

http://www.android.com/

#### open handset alliance

UI (User Interface) overview Supporting Multiple Screens Touch events and listeners

## User Interface Layout



- The Android user interface (UI) consists of screen views (one or more viewgroups of widgets), screen touch events, and key presses
- Resource directories the /app/res/\* folders in the project view
- General UI attributes Each UI object has 3 definable attributes that customize the look and feel of the UI
  - The dimension of the object: dp (density independent pixels), px, in, mm, pt and sp (scale independent pixels – as dp but scaled to font preference)
  - Text in the object (supports formatted strings, html and special chars)
  - The color of the object
- Alternate resources/localization (use the two first letters in language)
  - res/values-sv/strings.xml, res/values-da/strings.xml etc.
  - Missing values will fall back to the default res/values/strings.xml
- Multiple screen pixel densities (pixel count/screen size) in dpi (dots per inch)
  - res/drawable-x: ldpi(120), mdpi(160), hdpi(240), xhdpi(320), xxhdpi(480) and xxxhdpi(640)
  - Android at run-time determines the closest one to use and scales them
  - res/drawable is used for bitmaps which should not scale

## Layouts and attributes



- The user interface for Activities is defined via layouts. The layout defines the UI elements, their properties and their arrangement
  - A layout can be defined by XML, via Java code or a mix of both
  - The XML way is preferred since it is more flexible and easy to extend, customize and change - compared to using Java code during run-time
- These definitions are placed in the <projname>/res/layout/main.xml file and connected to the user interface if needed
  - Usually there is one /res/layout/<file>.xml file for every Activity screen
  - Symbol @ means that string should be expanded by the XML parser
  - Symbol + means that a unique "id" should be created in the name space

Resource	Reference in Java	Reference in XML
res/layout/main.xml	R.layout.main	@layout/main
res/drawable-hdpi/icon.png	R.drawable.icon	@drawable/icon
@+id/home_button	R.id.home_button	@id/home_button
<string name="hello"></string>	R.string.hello	@string/hello



http://developer.android.com/guide/topics/resources/index.html

- Android run on many devivces and in many regions. To reach the most users, your application should handle text, audio files, numbers, currency, and graphics in ways appropriate to the locales and screen size etc.
- The default resources are required and should define every string etc.

The text strings in res/values/strings.xml should use the default language, which is the language that you expect most of your application's users to speak. res/values/strings.xml (required directory) The default resource set must also include any default drawables and layouts, and can include other types of resources such as animations. res/drawable/(required directory holding at least one graphic file?) res/layout/ (required directory holding an XML file that defines the default layout) res/anim/ (required if you have any res/anim-<qualifiers> folders) res/xml/ (required if you have any res/xml-<qualifiers> folders) res/raw/ (required if you have any res/raw-<qualifiers> folders)

- When the user runs your program the Android system selects which resources to load, based on the device's locale etc.
  - If not found or partially not found it will fallback to the default resources
- Examples for lang, dimensions, orientation and styles etc.

res/layout-se/main.xml, res/values-se/strings.xml

res/values-ldpi/dimens.xml, res/values-9/styles.xml



## Views and ViewGroups



- The basic building block of a graphical layout is a View
- Each View is described by a View Object, which is responsible for drawing a rectangular area and handling events in that area
- The View is a base class for objects that interact with the user; they are called widgets
- A ViewGroup Object is a type of View that acts as a container to hold multiple Views (or other ViewGroups)



