Praktikum Entwicklung Mediensysteme (für Master)

Implementing a User Interface

Outline

- Introduction
- Programmatic vs. XML Layout
- Common Layout Objects
- Hooking into a Screen Element
- Listening for UI Notifications
- Applying a Theme to Your Application

Introduction

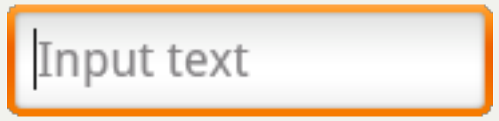
Implementing a User Interface

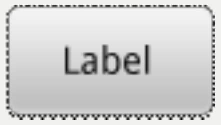
Introduction

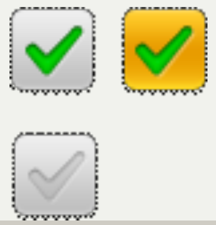
- Activity
  - Basic functional unit of an Android application
  - But by itself, it does not have any presence on the screen
- Views and Viewgroups
  - Basic units of user interface expression on the Android platform

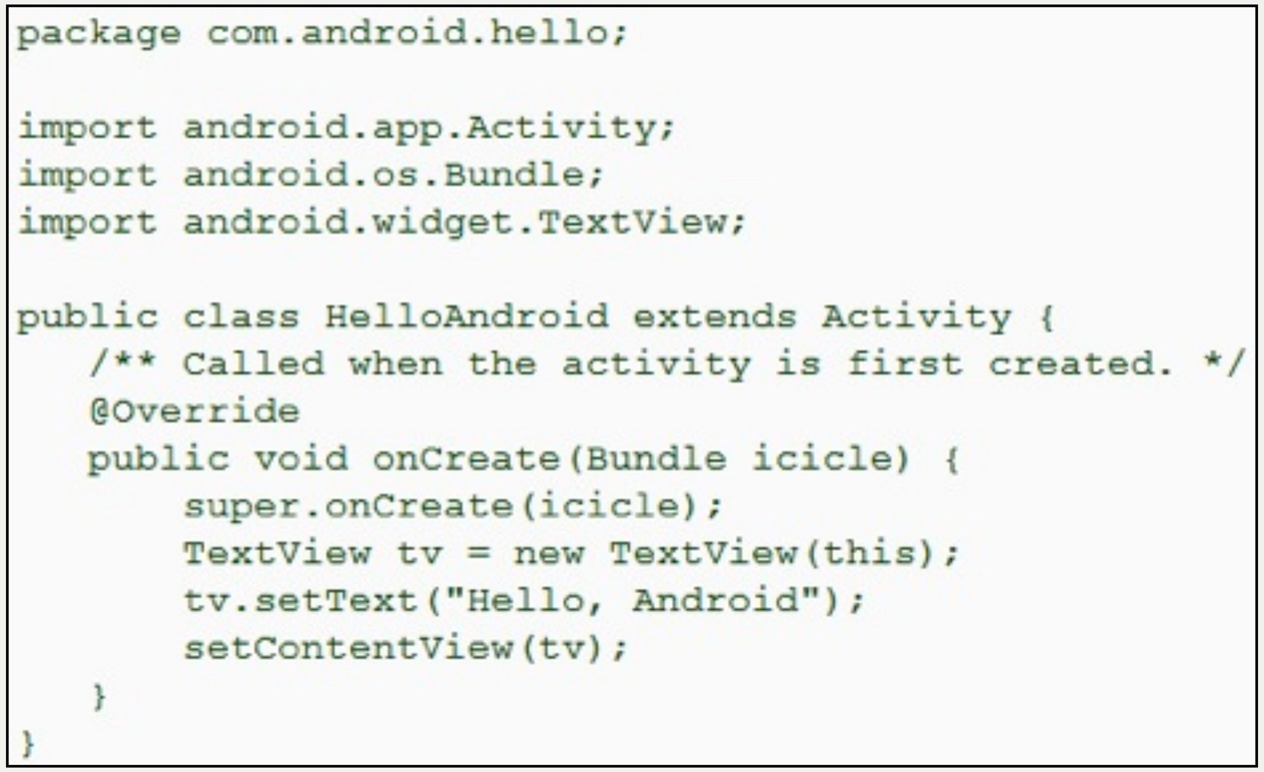
Beautiful View from up here...

