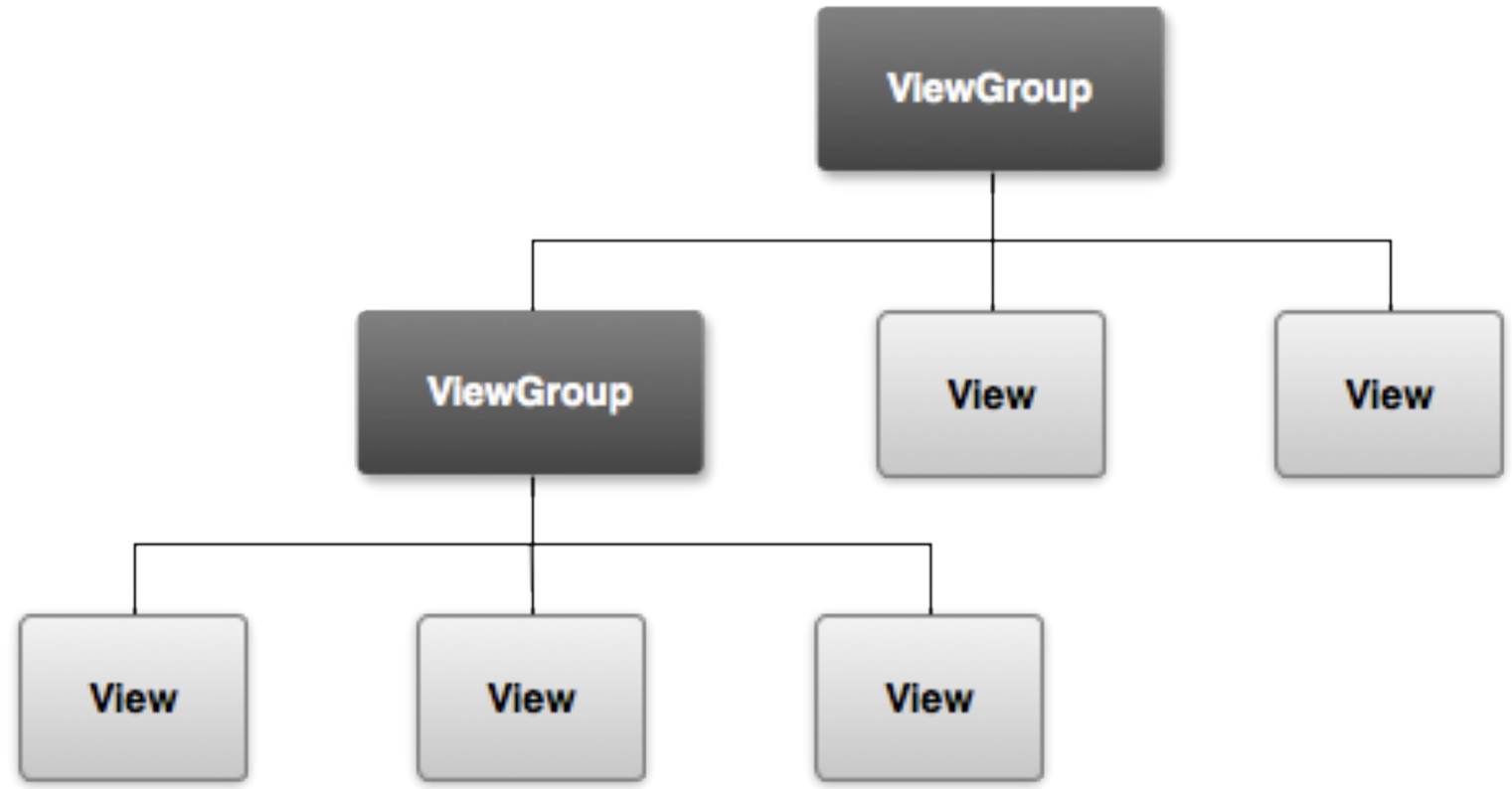
User Interface Software Projects

Assoc. Professor Donald J. Patterson
INF 134 Winter 2013

Android U/I design

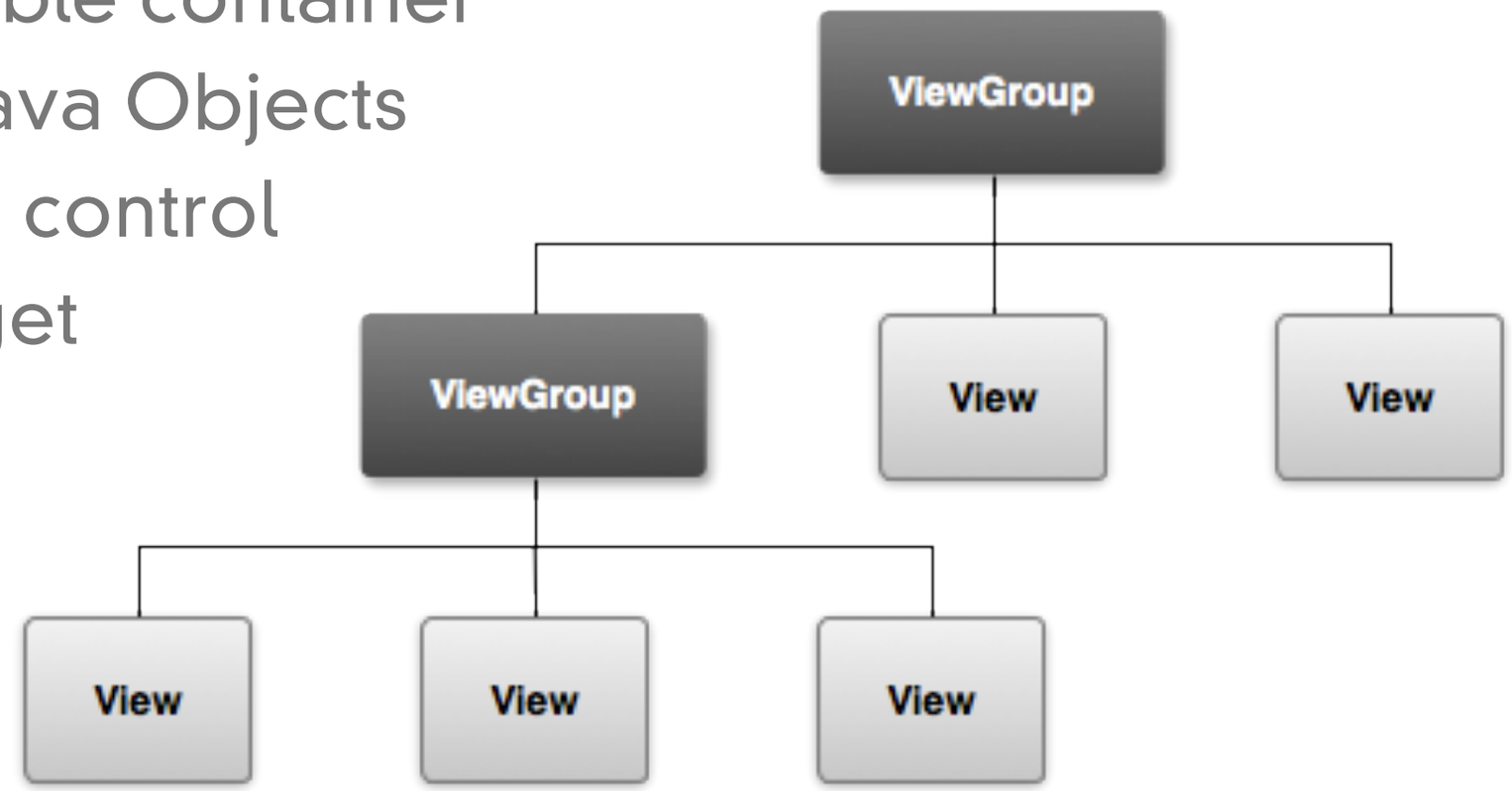
Android U/I Structure



http://developer.android.com/guide/topics/ui/overview.html

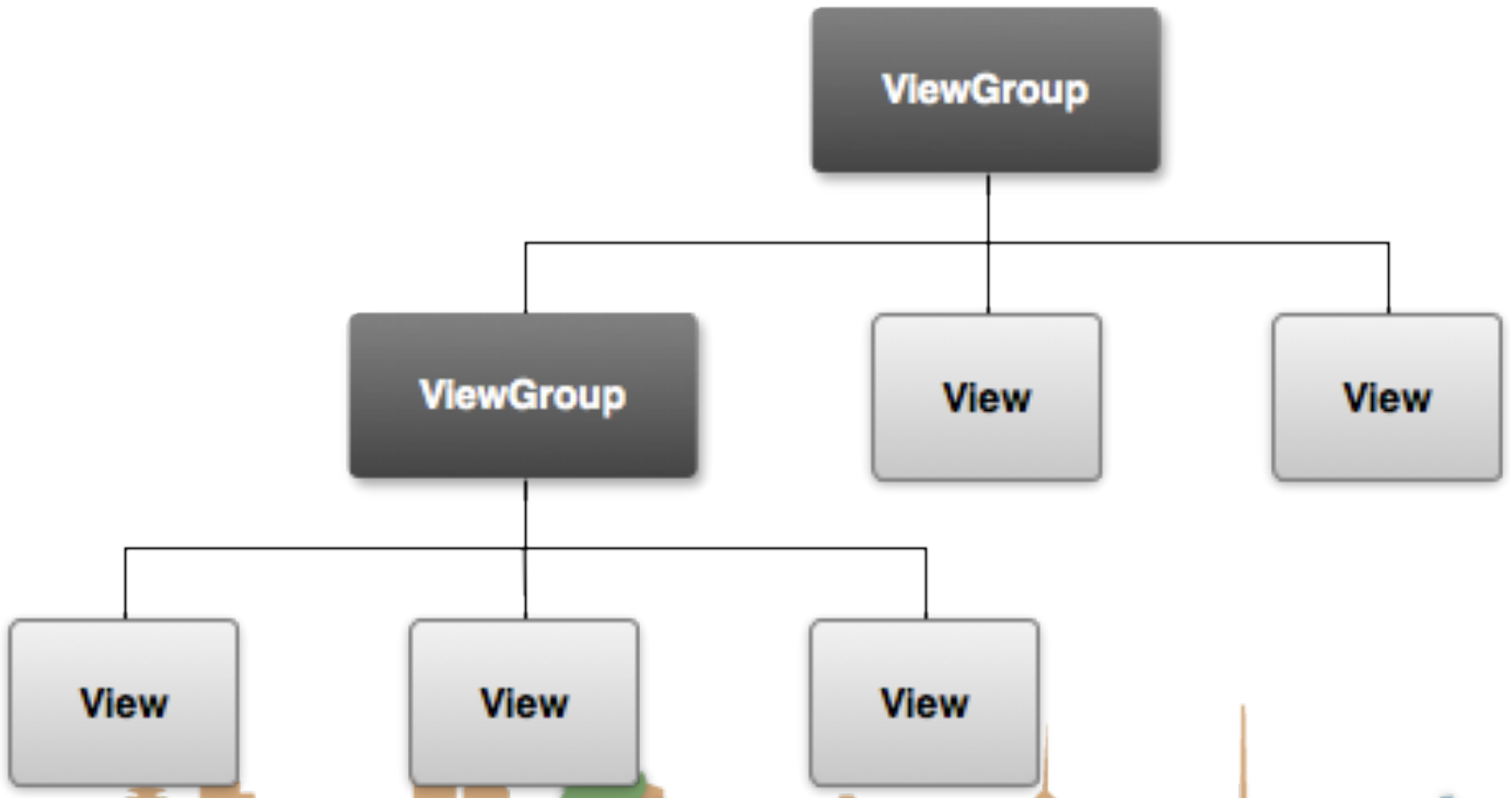
Android U/I Structure

- At the most abstract Android U/Is are hierarchies of
  - ViewGroup Java Objects
    - invisible container
  - View Java Objects
    - input control
    - widget



http://developer.android.com/guide/topics/ui/overview.html

Android U/I Structure

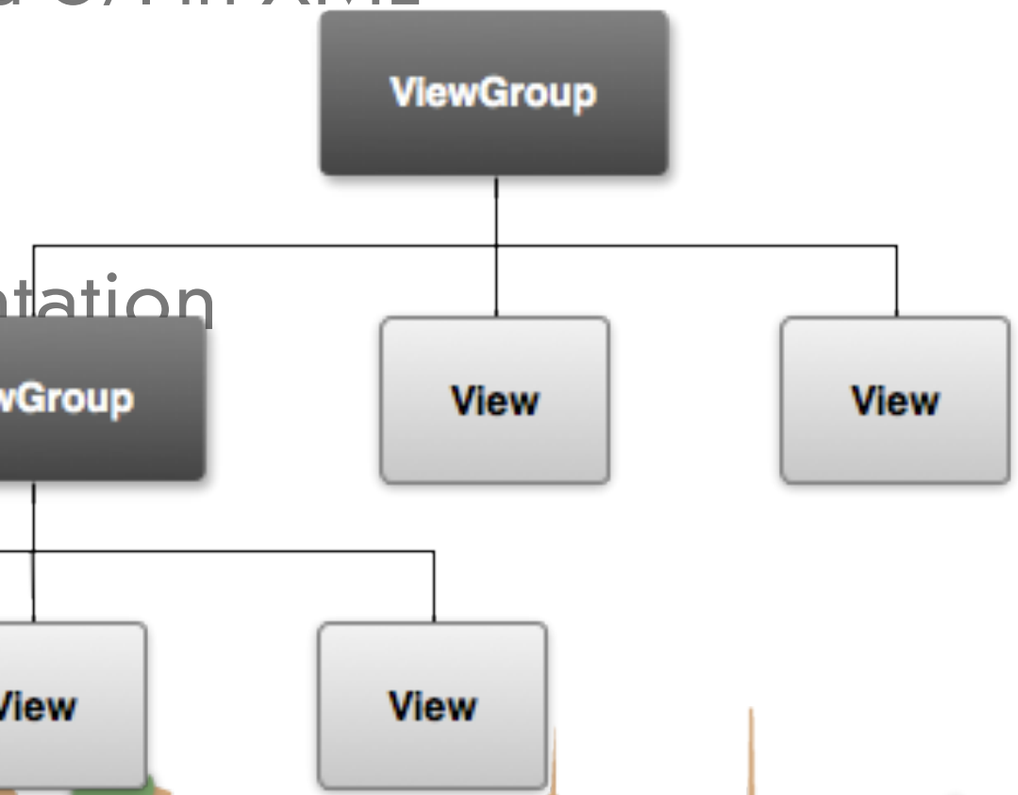


http://developer.android.com/guide/topics/ui/overview.html

Android U/I Structure

- You *can* create a U/I by instantiating Views and ViewGroups in your code
  - good for dynamic U/I designs
    - new View(...)