## Layout and Views 1





#### Layouts, resources and code [ 🕵 📜 AIN ACTIVITY

- The res/values/strings.xml normally contains all predefined strings in the project
- Activitys onCreate(), strings.xml and main.xml

```
Click To Launch Secondary Activity
public class MainActivity extends Activity
implements View.OnClickListener
                                        <?xml version="1.0" encoding="utf-8"?>
                                        <resources>
   private EditText mEditText1;
                                              <string name="hello">Hello World, MainActvity!</string>
   private Button mButton1;
                                              <string name="app name">MainActvity</string>
                                              <string name="mainactivity">MAIN ACTIVITY</string>
/** Called when the activity is first
                                        <string name="btnLaunchSecAct">Click To Launch Secondary Activity</string>
created. */
                                        </resources>
Override
public void onCreate(Bundle
                                             <?xml version="1.0" encoding="utf-8"?>
savedInstanceState)
                                             <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android</pre>
                                                 android: orientation="vertical"
   super.onCreate(savedInstanceState);
                                                 android:layout width="fill parent"
   setContentView(R.layout.main);
                                                 android:layout height="fill parent">
                                                 <TextView android:layout width="fill parent"
   mEditText1 =
                                                     android:layout height="wrap content"
(EditText) findViewById (R.id.editText1);
                                                                                                   main.xml
                                                     android:text="@string/mainactivity"/>
                                                 <EditText android:id="@+id/editText1"
   mButton1 =
                                                     android:layout width="fill parent"
(Button) findViewById (R.id.button1);
                                                     android: lavout height="wrap content"/>
                                                 <Button android:id="@+id/button1"
   mButton1.setOnClickListener(this);
                                                     android:layout width="fill parent"
}
                                                     android:layout height="wrap content"
11 . . .
                                                    android:text="@string/btnLaunchSecAct"/
                                             </LinearLayout>
```

# Layouts and Views 2 🏾 🕺 🛒 🐓

☐ main.xml ⊠		- 8
Editing config: default		Any locale 🔻 Android 4.0.3 👻 Create
2.7in QVGA    Portrait	▼ Normal ▼ Day time ▼ Theme	•
Palette		e $e$ $q$ $ $ $e$ $e$
🗀 Form Widgets		*
🗀 Text Fields		Almost
🗁 Layouts	MainActvity	everything
🔢 GridLayout 🧮 LinearLayout (Vertical)	MAIN ACTIVITY	everytillig
LinearLayout (Horizontal)		else is
📴 RelativeLayout 🔲 FrameLayout	Click To Launch Secondary Activity	Views/Midaets
🔄 Include Other Layout 🛛 😨 Fragment		views/viugets
🧱 TableLayout 📰 TableRow 🛤 Space	LinearLay	out
		_
🗀 Composite	RelativeLayout View	View
🗀 Images & Media	LinearLayout. LayoutParams	arLayout. utParams LayoutParams
🗀 Time & Date		
Transitions		
C Advanced	View View View	
Custom & Library Views	RelativeLayout. LayoutParam LayoutParam LayoutParam	Param
📰 Graphical Layout [ 🗐 main.xml		

## Layouts and Views examples 👲 🐔 🖆

#### Layouts

#### In general a view uses an Adapter to bind data to its layout

Linear Layout	Relative Layout	Table Layout	Grid View	Tab Layout	List View
Hello LinearLayout	Hello RelativeLayout	Helio TableLayout	Hello GridView		Hello ListView
red green blue yellow	Type here:	Open Ctrl-O Save Ctrl-S		Artists Albums Songs	American Samoa
	Cancel OK	Save As Ctrl-Shift-S		This is the Artists tab	El Salvador
		X Import X Export Ctrl-E			Saint Helena
		Quit			Saint Kitts and Nevis
					Saint Lucia
row one					Saint Pierre and Miquelon
row two					Saint Vincent and the Grenadines
rowthroo					Samoa
row three					San Marino Sa
row four					Saudi Arabia

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

#### Widgets & Other Views

Date Picker	Time Picker	Form Stuff	<u>Spinner</u>	Auto Complete	Gallery
Helio DatePicker 84-2008 change the date Mon, 04 August, 2008 + + + 04 Aug 2008 - Set Cancel	Hello TimePicker 1937 change the time 6:37 PM + + 6 37 PM - PM Set Cancel	Helio Form Stuff	Hello Spinner  Please select a planet: Mars	Hello auto complete         Country:       ca         Cambodia       Cameroon         Canada       Cape Verde         Cayman Islands       New Caledonia         Turks and Caicos Islands       Turks and Caicos Islands	Helio Gallery

# Controlling the Width and Height of UI Elements



- Each View object must specify a total width android:layout\_width and total height android:layout\_height in one of three ways
  - exact dimension Provides control, but does not scale to multiple screen types well
  - wrap\_content Just big enough to enclose the contents of the element plus padding
  - match\_parent (named fill\_parent before API 8) Size maximized to fill the element's parent, minus padding
- Margins (inside of an UI element border) is part of the size and Padding (outside of the UI element border) is specified as dp, it can be specified using one of two types of attributes
  - All Sets margins/padding equal on all four sides of an element
  - Left, Right, Top, Bottom Sets margin/padding on each side of an element separately
- Another attribute is android:layout\_weight, which can be assigned a number. It provides the Android system with a way to determine relative importance between different elements of a layout
  - Note! layout\_width or layout\_height should be set to "0" in this case

# Controlling W & H cont.

In the fourth row/tag below (Layout5), all buttons use layout\_width="0dp", also add **layout\_weight** and assign it the same value for all buttons

This gives the most satisfying layout!