- android.view.View
  - Stores layout and content for a specific rectangular area of the screen
  - Handles measuring and layout, drawing, focus change, scrolling, and key/gestures
  - Base class for widgets
    - Text
    - EditText
    - InputMethod
    - MovementMethod
    - Button
    - RadioButton
    - Checkbox
    - ScrollView









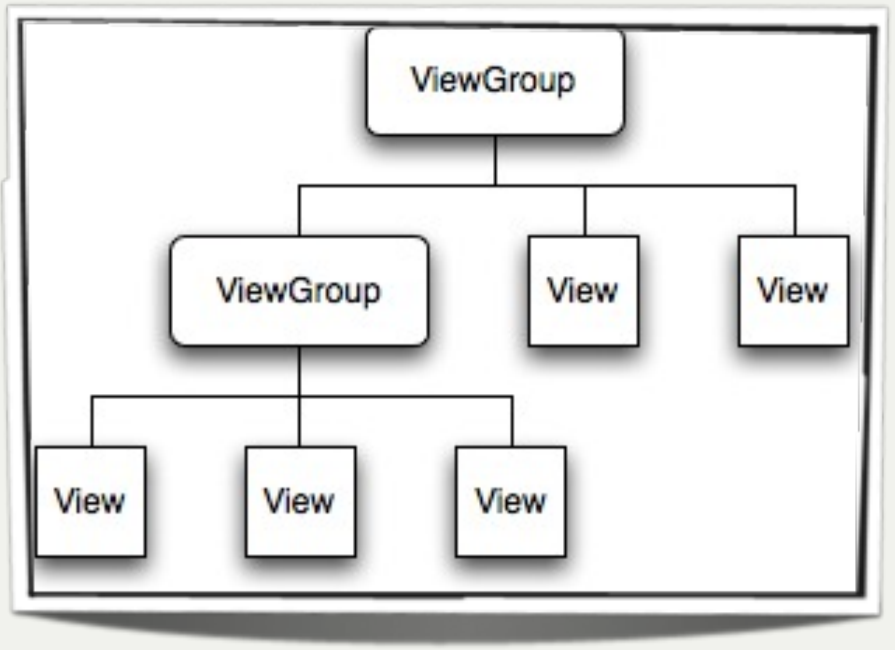
```