- It is faster and easier to build a U/I in XML
  - mostly static design
  - GUI assistance
  - separates code and presentation



http://developer.android.com/guide/topics/ui/overview.html

Android U/I Structure

public class

LinearLayout

extends ViewGroup

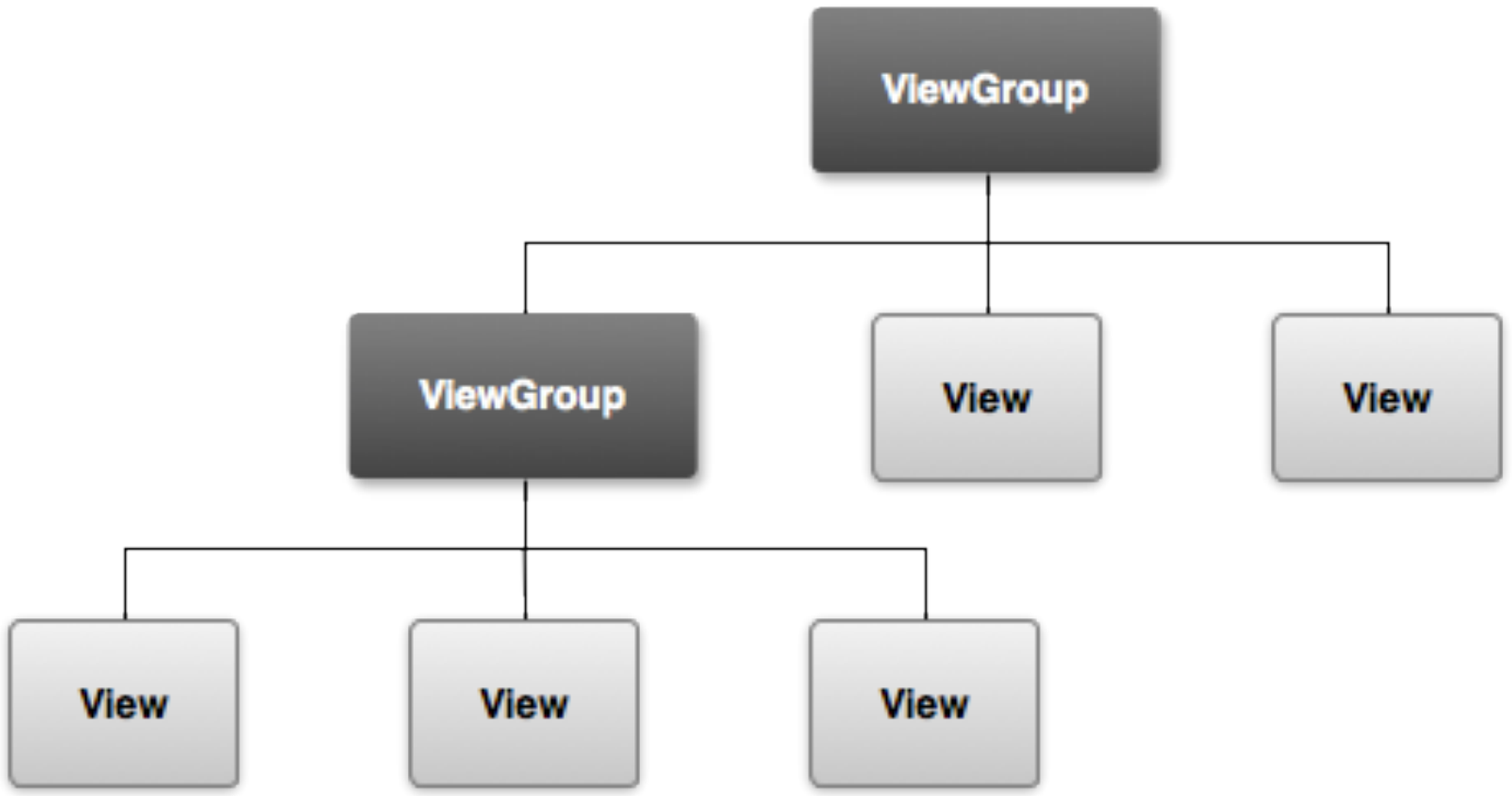
java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.LinearLayout

public class

TextView

extends View
implements ViewTreeObserver.OnPreDr

java.lang.Object
↳ android.view.View
↳ android.widget.TextView



http://developer.android.com/guide/topics/ui/overview.html

Android U/I Structure

public class

LinearLayout

extends ViewGroup

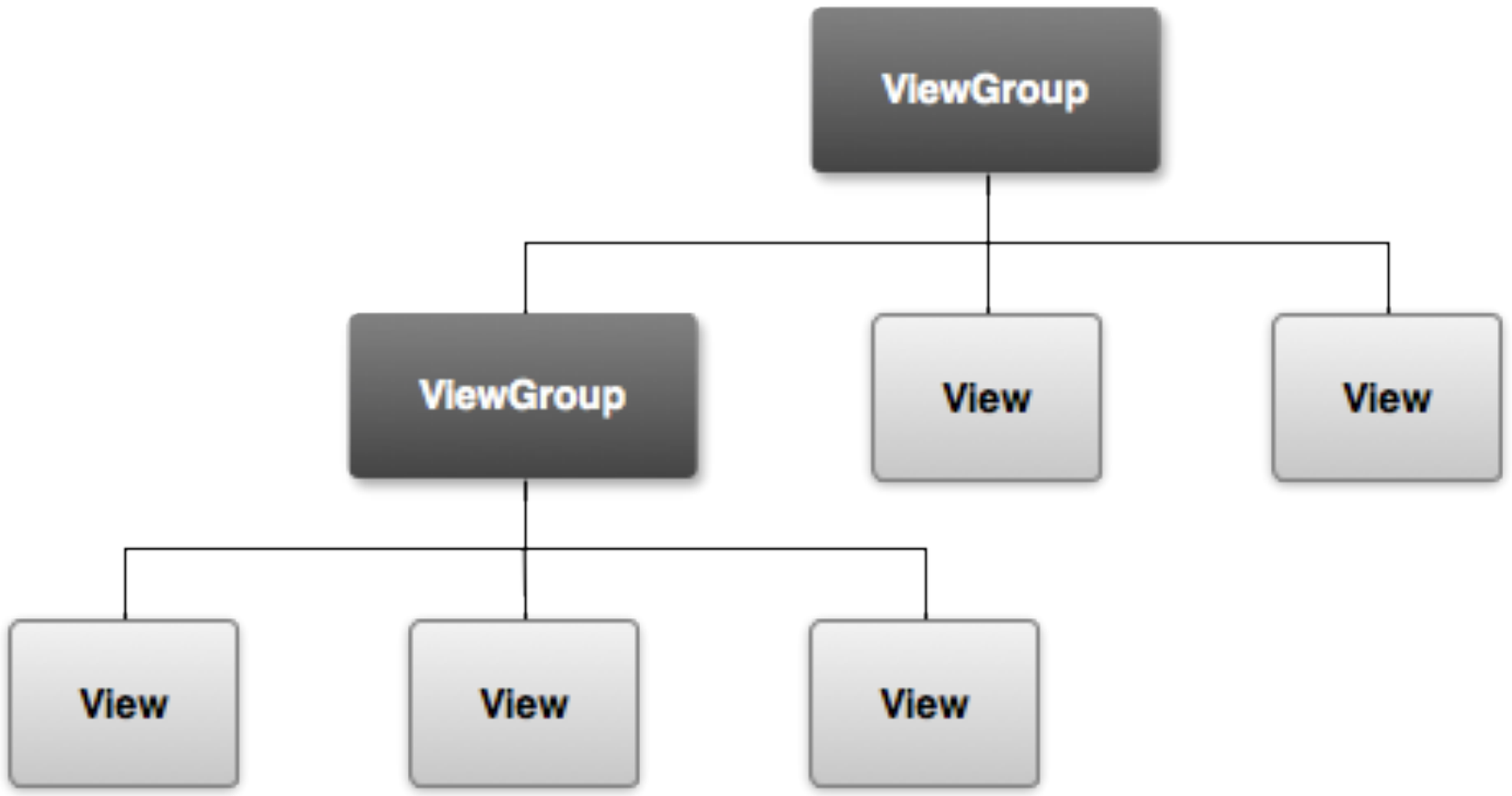
java.lang.Object
↳android.view.View
↳android.view.ViewGroup
↳android.widget.LinearLayout

public class

TextView

extends View
implements ViewTreeObserver.OnPreDr

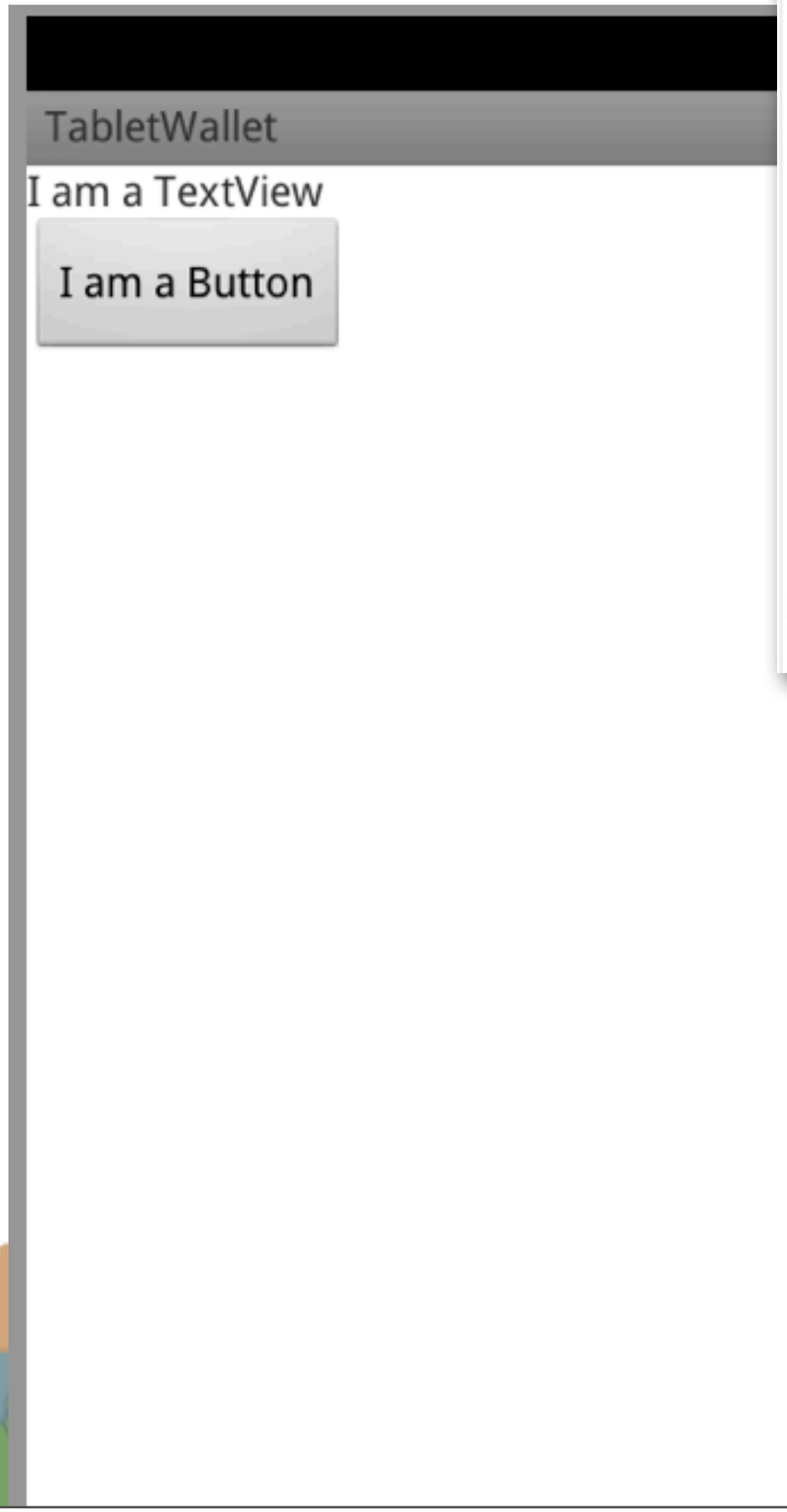
java.lang.Object
↳android.view.View
↳android.widget.TextView



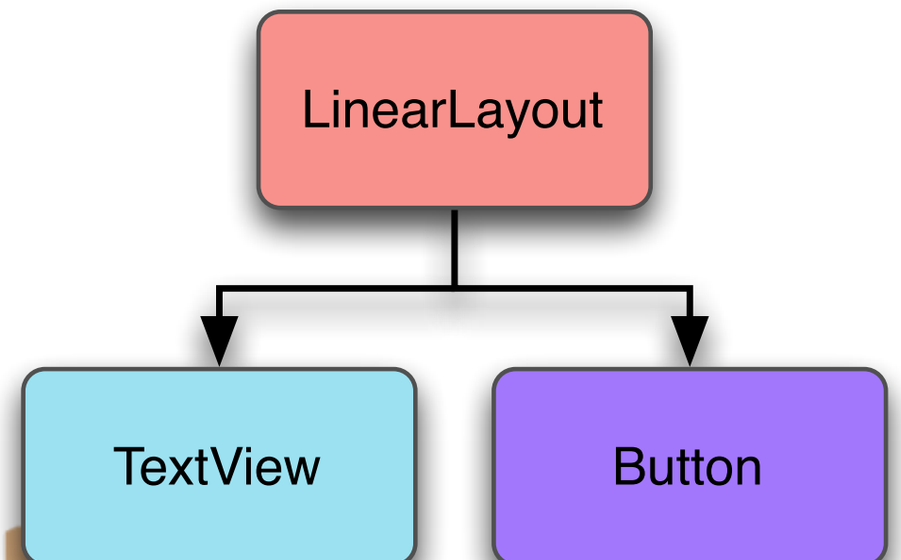
- Simpler hierarchies render faster

http://developer.android.com/guide/topics/ui/overview.html

Android U/I layouts




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

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am a TextView" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am a Button" />