<:Xmi Version- 1.0 encouring- ucr 0 :/
<linearlayout android:id="@+id/LinearLayout1 android:orientation=" td="" vertical"<=""></linearlayout>
<pre>xmlns:android="http://schemas.android.com/apk/res/android"</pre>
android:layout_width="fill_parent" android:layout_height="fill_parent">
<linearlayout <="" android:id="@+id/linearLayout2" td=""></linearlayout>
android:layout_width="fill_parent" android:layout_height="wrap_content">
<button <="" android:layout_width="wrap_content" android:text="add" td=""></button>
<pre>android:layout_height="wrap_content" /&gt;</pre>
<button <="" android:layout_width="wrap_content" android:text="subtract" td=""></button>
android:layout_height="wrap_content" />
<button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button>
android:layout_height="wrap_content" />
<button <="" android:layout_width="wrap_content" android:text="divide" td=""></button>
android:layout_height="wrap_content" />
<linearlayout <="" android:id="@+id/linearLayout3" td=""></linearlayout>
android:layout_width="fill_parent" android:layout_height="wrap_content">
<button <="" android:layout_width="wrap_content" android:text="add" td=""></button>
android:layout_height="wrap_content" />
<button <="" android:layout_width="wrap_content" android:text="subtract" td=""></button>
android:layout_height="wrap_content" />
<button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button>
android:layout_height="wrap_content" />
<button <="" android:layout_width="fill_parent" android:text="divide" td=""></button>
<pre>android:iayout_neight= wrap_concent /&gt; </pre>
<pre></pre>
android:layout width="fill parent" android:layout height="wrap content"
<pre>android.tayout_width= fift_patent android.tayout_nerght= wiap_content /</pre>
android:layout_beight="wrap_content"_android:paddingBight="20sp"/>
<button <="" android:layout="" android:text="subtract" td="" width="wrap content"></button>
(baccon anarora.cexe baberaet anarora.rayoue_wrach wrap_concent
android:layout_beight="wran_content"_android:paddingRight="20sp"/>
android:layout_height="wrap_content" android:paddingRight="20sp"/>
android:layout_height="wrap_content" android:paddingRight="20sp"/> <button <br="" android:layout_width="wrap_content" android:text="multiply">android:layout_height="wrap_content" android:paddingRight="20sp"/&gt;</button>
android:layout_height="wrap_content" android:paddingRight="20sp"/> <button <br="" android:layout_width="wrap_content" android:text="multiply">android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button_android:text="divide" <="" android:layout_width="wrap_content" td=""></button_android:text="divide"></button>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt;</pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt;</pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt;</pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>
<pre>android:layout_height="wrap_content" android:paddingRight="20sp"/&gt; <button <="" android:layout_width="wrap_content" android:text="multiply" td=""></button></pre>

### Layout Inspector (Hierarchy Viewer)



#### View the layout in a program. Only works with the current app in emulator. Above AS 2.2 - Tools > Android > $\bigcirc$

http://tools.android.com/tech-docs/layout-inspector



### Set Relative Layout and Layout ID 🛛 👘 👾 👘

Sometimes it is more convenient to set the layout relative to a starting object or parent object rather than absolute rules

Also, if the UI starts nesting LinearLayouts, it might be simpler to use relative layouts. This can be done using a <u>RelativeLayout view</u>