package com.android.hello;

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

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

Viewgroups

- android.view.Viewgroup
  - Contains and manages a subordinate set of views and other viewgroups
  - Base class for layouts

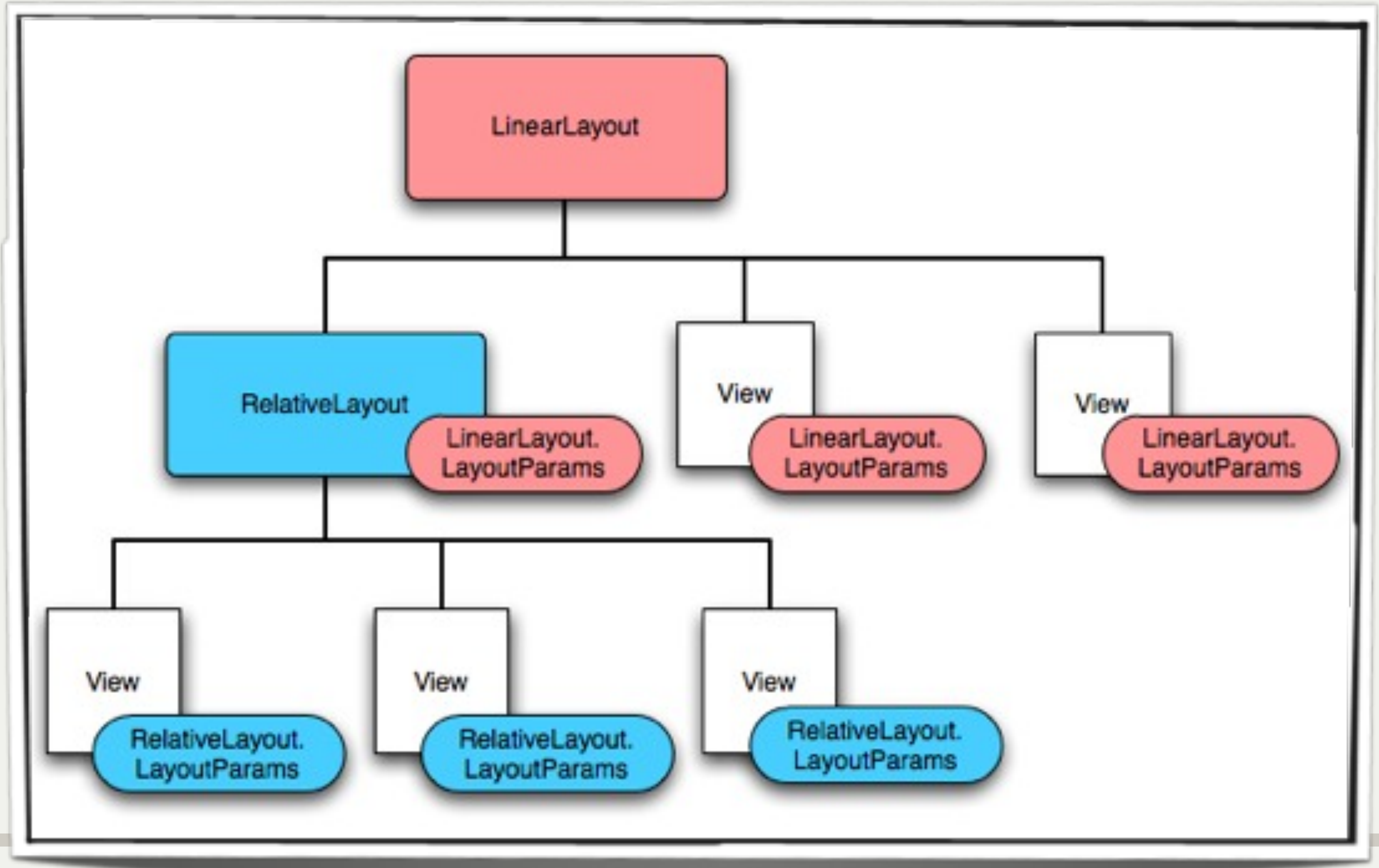
Tree-Structured UI

- An Activity in Android
  - Defined using a tree of view and viewgroup nodes