</LinearLayout>
```



/guide/topics/ui/overview.html

Android U/I Layouts

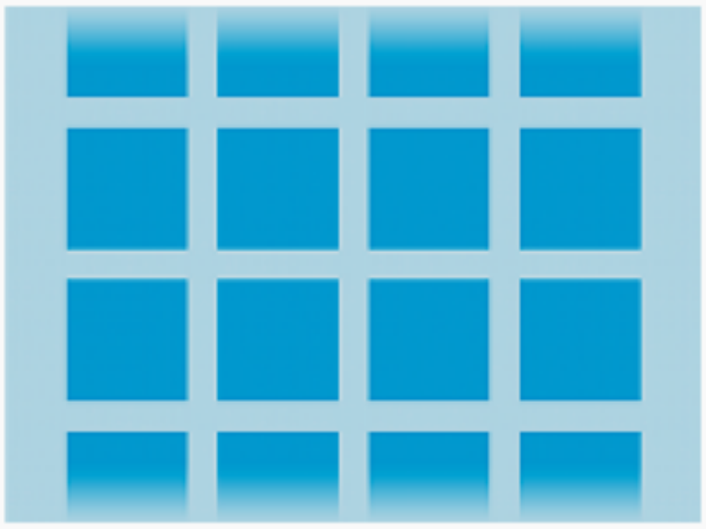
Linear Layout

Relative Layout

Web View

List View

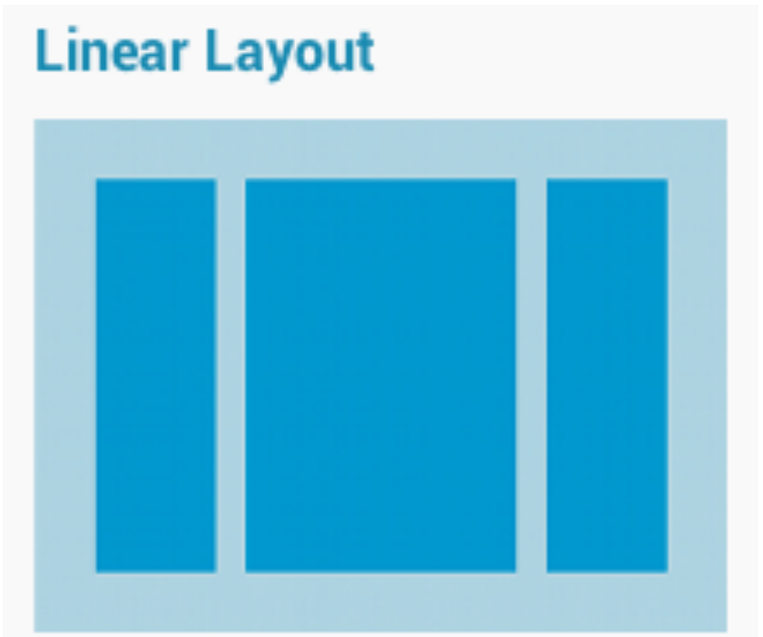
Grid View

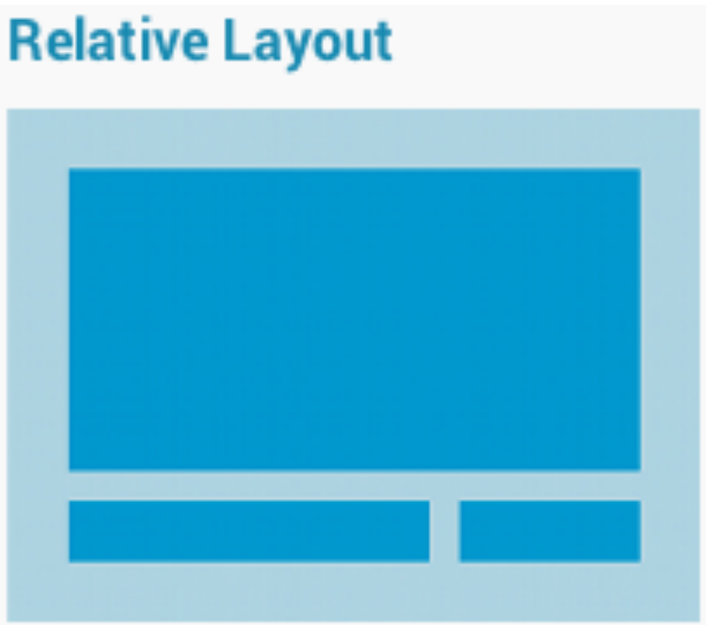


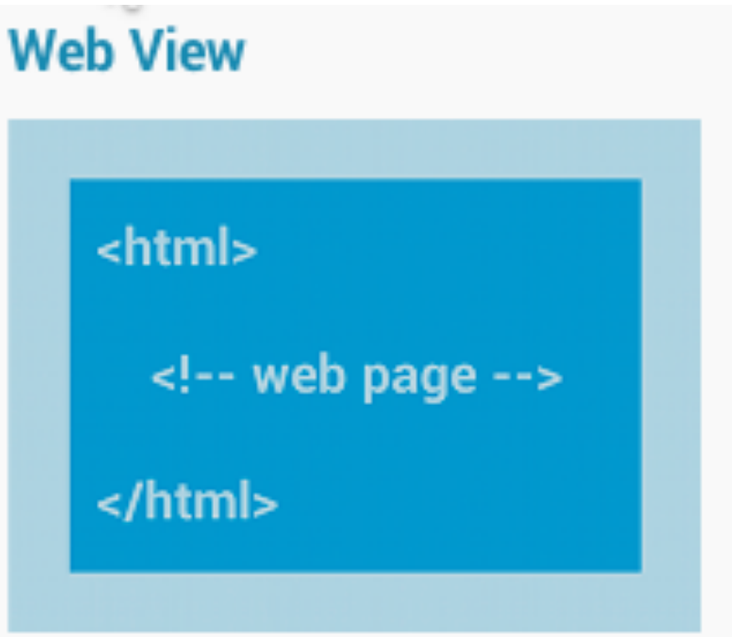
http://developer.android.com/guide/topics/ui/declaring-layout.html

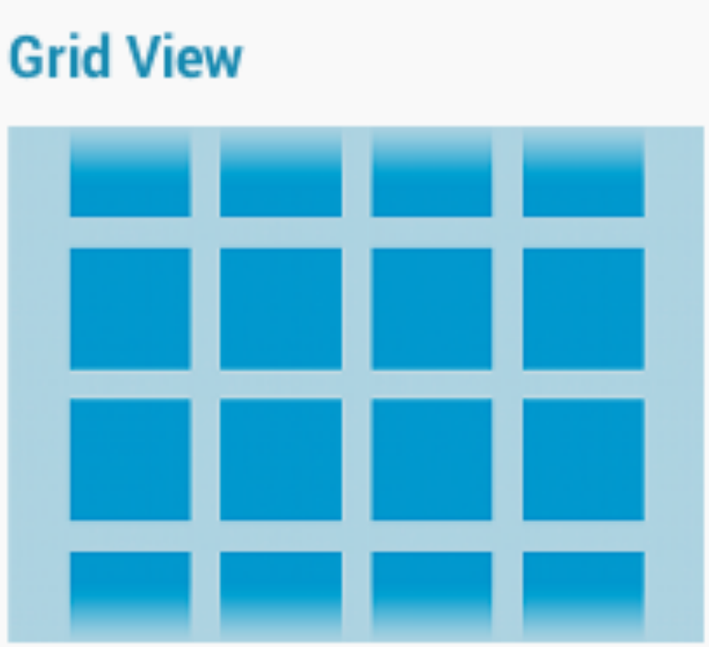
Android U/I Layouts

- A “Layout” is a ViewGroup that provides visual structure for U/I elements



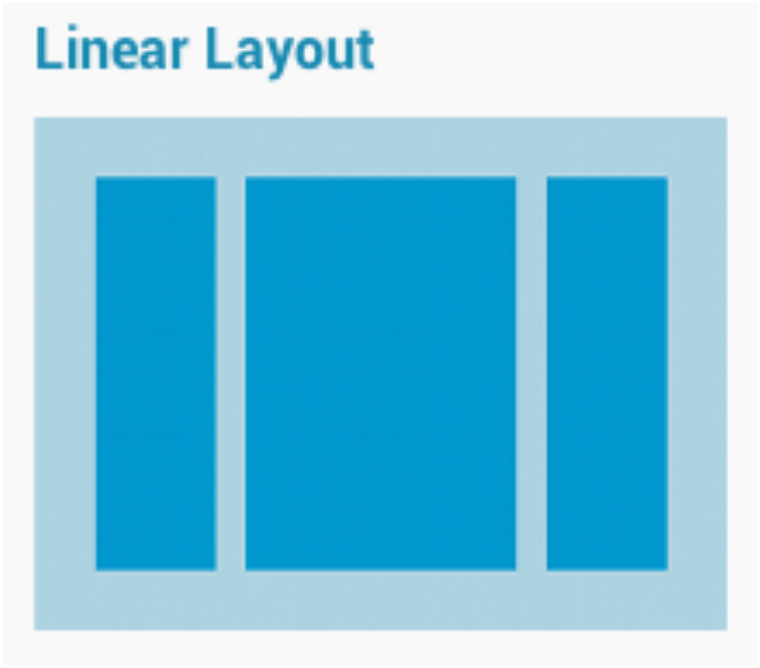






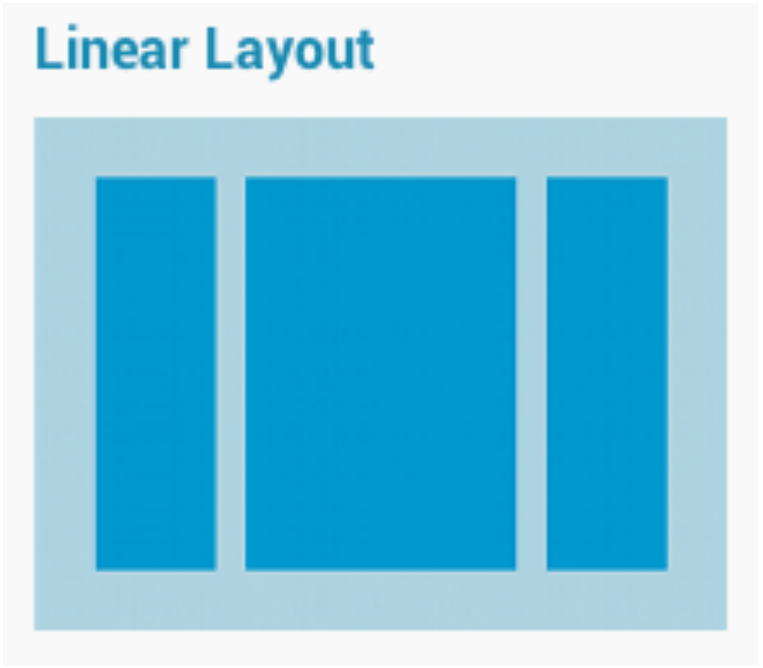
http://developer.android.com/guide/topics/ui/declaring-layout.html

Android U/I Layouts



http://developer.android.com/guide/topics/ui/declaring-layout.html

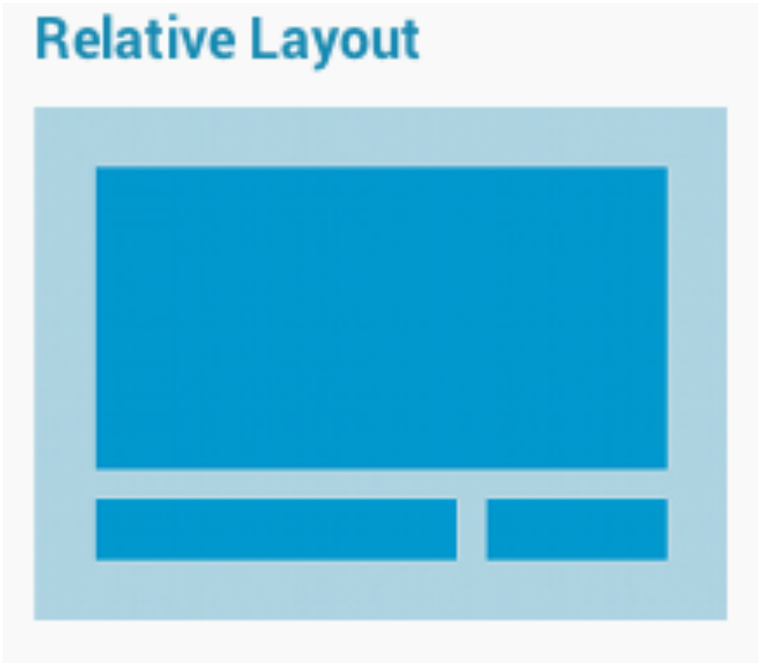
Android U/I Layouts



- Organizes children into horizontal or vertical rows
- Rows can have different relative widths
- A scroll bar is added if necessary

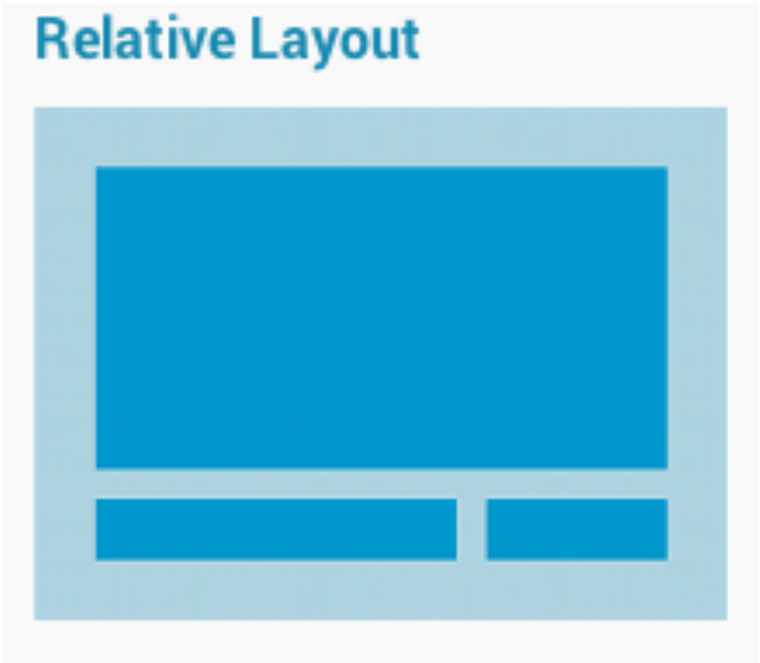
http://developer.android.com/guide/topics/ui/declaring-layout.html

Android U/I Layouts



http://developer.android.com/guide/topics/ui/declaring-layout.html

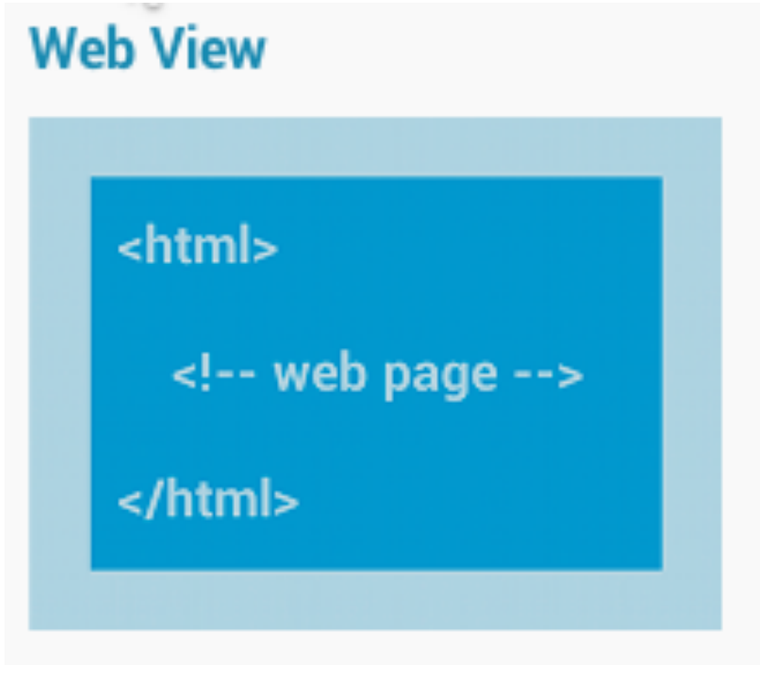
Android U/I Layouts



- Organizes children into containers that maintain relative positions
- Good for managing changing screen sizes
- For example, child B must be to the left of child C and both must be below child A

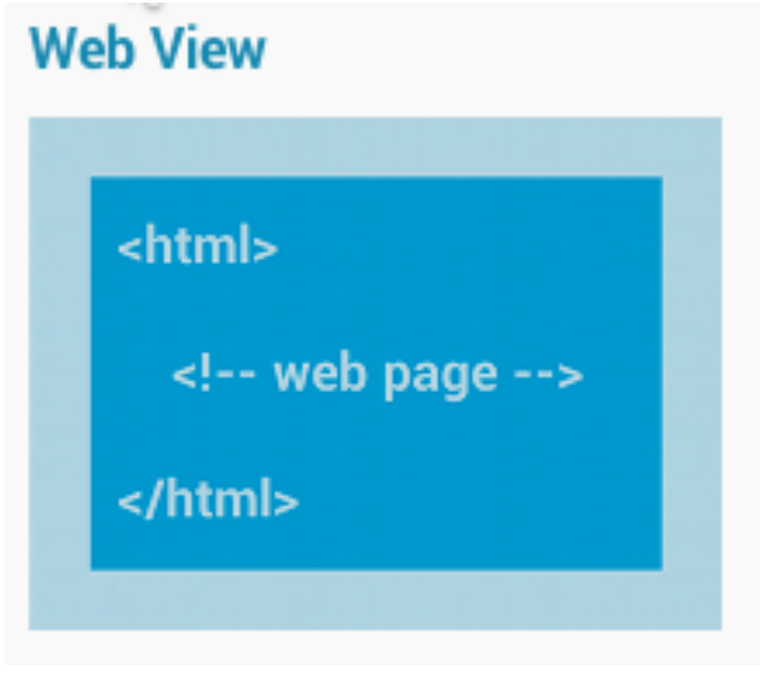
http://developer.android.com/guide/topics/ui/declaring-layout.html

Android U/I Layouts



http://developer.android.com/guide/topics/ui/declaring-layout.html

Android U/I Layouts



- Displays web pages

http://developer.android.com/guide/topics/ui/declaring-layout.html

Android U/I Layouts

http://developer.android.com/guide/topics/ui/declaring-layout.html

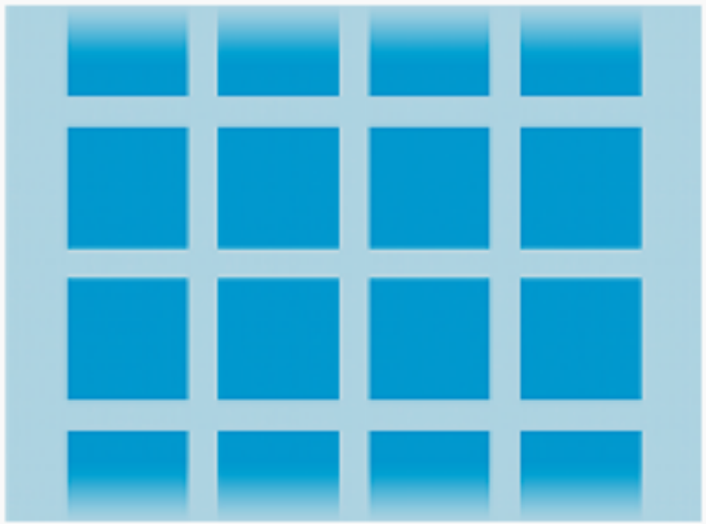
Android U/I Layouts

- Displays a scrolling single column list
- Requires an “Adapter”
- Adapter gets data for the ListView
- ListView makes an entry for every element in the data
- Adapter layouts the individual data points
- Good for data of unknown length
  - For example, a contact list

http://developer.android.com/guide/topics/ui/declaring-layout.html

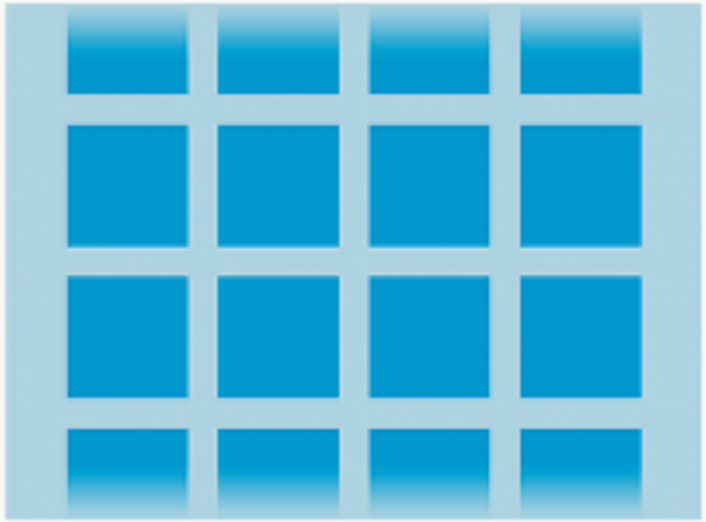
Android U/I Layouts

Grid View



http://developer.android.com/guide/topics/ui/declaring-layout.html

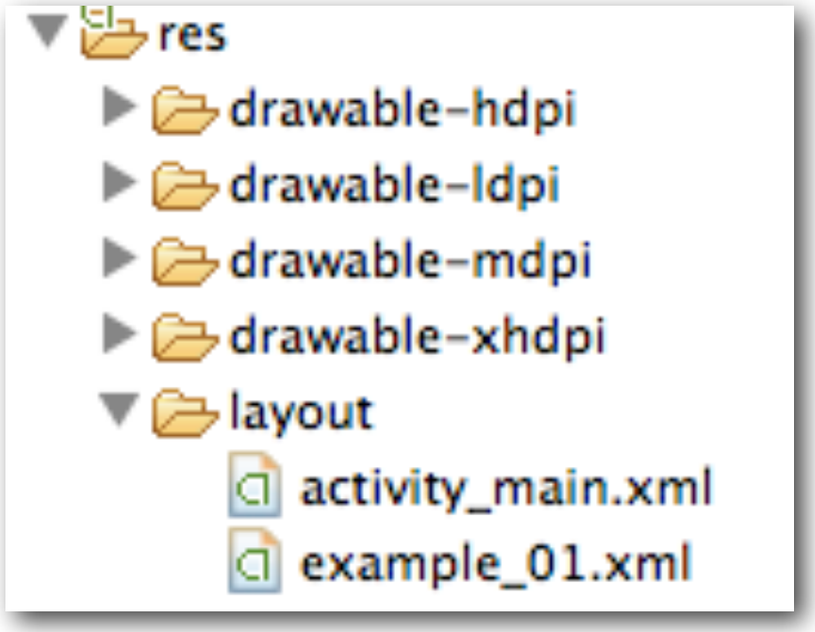
Android U/I Layouts



- Displays a scrolling grid of columns and rows
- Requires an “Adapter”
- Adapter gets data for the ListView
- ListView makes an entry for every element in the data
- Adapter layouts the individual data points
- Good for data of unknown length
  - For example, a list of album covers

http://developer.android.com/guide/topics/ui/declaring-layout.html

Activating Android U/I Layouts



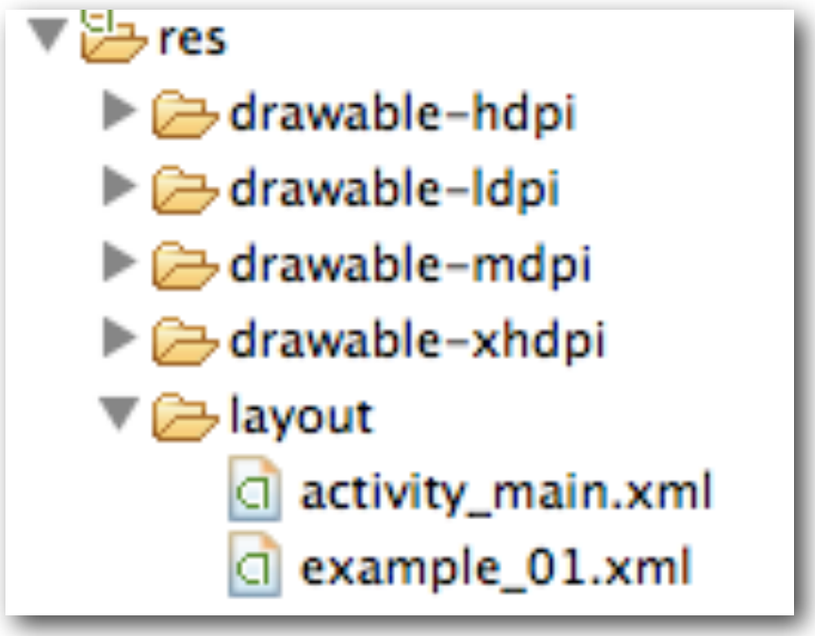