	Type here:		below: @id/entry alignParentRight: true
/pe here:	below: @id/label	Cancel	
Cancel OK		Calleer	
Cancer OK	Тур	pe here:	toLeftOf: @id/ok alignTop: @id/ok
		Ca	ancel OK
<pre><?xml version="1.0" encoding="utf-8"?> <relativelayout fill_parent")<="" pre="" xmlns:android="http:// android:layout_width="></relativelayout></pre>	/schemas.android.com/apk/res/andr nt" android:layout_height="wrap_o	oid" content" android:padding="1	0px">
<textview android:id="@+id/labe&lt;br&gt;android:layout_height=" th="" wr<=""><th>l" android:layout_width="fill_par cap_content" android:text="Type h</th><th>rent" S</th><th>Should be dp and sp</th></textview>	l" android:layout_width="fill_par cap_content" android:text="Type h	rent" S	Should be dp and sp
<edittext android:id="@+id/entry&lt;br&gt;android:layout_height=" wr<br="">android:layout_below="@ic</edittext>	y" android:layout_width="fill_par cap_content" android:background=" d/label" />	rent" '@android:drawable/editbox_k	for text size
<button and<br="" android:id="@+id/ok">android:layout_height="wr android:layout_alignParen</button>	droid:layout_width="wrap_content" rap_content" android:layout_margi tRight="true" android:layout_bel	" nLeft="10px" android:text=' .ow="@id/entry" />	"OK"
<button android:layout_width="wi&lt;br&gt;android:layout_height=" th="" wr<=""><td><pre>rap_content" rap_content" rap_content" android:text="Cancel</pre></td><td></td><td></td></button>	<pre>rap_content" rap_content" rap_content" android:text="Cancel</pre>		
android:layout_toLeftOf=" 	'&id/ok" android:layout_alignTop=	"@id/ok" />	

## Some rules for children in a Relative Layout



Relative Layout Rule	XML Attribute (All Start with the android: Tag)	Java Constant
Align this view's edge	layout_above	ABOVE
relative to anchor view's	layout_below	BELÓW
edge	layout_toRightOf	RIGHT_OF
<pre>android:layout_below="@id/TextView1"</pre>	layout_toLeftOf	LEFT_OF
Align this view's edge	layout_alignTop	ALIGN_TOP
with anchor view's edge	layout_alignBottom	ALIGN_BOTTOM
	layout_alignRight	ALIGN_RIGHT
	layout_alignLeft	ALIGN_LEFT
Align this view's text baseline with anchor view's text baseline	layout_alignBaseline	ALIGN_BASELINE
Align this view's edge	layout_alignParentTop	ALIGN_PARENT_TOP
with parent view's edge	layout_alignParentBottom	ALIGN_PARENT_BOTTOM
	layout_alignParentRight	ALIGN_PARENT_RIGHT
	layout_alignParentLeft	ALIGN_PARENT_LEFT
Center this view within	layout_centerInParent	CENTER_IN_PARENT
parent	layout_centerHorizontal	CENTER_HORIZONTAL
	layout centerVertical	CENTER VERTICAL

## **Constraint Layout**



- Allows you to create large and complex layouts with a flat view hierarchy by adding vertical and horizontal constraints. Loads faster than traditional layouts
- It is similar to RelativeLayout but more flexible and easier to use. With chains (spread, spread inside, weighted and packed) advanced views are possible
- Training/examples: https://developer.android.com/training/constraint-layout/



## Useful UI (TextView) Attributes



Dessible and

<TextView android:text="@string/myTextString" android:id="@+id/my\_text\_label" android:background="@android:drawable/editbox\_background" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="48sp" />

#### Dimension

#### **Possible Values**

Any number followed by one of the following dimensions:

px-Actual pixels on the screen

dp (or dip)—Device-independent pixels relative to a 160dpi screen
 sp—Device-independent pixels scaled by user's font size preference
 in—Inches based on physical screen size
 mm—Millimeters based on physical screen size
 pt—1/72 of an inch based on physical screen size

#### Color

Possible values are a 12-bit color #rgb, 16-bit color with alpha opacity #argb, 24bit color #rrggbb, or 32-bit color with alpha opacity #aarrggbb. It is also possible to utilize the predefined colors in the Color class within Java files, such as Color.CYAN, Color.GREEN, etc.