- setContentView() method
  - Called by the Activity to attach the tree to the screen for rendering



LayoutParams

- Every viewgroup class uses a nested class that extends ViewGroup.LayoutParams
  - Contains property types that defines the child's size and position

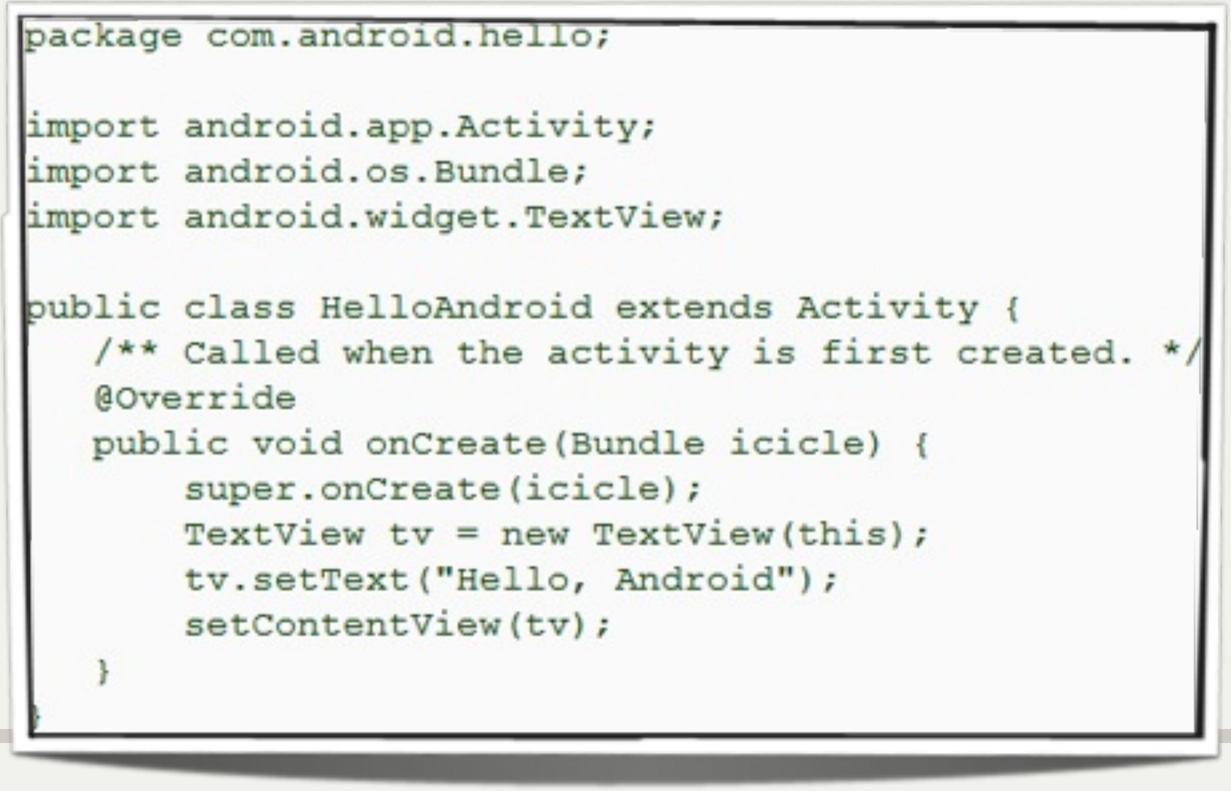

LinearLayout
RelativeLayout
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
