CHAPTER 5. USER INTERFACES USING XML LAYOUTS

The View Class

- The View class represents the basic building block for user interface components.
- Each View occupies a rectangular area on the screen and is responsible for drawing and event handling.
- View is the base class for widgets, used to create interactive UI components (buttons, text fields, etc.).
- The ViewGroup subclass is the base class for layouts, which are invisible containers that hold other Views (or ViewGroups) and define their layout characteristics.

Using View

All views in a window are organized in a tree structure.

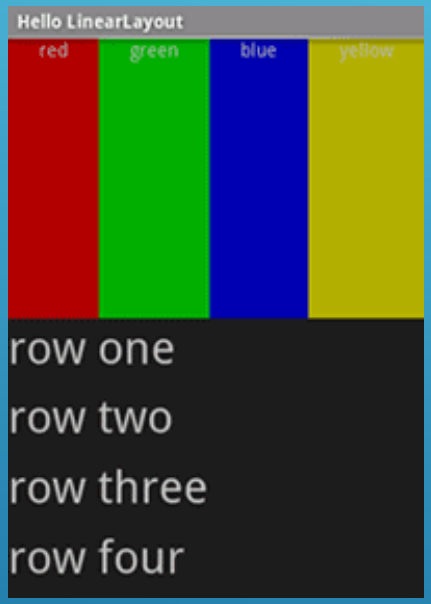
You can add views from the source code or define a tree structure of views in one or more XML layout files.

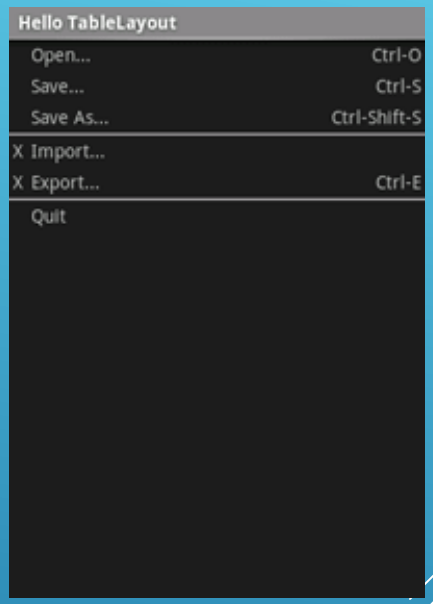
Once a view tree has been created, there are some operations that may need to be done :

1. **Set properties**: for example, pre-assign text lines in a TextView. Known properties can be preset in XML layout files.
2. **Set focus:** mechanism to move focus in response to user input. To request focus for a specific view, call requestFocus()..
3. **Set up listeners:** View allows to place listeners, these listeners are called when an event occurs for the view. For example, a Button uses a listener to listen to the button click event.
4. **Set visibility:** we can show or hide a view by using *setVisibility(int)*.

A brief sample of UI compone

Layouts





Linear Layout
A LinearLayout is a GroupView that will lay child View elements vertically or horizontally.

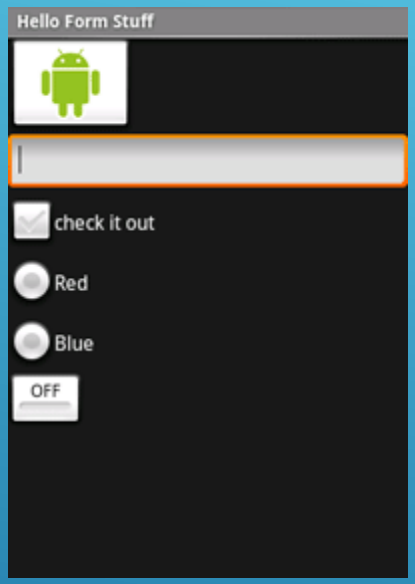
Relative Layout
A RelativeLayout is a ViewGroup that allows you to layout child elements in positions relative to the parent or siblings elements.

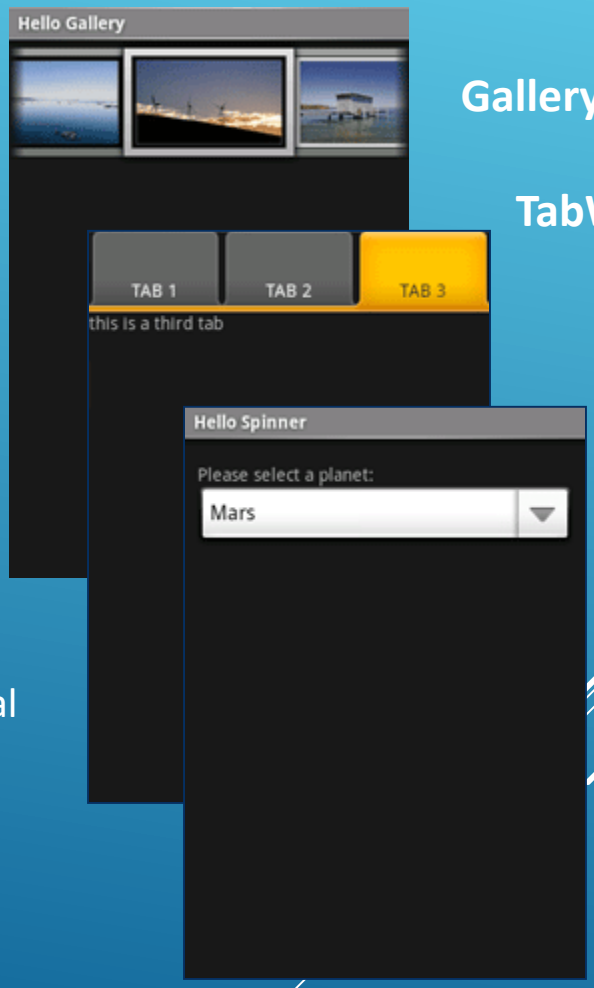
Table Layout
A TableLayout is a ViewGroup that will lay child View elements into rows and columns.

4

A brief sample of UI compone

Widgets





GalleryView

TabWidget

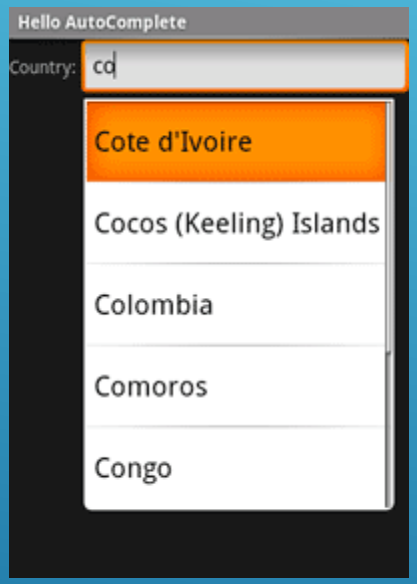
Spinner

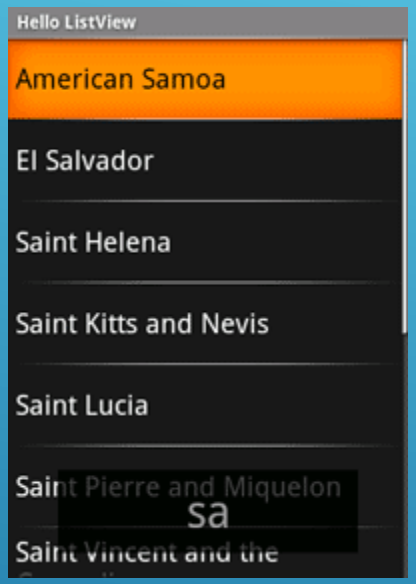
DatePicker
A *DatePicke* is a widget that allows the user to select a month, day and year.

Form Controls
Includes a variety of typical form widgets, like:
image buttons,
text fields,
checkboxes and
radio buttons.

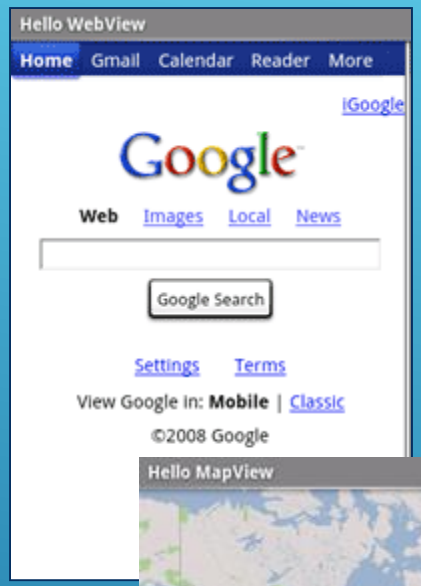
5

A brief sample of UI compone





WebView



MapView

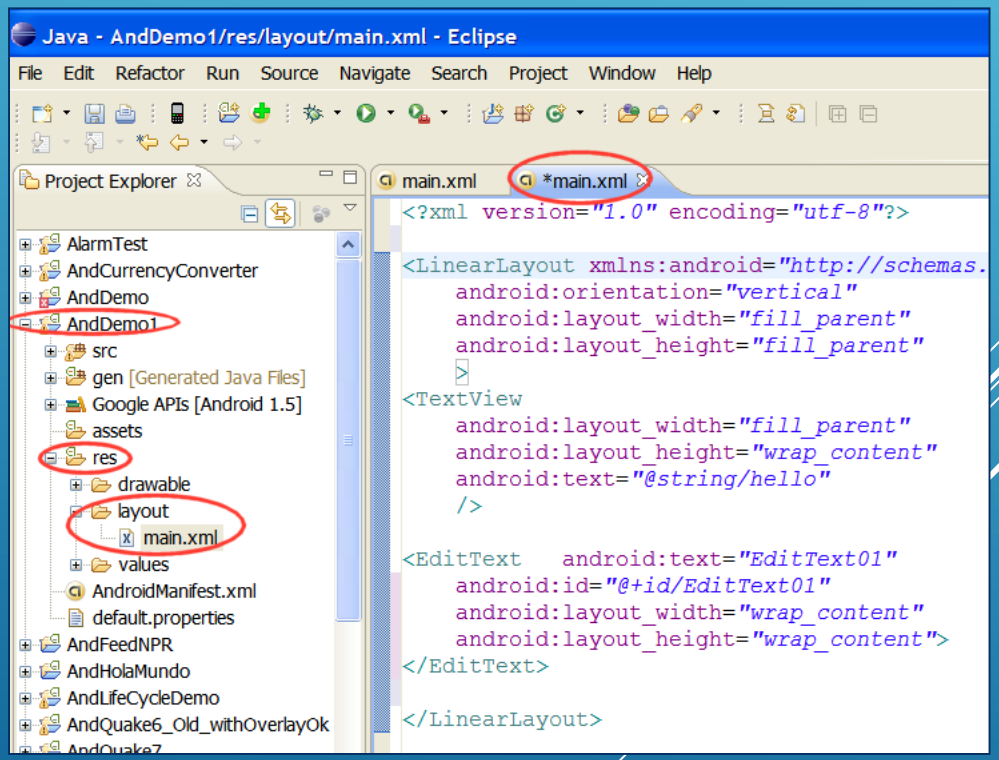
AutoCompleteTextView
It is a version of the *EditText* widget that will provide auto-complete suggestions as the user types. The suggestions are extracted from a collection of strings.

