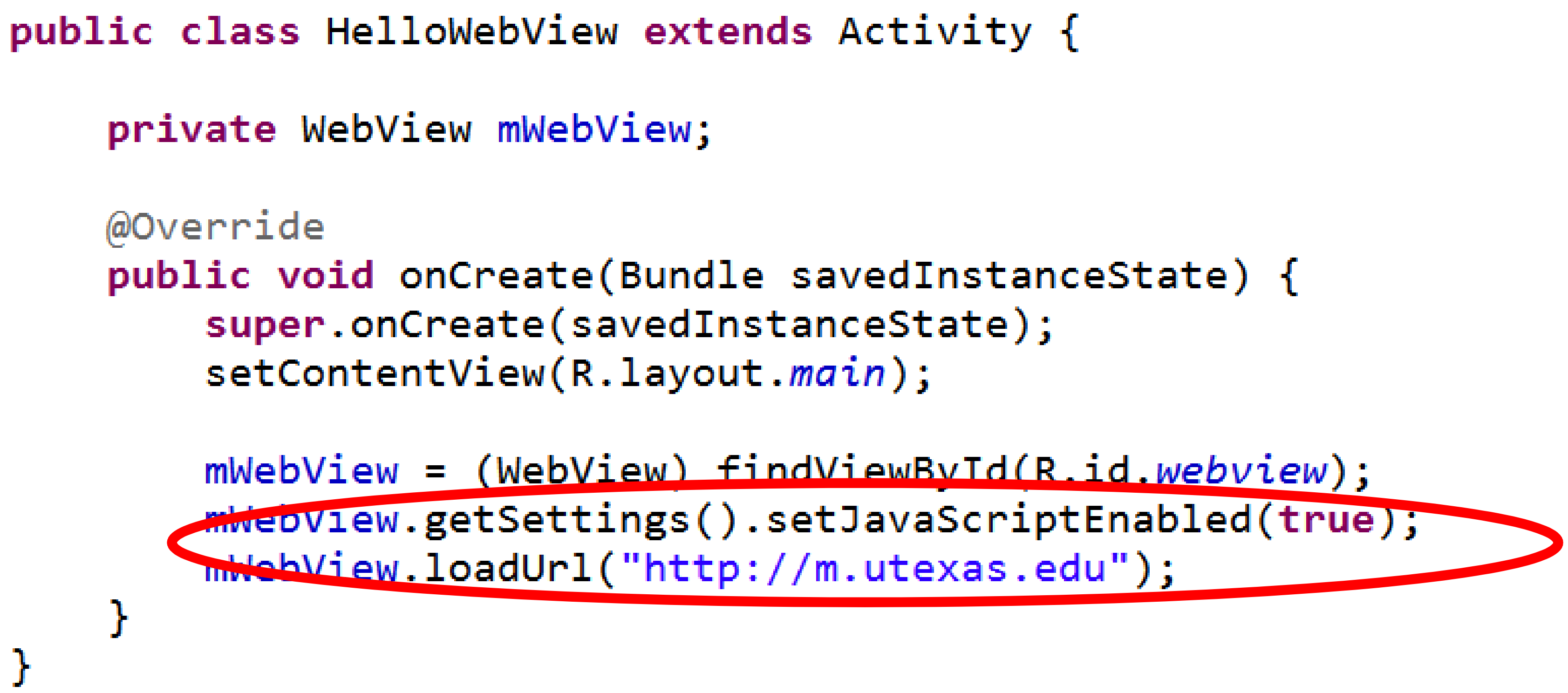
- Simple app to view and navigate web pages
- XML code (e.g in res/layout/main.xml)



```xml
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

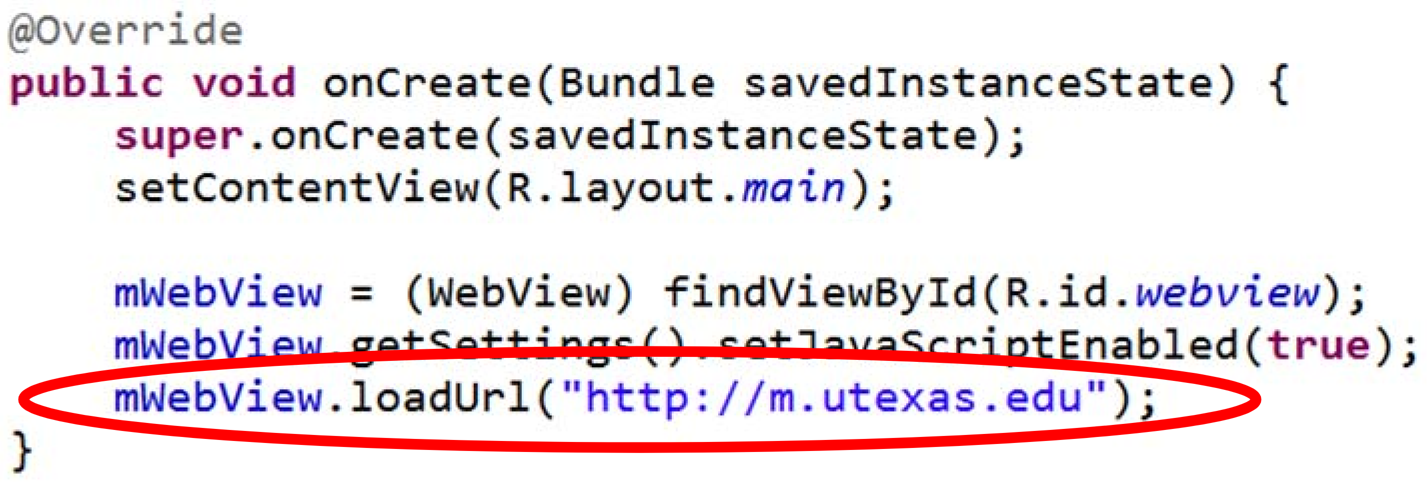
WebView Activity

- In onCreate, use loadURL to load website
- If website contains Javascript, enable Javascript