```
public class MainActivity extends Activity {

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

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.example_01);
    }
```

http://developer.android.com/guide/topics/ui/overview.html

Activating Android U/I Layouts

- When you create your U/I with XML you can handle different devices in separate layout files
- After determining what orientation and device you have, you can set the correct Content view
- XML U/Is are compiling into static Java objects




```
public class MainActivity extends Activity {

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

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.example_01);
    }
```

http://developer.android.com/guide/topics/ui/overview.html

Activating Android U/I Layouts

```
<ViewFlipper android:id="@+id/viewFlipper">
    <LinearLayout>
        // Here would go the content of R.layout.imagegraph
    </LinearLayout>
    <LinearLayout>
        // Here would go the content of R.layout.editgraph
    </LinearLayout>
</ViewFlipper>
```

```
ViewFlipper vf = (ViewFlipper) findViewById( R.id.viewFlipper );
vf.showNext();
```

http://stackoverflow.com/questions/7017428/switching-between-2-layouts-in-android-activity

Activating Android U/I Layouts

- To switch between “screens” in an activity look at the “ViewFlipper” class
- ViewFlipper wraps multiple screens in one U/I and let’s you switch between them

```
<ViewFlipper android:id="@+id/viewFlipper">
    <LinearLayout>
        // Here would go the content of R.layout.imagegraph
    </LinearLayout>
    <LinearLayout>
        // Here would go the content of R.layout.editgraph
    </LinearLayout>
</ViewFlipper>
```

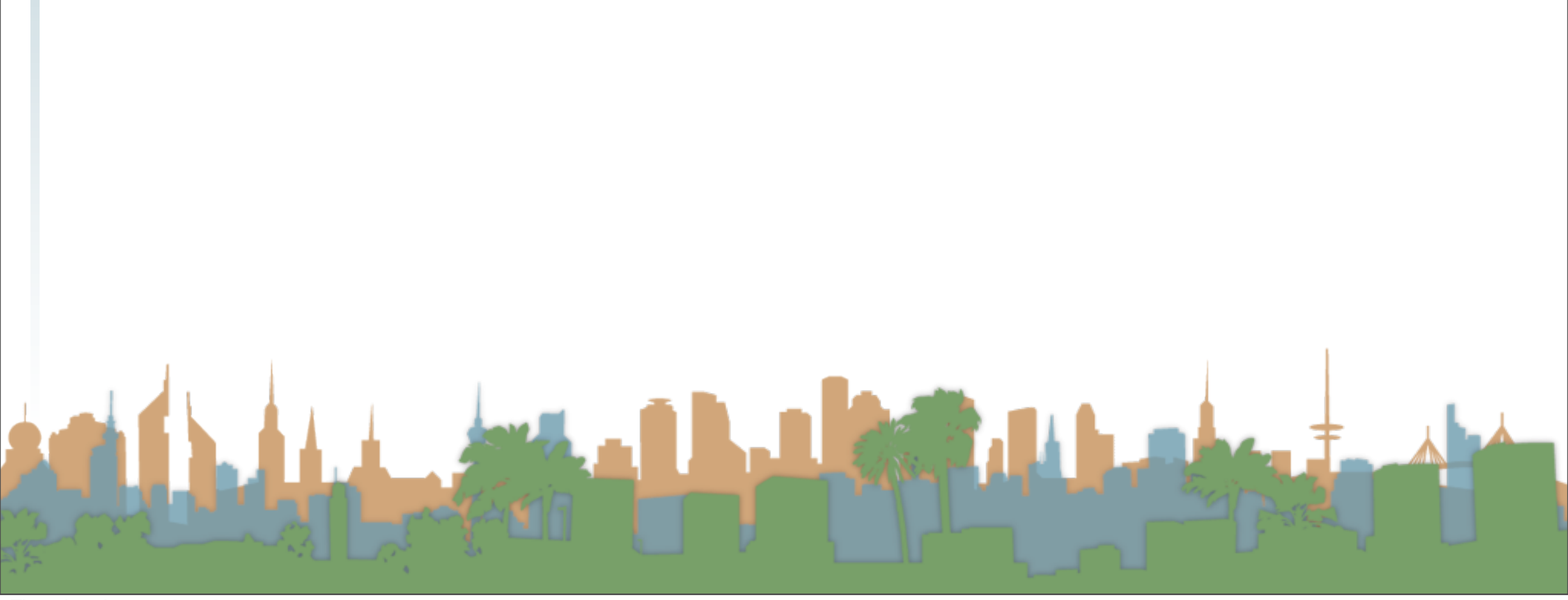
```
ViewFlipper vf = (ViewFlipper) findViewById( R.id.viewFlipper );
vf.showNext();
```

http://stackoverflow.com/questions/7017428/switching-between-2-layouts-in-android-activity

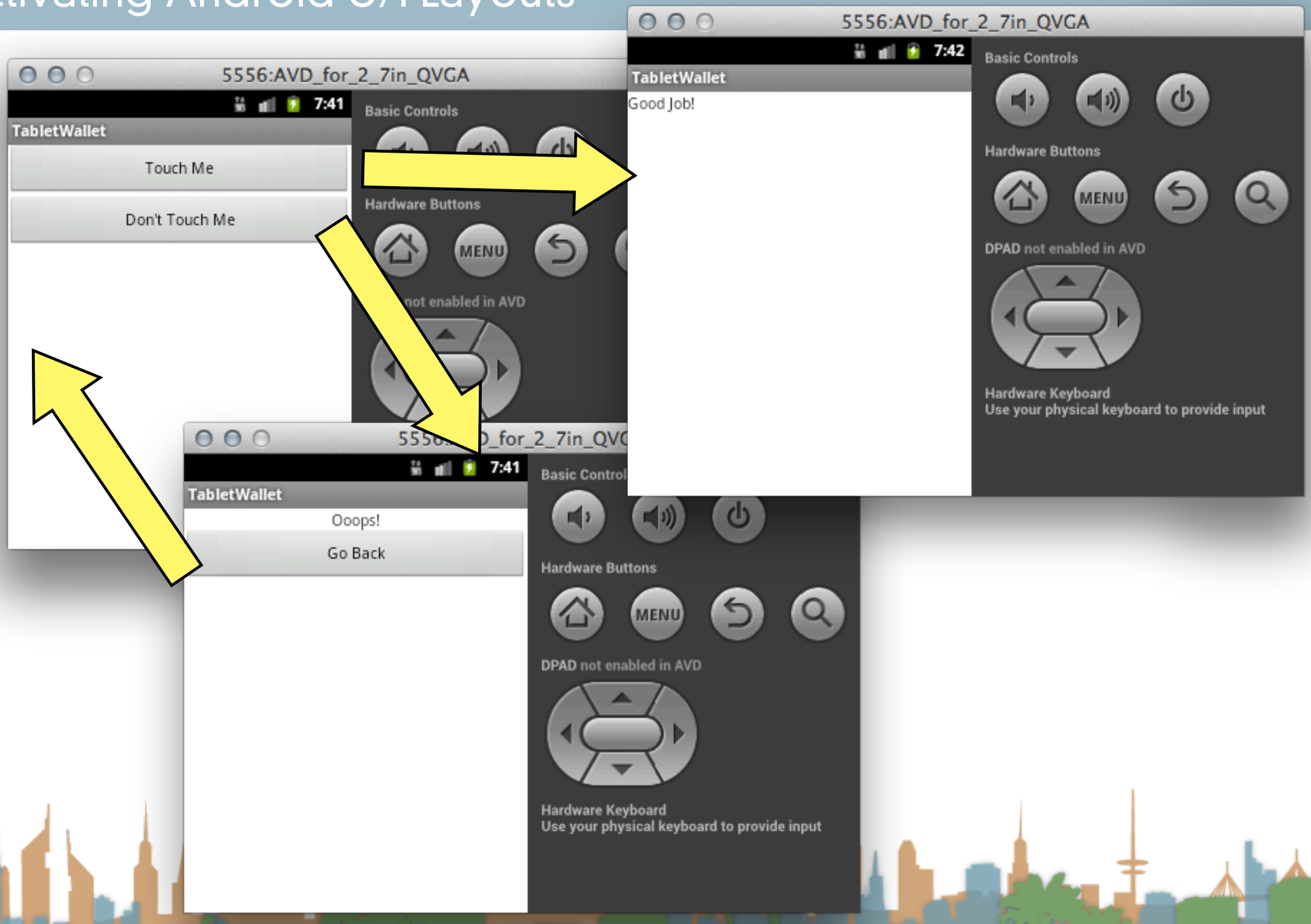
Activating Android U/I Layouts

Activating Android U/I Layouts

- Run demo

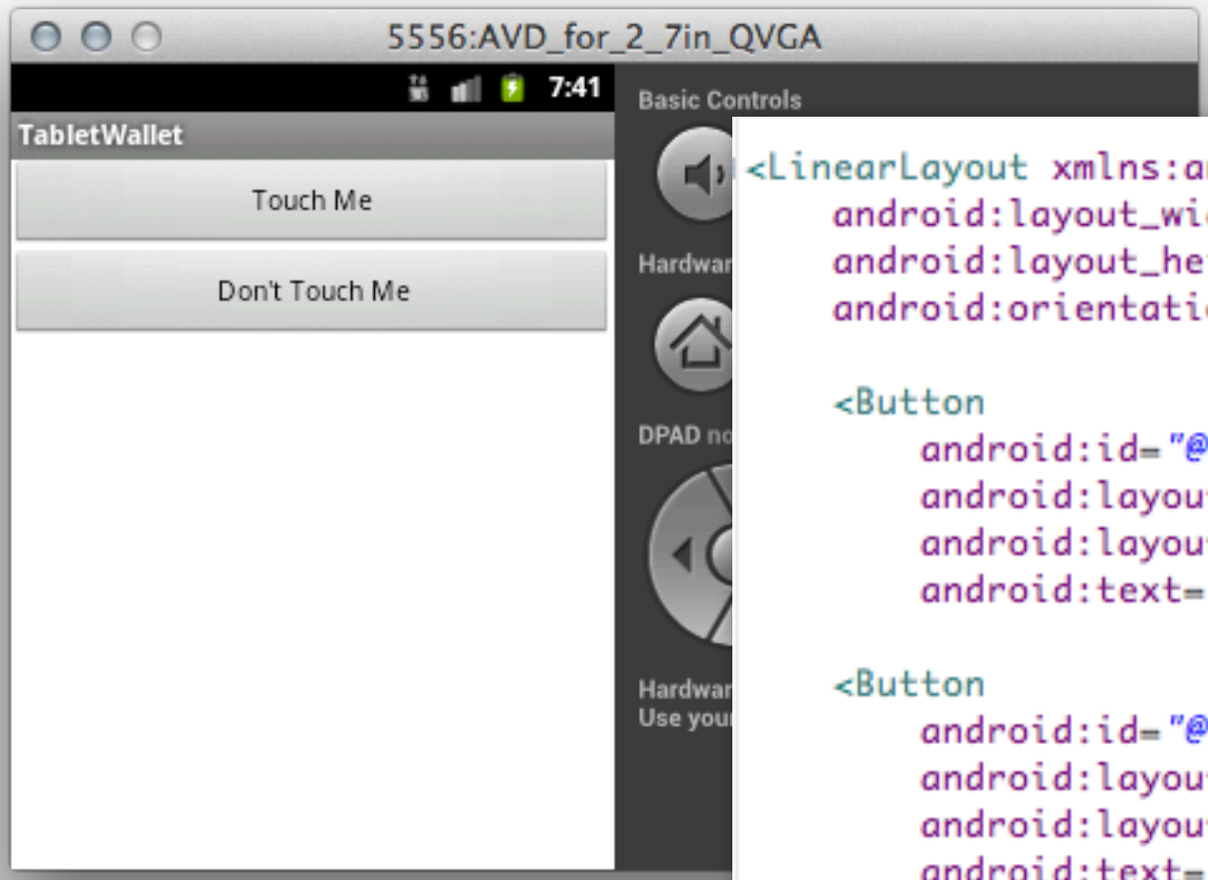


Activating Android U/I Layouts



http://developer.android.com/guide/topics/ui/overview.html

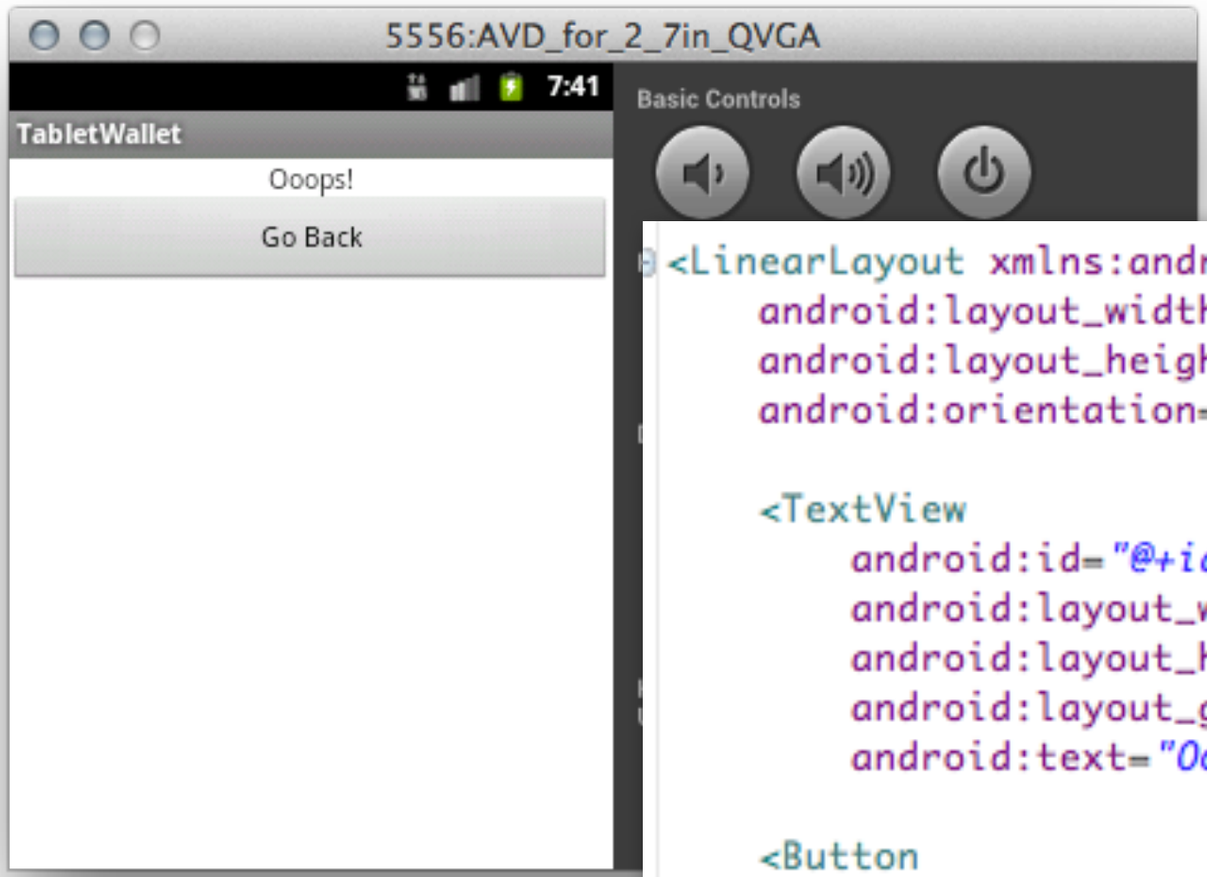
Activating Android U/I Layouts




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

    <Button
        android:id="@+id/touch"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Touch Me" />

    <Button
        android:id="@+id/notouch"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Don&apos;t Touch Me" />

</LinearLayout>
```