ListView
A *ListView* is a View that shows items in a vertically scrolling list. The items are acquired from a *ListAdapter*.

What is an XML Layout?

XML-based layout là một đặc tả về các UI component (widget), quan hệ giữa chúng với nhau và với container chứa chúng – tất cả được viết theo định dạng XML.

Android coi các XML-based layout là các ***resource (tài nguyên)***, và các file layout được lưu trong thư mục **res/layout** trong project của ta.



What is an XML Layout?

Mỗi file **XML** chứa một **cấu trúc phân cấp dạng cây**, đặc tả layout của các widget và các container thành phần của một View.

Các thuộc tính của mỗi phần tử XML là các *tính chất*, mô tả bề ngoài của widget hoặc hoạt động của một container.

Example:
Nếu một phần tử *Button* có một thuộc tính có giá trị

android:textStyle = "bold"

Nghĩa là phần text hiện trên mặt nút cần được vẽ bằng font chữ đậm (bold).

An example

Ứng dụng có một nút bấm chiếm toàn bộ màn hình.
Khi nhấn nút, phần text của nút cho biết thời gian hiện hành.


```
import
import
import
import
import
import

public class          extends
       btn

@Override
public void
   super
                                main
   btn                                    myButton
   btn                        new
      @Override
      public void



    // onCreate
//
private void
   btn          new
```

An example

This is the XML-Layout definition

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

Phần tử gốc(root) cần khai báo Android XML namespace:

xmlns:android="http://schemas.android.com/apk/res/android"

Tất cả các phần tử khác sẽ là con của root và sẽ thừa kế khai báo namespace đó.

Vì ta muốn gọi đến nút đó từ bên trong mã Java, ta cần cho nó một id qua thuộc tính **android:id** .

An example *cont.*

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

Các thuộc tính còn lại của thực thể Button này là:

- **android:text** giá trị khởi tạo của chuỗi text cần hiện trên mặt nút (ở đây là xâu rỗng)

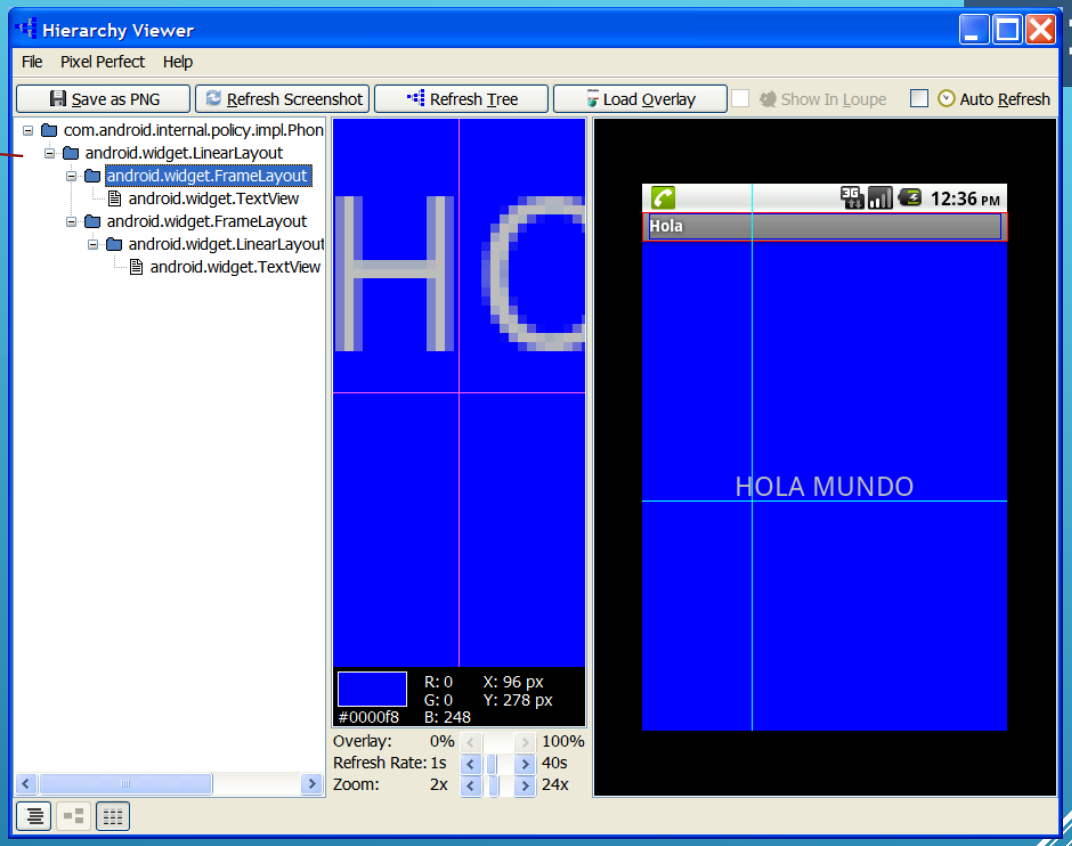
- **android:layout_width** và **android:layout_height** báo cho Android rằng chiều rộng và chiều cao của nút chiếm toàn bộ container (parent), ở đây là toàn bộ màn hình.

11

UI Hierarchy

Look for your SDK folder, usually:
C:/your_sdk_path/android_sdk_windows/tools

UI Tree



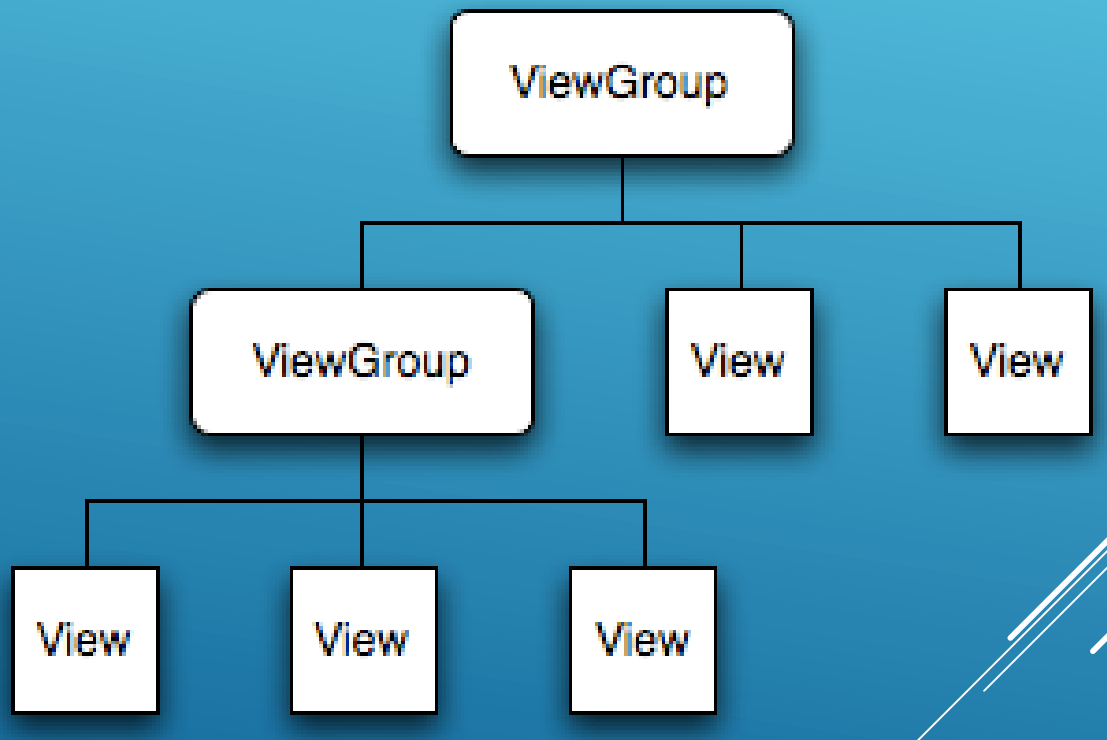
The utility ***HierarchyViewer*** displays the UI structure of the current screen shown on the emulator or device.