#### **Default Values in Bold in the Last Column**

TextView Attribute	XML Element	Java Method	Default Values
Display string	android:text	setText(CharSequence)	Any string
Font size	android:textSize	setTextSize(float)	Any dimension
Font color	android:textColor	setTextColor(int)	Any color
Background color	N/A	setBackgroundColor(int)	Any color
Font style	android:textStyle	setTypeface(Typeface)	bold italic bold italic
Font type	android:typeface	setTypeface(Typeface)	normal sans serif monospace
Text placement in display area	android:gravity	setGravity(int)	top bottom <b>left</b> right

(more...)

# Layouts and Views tips

http://developer.android.com/guide/topics/manifest/activity-element.html



• To force single instance or single task mode - set the following for the Activity in the MAIN/LAUNCHER intent-filter in androidmanifest.xml

android:launchMode="singleInstance"
android:launchMode="singleTask"
android:launchMode="singleTop"

The "singleInstance" mode is identical to "singleTask" except if "singleInstance" starts another activity, that activity will be launched into a different task regardless of its launch mode.

- Retain the task state return to the last state at re-launch android:alwaysRetainTaskState="true"
- To force screen orientation add to the Activity element android:screenOrientation="portrait" android:screenOrientation="landscape"
- Try out the Api Demos app in the emulator to examine all possible GUI combinations/controls source at:
  - https://android.googlesource.com/platform/development/ > version > samples
- User Interface Guidelines for the interaction and visual design of Android applications
  - New: http://developer.android.com/design/index.html
- http://petrnohejl.github.io/Android-Cheatsheet-For-Graphic-Designers/

### Supporting Multiple Screen Sizes 1

http://developer.android.com/guide/practices/screens\_support.html



- Density-independent pixel (dp) A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way (dpi/ppi = dots/pixels per inch)
  - − For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels  $\rightarrow$  pixels per inch = dp \* (dpi / 160). For a xxhdpi (3.0x) screen it is 3 pixels
- Layout-small, layout-large or layout-xlarge is deprecated
- Use smallestWidth sw<N>dp after Android 3.2 note dp
  - For example: res/layout-sw600dp/ defines the smallest available width required by your layout resources



### Supporting Multiple Screen Sizes 2

http://developer.android.com/guide/practices/screens support.html



- We also have w<N>dp and h<N>dp which is minimum available width and height in dp units
- Example 800x1280px 7" tablet  $\rightarrow$  215,6 ppi falls into hdpi (1,5x)
  - We should use 800/1,5 = sw533dp, but if we set layout-sw480dp we handle 720x1280 pixel tablets as well
- 1080x1920px 5,2" phone  $\rightarrow$  423,6 ppi falls into xxhdpi (3.0x)
  - 1080/3 = 360 so if we would like another layout on this phone we must set layout-sw360dp
- We could also use the w<N>dp (1920/3 = 640), setting layoutw600dp gives any screen with 600dp available width  $d_p = \sqrt{w_p^2 + h_p^2}$ or more the desired layout whether the device is in 2. Calculate PPI: landscape or portrait orientation

Calculation of screen PPI (Pixels Per Inch)

- d<sub>p</sub> is diagonal resolution in pixels
- *w*<sub>p</sub> is width resolution in pixels
- h<sub>n</sub> is height resolution in pixels
- d<sub>i</sub> is diagonal size in inches (this is the number advertised as the size of the display).

Calculate diagonal resolution in pixels using the Pythagorean theorem:

$$PPI = \frac{d_p}{d_i}$$

where

#### Supporting Multiple Screen Sizes Wrap-up



- Get and set the density
  - From a command shell get the density with: adb shell wm density
  - To set the density just put a number after as: adb shell wm density 320
  - With Android Nougat users can control the DPI under Settings > Display > Display size
- Calculate the density
  - The physical density of a 5.2 inch screen with 1920 x 1080 is 480
    - This translates into: 480/160 = xxhdpi 3
  - The formula is actual-dpi / 160. (Everything is scaled to 160 dpi)
  - To get the physical density = sqrt((wp \* wp) + (hp \* hp)) / di
  - Where:
    - wp is width resolution in pixels, hp is height resolution in pixels, and di is diagonal size in inches. Example a 8.3 inch tablet with 1920 x 1200 pixels
    - (1200<sup>2</sup> + 1920<sup>2</sup>)<sup>0.5</sup> = 2265 / 8.3 = 273 dpi or ppi
    - 273/160 = 1.7. Will fall into xhdpi 2 or hdpi 1.5 (1200/2 = sw-600dp)