http://developer.android.com/guide/topics/ui/overview.html

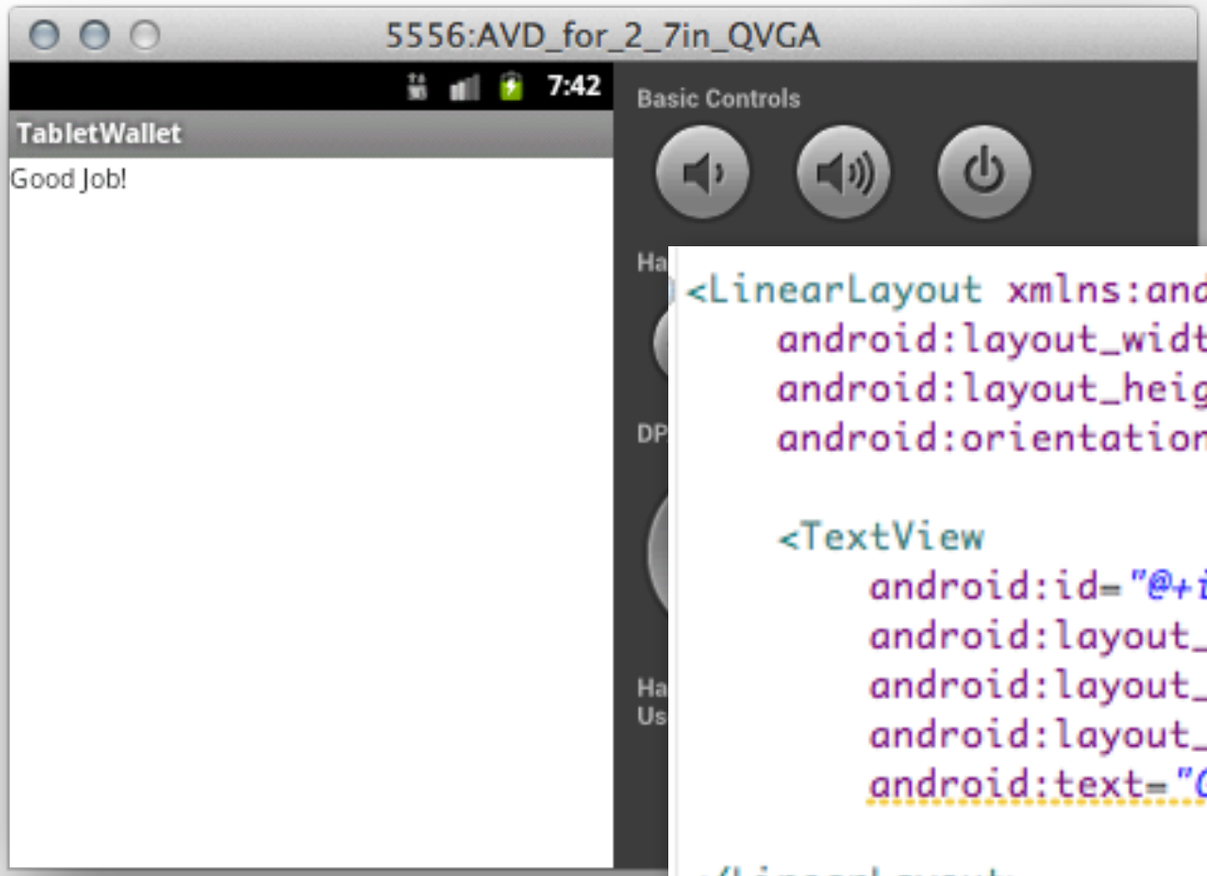
Activating Android U/I Layouts




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

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Ooops!" />

    <Button
        android:id="@+id/back"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Go Back" />

</LinearLayout>
```

http://developer.android.com/guide/topics/ui/overview.html

Activating Android U/I Layouts




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

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Good Job!" />

</LinearLayout>
```

http://developer.android.com/guide/topics/ui/overview.html

Activating Android U/I Layouts

- post_it_sub_bad.xml
- post_it_sub_good.xml
- post_it_sub_main.xml
- post_it_wrapper.xml

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

    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/post_it_sub_main" />

    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/post_it_sub_bad" />

    <include
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        layout="@layout/post_it_sub_good" />

</ViewFlipper>
```

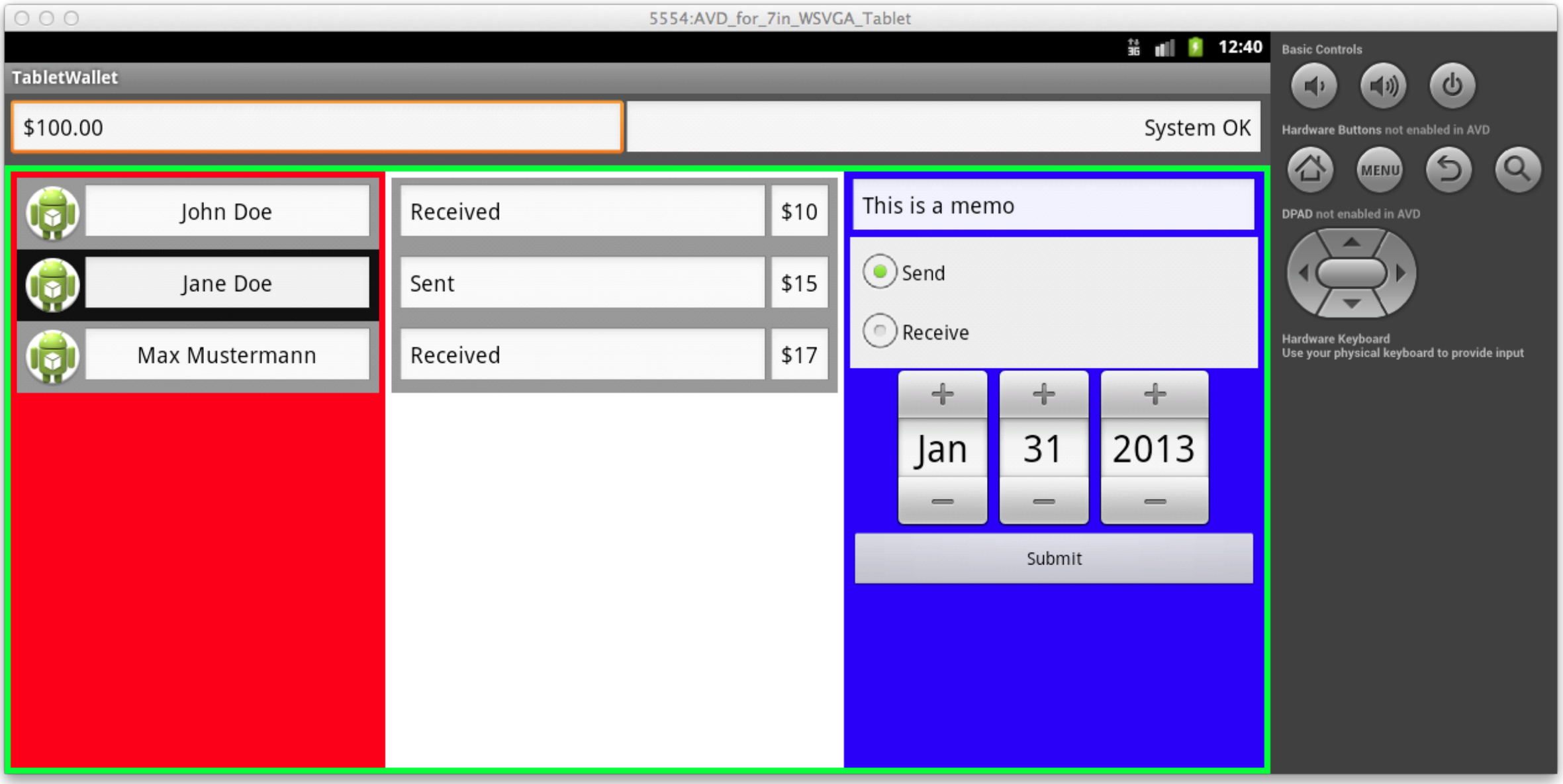
http://developer.android.com/guide/topics/ui/overview.html

Activating Android U/I Layouts


```
public class MainActivity extends Activity {

    private ViewFlipper switcher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.post_it_wrapper);
        switcher = (ViewFlipper) findViewById(R.id.profileSwitcher);

        OnClickListener switchToGood = new OnClickListener() {
            public void onClick(View v) {
                switcher.setDisplayedChild(2);
            }
        };
        OnClickListener switchToBad = new OnClickListener() {
            public void onClick(View v) {
                switcher.setDisplayedChild(1);
            }
        };
        OnClickListener switchToMain = new OnClickListener() {
            public void onClick(View v) {
                switcher.setDisplayedChild(0);
            }
        };

        Button button = (Button)findViewById(R.id.touch);
        button.setOnClickListener(switchToGood);

        button = (Button)findViewById(R.id.notouch);
        button.setOnClickListener(switchToBad);

        button = (Button)findViewById(R.id.back);
        button.setOnClickListener(switchToMain);