(Execute app on emulator, execute HierarchyViewer, click on Emulator > Refresh Screenshot)

12

Android Layouts

Each element in the XML Layout is either a ***View*** or ***ViewGroup*** object



Android Layouts

Displaying the Application's View

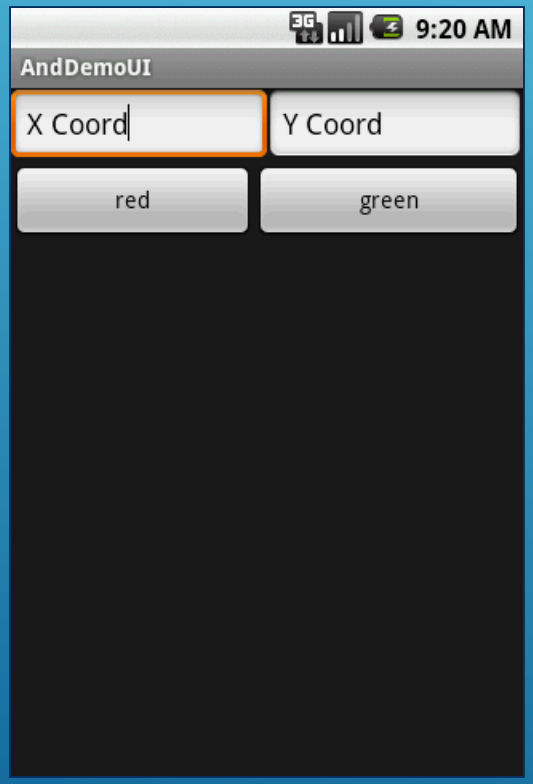
The Android UI Framework paints the screen by walking the View tree by asking each component to draw itself in a *pre-order traversal* way.

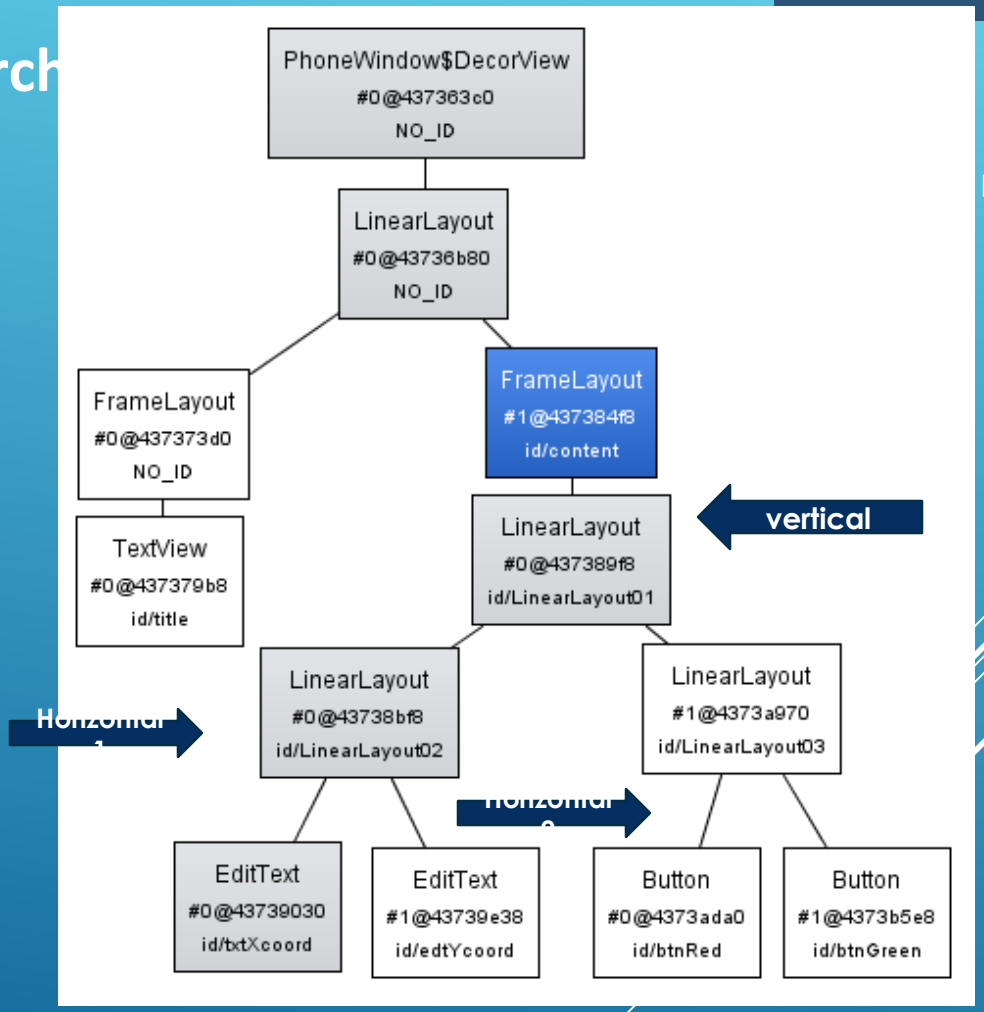
Each component draws itself and then asks each of its children to do the same.

See: Android – Application Development, by R. Rogers et al. O'Reilly Pub. 2009, ISBN 978-0-596-52147-0

Android Layouts

Example: Display UI Hierarch r8.





See: Android – Application Development, by R. Rogers et al. O'Reilly Pub. 2009, ISBN 978-0-596-52147-0

Android Layouts

Example: Display UI Hierarchy (Using SDK Revision 8)

UI Tree

Hierarchy Viewer

File Pixel Perfect Help

Save as PNG | Refresh Screenshot | Refresh Tree | Load Overlay | Show In Loupe | Auto Refresh

- com.android.internal.policy.impl.PhoneWind
  - android.widget.LinearLayout
    - android.widget.FrameLayout
      - android.widget.TextView
    - android.widget.FrameLayout
      - android.widget.LinearLayout
        - android.widget.LinearLayout
          - android.widget.EditText
          - android.widget.EditText
        - android.widget.LinearLayout
          - android.widget.Button
          - android.widget.Button

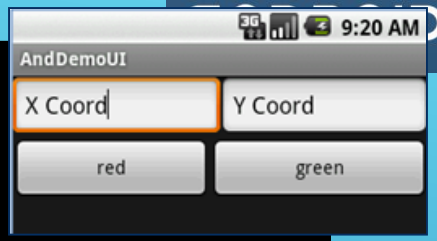
R: 0 G: 0 B: 0 #000000 X: 161 px Y: 112 px

Overlay: 0% 100%
Refresh Rate: 1s 40s
Zoom: 2x 24x

12:22 PM
Hola
X coord
Y coord
go GREEN
go BLUE

16

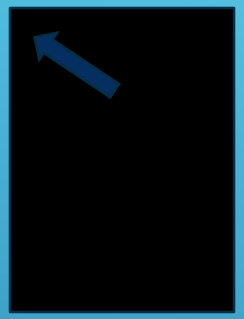
Example: Display UI Hierarchy




```xml
<?xml version "1.0" encoding "utf-8"?>
<LinearLayout android:id "@+id/LinearLayout01"
android:layout_width "fill_parent" android:layout_height "fill_parent"
android:orientation "vertical"
xmlns:android "http://schemas.android.com/apk/res/android">
     <LinearLayout android:id "@+id/LinearLayout02"
     android:layout_width "fill_parent" android:layout_height "wrap_content">
          <EditText android:id "@+id/txtXcoord" android:layout_width "wrap_content"
               android:layout_height "wrap_content" android:text "X Coord"
               android:layout_weight "1">
          </EditText>
          <EditText android:id "@+id/edtYcoord" android:layout_width "wrap_content"
               android:layout_height "wrap_content" android:text "Y Coord"
               android:layout_weight "1">
          </EditText>
     </LinearLayout>
     <LinearLayout android:id "@+id/LinearLayout03"
     android:layout_width "fill_parent" android:layout_height "wrap_content">
          <Button android:id "@+id/btnRed" android:layout_width "wrap_content"
               android:layout_height "wrap_content" android:text "red"
               android:layout_weight "1">
          </Button>
          <Button android:id "@+id/btnGreen" android:layout_width "wrap_content"
               android:layout_height "wrap_content" android:text "green"
               android:layout_weight "1">
          </Button>
     </LinearLayout>
</LinearLayout>
```

Common Layouts

There are five basic types of Layouts:
Frame, Linear, Relative, Table, and **Absolute.**



1. FrameLayout

FrameLayout is the simplest type of layout object. It's basically a *blank space on your screen* that you can later fill with a single object — for example, a picture that you'll swap in and out.

All child elements of the FrameLayout are *pinned to the top left corner of the screen*; you cannot specify a different location for a child view.

Subsequent child views will simply be drawn over previous ones, partially or totally obscuring them (unless the newer object is transparent).

18

Common Layouts

2. LinearLayout

LinearLayout aligns all children in a single direction — *vertically* or *horizontally* depending on the **android:orientation** attribute.

All children are stacked one after the other, so a

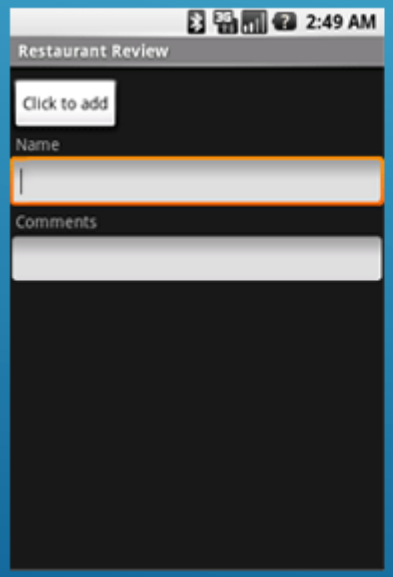
- *vertical* list will only have one child per row, no matter how wide they are, and a
- *horizontal* list will only be one row high (the height of the tallest child, plus padding).

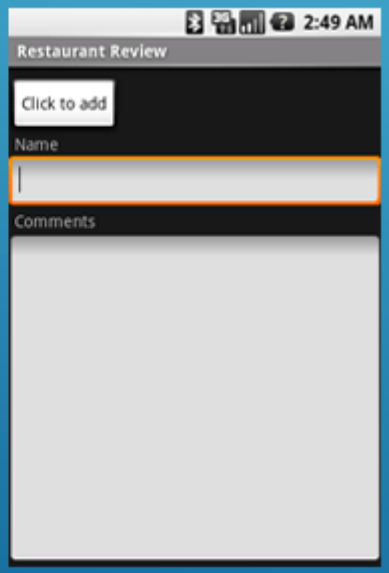
A LinearLayout respects *margins* between children and the *gravity* (right, center, or left alignment) of each child.