```
public class HelloWebView extends Activity {

    private WebView mWebView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mWebView = (WebView) findViewById(R.id.webview);
        mWebView.getSettings().setJavaScriptEnabled(true);
        mWebView.loadUrl("http://m.utexas.edu");
    }
}
```

loadUrl( )

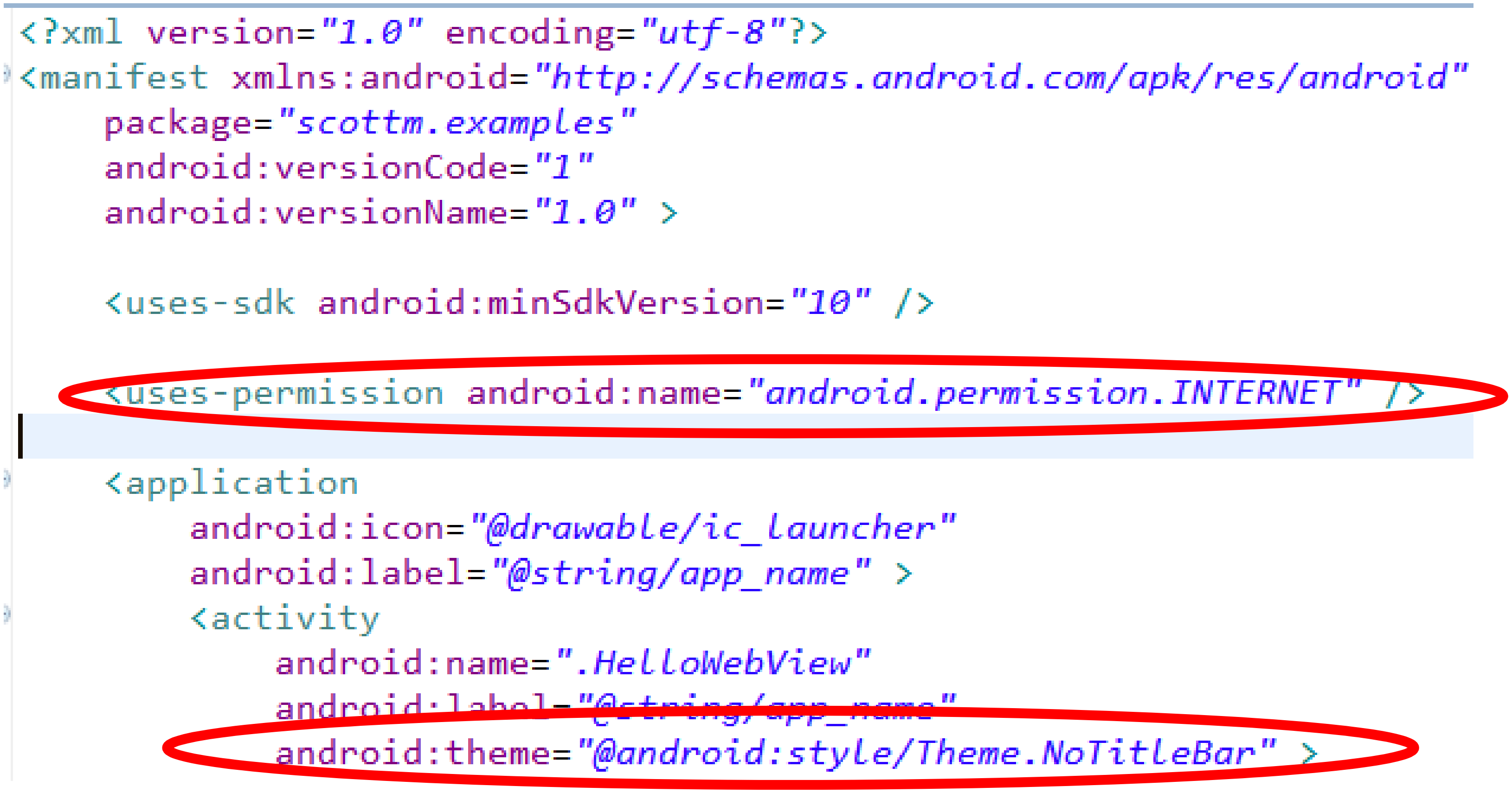


```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mWebView = (WebView) findViewById(R.id.webview);
    mWebView.getSettings().setJavaScriptEnabled(true);
    mWebView.loadUrl("http://m.utexas.edu");
}
```

- loadUrl( ) Works with
  - **http://** and **https://** URLs
  - **file//** URLs pointing to local filesystem
  - **file:///** android_asset/ URLs pointing to app's assets (later)
  - **content://** URLs pointing to content provider that is streaming published content

WebView Example

- Add permission to AndroidManifest.xml for app to use Internet
- Also change style so no title bar

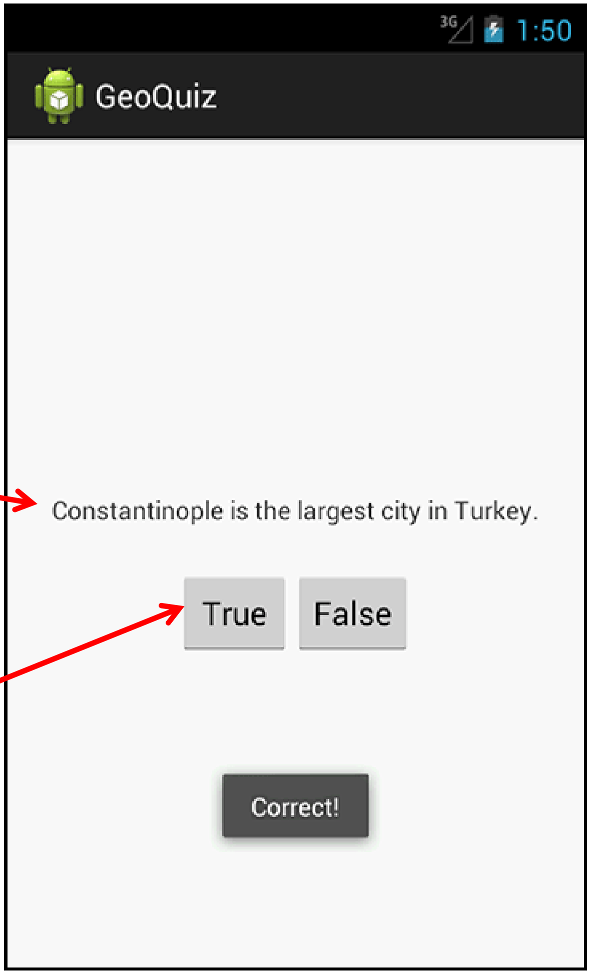