RelativeLayout. LayoutParams
View
RelativeLayout. LayoutParams
View
RelativeLayout. LayoutParams

Introduction

Implementing a User Interface

Programmatic UI Layout

- Programmatic UI Layout
  - Constructing and building the applications UI directly from source code
  - Disadvantage
    - small changes in layout can have a big effect on the source code

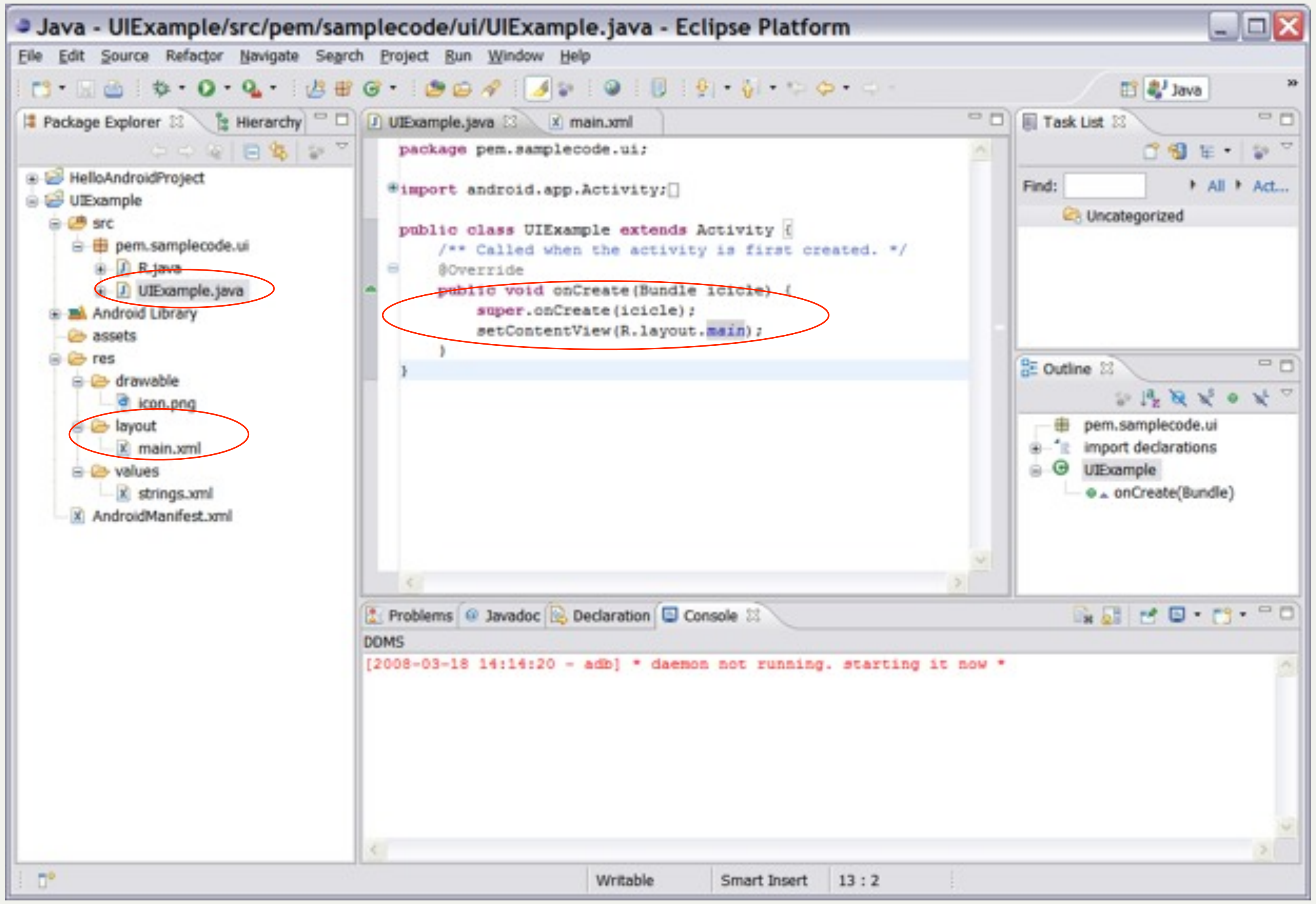