Common Layouts

2. LinearLayout

You may attribute a **weight** to children of a LinearLayout.
Weight gives an "importance" value to a view, and allows it to expand to fill any remaining space in the parent view.





Example:
The following two forms represent a LinearLayout with a set of elements: a button, some labels and text boxes. The text boxes have their width set to *fill_parent*; other elements are set to *wrap_content*. The gravity, by default, is left.

The difference between the two versions of the form is that the form on the left has **weight** values unset (**0** by default), while the form on the right has the comments text box weight set to **1**. If the Name textbox had also been set to 1, the Name and Comments text boxes would be the same height.

Common Layouts

3. TableLayout

1. TableLayout positions its children into **rows** and **columns**.
2. TableLayout containers do not display border lines.
3. The table will have *as many columns as the row with the most cells*.
4. A cell could be empty, but *cannot span columns*, as they can in HTML.
5. A *TableRow* object defines a single row in the table.
6. A row has zero or more cells, each cell is defined by any kind of other View.
7. A cell may also be a ViewGroup object.

Common Layouts

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/a
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:stretchColumns="*">
      <TableRow>
            <TextView android:text="Open…"
                 android:padding="3dip" />
            <TextView android:text="Ctrl-O"
                  android:gravity="right"
                 android:padding="3dip" />
      </TableRow>
      <TableRow>
            <TextView android:text="Save As…"
                 android:padding="3dip" />
            <TextView      android:text="Ctrl-Shi
                 android:gravity="right"
                 android:padding="3dip" />
      </TableRow>
</TableLayout>
```

TableLayout Example
The following sample layout has two rows and two cells in each. The accompanying screenshot shows the result, with cell borders displayed as dotted lines (*added for visual effect*).

Common Layouts

4. RelativeLayout

1. RelativeLayout lets child views specify their *position relative to the parent view or to each other* (specified by ID).

2. You can align two elements by *right border*, or make one *below* another, *centered* in the screen, *centered left*, and so on.

3. Elements are *rendered in the order given*, so if the first element is centered in the screen, other elements aligning themselves to that element will be aligned relative to screen center.

4. Also, because of this ordering, if using XML to specify this layout, the element that you will reference (in order to position other view objects) must be listed in the XML file before you refer to it from the other views via its reference ID.

Common Layouts

4. RelativeLayout

5. The defined RelativeLayout parameters are (android:layout_...) :

- width, height,
- below, above
- alignTop, alignParentTop,
- alignBottom, alignParentBottom
- toLeftOf, toRightOf
- padding [Bottom | Left | Right | Top], and
- margin [Bottom | Left | Right | Top].

For example, assigning the parameter

android:layout_toLeftOf="@+id/my_button"

to a TextView would place the TextView to the left of the View with the ID *my_button*

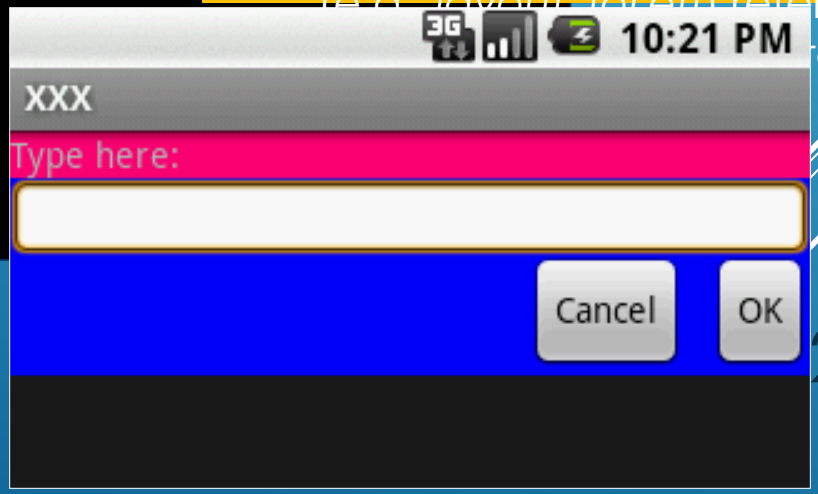
24

Common Layouts

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000ff"
        android:padding="10px" >

  <TextView android:id="@+id/label"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:background="#ffff0077"
      android:text="Type here:" />

  <EditText android:id="@+id/entry"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_below="@+id/label" />
```

RelativeLayout Example

The example below shows an XML file and the resulting screen in the UI. Note that the attributes that refer to relative elements (e.g. *layout_toLeft*) refer to the relative



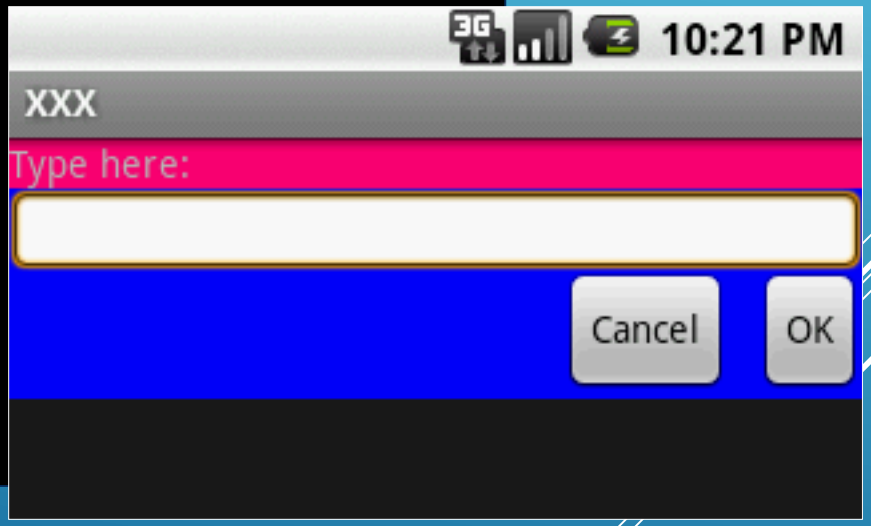
Continue next page

Common Layouts

```
<Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10px"
        android:text="OK" />

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

</RelativeLayout>
```

RelativeLayout Example

Cont.



A Detailed List of Widgets

For a detailed list consult:

http://developer.android.com/reference/android/widget/package-summary.html

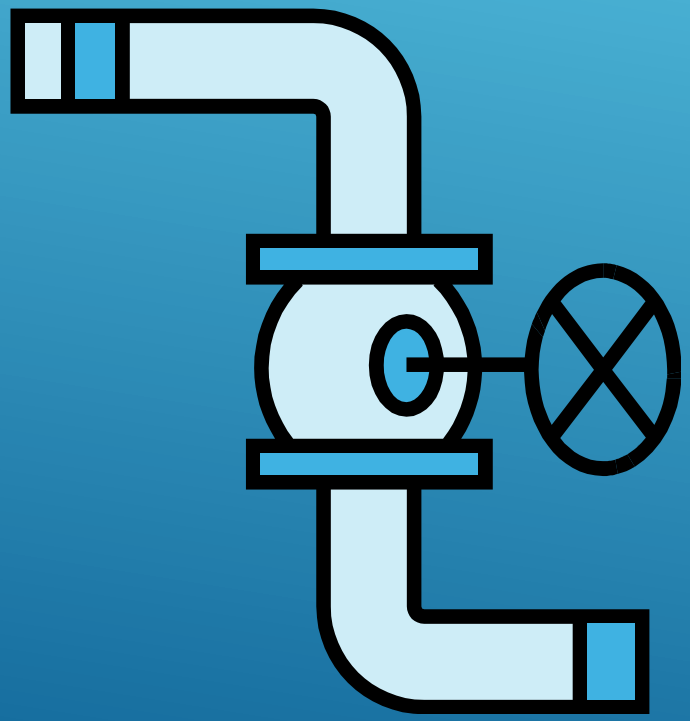
AbsListView
AbsListView.LayoutParams
AbsoluteLayout
AbsoluteLayout.LayoutParams
AbsSeekBar
AbsSpinner
AdapterView<T extends Adapter>
AdapterContextMenuInfo
AlphabetIndexer
AnalogClock
ArrayAdapter<T>
AutoCompleteTextView
BaseAdapter
BaseExpandableListAdapter
Button
CheckBox
CheckedTextView
Chronometer
CompoundButton
CursorAdapter
CursorTreeAdapter
DatePicker
DialerFilter
DigitalClock
EditText
ExpandableListView
ExpandableListContextMenuInfo
Filter
Filter.FilterResults
FrameLayout
FrameLayout.LayoutParams
Gallery
Gallery.LayoutParams
GridView
HeaderViewListAdapter
HorizontalScrollView
ImageButton
ImageSwitcher
ImageView
LinearLayout
LinearLayout.LayoutParams
ListView
ListView.FixedViewInfo
MediaController
MultiAutoCompleteTextView
CommaTokenizer
PopupWindow
ProgressBar
RadioButton
RadioGroup
RadioGroup.LayoutParams
RatingBar
RelativeLayout
RelativeLayout.LayoutParams
RemoteViews
ResourceCursorAdapter
ResourceCursorTreeAdapter
Scroller
ScrollView
SeekBar
SimpleAdapter
SimpleCursorAdapter
SimpleCursorTreeAdapter
SimpleExpandableListAdapter
SlidingDrawer
Spinner
TabHost
TabHost.TabSpec
TableLayout
TableLayout.LayoutParams
TableRow
TableRow.LayoutParams
TabWidget
TextSwitcher
TextView
TextView.SavedState
TimePicker
Toast
ToggleButton
TwoLineListItem
VideoView
ViewAnimator
ViewFlipper
ViewSwitcher
ZoomButton
ZoomControls

27

Attaching Layouts to Java Co

PLUMBING. Ta phải 'nối' các phần từ XML với các đối tượng tương đương trong activity. Nhờ đó, ta có thể thao tác với UI từ mã chương trình.