```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="scottm.examples"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloWebView"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar" >
```

Android UI Design Example

GeoQuiz App
Reference: Android Nerd Ranch, pgs 1-30

- App presents questions to test user's knowledge of geography
- User answers by pressing **True** or **False** buttons
- How to get this book?



1:50

GeoQuiz

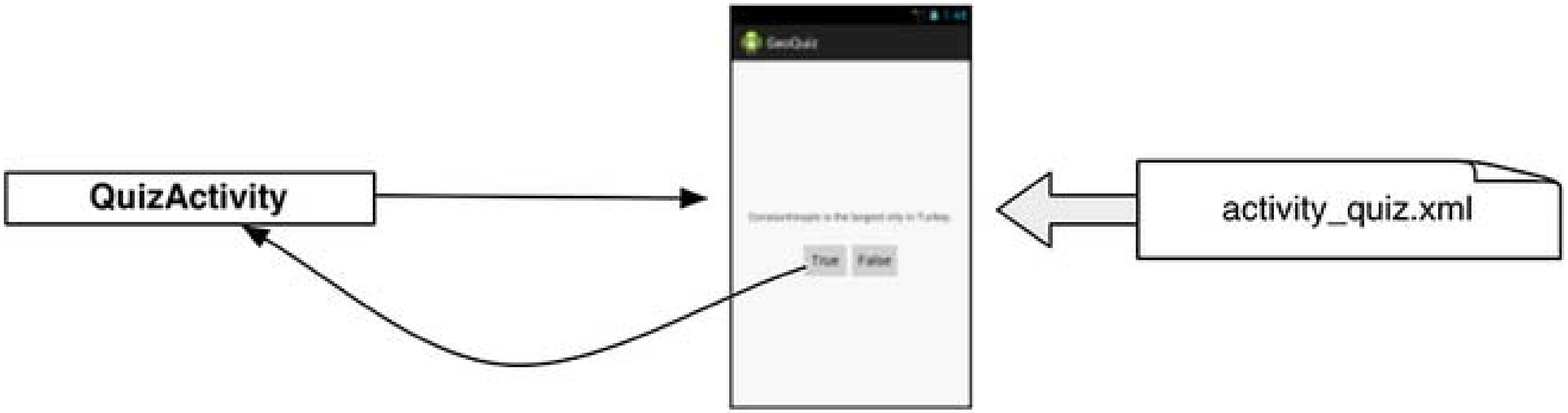
Constantinople is the largest city in Turkey.

Question

User responds by clicking True or False

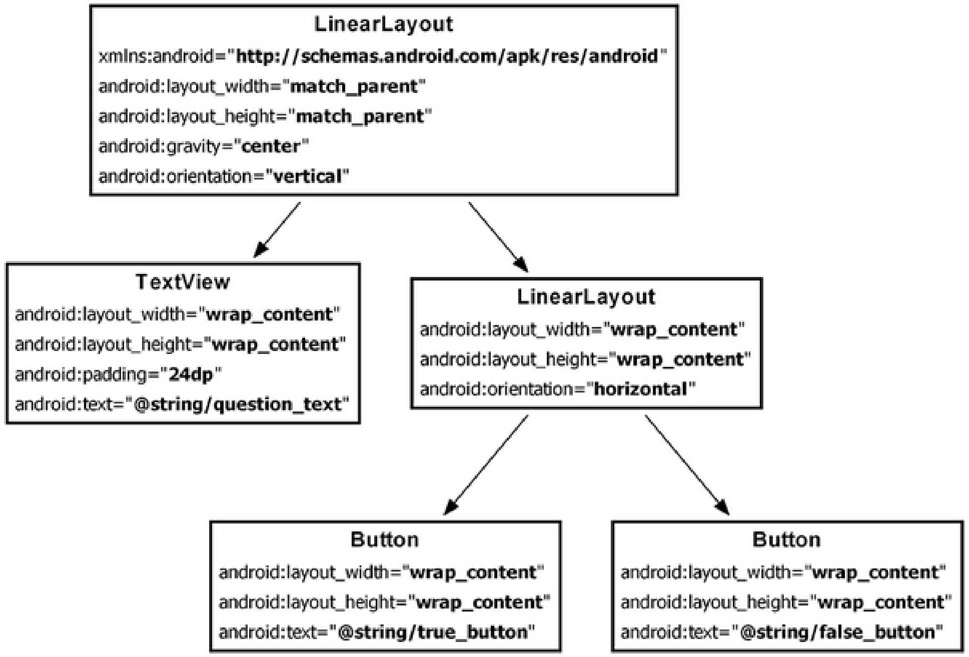
GeoQuiz App

- 2 main files:
  - **activity_quiz.xml:** to format app screen
  - **QuizActivity.java:** To present question, accept True/False response
- **AndroidManifest.xml** also auto-generated



GeoQuiz: Plan Out App Widgets

- 5 Widgets arranged hierarchically



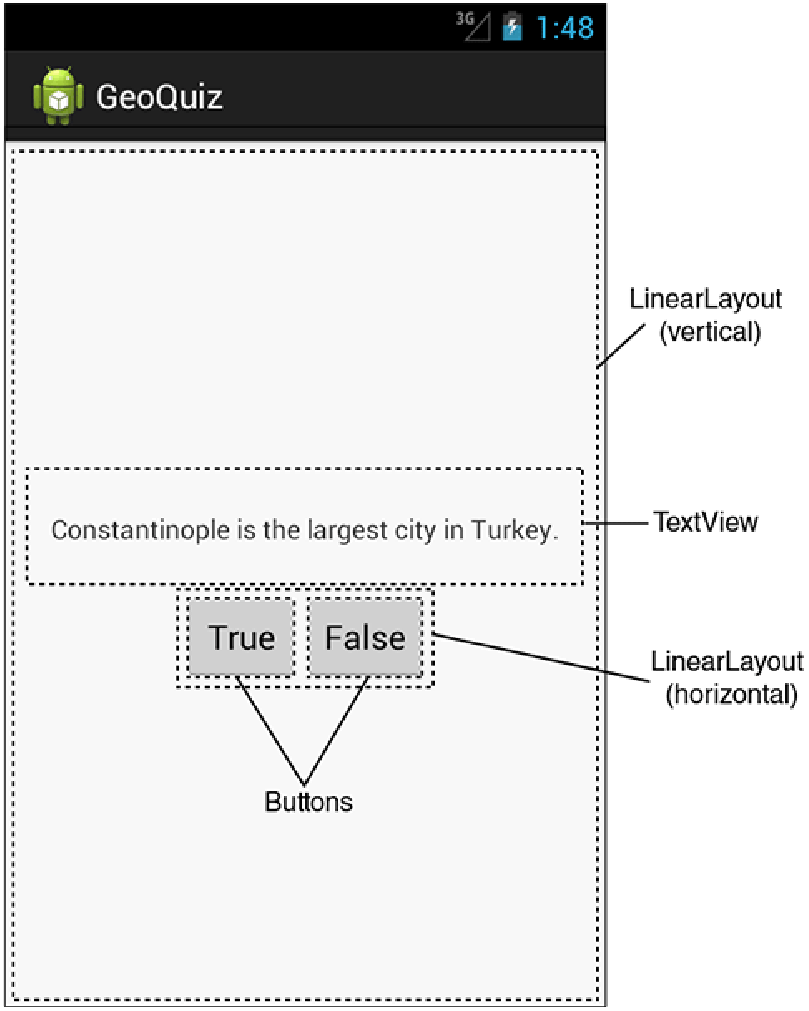
LinearLayout
xmlns:android="**http://schemas.android.com/apk/res/android**"
android:layout_width="**match_parent**"
android:layout_height="**match_parent**"
android:gravity="**center**"
android:orientation="**vertical**"

TextView
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:padding="**24dp**"
android:text="**@string/question_text**"

LinearLayout
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:orientation="**horizontal**"

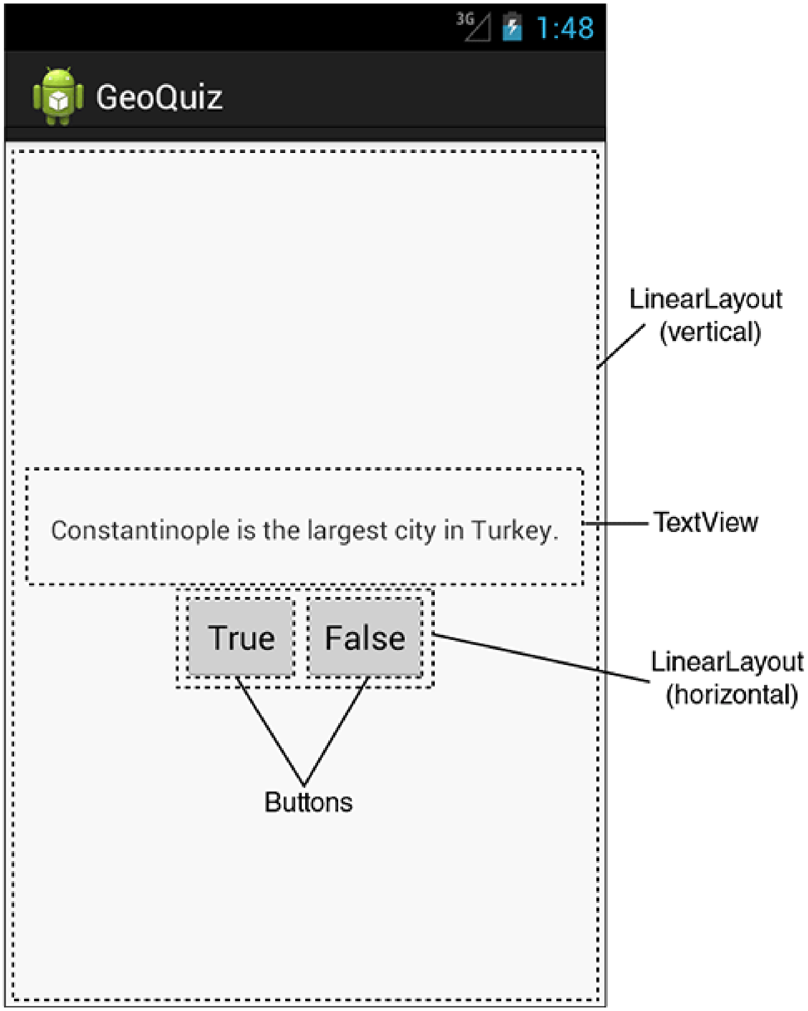
Button
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/true_button**"

Button
android:layout_width="**wrap_content**"
android:layout_height="**wrap_content**"
android:text="**@string/false_button**"


3G
1:48
GeoQuiz
Constantinople is the largest city in Turkey.
True
False
LinearLayout (vertical)
TextView
LinearLayout (horizontal)
Buttons

GeoQuiz: activity_quiz.xml File listing

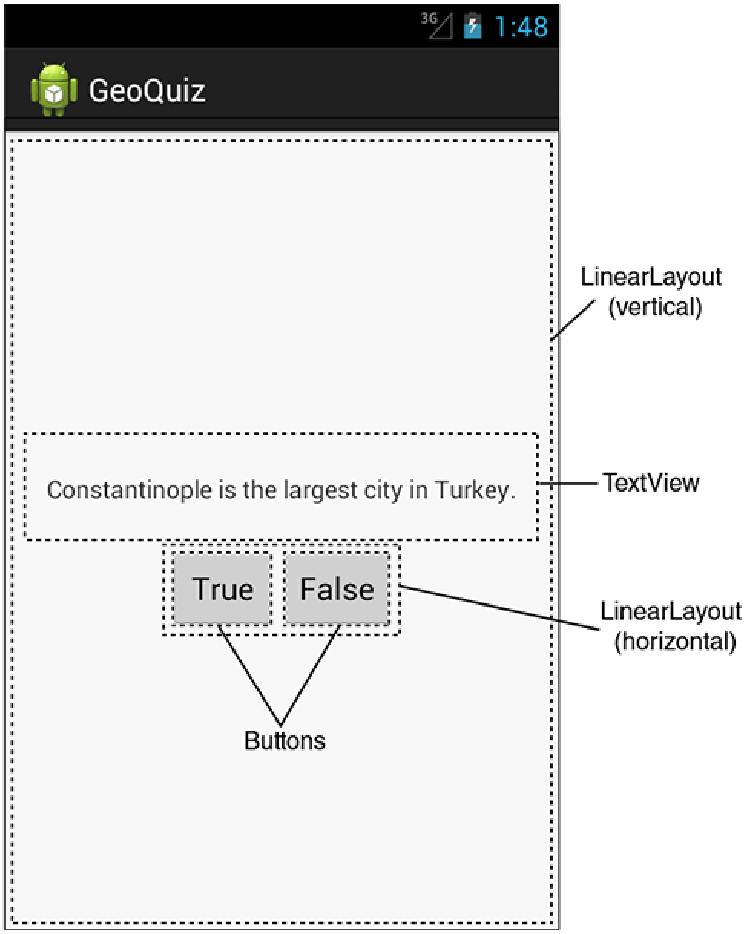
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:orientation="vertical" >

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/true_button" />

    <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/false_button" />

  </LinearLayout>

</LinearLayout>
```