```
package com.android.hello;

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

public class HelloAndroid extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle icicle) {
       super.onCreate(icicle);
       TextView tv = new TextView(this);
       tv.setText("Hello, Android");
       setContentView(tv);
   }
}
```

Upgrading UI to XML Layout

- XML-based Layout
  - Inspired by web development model where the presentation of the application's UI is separated from the logic
  - Two files to edit
    - Java file – application logic
    - XML file – user interface

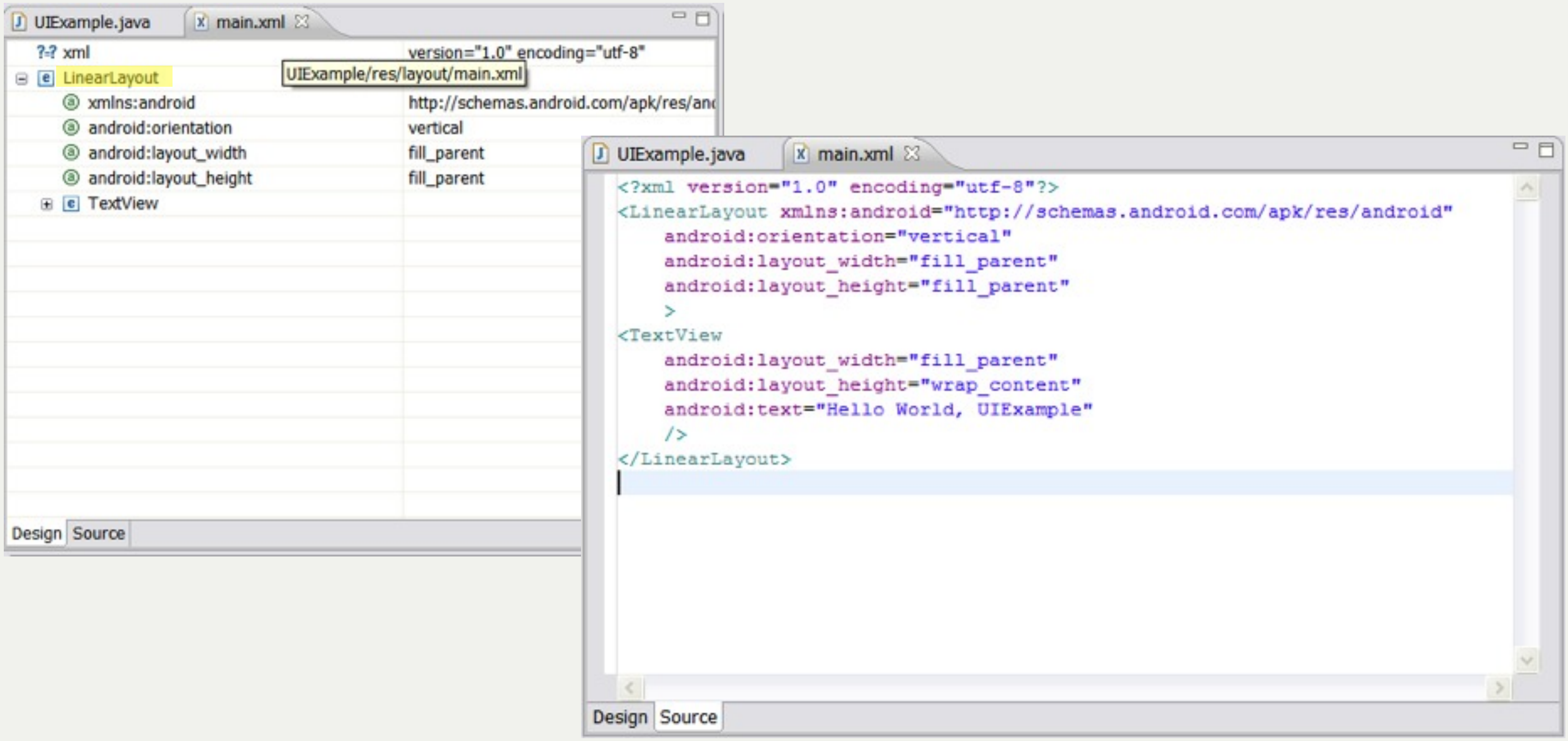
Upgrading UI to XML Layout



```
package pem.samplecode.ui;

import android.app.Activity;