		\$ ♥∡  🗎 7:	09 🖬		3 ♥⊿	7:09	-		* 1	7:
Setti	ngs	۹	: ≡	Display		:		Display size		
•	Wi-Fi 'tsunami'		Slee After	p 5 minutes of inactivity					Hi Pete!	
*	Bluetooth Disconnected		Pres Quick scree	is power button twice for kly open camera without ur en	or camera nlocking your	٠	P	Hey, want to grab	Tue 6:00PM	
0	Data usage 9.82 MB of data used		Scre	en saver k				catch up today? Tue 6:01PM		
	More		Amb	vient display e screen when you pick up	device or receive	•		Sounds great. I kn place not too far f Tue 6:02PM	ow of a good rom here.	A
Device			notifi	cations						
ø	Display Adaptive brightness is ON		Font Defai	i size ult			Pr	Perfect!		
	Notifications All apps allowed to send		Disp Defa	ilay size ult				•••		
	Sound Ringer volume at 43%		Whe	n device is rotated te the contents of the scree	20		-	Defa	ult	• +
ă	Apps		Cast	t			Make Some	the items on your scr apps on your screen	een smaller or la may change posi	rger. tion.
				⊲ 0				⊲ C		

### Handle View touch events 1



 There are four different ways to add listeners for handling touch events (button onClick and onLongClick)

```
Inner Class (btn1)
Qoverride
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
                                                                               Anonymous Inner Class (btn2)
     // method 1 - uses an inner class named btn1Listener...
     Button btn1 = (Button)findViewById(R.id.Button01);
     btn1.setOnClickListener(btn1Listener);
                                                                              Implementing an Interface (btn3)
     // method 2 - use an anonymous inner class as a listener...
     Button btn2 = (Button) findViewById(R.id.Button02);
                                                                               Calling From XML Layout (btn4)
     btn2.setOnClickListener(new View.OnClickListener() {
          QOverride
          public void onClick(View v) {
               showToastMessage("You clicked btn2 - uses an anonymous inner class");
     });
     // method 3 - note that this activity implements the View.OnClickListener interface
     // which means that we must implement the onClick() method (which you'll find below)..
     Button btn3 = (Button) findViewById(R.id.Button03);
     btn3.setOnClickListener(this);
     // method 4 - look at the method btn4Listener() below
```

### Handle View touch events 2



See the ButtonClickTest project

```
// Method 1 - here's the inner class used as a listener for btn1...
private View.OnClickListener btn1Listener = new View.OnClickListener() {
     Override
    public void onClick(View v) {
          showToastMessage("You clicked btn1 - uses an inner class named btn1Listener");
};
// Method 3 - here's a method that you must have when your activity implements the
// public class ButtonClickTest extends Activity implements View.OnClickListener interface...
Qoverride
public void onClick(View v) {
     showToastMessage("you clicked on a btn3, which uses this Activity as the listener");
// Method 4 - here's the handler for btn4 (declared in the XML layout file, btn4 properties)
// note: this method (On Click property) only works with android api level 7 and higher,
// it must be public and must take a single parameter which is a View
public void btn4Listener(View v) {
     showToastMessage("You clicked btn4 - listener was set up in the XML layout");
private void showToastMessage(String msg) {
     Toast toast = Toast.makeText(this /*getApplicatonContext()*/, msg, Toast.LENGTH SHORT);
     toast.show();
```

### Handle View touch events 3



• Examples using the (View v) parameter

```
// working with just one clicklistener and if you have declared
// b*.setOnClickListener(myOnlyhandler);, you can do...
private View.OnClickListener myOnlyhandler = new View.OnClickListener() {
    public void onClick(View v) {
         if( button1.getId() == ((Button)v).getId() ){
              // it was the first button
         }
         else if( button2.getId() == ((Button)v).getId() ){
              // it was the second button
         }
    }
}
// if you defined android:onClick="ButtonOnClick" for all buttons in your XML, you can do...
public void ButtonOnClick(View v) {
    switch (v.getId()) {
       case R.id.button1:
         doSomething1();
         break;
       case R.id.button2:
         doSomething2();
         break;
     }
```