Constantinople is the largest city in Turkey.

GeoQuiz: strings.xml File listing

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

  <string name="app_name">GeoQuiz</string>
  ~~<string name="hello_world">Hello, world!</string>~~
  <string name="question_text">Constantinople is the largest city in
Turkey.</string>
  <string name="true_button">True</string>
  <string name="false_button">False</string>
  <string name="menu_settings">Settings</string>

</resources>
```



Constantinople is the largest city in Turkey.

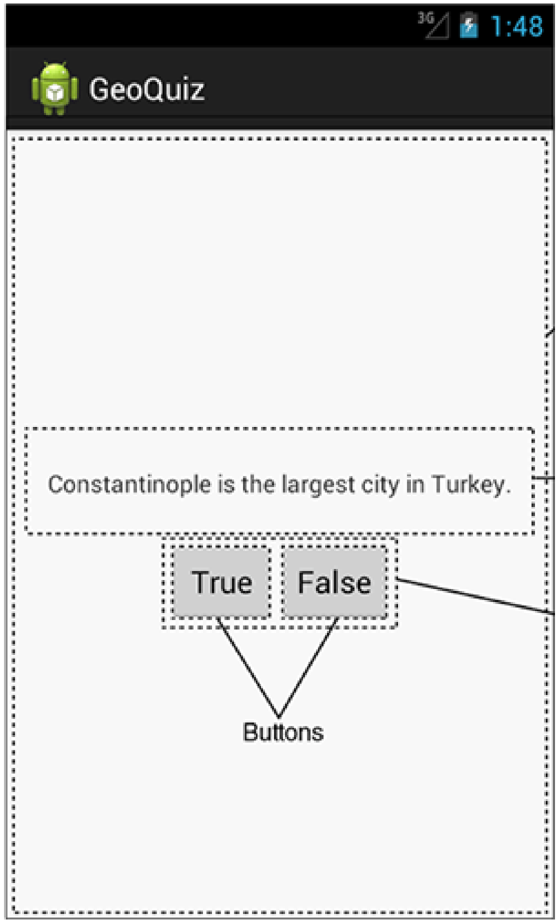
QuizActivity.java

- Initial QuizActivity.java code

```
package com.bignerdranch.android.geoquiz;

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

public class QuizActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
    }
```

onCreate Method is called once Activity is created

specify layout XML file

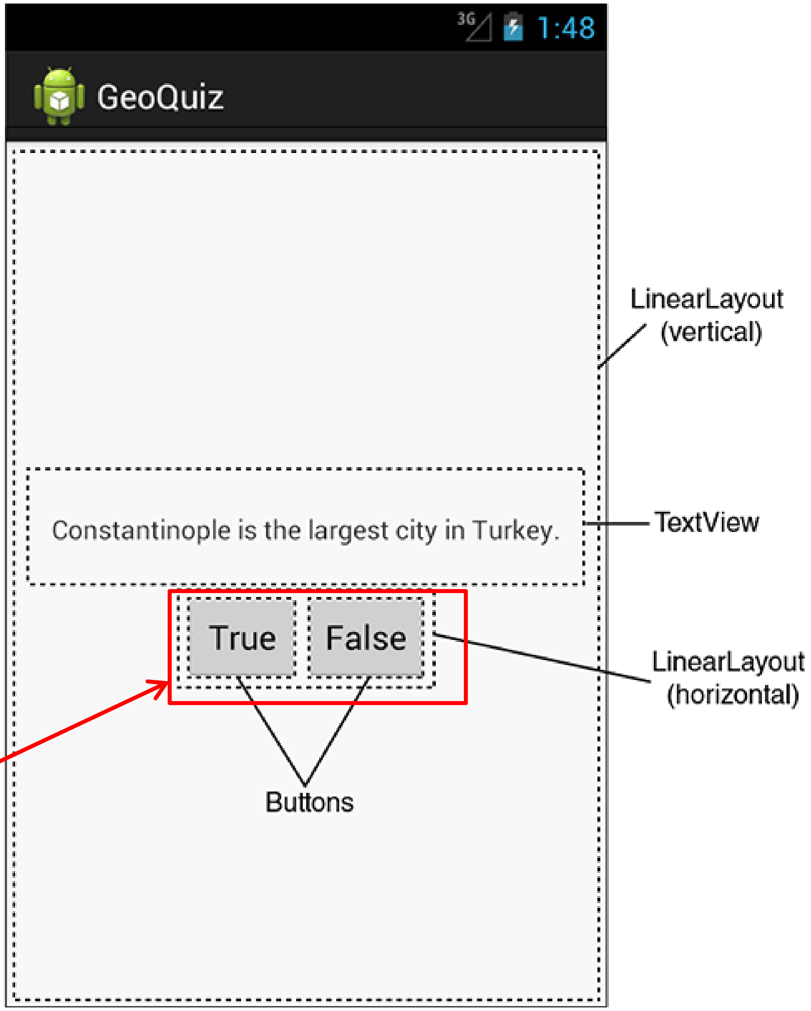


Constantinople is the largest city in Turkey.

- Would like java code to respond to True/False buttons being clicked

Responding to True/False Buttons in Java

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
... >

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="24dp"
    android:text="@string/question_text" />

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

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

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

  </LinearLayout>

</LinearLayout>
```



Constantinople is the largest city in Turkey.