    }
```

http://developer.android.com/guide/topics/ui/overview.html

Assignment #3

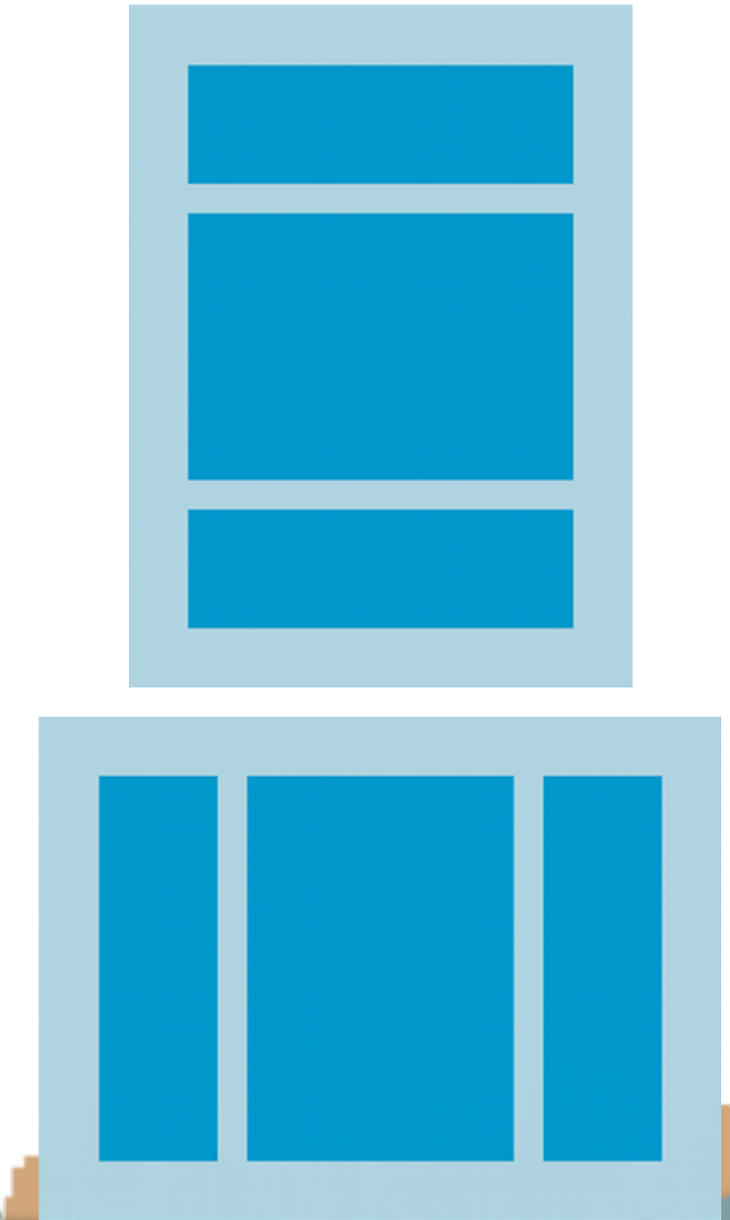


Assignment #3

Assignment #3

- Other things you need to know for Assignment #3
  - LinearLayout
    - H vs V
  - LinearLayout Weights
  - Properties associated with relative Layouts
  - Background Colors
  - Horizontal vs Vertical Scroll bars

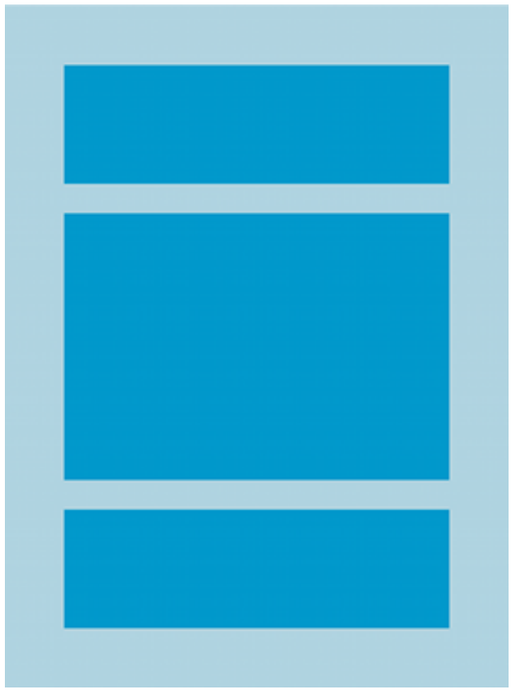
Assignment #3



http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

- LinearLayout Horizontal or Vertical
  - LinearLayout may specify the XML attribute, android:orientation
    - “vertical”



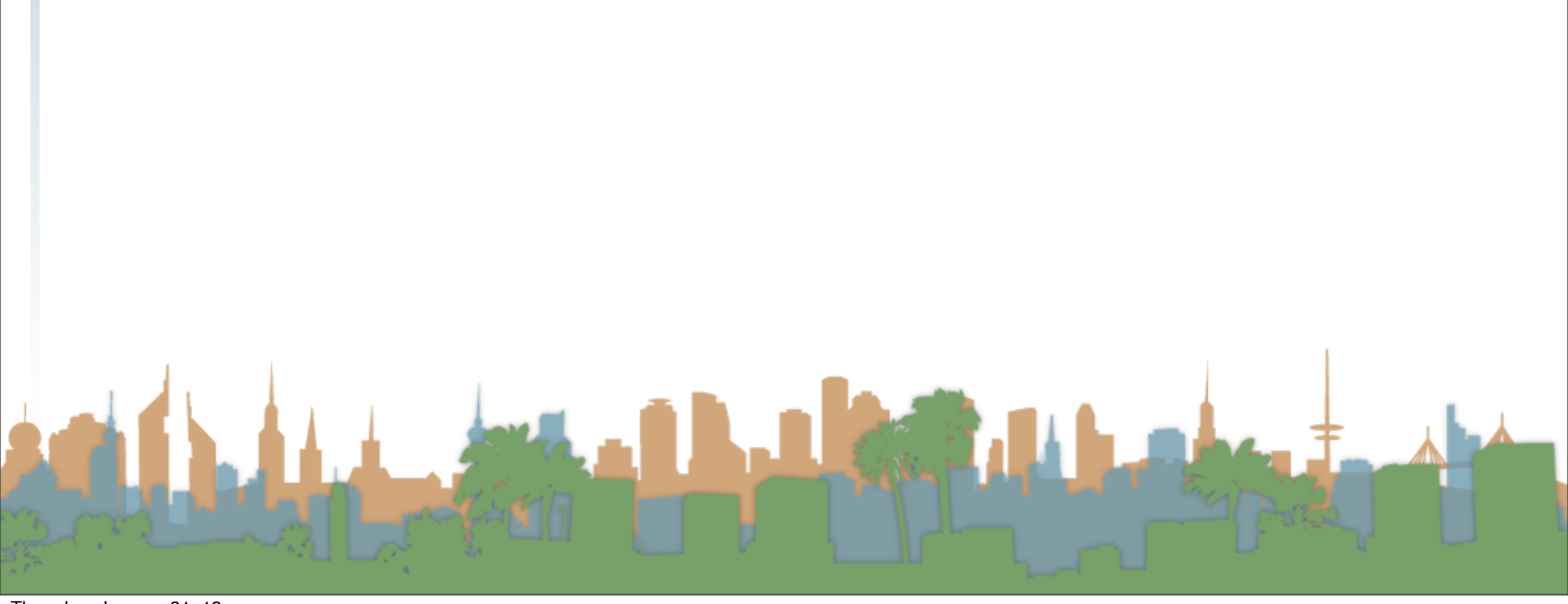
    - “horizontal”

http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

Assignment #3

- Other things you need to know for Assignment #3
  - LinearLayout
    - H vs V
  - LinearLayout Weights
  - Properties associated with RelativeLayouts
  - Background Colors
  - Horizontal vs Vertical Scroll bars

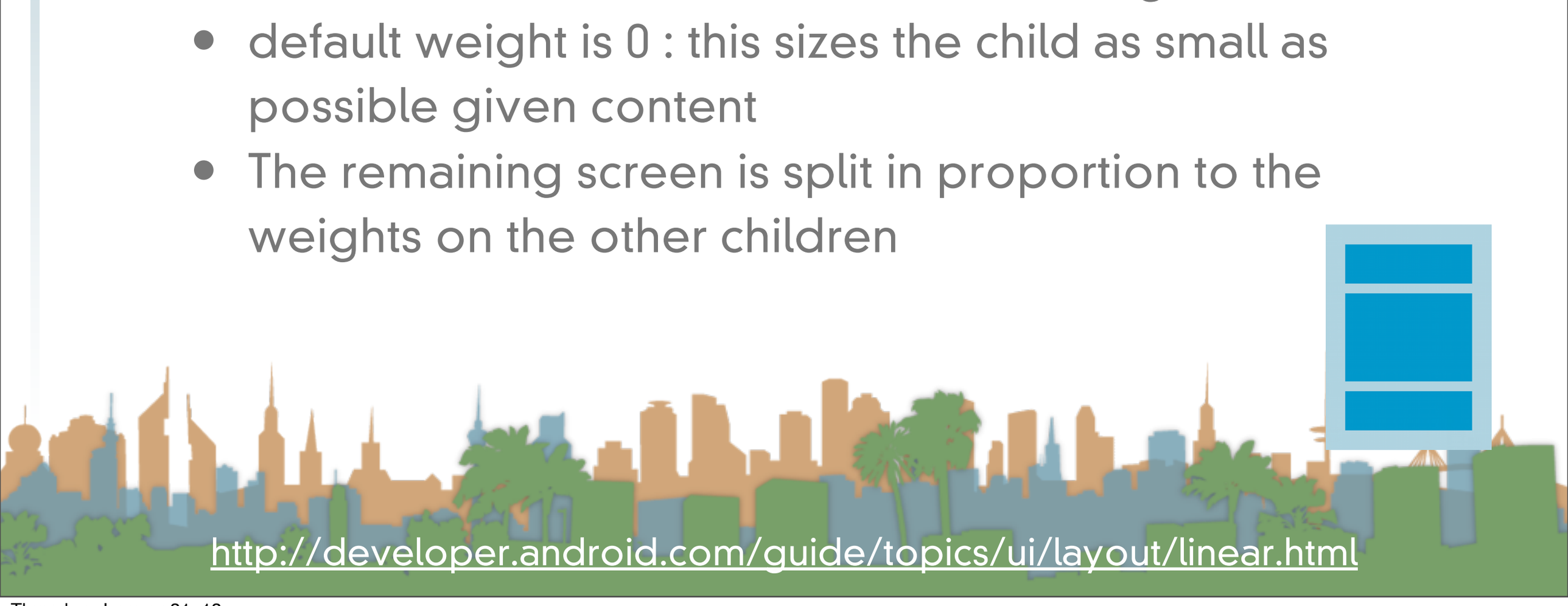


Assignment #3

http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

- LinearLayout Weights
  - children of LinearLayout Weights may use the XML attribute, android:layout_weights
  - This gives Android hints about how to treat your U/I when the screen changes size or rotates
    - The larger the value the more "important" the child is and the more screen real estate it will get.
    - default weight is 0 : this sizes the child as small as possible given content
    - The remaining screen is split in proportion to the weights on the other children



http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

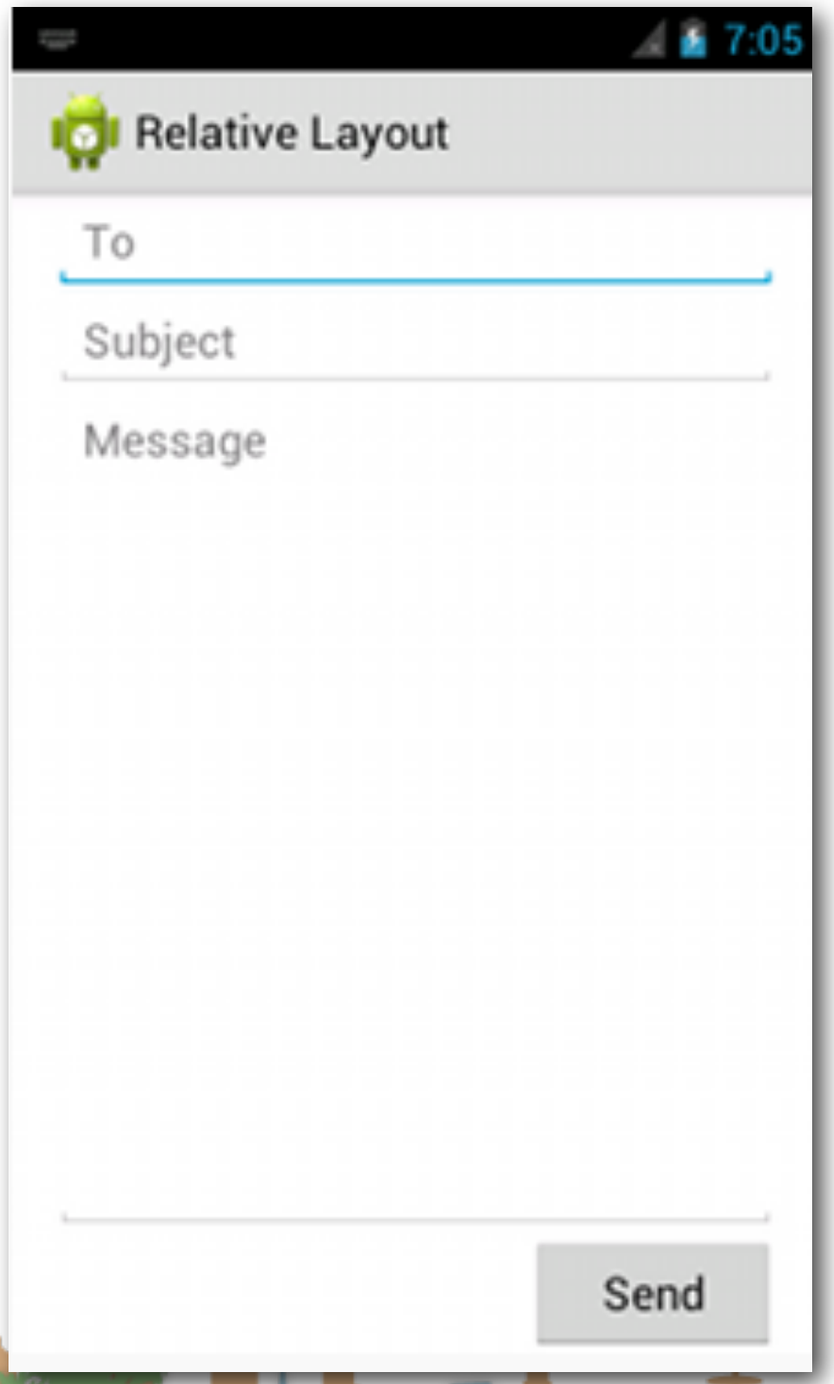
Assignment #3

- LinearLayout Weights: Example
  - 3 text fields
    - A: weight 1
    - B: weight 1
    - C: no weight
  - C will not grow and only occupy the space required by it's content
  - A and B will grow and split the remaining space equally
  - if C's weight was 2
    - C would get 1/2 the screen
    - A and B would get 1/4 each

http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

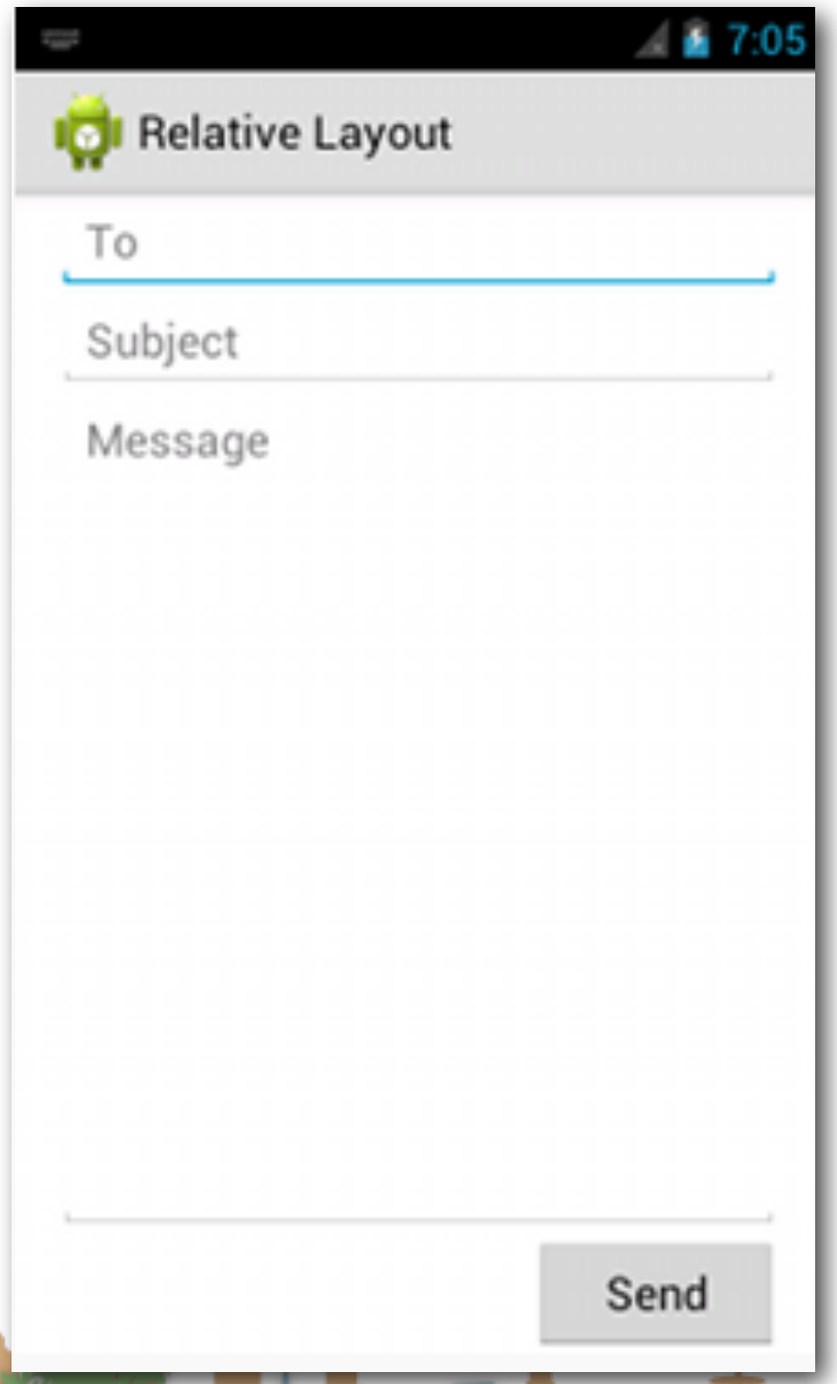


http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

- LinearLayout Weights: Example

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



http://developer.android.com/guide/topics/ui/layout/linear.html

Assignment #3

android:layout_gravity

Standard gravity constant that a child supplies to its parent. Defines how the child view should be positioned, on both the X and Y axes, within its enclosing layout.

Must be one or more (separated by '|') of the following constant values.

Constant	Value	Description
`top`	0x30	Push object to the top of its container, not changing its size.
`bottom`	0x50	Push object to the bottom of its container, not changing its size.
`left`	0x03	Push object to the left of its container, not changing its size.
`right`	0x05	Push object to the right of its container, not changing its size.
`center_vertical`	0x10	Place object in the vertical center of its container, not changing its size.
`fill_vertical`	0x70	Grow the vertical size of the object if needed so it completely fills its container.
`center_horizontal`	0x01	Place object in the horizontal center of its container, not changing its size.
`fill_horizontal`	0x07	Grow the horizontal size of the object if needed so it completely fills its container.
`center`	0x11	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
`fill`	0x77	Grow the horizontal and vertical size of the object if needed so it completely fills its container.
`clip_vertical`	0x80	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
`clip_horizontal`	0x08	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
`start`	0x00800003	Push object to the beginning of its container, not changing its size.
`end`	0x00800005	Push object to the end of its container, not changing its size.

http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html

Assignment #3

- Other attributes for LinearLayout beyond weight

android:layout_gravity

Standard gravity constant that a child supplies to its parent. Defines how the child view should be positioned, on both the X and Y axes, within its enclosing layout.

Must be one or more (separated by '|') of the following constant values.

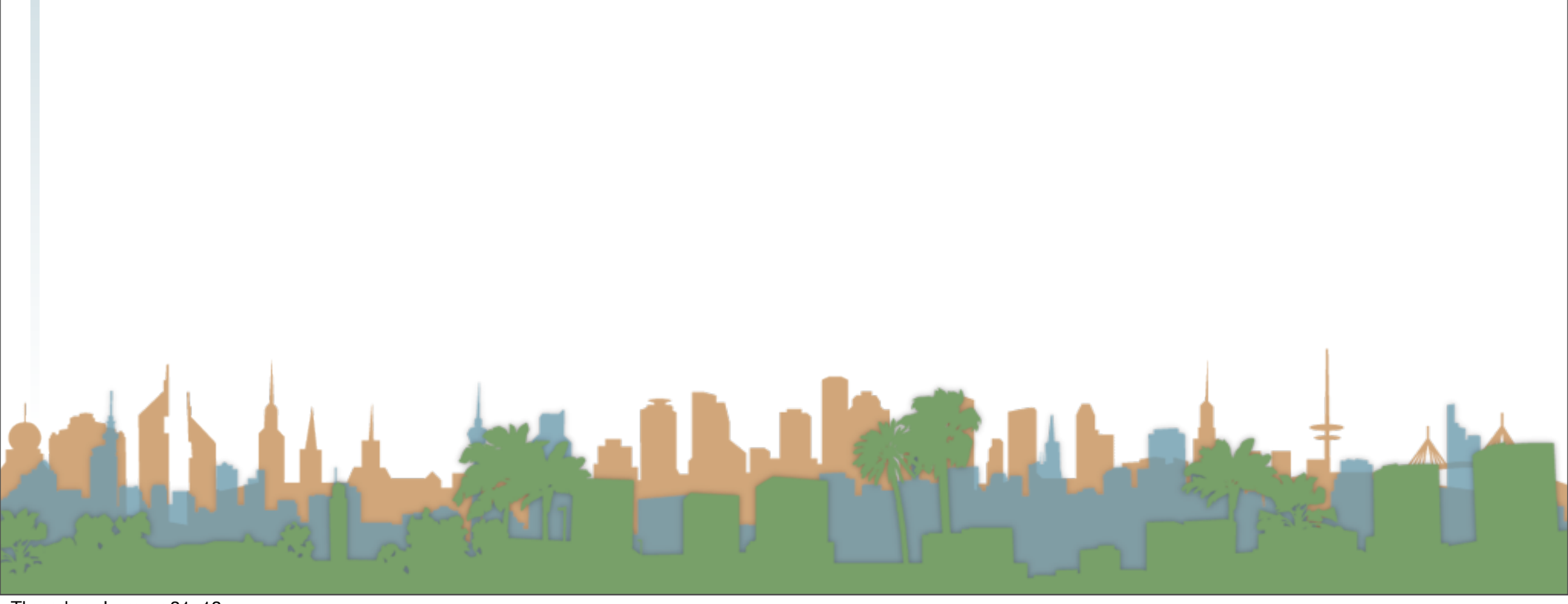
Constant	Value	Description
`top`	0x30	Push object to the top of its container, not changing its size.
`bottom`	0x50	Push object to the bottom of its container, not changing its size.