XLM Layout
<xml….
. . .
. . .
</xml>



JAVA code
public class ….
{
. . .
. . .
}

28

Attaching Layouts to Java Co

Giả sử UI đã được tạo tại ***res/layout/main.xml***. Ứng dụng có thể gọi layout này bằng lệnh

```
setContentView(R.layout.main);
```

Có thể truy nhập các widget, chẳng hạn ***myButton***, bằng lệnh *findViewByID(…)* như sau

```
Button btn = (Button) findViewByID(R.id.myButton);
```

Trong đó, **R** là một lớp được sinh tự động để theo dõi các tài nguyên của ứng dụng. Cụ thể, **R.id…** là các widget được định nghĩa trong layout XML.

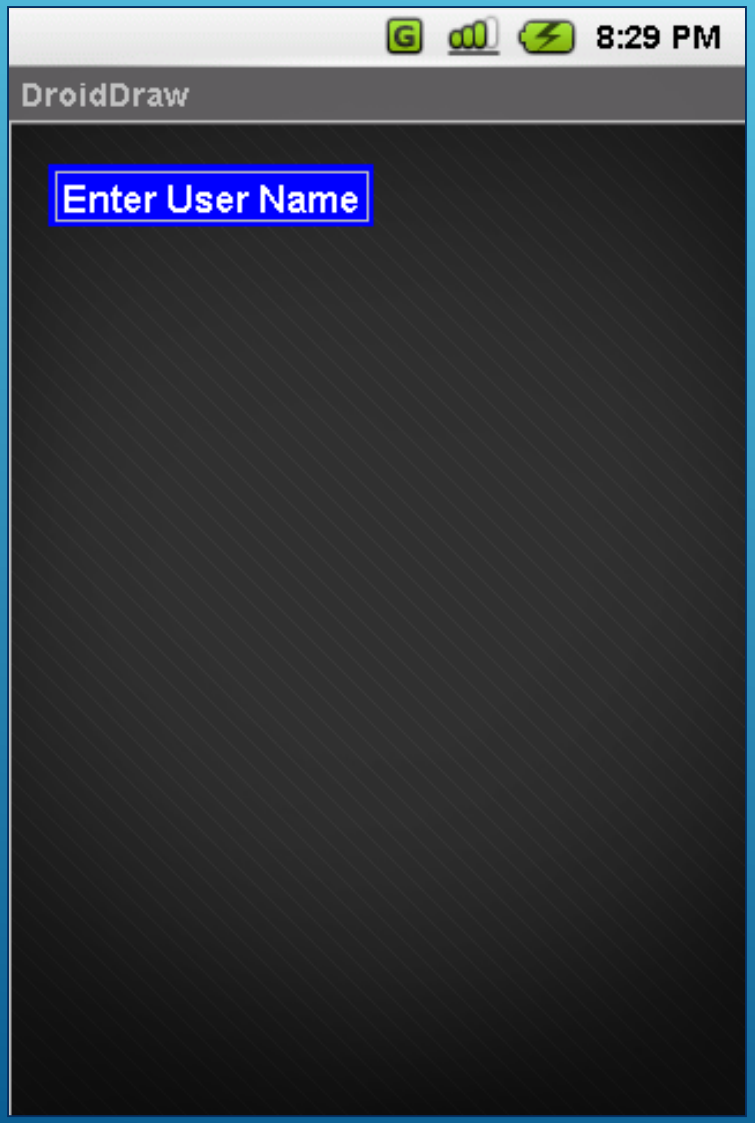
Attaching Layouts to Java Co

Gắn Listener cho Widget (event handling)

Button trong ví dụ của ta có thể dùng được sau khi gắn một listener cho sự kiện click:

```
btn.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        updateTime();
    }
});

private void updateTime() {
    btn.setText(new Date().toString());
}
```

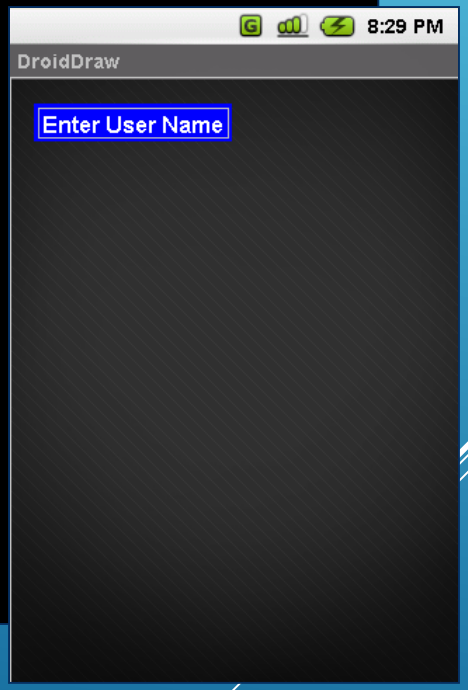
Basic Widgets: Labels



- A label is called in android a **TextView**.
- TextViews are typically used to display a caption.
- TextViews are *not* editable, therefore they take no input.

Basic Widgets: Labels

```
<?xml version "1.0" encoding "utf-8"?>
<LinearLayout
android:id "@+id/myLinearLayout"
android:layout_width "fill_parent"
android:layout_height "fill_parent"
xmlns:android "http://schemas.android.com/apk/res/android"
>
    <TextView
    android:id "@+id/myTextView1"
    android:layout_width "wrap_content"
    android:layout_height "wrap_content"
    android:background "#ff0000ff"
    android:padding "3dp"
    android:text "Enter User Name"
    android:textSize "16sp"
    android:textStyle "bold"
    android:gravity "center">
    </TextView>

</LinearLayout>
```



32

Basic Widgets: Labels/TextViews

http://developer.android.com/reference/android/widget/TextView.h

Attribute Name	*Related Method*	*Description*
android:autoLink	setAutoLinkMask(int)	Controls whether links such as urls and email addresses are automatically found and converted to clickable links.
android:autoText	setKeyListener(KeyListener)	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:bufferType	setText(CharSequence,TextView.BufferType)	Determines the minimum type that getText() will return.
android:capitalize	setKeyListener(KeyListener)	If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types.
android:cursorVisible	setCursorVisible(boolean)	Makes the cursor visible (the default) or invisible Must be a boolean value, either "`true`" or "`false`".
android:digits	setKeyListener(KeyListener)	If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept.
android:drawableBottom	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	The drawable to be drawn below the text.
android:drawableLeft	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	The drawable to be drawn to the left of the text.
android:drawablePadding	setCompoundDrawablePadding(int)	The padding between the drawables and the text.
android:drawableRight	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	The drawable to be drawn to the right of the text.
android:drawableTop	setCompoundDrawablesWithIntrinsicBounds(Drawable,Drawable,Drawable,Drawable)	The drawable to be drawn above the text.
android:editable		If set, specifies that this TextView has an input method.
android:editorExtras	setInputExtras(int)	Reference to an <input-extras> XML resource containing additional data to supply to an input method, which is private to the implementation of the input method.
android:ellipsize	setEllipsize(TextUtils.TruncateAt)	If set, causes words that are longer than the view is wide to be ellipsized instead of broken in the middle.
android:ems	setEms(int)	Makes the TextView be exactly this many ems wide Must be an integer value, such as "`100`".
android:freezesText	setFreezesText(boolean)	If set, the text view will include its current complete text inside of its frozen icicle in addition to meta-data such as the current cursor position.

33

Basic Widgets: Labels/TextViews *cont.*

http://developer.android.com/reference/android/widget/TextView.h

Attribute Name	*Related Method*	*Description*
android:gravity	setGravity(int)	Specifies how to align the text by the view's x and/or y axis when the text is smaller than the view.
android:height	setHeight(int)	Makes the TextView be exactly this many pixels tall.
android:hint	setHint(int)	Hint text to display when the text is empty.
android:imeActionId	setImeActionLabel(CharSequence,int)	Supply a value for EditorInfo.actionId used when an input method is connected to the text view.
android:imeActionLabel	setImeActionLabel(CharSequence,int)	Supply a value for EditorInfo.actionLabel used when an input method is connected to the text view.
android:imeOptions	setImeOptions(int)	Additional features you can enable in an IME associated with an editor, to improve the integration with your application.
android:includeFontPadding	setIncludeFontPadding(boolean)	Leave enough room for ascenders and descenders instead of using the font ascent and descent strictly.
android:inputMethod	setKeyListener(KeyListener)	If set, specifies that this TextView should use the specified input method (specified by fully-qualified class name).
android:inputType	setRawInputType(int)	The type of data being placed in a text field, used to help an input method decide how to let the user enter text.
android:lineSpacingExtra	setLineSpacing(float,float)	Extra spacing between lines of text.
android:lineSpacingMultiplier	setLineSpacing(float,float)	Extra spacing between lines of text, as a multiplier.
android:lines	setLines(int)	Makes the TextView be exactly this many lines tall Must be an integer value, such as "`100`".
android:linksClickable	setLinksClickable(boolean)	If set to false, keeps the movement method from being set to the link movement method even if autoLink causes links to be found.
android:marqueeRepeatLimit	setMarqueeRepeatLimit(int)	The number of times to repeat the marquee animation.
android:maxEms	setMaxEms(int)	Makes the TextView be at most this many ems wide Must be an integer value, such as "`100`".
android:maxHeight	setMaxHeight(int)	Makes the TextView be at most this many pixels tall Must be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`".
android:maxLength	setFilters(InputFilter)	Set an input filter to constrain the text length to the specified number.
android:maxLines	setMaxLines(int)	Makes the TextView be at most this many lines tall Must be an integer value, such as "`100`".