Write code in Java file to specify app's response when True/False buttons are clicked

2 Ways to Respond to Button Clicks

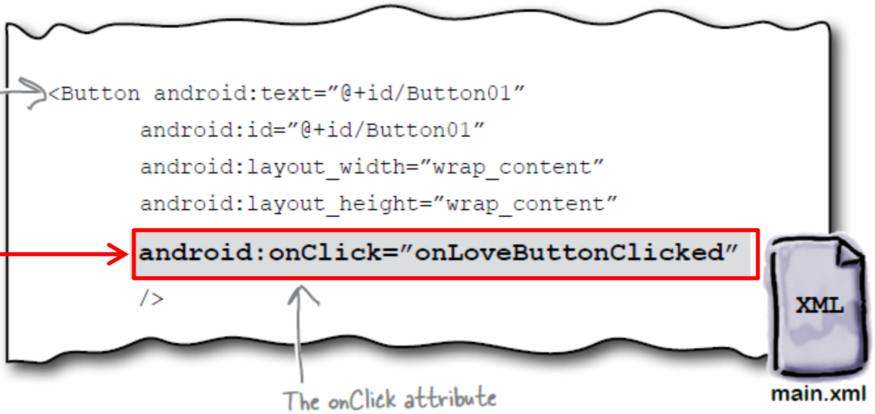
1. In XML: set android:onClick attribute

2. In java create a ClickListener object, override onClick method
   - typically done with anonymous inner class

Approach 1: Button that responds to Clicks
Reference: Head First Android

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

The Button definition from main.xml

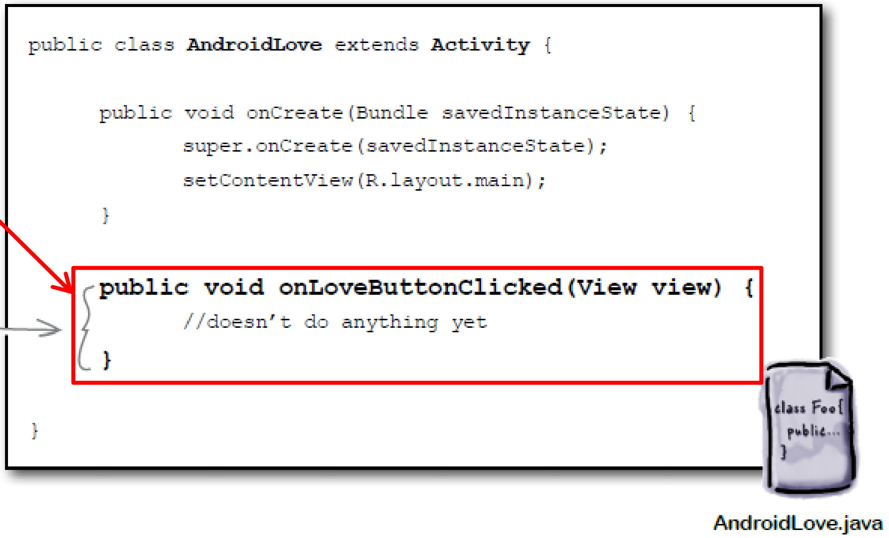


```
<Button android:text=”@+id/Button01”
        android:id=”@+id/Button01”
        android:layout_width=”wrap_content”
        android:layout_height=”wrap_content”
        android:onClick=”onLoveButtonClicked”
        />
```

The onClick attribute added to the Button. Pointing to the onLoveButtonClicked method.

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

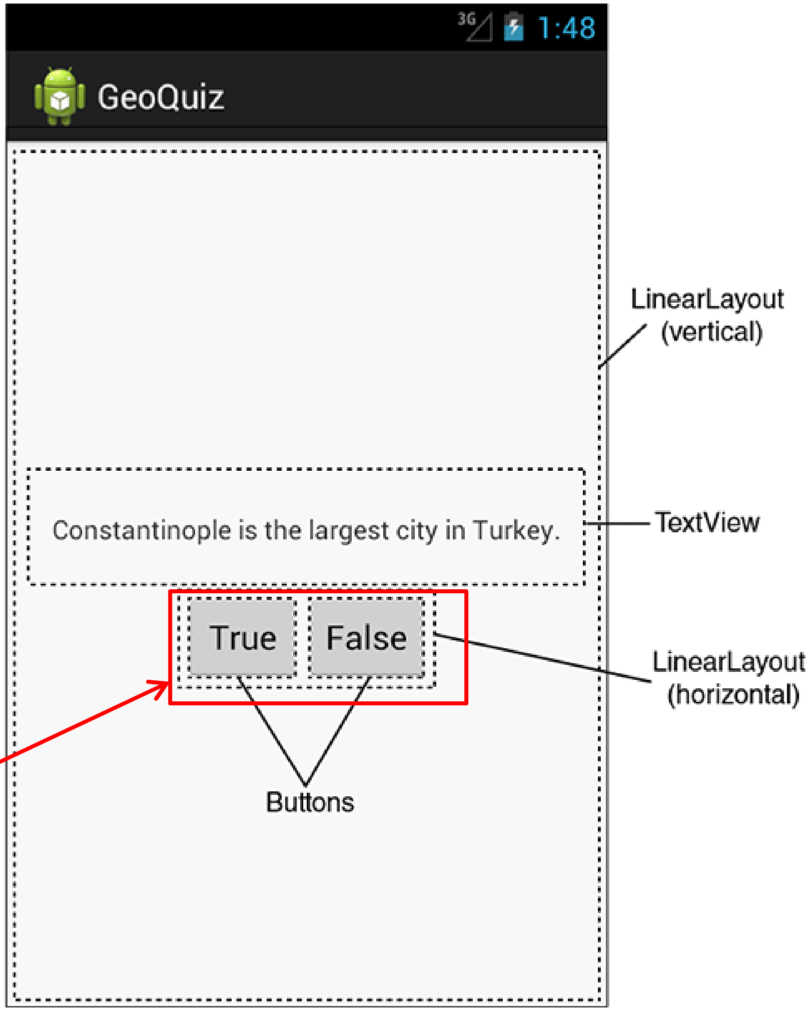
The new onLoveVuttonClicked method that's referenced from the android:onClick Button attribute.



```
public class AndroidLove extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onLoveButtonClicked(View view) {
        //doesn't do anything yet
    }

}
```

Approach 2: Create a ClickListener object, override onClick

- First, get reference to Button in our Java file. How?



3G 1:48
GeoQuiz
Constantinople is the largest city in Turkey.
True
False
LinearLayout (vertical)
TextView
LinearLayout (horizontal)
Buttons

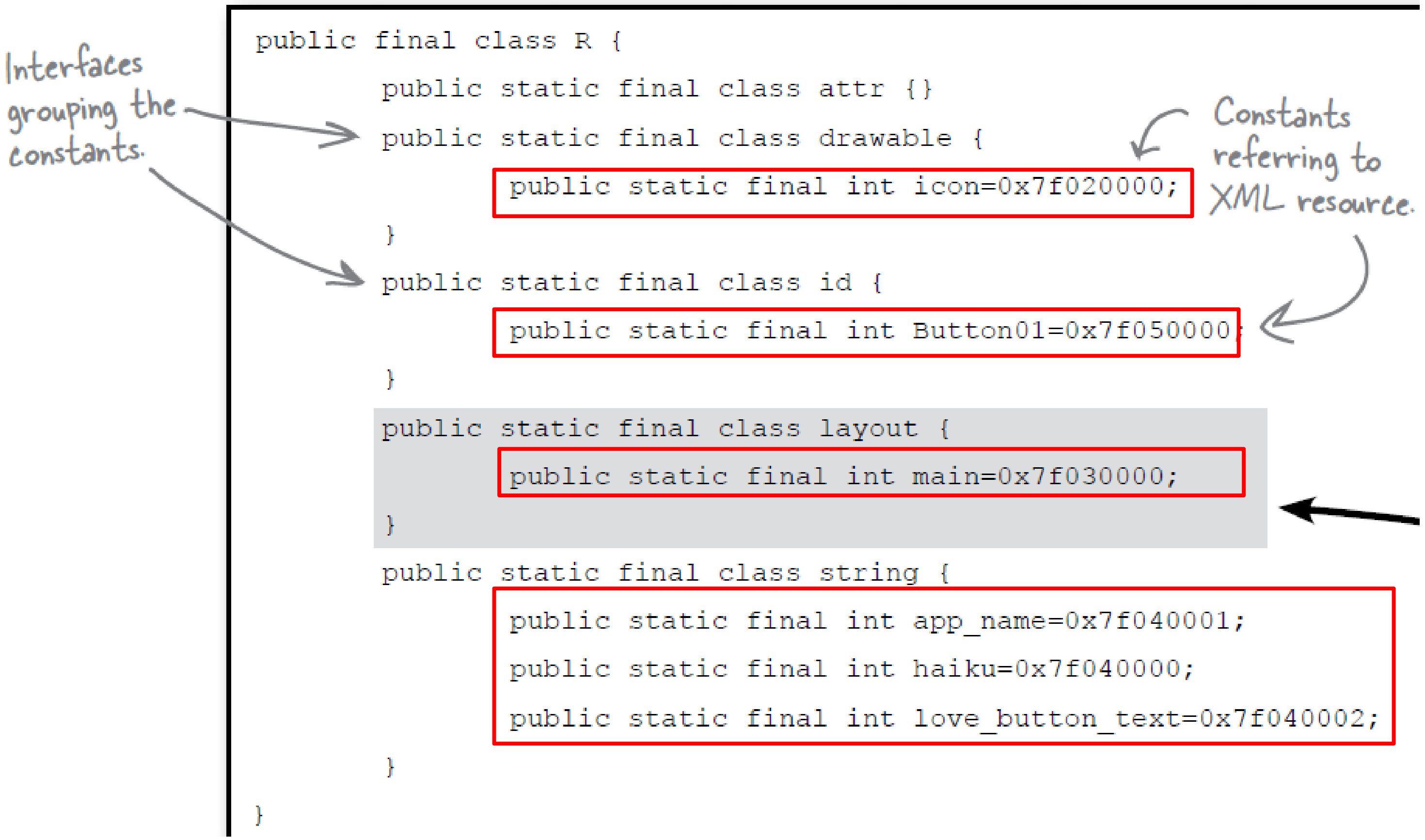
```
<Button
   android:id="@+id/true_button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/true_button" />
```