## Event Listeners interfaces and methods



- onClick() from View.OnClickListener
- onLongClick() from View.OnLongClickListener
- onFocusChange() from View.OnFocusChangeListener
- onKey() from View.OnKeyListener
  - Called when the view has focus and a key is pressed
- onTouch() From View.OnTouchListener gestures
- Methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction
- More about Views in the API documents
  - http://developer.android.com/reference/android/view/View.html

# The Application Context



- The class android.content.Context provides the connections to the current "context"
- As Activities and Services extends this class you can directly access the Context via
   Return the context of the single, global Application object of the
  - this (current Context)
  - getApplicationContext()

Return the context of the single, global Application object of the current process. This generally should only be used if you need a Context whose

This generally should only be used if you need a Context whose lifecycle is separate from the current context, that is tied to the lifetime of the process rather than the current component.

- Or explicitly use ClassName.this
- Using the this reference to current object is recommended but it is not always resolved correctly
  - For example in inner classes and anonymous inner classes as in the previous ButtonClickTest example
- The static abstract Context class is usually passed to the getSystemService() which allows to receive a manager object for different hardware parts in the system as Sensors etc.

```
String ns = Context.NOTIFICATION_SERVICE;
mNManager = (NotificationManager) getSystemService(ns);
```

## AS File > New > ...



- Or right click on an object anywhere in the AS environment (folders, files, UI, etc.)
- Wizards for most common stuff

. . .

Layouts,
 XML values file,
 Service,
 ContentProvider,

👧 Mapsproject - [C:\hjc	>\AndroidStudio\N	/apsproject] - Android Studio 2.1.	3
<u>File</u> <u>E</u> dit <u>V</u> iew <u>N</u> avigate	<u>C</u> ode Analy <u>z</u> e <u>R</u> e	efactor <u>B</u> uild R <u>u</u> n <u>T</u> ools VC <u>S W</u> i	ndow <u>H</u> elp
New Open Open <u>R</u> ecent	,	New Project Import Project Project from Version Control •	D D D D D D D D D D D D D D D D D D D
Close Project		New Module	
Se <u>t</u> tings	Ctrl+Alt+S	Import Module	
Project Structure	Ctrl+Alt+Shift+S	Import Sample	
Other Settings	•	C Java Class	
Import Settings		Android resource file	
Export Settings		Android resource directory	
Settings Repository		File	
📙 <u>S</u> ave All	Ctrl+S	Package	
Synchronize	Ctrl+Alt+Y	S C++ Class	
Invalidate Caches / Re	estart	C/C++ Source File	
Export to <u>H</u> TML		C/C++ Header File	
Print		🖷 Image Asset	
Add to Favorites	•	🖷 Vector Asset	
File Encoding		Singleton	
Line Separators	•	Edit File Templates	Search Ever
Make File Read-only		📫 AIDL 🔸	
Power Save Mode		🛑 Activity 🔹	🖷 Gallery
Exit		🖷 Android Auto 🔹	Always On Wear Activity (Requires minSdk >= 20)
		ا <b>ب</b> Folder ۲۰	Android TV Activity
		Fragment	Basic Activity
		Google	Blank Wear Activity (Requires minSdk >= 20)
		•••• Other	Empty Activity
		Service	Fullscreen Activity
		Wear	Login Activity
		······································	
			Scrolling Activity
			Settings Activity
			Tabbeu Activity

### Eclipse File > New > Other... 👚 🛫 👘



- Wizards for most common stuff
  - Layouts, XML values file, Service, ContentProvider, ...

New	New Android Object
Select a wizard         Create an Android object such as a Service, an Activity, a Broadcast Receiver etc.         Wizards:         type filter text         ▷ ➢ General	Create Android Object Select which template to use
<ul> <li>Android</li> <li>Android Activity</li> <li>Android Application Project</li> <li>Android Icon Set</li> <li>Android Object</li> <li>Android Project from Existing Code</li> <li>Android Sample Project</li> <li>Android Test Project</li> <li>Android XML File</li> <li>Android XML Layout File</li> </ul>	SettingsActivity BroadcastReceiver ContentProvider CustomView Service
Android XML Values File	New Master/Detail Flow Creates a new master/detail flow, which is two columns on tablets, and one column on smaller screens. This creates a master fragment, detail fragment, and two activities.
A Back     Next >     Finish     [	A Back Next > Finish Cancel