`left`	0x03	Push object to the left of its container, not changing its size.
`right`	0x05	Push object to the right of its container, not changing its size.
`center_vertical`	0x10	Place object in the vertical center of its container, not changing its size.
`fill_vertical`	0x70	Grow the vertical size of the object if needed so it completely fills its container.
`center_horizontal`	0x01	Place object in the horizontal center of its container, not changing its size.
`fill_horizontal`	0x07	Grow the horizontal size of the object if needed so it completely fills its container.
`center`	0x11	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
`fill`	0x77	Grow the horizontal and vertical size of the object if needed so it completely fills its container.
`clip_vertical`	0x80	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
`clip_horizontal`	0x08	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
`start`	0x00800003	Push object to the beginning of its container, not changing its size.
`end`	0x00800005	Push object to the end of its container, not changing its size.

http://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html

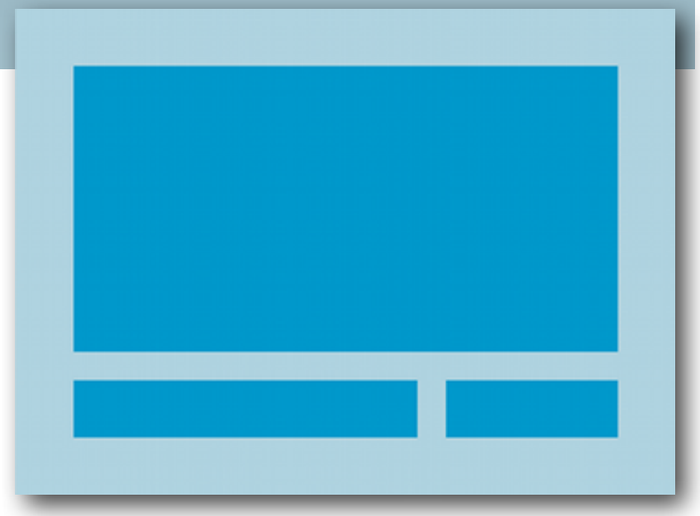
Assignment #3

Assignment #3

- Other things you need to know for Assignment #3
  - LinearLayout
    - H vs V
  - LinearLayout Weights
  - Properties associated with RelativeLayouts
  - Background Colors
  - Horizontal vs Vertical Scroll bars

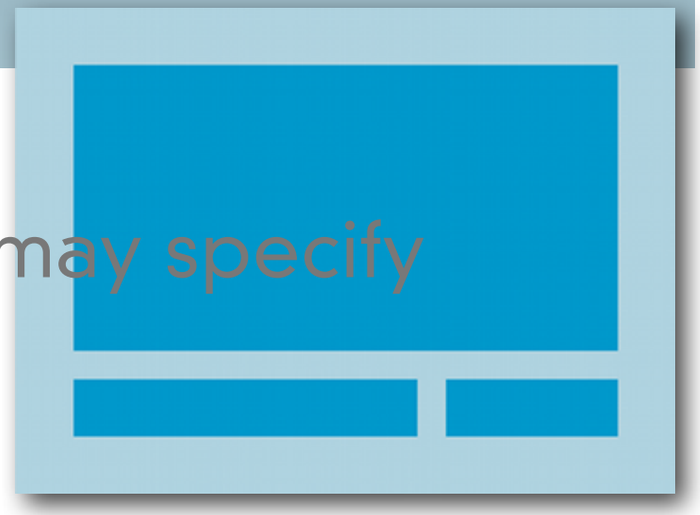


Assignment #3



http://developer.android.com/guide/topics/ui/layout/relative.html

Assignment #3

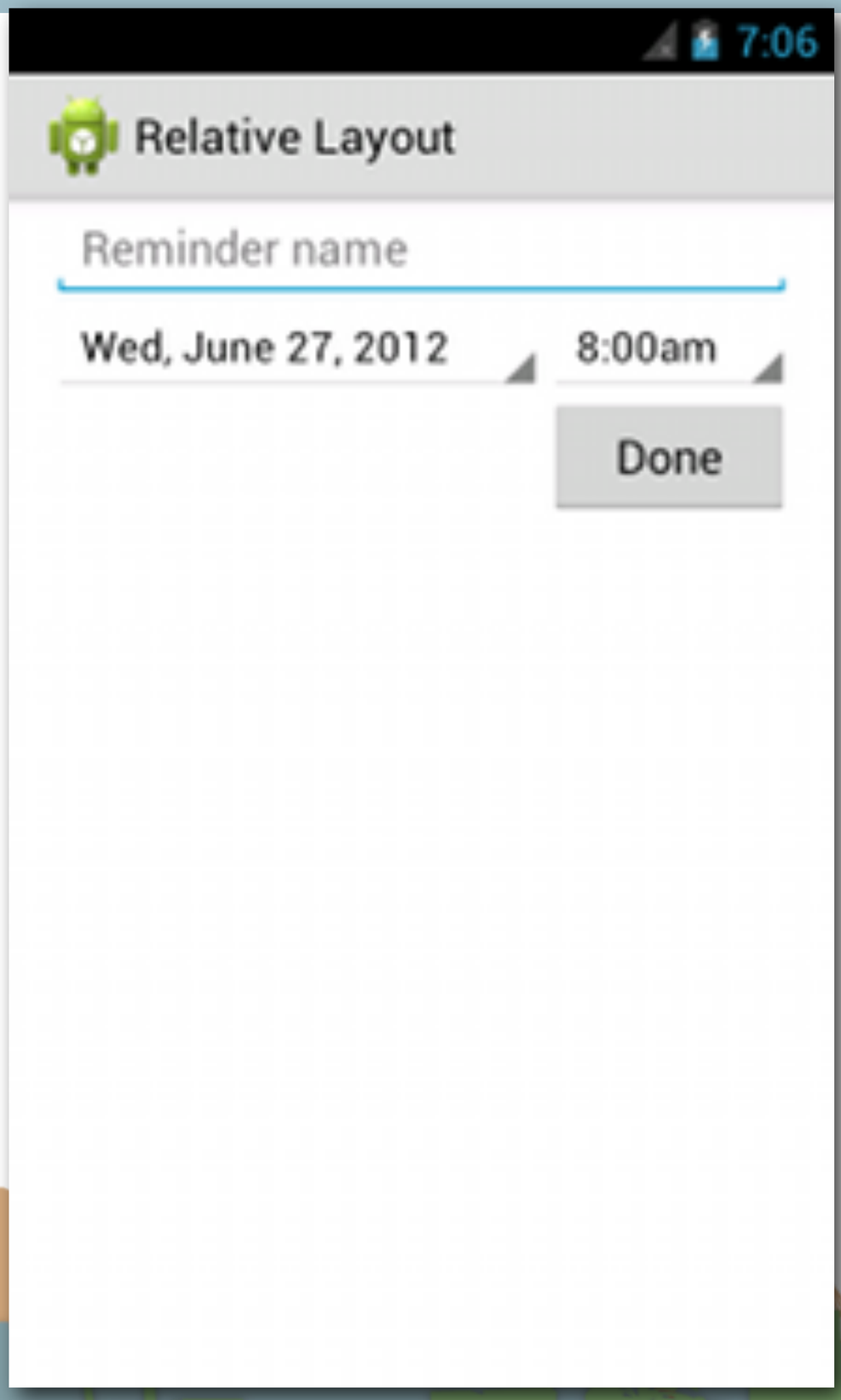


- RelativeLayout
  - Children of a RelativeLayout ViewGroup may specify the XML attributes to control layout, e.g.:
    - “android:layout_alignParentTop”
      - If "true", makes the top edge of this view match the top edge of the parent.
    - “android:layout_centerVertical
      - If "true", centers this child vertically within its parent.
    - “android:layout_below
      - Positions the top edge of this view below the view specified with a resource ID, possibly a sibling
    - “android:layout_toRightOf
      - Positions the left edge of this view to the right of the view specified with a resource ID, possibly a sibling

http://developer.android.com/guide/topics/ui/layout/relative.html

Assignment #3

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

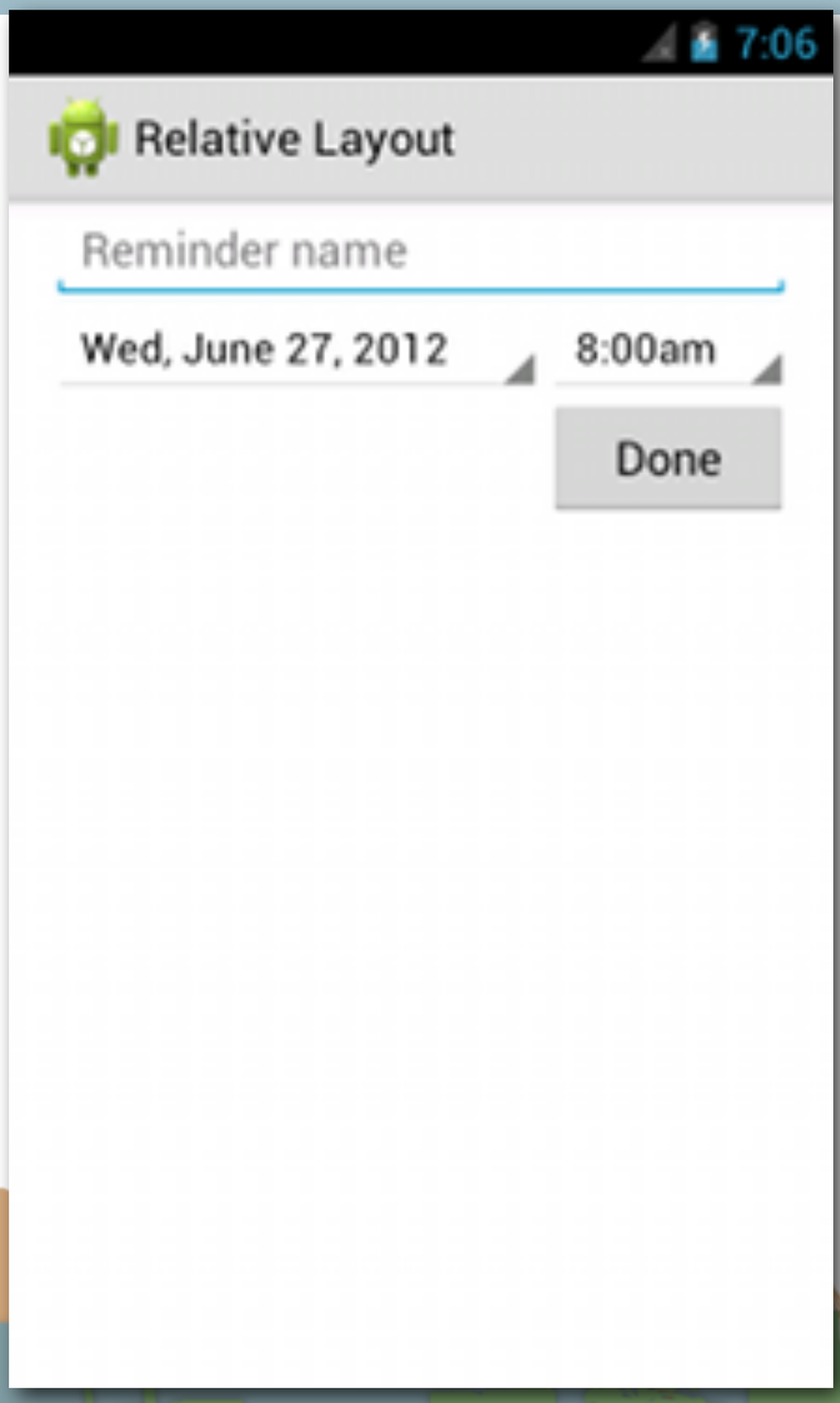


http://developer.android.com/guide/topics/ui/layout/relative.html

Assignment #3

- RelativeLayout Example

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



http://developer.android.com/guide/topics/ui/layout/relative.html

Assignment #3

XML Attributes		
Attribute Name	*Related Method*	*Description*
android:layout_above		Positions the bottom edge of this view above the given anchor view ID.
android:layout_alignBaseline		Positions the baseline of this view on the baseline of the given anchor view ID.
android:layout_alignBottom		Makes the bottom edge of this view match the bottom edge of the given anchor view ID.
android:layout_alignEnd		Makes the end edge of this view match the end edge of the given anchor view ID.
android:layout_alignLeft		Makes the left edge of this view match the left edge of the given anchor view ID.
android:layout_alignParentBottom		If true, makes the bottom edge of this view match the bottom edge of the parent.
android:layout_alignParentEnd		If true, makes the end edge of this view match the end edge of the parent.
android:layout_alignParentLeft		If true, makes the left edge of this view match the left edge of the parent.
android:layout_alignParentRight		If true, makes the right edge of this view match the right edge of the parent.
android:layout_alignParentStart		If true, makes the start edge of this view match the start edge of the parent.
android:layout_alignParentTop		If true, makes the top edge of this view match the top edge of the parent.
android:layout_alignRight		Makes the right edge of this view match the right edge of the given anchor view ID.
android:layout_alignStart		Makes the start edge of this view match the start edge of the given anchor view ID.
android:layout_alignTop		Makes the top edge of this view match the top edge of the given anchor view ID.
android:layout_alignWithParentIfMissing		If set to true, the parent will be used as the anchor when the anchor cannot be be found for layout_toLeftOf, layout_toRightOf, etc.