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

Need reference to Buttons

R.Java Constants

- During compilation, XML resources (drawables, layouts, strings, views with IDs, etc) are assigned constants
- Sample R.Java file



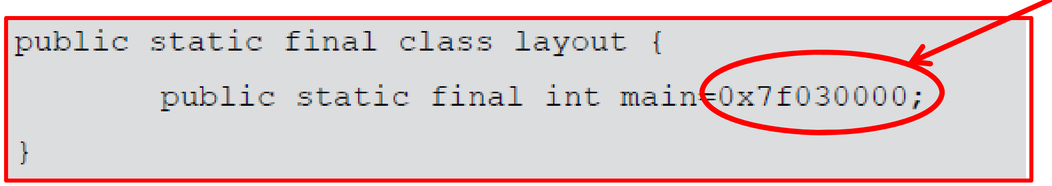
```
public final class R {
        public static final class attr {}
        public static final class drawable {
                public static final int icon=0x7f020000;
        }
        public static final class id {
                public static final int Button01=0x7f050000;
        }
        public static final class layout {
                public static final int main=0x7f030000;
        }
        public static final class string {
                public static final int app_name=0x7f040001;
                public static final int haiku=0x7f040000;
                public static final int love_button_text=0x7f040002;
        }
}
```

Interfaces grouping the constants.

Constants referring to XML resource.

Referring to Resources in Java File

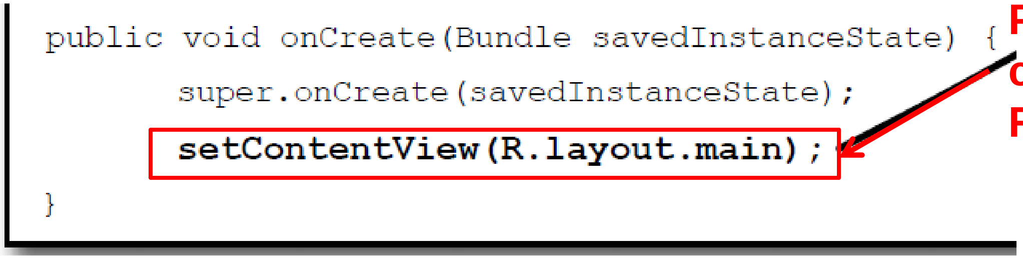
- Can refer to resources in Java file using these constants
- Example



```
public static final class layout {
        public static final int main=0x7f030000;
}
```

Constant assigned to R.layout.main at runtime

- In java file, R.java the constant corresponding to main.xml is argument of setContentView



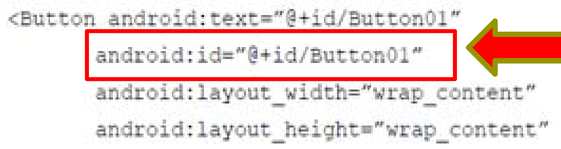
```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
}
```

Pass in layout file as constant assigned to R.layout.main

Referencing Widgets by ID

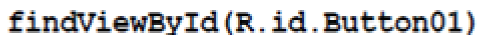
- To reference a widget in Java code, you need its **android:id**

In XML file, give the widget/view an ID i.e. assign android:id



```
<Button android:text=”@+id/Button01”
        android:id=”@+id/Button01”
        android:layout_width=”wrap_content”
        android:layout_height=”wrap_content”
```

In java file, to reference/manipulate view/widget use its ID to find it (call findviewbyID( ))

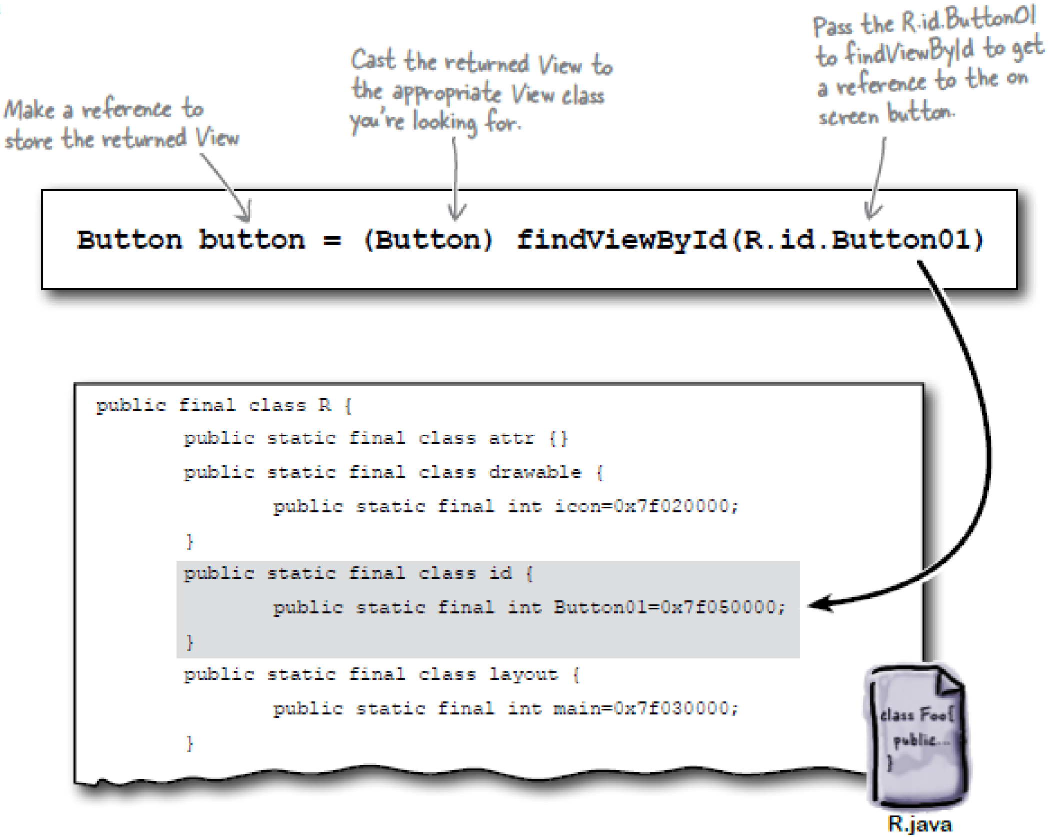


```
findViewById(R.id.Button01)
```

Getting View References

- **findViewById** method is part of Activity class so it can be called in our java file (e.g. MainActivity.java)

- A generic view is returned (not subclasses e.g. buttons, TextView), so needs to cast



Make a reference to store the returned View

Cast the returned View to the appropriate View class you're looking for.