34

Basic Widgets: Labels/TextViews *cont.*

http://developer.android.com/reference/android/widget/TextView.h

Attribute Name	*Related Method*	*Description*
android:maxWidth	setMaxWidth(int)	Makes the TextView be at most this many pixels wide Must be a dimension value, which is a floating point number appended with a unit such as "14.5sp".
android:minEms	setMinEms(int)	Makes the TextView be at least this many ems wide Must be an integer value, such as "100".
android:minHeight	setMinHeight(int)	Makes the TextView be at least this many pixels tall Must be a dimension value, which is a floating point number appended with a unit such as "14.5sp".
android:minLines	setMinLines(int)	Makes the TextView be at least this many lines tall Must be an integer value, such as "100".
android:minWidth	setMinWidth(int)	Makes the TextView be at least this many pixels wide Must be a dimension value, which is a floating point number appended with a unit such as "14.5sp".
android:numeric	setKeyListener(KeyListener)	If set, specifies that this TextView has a numeric input method.
android:password	setTransformationMethod(TransformationMethod)	Whether the characters of the field are displayed as password dots instead of themselves.
android:phoneNumber	setKeyListener(KeyListener)	If set, specifies that this TextView has a phone number input method.
android:privateImeOptions	setPrivateImeOptions(String)	An addition content type description to supply to the input method attached to the text view, which is private to the implementation of the input method.
android:scrollHorizontally	setHorizontallyScrolling(boolean)	Whether the text is allowed to be wider than the view (and therefore can be scrolled horizontally).
android:selectAllOnFocus	setSelectAllOnFocus(boolean)	If the text is selectable, select it all when the view takes focus instead of moving the cursor to the start or end.
android:shadowColor	setShadowLayer(float,float,float,int)	Place a shadow of the specified color behind the text.
android:shadowDx	setShadowLayer(float,float,float,int)	Horizontal offset of the shadow.
android:shadowDy	setShadowLayer(float,float,float,int)	Vertical offset of the shadow.
android:shadowRadius	setShadowLayer(float,float,float,int)	Radius of the shadow.

35

Basic Widgets: Labels/TextViews *cont.*

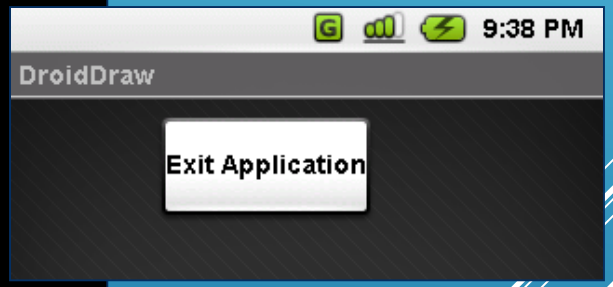
http://developer.android.com/reference/android/widget/TextView.h

Attribute Name	*Related Method*	*Description*
android:singleLine	setTransformationMethod(TransformationMethod)	Constrains the text to a single horizontally scrolling line instead of letting it wrap onto multiple lines, and advances focus instead of inserting a newline when you press the enter key.
android:text	setText(CharSequence)	Text to display.
android:textColor	setTextColor(ColorStateList)	Text color.
android:textColorHighlight	setHighlightColor(int)	Color of the text selection highlight.
android:textColorHint	setHintTextColor(int)	Color of the hint text.
android:textColorLink	setLinkTextColor(int)	Text color for links.
android:textScaleX	setTextScaleX(float)	Sets the horizontal scaling factor for the text Must be a floating point value, such as "1.2".
android:textSize	setTextSize(float)	Size of the text.
android:textStyle	setTypeface(Typeface)	Style (bold, italic, bolditalic) for the text.
android:typeface	setTypeface(Typeface)	Typeface (normal, sans, serif, monospace) for the text.
android:width	setWidth(int)	Makes the TextView be exactly this many pixels wide.

Basic Widgets: Buttons

- A **Button** widget allows the simulation of a clicking action on a GUI.
- Button is a subclass of TextView. Therefore formatting a Button's face is similar to the setting of a TextView.

```
...
<Button
android:id "@+id/btnExitApp"
android:layout_width "wrap_content"
android:layout_height "wrap_content"
android:padding "10px"
android:layout_marginLeft "5px"
android:text "Exit Application"
android:textSize "16sp"
android:textStyle "bold"
android:gravity "center"
android:layout_gravity "center_horizontal"
>
</Button>
```



37

Bài tập!

Cài đặt một trong các project sau
bằng các text box đơn giản
(EditText, TextView)
và các Button:

1.Tính lãi suất gửi tiền tiết kiệm (tiền gốc, lãi suất, thời hạn -> lãi)
2.Tính điểm tổng kết môn học Lập trình nhúng từ 3 điểm thành phần
3.Simple Flashlight (các nút bấm để đổi màu màn hình)

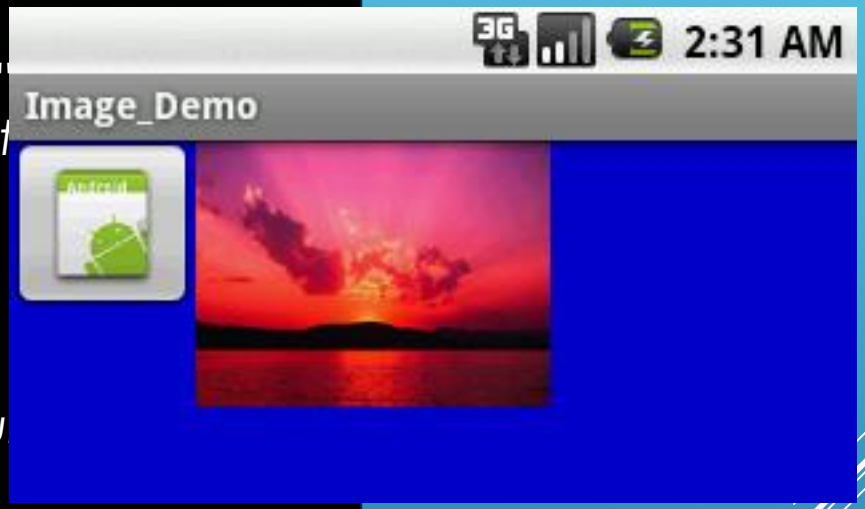
Chú ý: Activity label và tên project bắt đầu bằng username bitbucket. Tự thiết kế bố cục các view trên màn hình giao diện.
Nộp: toàn bộ mã nguồn + một vài trang màn hình tiêu biểu.
Hạn nộp: trước giờ thực hành tuần sau.

Basic Widgets: Images

- **ImageView** and **ImageButton** are two Android widgets that allow embedding of images in your applications.
- Both are *image-based widgets* analogue to *TextView* and *Button*, respectively.
- Each widget takes an **android:src** or **android:background** attribute (in an XML layout) to specify what picture to use.
- Pictures are usually reference a *drawable* resource.
- **ImageButton**, is a subclass of ImageView. It adds the standard *Button* behavior for responding to *click* events.

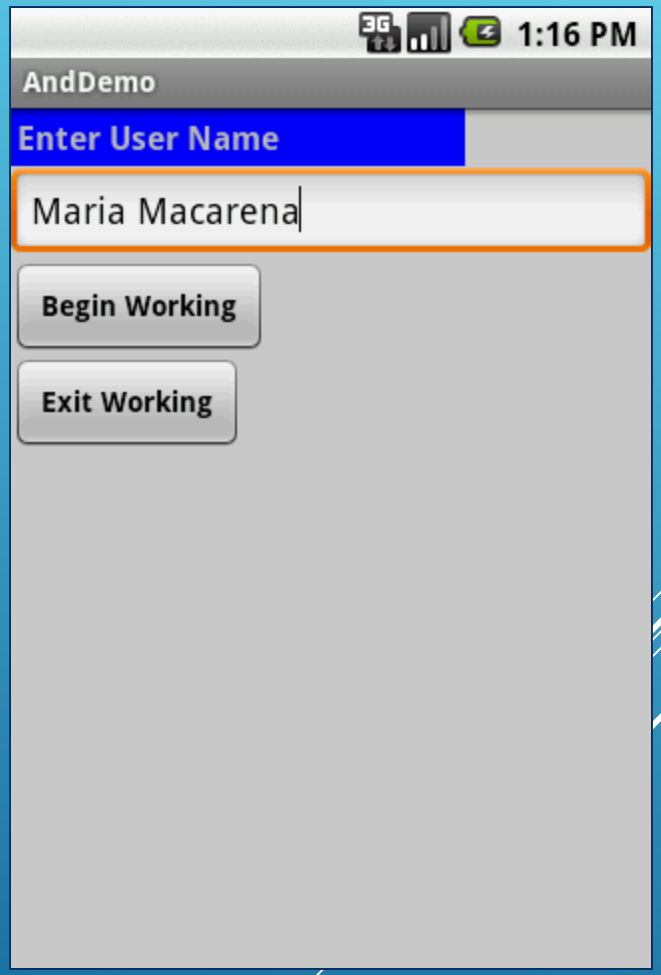
Basic Widgets: Images

```
. . .
<ImageButton
  android:id="@+id/myImageBtn1"
  android:src="@drawable/icon"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content
>
</ImageButton>
<ImageView
  android:id="@+id/myImageView1"
  android:src="@drawable/microsoft_su
  android:layout_width="150px"
  android:layout_height="120px"
  android:scaleType="fitXY"
>
</ImageView>
```



Basic Widgets: EditText

- The **EditText** (or *textBox*) widget is an extension of *TextView* that allows updates.
- The control configures itself to be *editable*.
- Important Java methods are: txtBox.setText("someValue") and txtBox.getText().toString()



Basic Widgets: EditText

In addition to the standard TextView properties EditText has many others features such as:

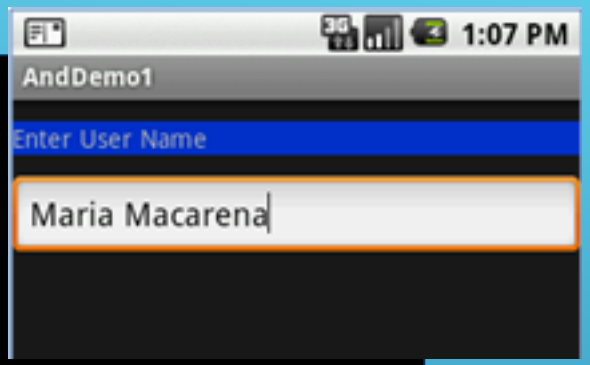
- **android:autoText**, (true/false) provides automatic spelling assistance
- **android:capitalize**, (*words/sentences*) automatic capitalization
- **android:digits**, to configure the field to accept only certain digits
- **android:singleLine**, is the field for single-line / multiple-line input
- **android:password**, (*true/false*) controls field's visibility
- **android:numeric**, (*integer, decimal, signed*) controls numeric format
- **android:phonenumber**, (true/false) Formatting phone numbers