android:layout_below		Positions the top edge of this view below the given anchor view ID.
android:layout_centerHorizontal		If true, centers this child horizontally within its parent.
android:layout_centerInParent		If true, centers this child horizontally and vertically within its parent.
android:layout_centerVertical		If true, centers this child vertically within its parent.
android:layout_toEndOf		Positions the start edge of this view to the end of the given anchor view ID.
android:layout_toLeftOf		Positions the right edge of this view to the left of the given anchor view ID.
android:layout_toRightOf		Positions the left edge of this view to the right of the given anchor view ID.
android:layout_toStartOf		Positions the end edge of this view to the start of the given anchor view ID.

http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html

Assignment #3

- RelativeLayout Example

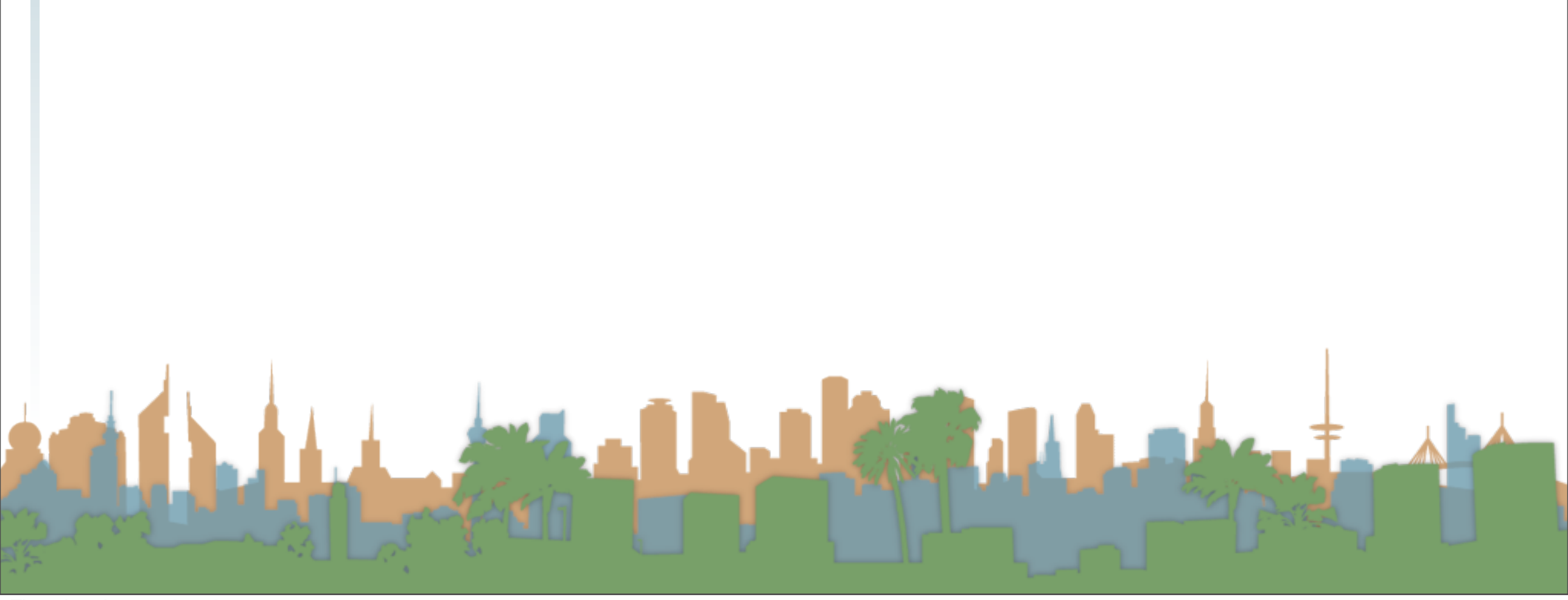
XML Attributes		
Attribute Name	*Related Method*	*Description*
android:layout_above		Positions the bottom edge of this view above the given anchor view ID.
android:layout_alignBaseline		Positions the baseline of this view on the baseline of the given anchor view ID.
android:layout_alignBottom		Makes the bottom edge of this view match the bottom edge of the given anchor view ID.
android:layout_alignEnd		Makes the end edge of this view match the end edge of the given anchor view ID.
android:layout_alignLeft		Makes the left edge of this view match the left edge of the given anchor view ID.
android:layout_alignParentBottom		If true, makes the bottom edge of this view match the bottom edge of the parent.
android:layout_alignParentEnd		If true, makes the end edge of this view match the end edge of the parent.
android:layout_alignParentLeft		If true, makes the left edge of this view match the left edge of the parent.
android:layout_alignParentRight		If true, makes the right edge of this view match the right edge of the parent.
android:layout_alignParentStart		If true, makes the start edge of this view match the start edge of the parent.
android:layout_alignParentTop		If true, makes the top edge of this view match the top edge of the parent.
android:layout_alignRight		Makes the right edge of this view match the right edge of the given anchor view ID.
android:layout_alignStart		Makes the start edge of this view match the start edge of the given anchor view ID.
android:layout_alignTop		Makes the top edge of this view match the top edge of the given anchor view ID.
android:layout_alignWithParentIfMissing		If set to true, the parent will be used as the anchor when the anchor cannot be be found for layout_toLeftOf, layout_toRightOf, etc.
android:layout_below		Positions the top edge of this view below the given anchor view ID.
android:layout_centerHorizontal		If true, centers this child horizontally within its parent.
android:layout_centerInParent		If true, centers this child horizontally and vertically within its parent.
android:layout_centerVertical		If true, centers this child vertically within its parent.
android:layout_toEndOf		Positions the start edge of this view to the end of the given anchor view ID.
android:layout_toLeftOf		Positions the right edge of this view to the left of the given anchor view ID.
android:layout_toRightOf		Positions the left edge of this view to the right of the given anchor view ID.
android:layout_toStartOf		Positions the end edge of this view to the start of the given anchor view ID.

http://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html

Assignment #3

Assignment #3

- Other things you need to know for Assignment #3
  - LinearLayout
    - H vs V
  - LinearLayout Weights
  - Properties associated with RelativeLayouts
  - Background Colors



Assignment #3

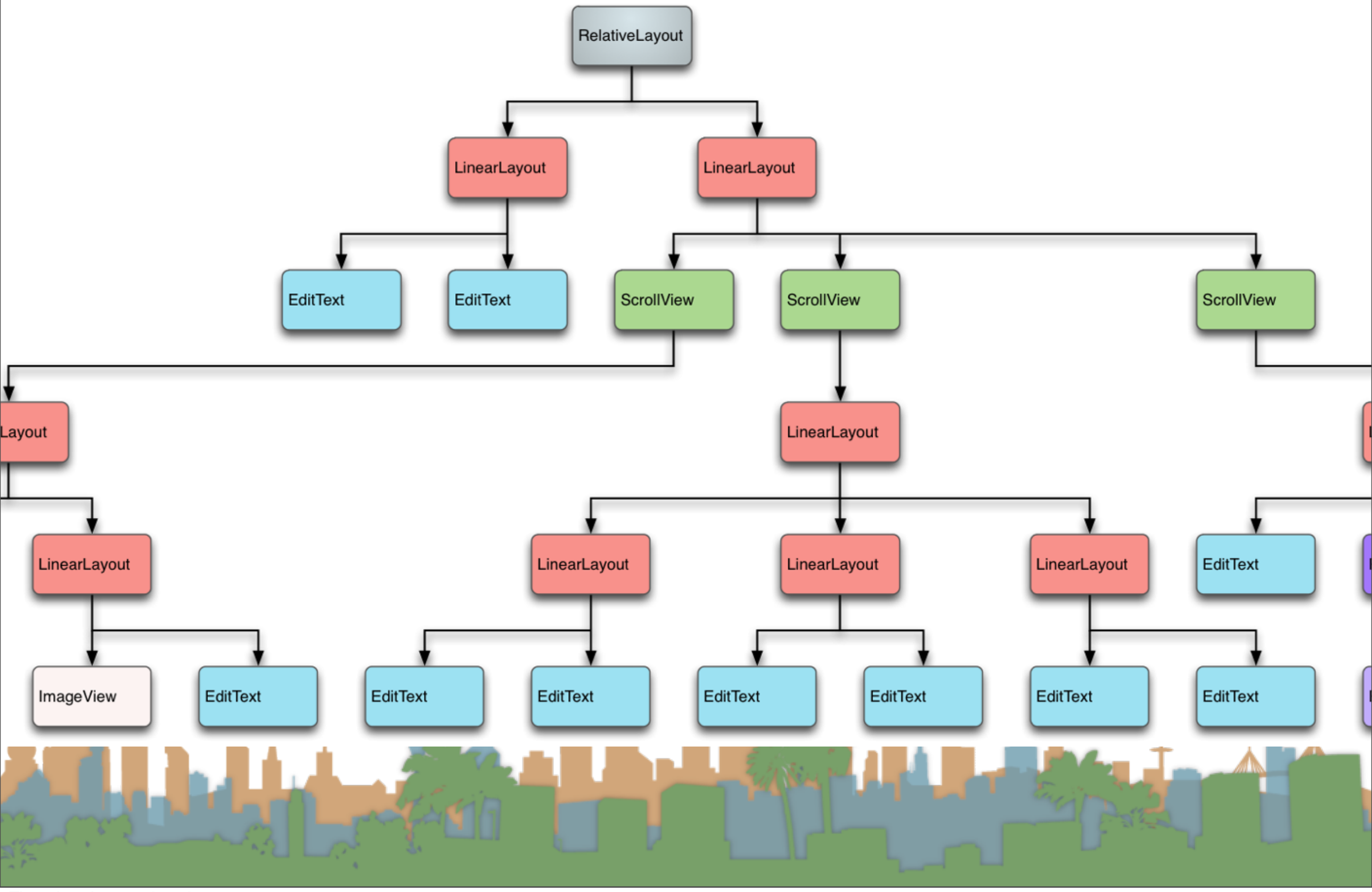
http://developer.android.com/reference/android/view/View.html#attr_android:background

Assignment #3

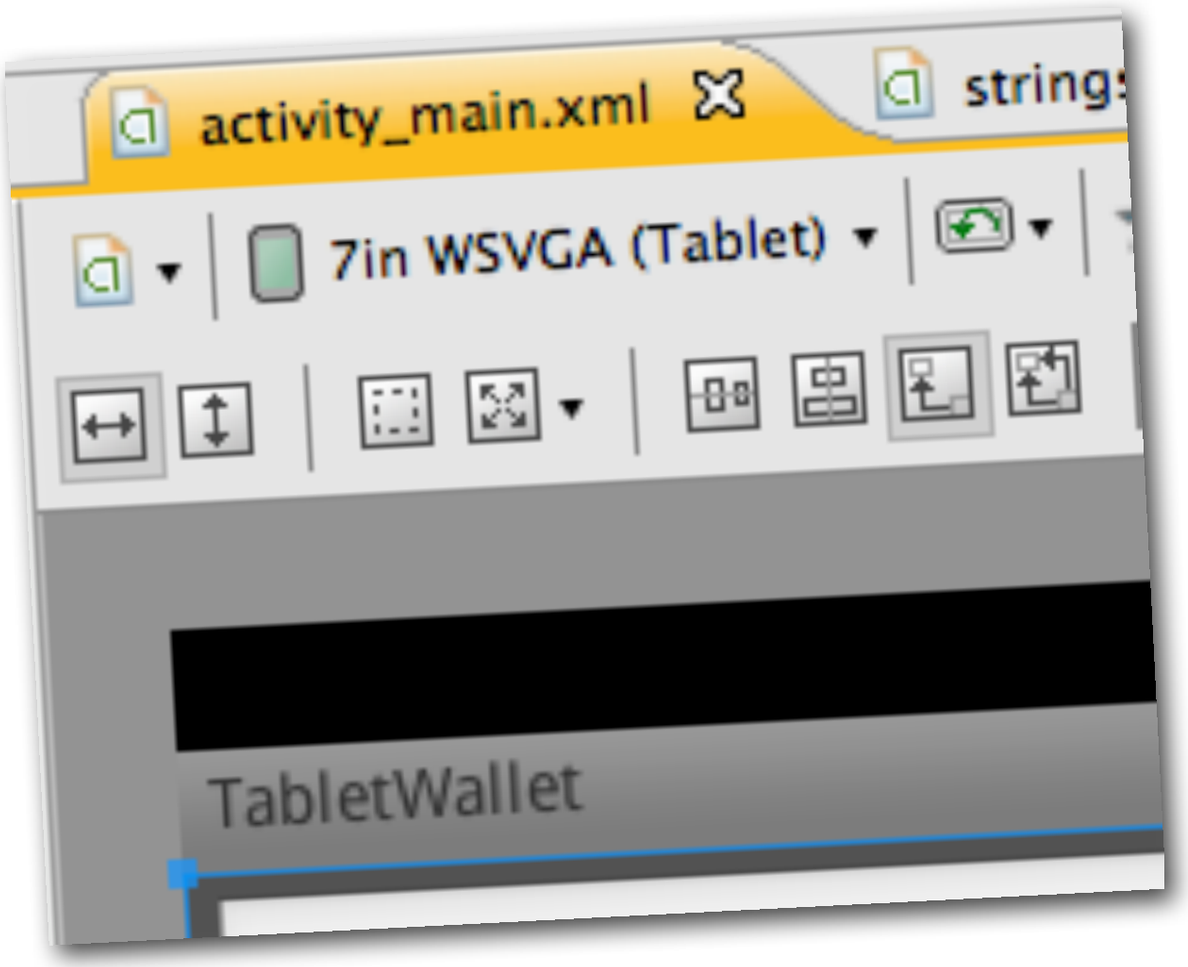
- To set the background color of a View use:
  - android:background="#555"
- The background can be a color in hex notation (0-F)
  - "#rgb"
  - "#argb"
  - "#rrggbb"
  - "#aarrggbb"
- The background can be a resource (like an image)
  - resource: "@[+][package:]type:name"
  - theme: "?[package:][type:]name"

http://developer.android.com/reference/android/view/View.html#attr_android:background

Assignment #3



Assignment #3



Assignment #3

- Other things you need to know for Assignment #3
  - How to build a U/I for a different sized device