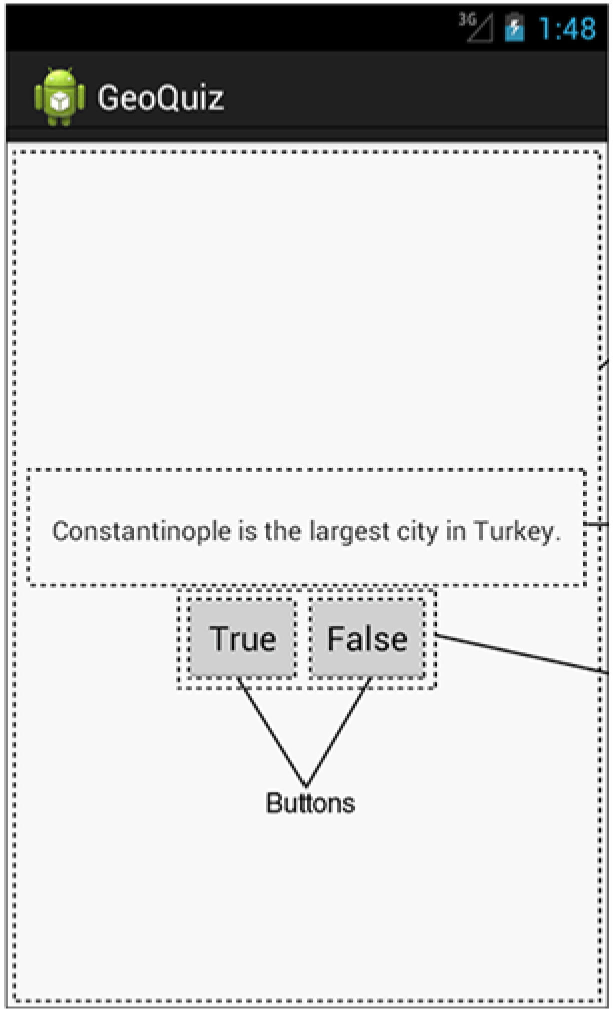
Pass the R.id.Button01 to findViewById to get a reference to the on screen button.

```
Button button = (Button) findViewById(R.id.Button01)
```

```
public final class R {
        public static final class attr {}
        public static final class drawable {
                public static final int icon=0x7f020000;
        }
        public static final class id {
                public static final int Button01=0x7f050000;
        }
        public static final class layout {
                public static final int main=0x7f030000;
        }
```

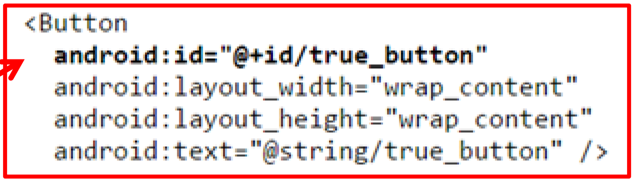
QuizActivity.java: Getting References to Buttons

- To get reference to buttons in java code

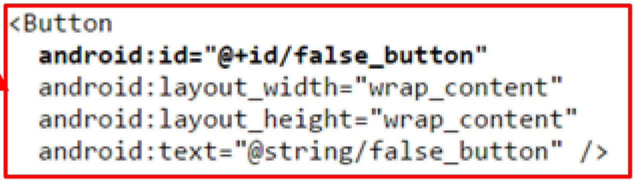


```
public class QuizActivity extends Activity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mFalseButton = (Button)findViewById(R.id.false_button);
    }

...
}
```

Declaration in XML



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



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

QuizActivity.java: Setting Listeners

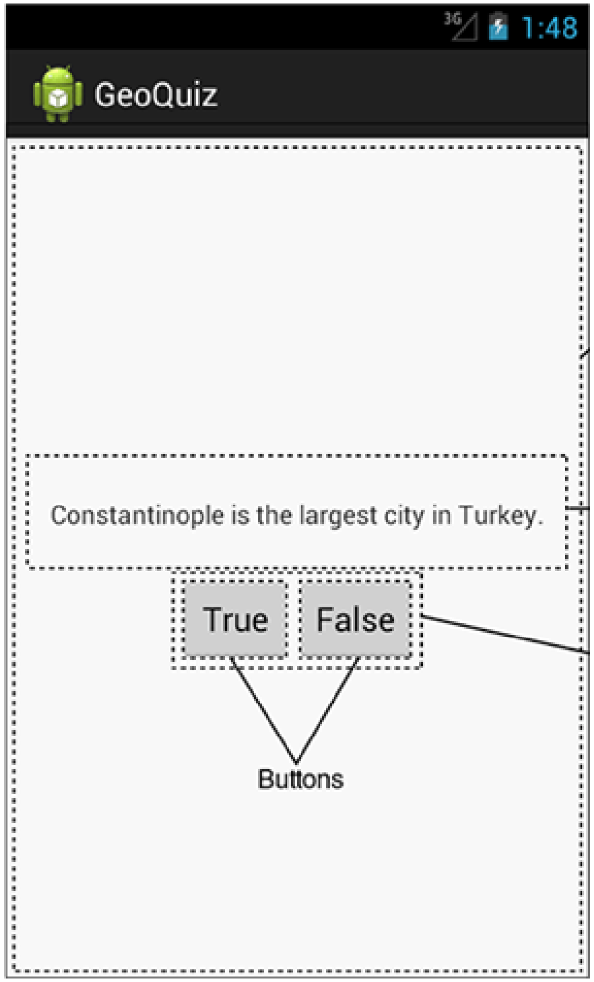
- Set listeners for **True** and **False** button

```
...

    mTrueButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Does nothing yet, but soon!
        }
    });

    mFalseButton = (Button)findViewById(R.id.false_button);
    mFalseButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Does nothing yet, but soon!
        }
    });
}
```

1.Set Listener Object
For mTrueButton

3. Overide onClick method
(insert your code to do
whatever you want as
mouse response here)

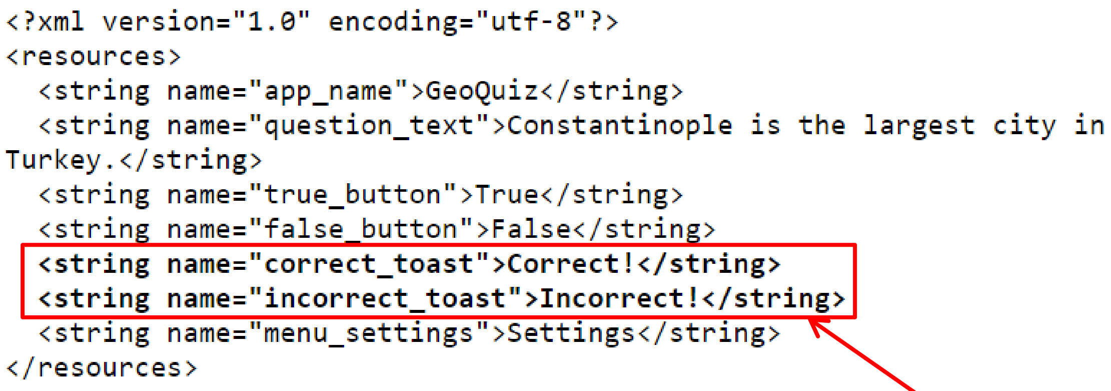
2. Create listener
object as anonymous
(unnamed) inner object



Constantinople is the largest city in Turkey.