Basic Widgets: EditViews

Example

```
...
<EditText
    android:id="@+id/txtUserName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:autoText="true"
    android:capitalize="words"
    android:hint="First Last Name"
>
</EditText>
...
```

Upper case words

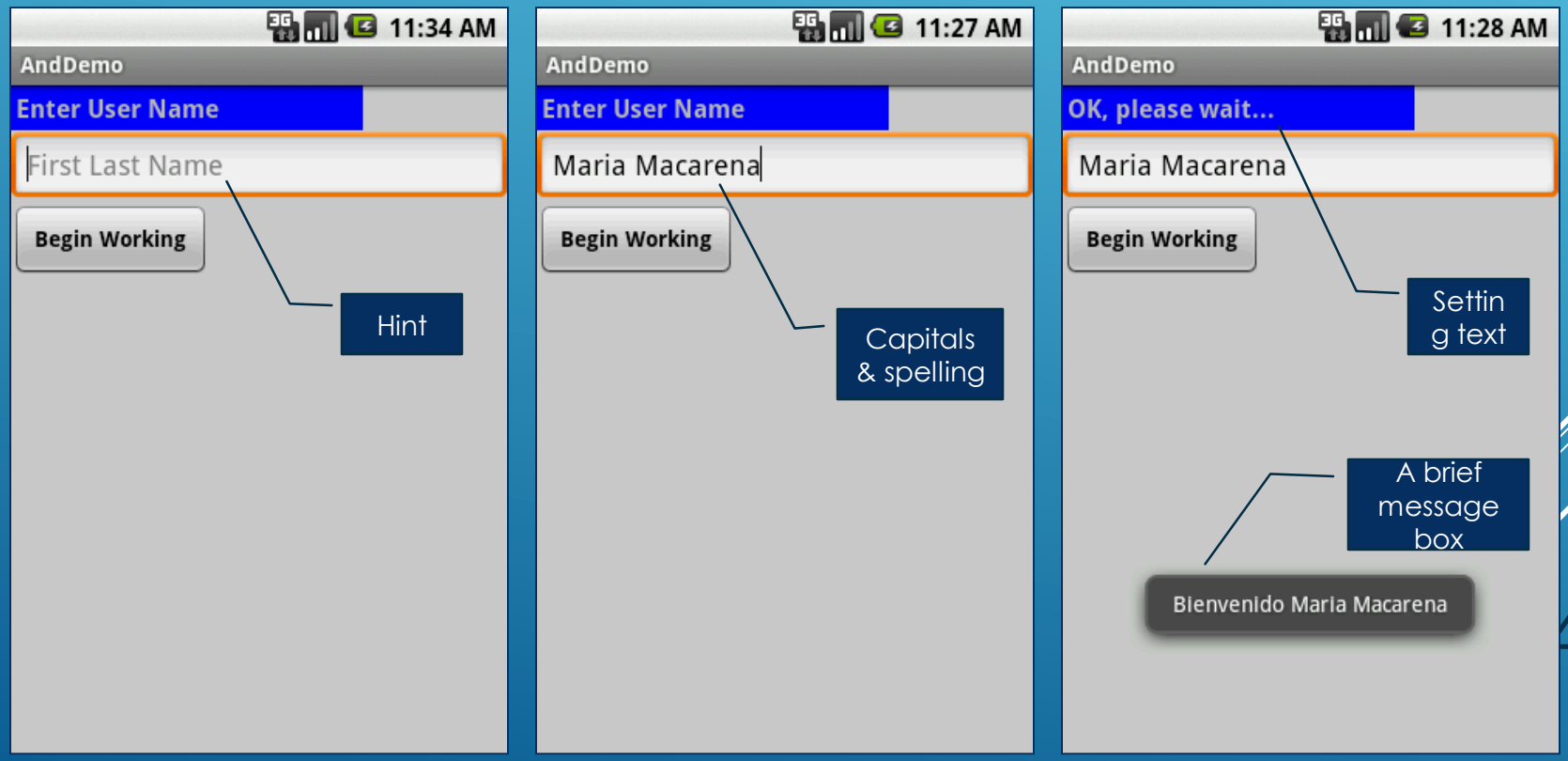


Enter "teh"
It will be changed to:
"The"

Suggestion (grey)

Basic Widgets: Example 1

In this little example we will use an **AbsoluteLayout** holding a label(**TexView**), a textBox (**EditText**), and a **Button**.
We will use the view as a sort of simplified login screen.



Basic Widgets: Example 1

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

<TextView
android:id="@+id/labelUserName"
android:layout_width="227px"
android:layout_height="wrap_content"
android:background="#ff0000ff"
android:padding="3px"
android:text="Enter User Name"
android:textSize="16sp"
android:textStyle="bold"
>
</TextView>
```

```
<EditText
android:id="@+id/txtUserName"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textSize="18sp"
android:autoText="true"
android:capitalize="words"
android:hint="First Last Name"
>
</EditText>

<Button
android:id="@+id/btnBegin"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=" Begin Working "
android:textSize="14px"
android:textStyle="bold"
>
</Button>

</LinearLayout>
```

Basic Widgets: Example 1

Android's Application (1 of 2)

```
package
import
import
import
import
import
import
import
import
////////////////////////////////////////////////////////////////////////
// "LOGIN" - a gentle introduction to UI controls

public class          extends
             labelUserName
             txtUserName
           btnBegin

    @Override
    public void
        super
                                main

        //binding the UI's controls defined in "main.xml" to Java code
        labelUserName                                   labelUserName
        txtUserName                                   txtUserName
        btnBegin                                  btnBegin
```

Basic Widgets: Example 1

Android's Application (2 of 2)

```
//LISTENER: wiring the button widget to events-&-code
btnBegin                    new
@Override
public void
                    txtUserName
    if                  "Maria Macarena"
        labelUserName        "OK, please wait..."

            "Bienvenido "
                LENGTH_SHORT


        "Bienvenido "
            LENGTH_SHORT


    // onClick

  //onCreate

//class
```

Basic Widgets: Example 1

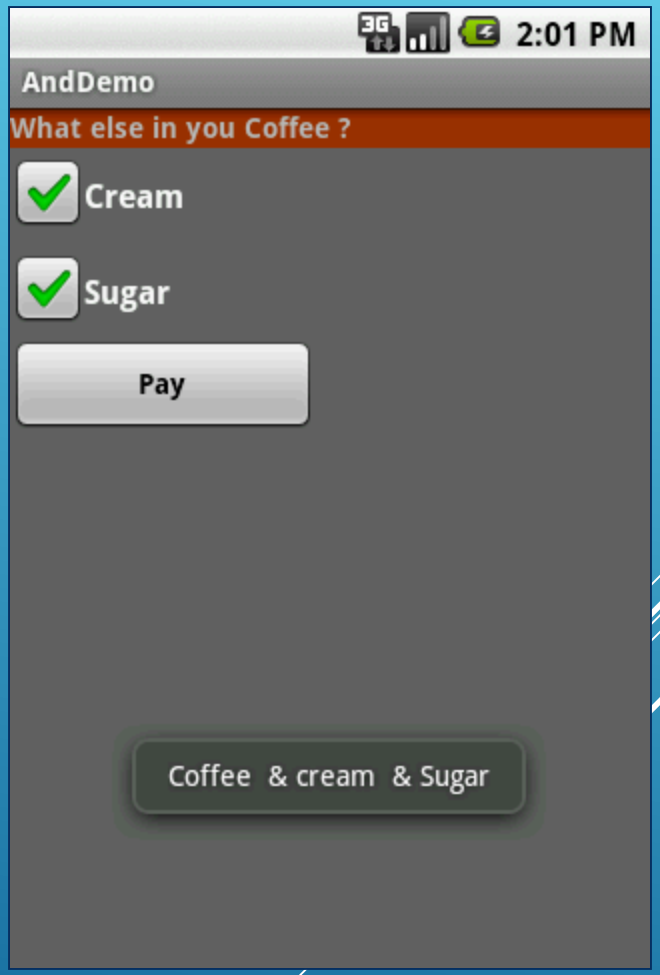
Note: Another way of defining a Listener for multiple button widgets

```
package cis493.gui;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.*;

public class AndDemo extends Activity implements OnClickListener {
    Button btnBegin;
    Button btnExit;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //binding the UI's controls defined in "main.xml" to Java code
        btnBegin = (Button) findViewById(R.id.btnBegin);
        btnExit = (Button) findViewById(R.id.btnExit);

        //LISTENER: wiring the button widget to events-&-code
        btnBegin.setOnClickListener(this);
        btnExit.setOnClickListener(this);
    }//onCreate

    @Override
    public void onClick(View v) {
        if (v.getId()==btnBegin.getId() ){
            Toast.makeText(getApplicationContext(), "1-Begin", 1).show();
        }
        if (v.getId()==btnExit.getId() ){
            Toast.makeText(getApplicationContext(), "2-Exit", 1).show();
        }
    }//onClick
}//class
```

Basic Widgets: CheckBox

A checkbox is a specific type of two-states button that can be either *checked* or *unchecked*.

A example usage of a checkbox inside your activity would be the following:



Example 2: CheckBox

Complete code for the checkBox demo (1 of 3)

Layout: main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/linearLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff666666"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>

<TextView
android:id="@+id/labelCoffee"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff993300"
android:text="What else in you Coffee ?"
android:textStyle="bold"
>
</TextView>
```

```
<CheckBox
android:id="@+id/chkCream"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Cream"
android:textStyle="bold"
>
</CheckBox>
<CheckBox
android:id="@+id/chkSugar"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Sugar"
android:textStyle="bold"
>
</CheckBox>
<Button
android:id="@+id/btnPay"
android:layout_width="153px"
android:layout_height="wrap_content"
android:text="Pay"
android:textStyle="bold"
>
</Button>
</LinearLayout>
```

Example 2: CheckBox

Complete code for the checkBox demo (2 of 3)

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class AndDemo extends Activity {
    CheckBox chkCream;
    CheckBox chkSugar;
    Button btnPay;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //binding XMl controls with Java code
        chkCream = (CheckBox)findViewById(R.id.chkCream);
        chkSugar = (CheckBox)findViewById(R.id.chkSugar);
        btnPay = (Button) findViewById(R.id.btnPay);
```

Example 2: CheckBox

Complete code for the checkBox demo (1 of 2)

```
        //LISTENER: wiring button-events-&-code
        btnPay.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        String msg = "Coffee ";
                        if (chkCream.isChecked()) {
                                msg += " & cream ";
                        }
                        if (chkSugar.isChecked(){
                                msg += " & Sugar";
                        }
                        Toast.makeText(getApplicationContext(),
                                        msg, Toast.LENGTH_SHORT).show();
                        //go now and compute cost...