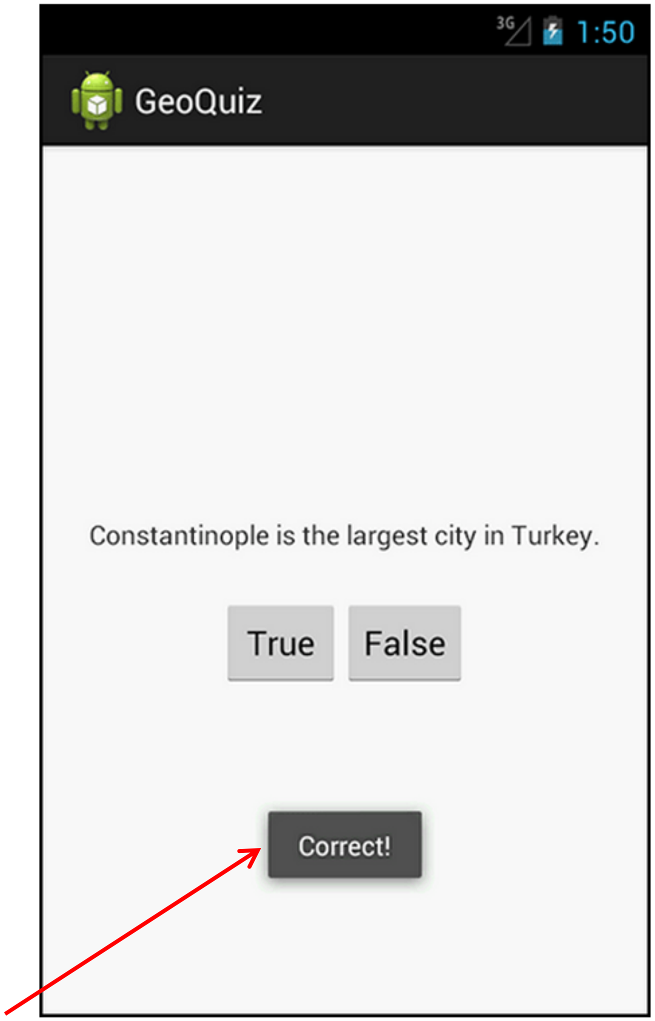
QuizActivity.java: Adding a Toast

- A toast is a short pop-up message
- After user clicks True or False button, our app will pop-up a toast to inform the user if they were right or wrong
- First, we need to add toast strings (Correct, Incorrect) to strings.xml



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">GeoQuiz</string>
  <string name="question_text">Constantinople is the largest city in
Turkey.</string>
  <string name="true_button">True</string>
  <string name="false_button">False</string>
  <string name="correct_toast">Correct!</string>
  <string name="incorrect_toast">Incorrect!</string>
  <string name="menu_settings">Settings</string>
</resources>
```

A toast



Constantinople is the largest city in Turkey.

QuizActivity.java: Adding a Toast

- To create a toast, call the method:

```
public static Toast makeText(Context context, int resId, int duration)
```

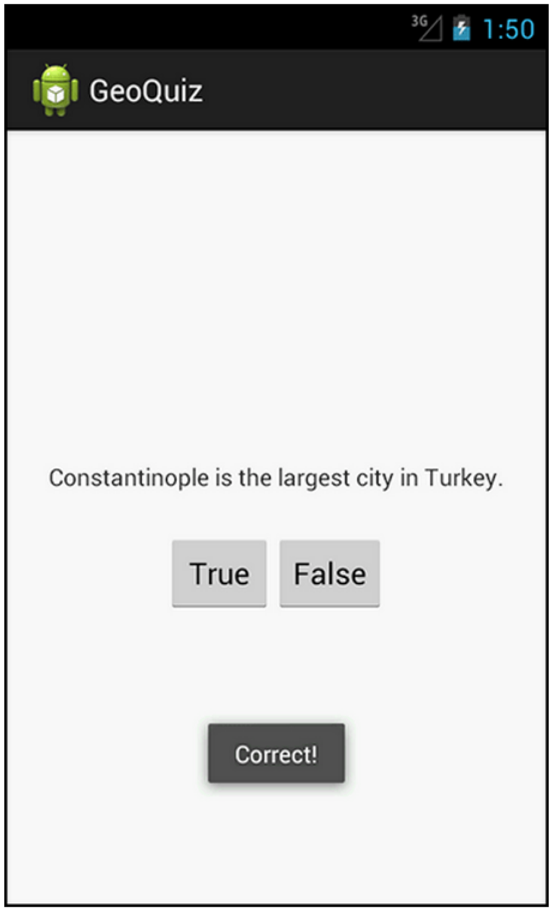
Instance of Activity (Activity is a subclass of context)

Resouce ID of the string that toast should display

Constant to specifiy how long toast should be visible

- After creating toast, call **toast.show( )** to display it
- For example to add a toast to our onClick( ) method:

```
public void onClick(View v) {
    Toast.makeText(QuizActivity.this,
                   R.string.incorrect_toast,
                   Toast.LENGTH_SHORT).show();
}
```



Constantinople is the largest city in Turkey.

QuizActivity.java: Adding a Toast

- Code for adding a toast

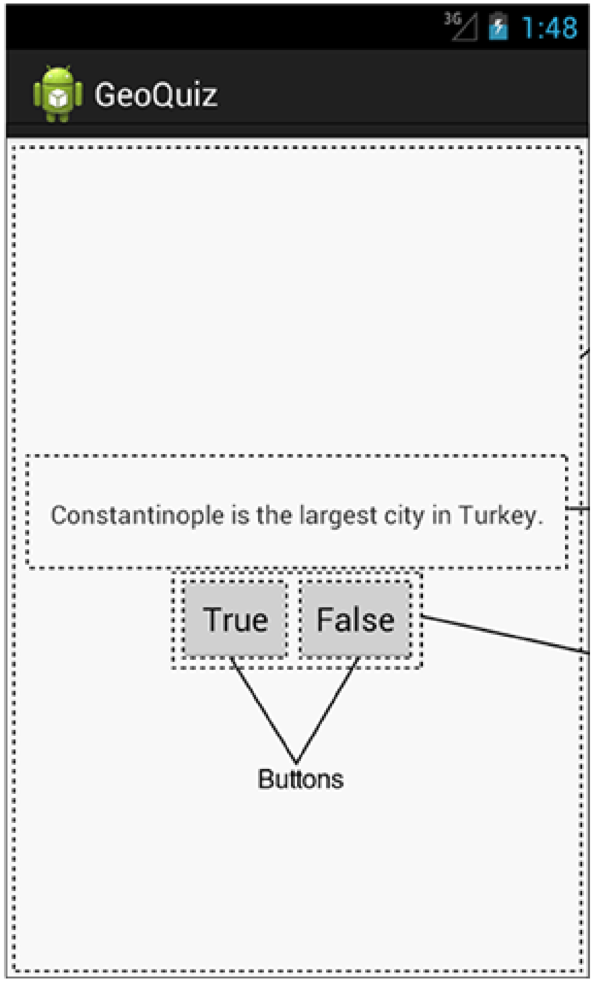
```
...
mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
                       R.string.incorrect_toast,
                       Toast.LENGTH_SHORT).show();
    }
});
```

```
mFalseButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(QuizActivity.this,
                       R.string.correct_toast,
                       Toast.LENGTH_SHORT).show();
    }
});
```

1.Set Listener Object For mTrueButton

3. Overide onClick method Make a toast

2. Create listener object as anonymous innner object



Constantinople is the largest city in Turkey.

QuizActivity.java: Complete Listing

```
package com.bignerdranch.android.geoquiz;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class QuizActivity extends Activity {

    Button mTrueButton;
    Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);

        mTrueButton = (Button)findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                        R.string.incorrect_toast, Toast.LENGTH_SHORT)
                        .show();
            }
        });
```

QuizActivity.java: Complete Listing (Contd)

```
        mFalseButton = (Button)findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Toast.makeText(QuizActivity.this,
                        R.string.correct_toast, Toast.LENGTH_SHORT)
                        .show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

     // Inflate the menu;
     // this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.activity_quiz, menu);
        return true;
    }

}
```

Used if app has an
Action bar menu

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