                }//onClick


        });
    }//onCreate
}//class
```

Basic Widgets: RadioButton

- A radio button is a two-states button that can be either *checked* or *unchecked*.
- When the radio button is unchecked, the user can press or click it to check it.
- Radio buttons are normally used together in a **RadioGroup**.

- When several radio buttons live inside a radio group, checking one radio button *unchecks* all the others.
- RadioButton inherits from … TextView. Hence, all the standard TextView properties for *font face, style, color,* etc. are available for controlling the look of radio buttons.
- Similarly, you can call ***isChecked()*** on a RadioButton to see if it is selected, **toggle()** to select it, and so on, like you can with a CheckBox.

Basic Widgets: RadioButton

Example

We extend the previous example by adding a *RadioGroup* and three *RadioButtons*. Only new XML and Java code is shown:

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

<RadioGroup
android:id="@+id/radGroupCoffeeType"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<TextView
android:id="@+id/labelCoffeeType"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff993300"
android:text="What type of coffee?"
android:textStyle="bold"
>
</TextView>
<RadioButton
android:id="@+id/radDecaf"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Decaf"
>
</RadioButton>
<RadioButton
android:id="@+id/radExpresso"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Expresso"
>
</RadioButton>
<RadioButton
android:id="@+id/radColombian"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Colombian"
>
</RadioButton>
</RadioGroup>

...

</LinearLayout>
```

Basic Widgets: RadioButton

Android Activity (1 of 3)

```
package
// example using RadioButtons
import
import
import
import
import
import
import
import
import

public class            extends
            chkCream
            chkSugar
          btnPay
              radCoffeeType
               radDecaf
               radExpresso
               radColombian
```


55

Basic Widgets: RadioButton

Android Activity (2 of 3)

```
@Override
   public void
       super
                                   main
       //binding XMl controls to Java code
       chkCream                                    chkCream
       chkSugar                                    chkSugar
       btnPay                                   btnPay

       radCoffeeType                                   radGroupCoffeeType
       radDecaf                                     radDecaf
       radExpresso                                     radExpresso
       radColombian                                    radColombian
```

Basic Widgets: RadioButton

```
//LISTENER: wiring button-events-&-code
btnPay                    new
  @Override
  public void
                  "Coffee "
    if  chkCream
            " & cream "

    if  chkSugar
            " & Sugar"

    // get radio buttons ID number
    int           radCoffeeType
    // compare selected's Id with individual RadioButtons ID
    if  radColombian
            "Colombian "
    // similarly you may use .isChecked() on each RadioButton
    if  radExpresso
            "Expresso "

                                                LENGTH_SHORT
    // go now and compute cost...
     // onClick

 // onCreate
// class
```

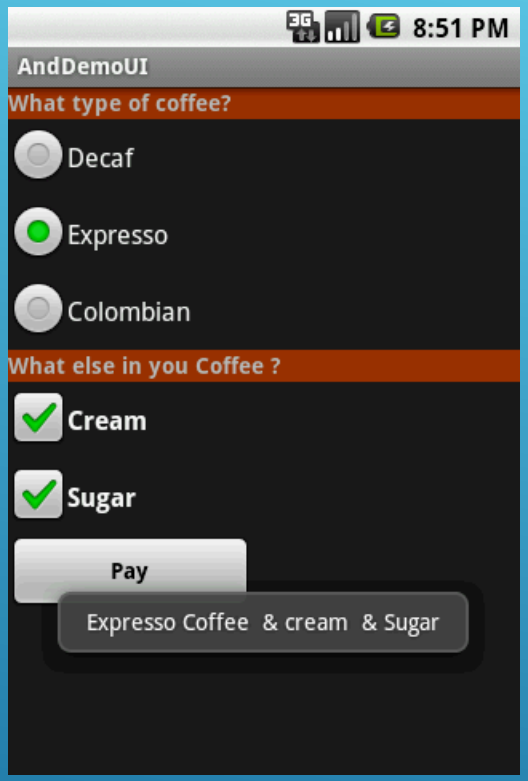
Basic Widgets: RadioButton

Example

This UI uses *RadioButtons* and *CheckBoxes* to define choices

RadioGroup

Summary of choices



58

UI – OTHER FEATURES

All *widgets* extend **View** therefore they acquire a number of useful View properties and methods including:

XML Controls the focus sequence:

- android:visibility
- Android:background

Java methods

- myButton.requestFocus()
- myTextBox.isFocused()
- myWidget.setEnabled()
- myWidget.isEnabled()

Questions ?

Resource: DroidDraw

www.droidDraw.org

DroidDraw

File Edit Project Help

Generate Load Undo Redo

Screen

Root Layout: LinearLayout

Screen Size: HVGA Portrait

8:22 AM

DroidDraw

ABC Droid Corporation of Ohio

Button EditText

Widgets Layouts Properties Strings Colors Arrays Support

Properties for: EditText

Id @+id/widget34

Width wrap_content

Height wrap_content

Background Color

Apply

Output

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff009999"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/widget29"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

Android Asset Studio – Beta (Accessed: 18-Jan-2011)

AAS Link: http://code.google.com/p/android-ui-utils/
Icon Gen http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html
Pencil 1.2 http://pencil.evolus.vn/en-US/Home.aspx
Video: http://www.youtube.com/watch?v=EaT7sYr_f0k&feature=player_embedded

WARNING: These utilities are currently in beta.

Utilities that help in the design and development of Android application user interfaces. This library currently consists of three individual tools for designers and developers:

1. UI Prototyping Stencils

A set of stencils for the Pencil GUI prototyping tool, which is available as an add-on for Firefox or as a standalone download.

2. Android Asset Studio

Try out the beta version: Android Asset Studio (shortlink: http://j.mp/androidassetstudio)

A web-based set of tools for generating graphics and other assets that would eventually be in an Android application's res/ directory.

Currently available asset generators area available for:

Launcher icons
Menu icons
Tab icons
Notification icons

Support for creation of XML resources and nine-patches is planned for a future release.

3. Android Icon Templates

A set of Photoshop icon templates that follow the icon design guidelines, complementing the official Android Icon Templates Pack.

Questions - Measuring Graphic Elements

Q. What is **dpi ?**

Stands for **dots per inch.** You can compute it using the following formula:

dpi = sqrt (width_pixels^2 + height_pixels^2) / diagonal_inches

G1 (base device 320x480) 155.92 dpi (3.7 in diagonally)

Nexus (480x800) 252.15 dpi

Q. What is my Emulator's Screen Resolution?

When creating an AVD you could set the entry "**Abstracted LCD density**" parameter to anything. Its default value is 160 dpi (use 260 for Nexus).

Q. How Android deals with screen resolutions?

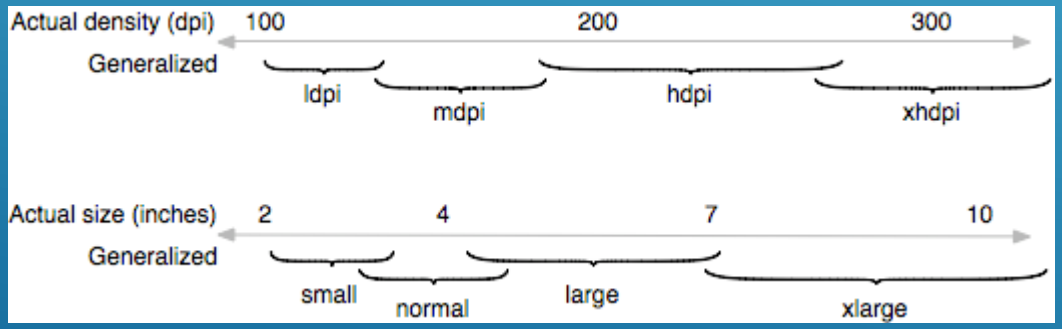


Illustration of how the Android platform maps actual screen densities and sizes to generalized density and size configurations.

Questions - Measuring Graphic Elements

Q. What do I gain by using screen densities?

More homogeneous results as shown below

Examples of density independence on WVGA high density (left), HVGA medium density (center), and QVGA low density (right).

Q. How to set different density/size screens in my application?

The following manifest fragments declares support for small, normal, large, and xlarge screens in any density.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <supports-screens
        android:smallScreens="true"
        android:normalScreens="true"
        android:largeScreens="true"
        android:xlargeScreens="true"
        android:anyDensity="true" />
    ...
</manifest>
```

Questions - Measuring Graphic Elements

Q. Give me an example on how to use dip units.

Assume you design your interface for a G1 phone having 320x480 pixels (Abstracted LCD density is 160 – See your AVD entry)

You want a button to be hand-placed in the middle of the screen.

You could allocate the 320 horizontal pixels as [100 + 120 + 100]. The XML would be

```
<Button>
    android:layout_height="wrap_content"
    android:layout_width="120dip"
    android:layout_x="100dip"
    android:layout_y="240dip"
    android:text="Go"
    android:id="@+id/btnGo"
</Button>
```

Instead of using pixels (px) you should use dip. If the application is deployed on a higher resolution screen (more pixels in 1 dip) the button is still mapped to the middle of the screen.