public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
    }
}
```

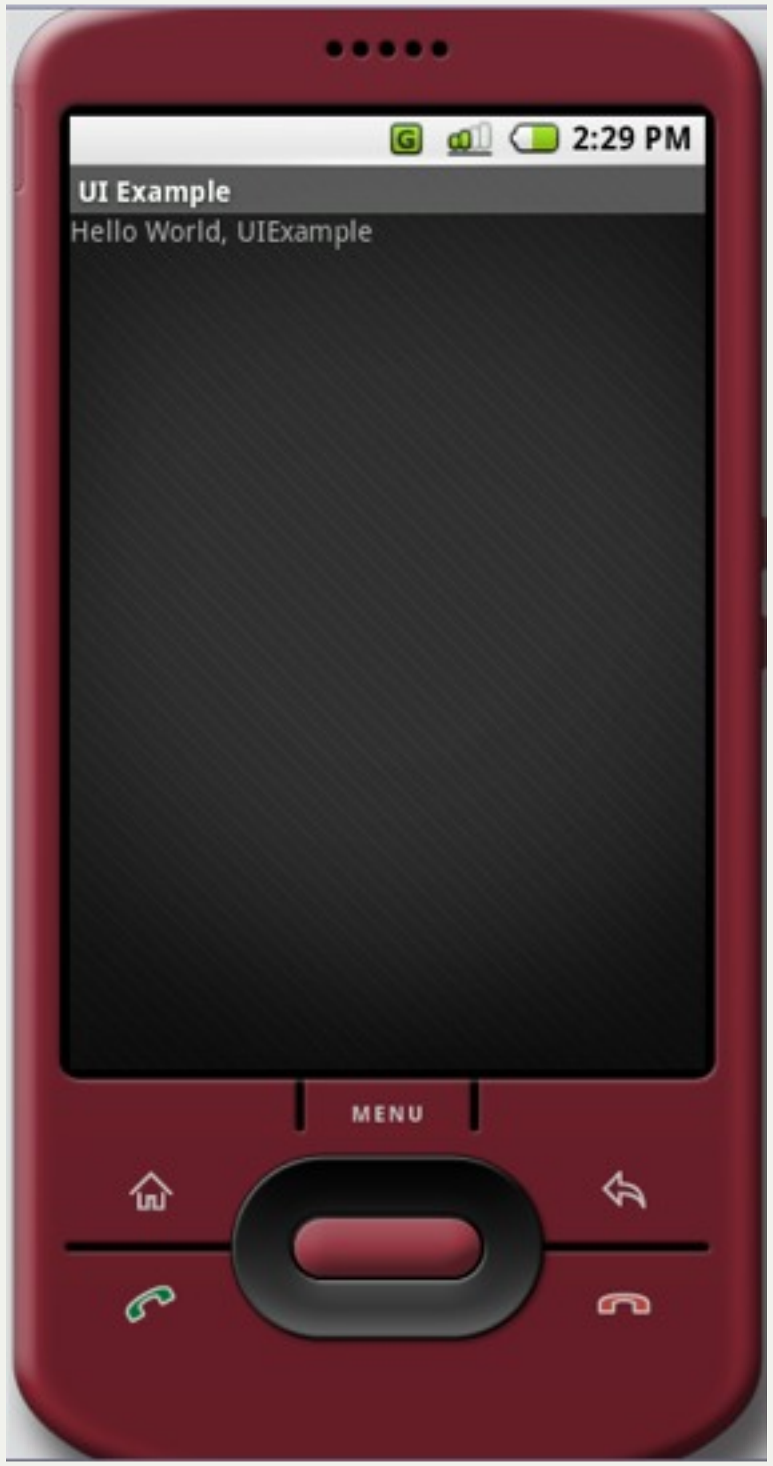
```
DDMS
[2008-03-18 14:14:20 - adb] * daemon not running. starting it now *
```

Upgrading UI to XML Layout



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

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



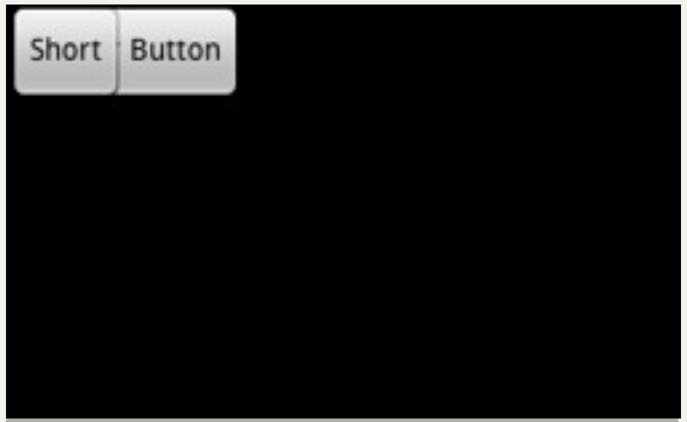
UI Example

Hello World, UIExample

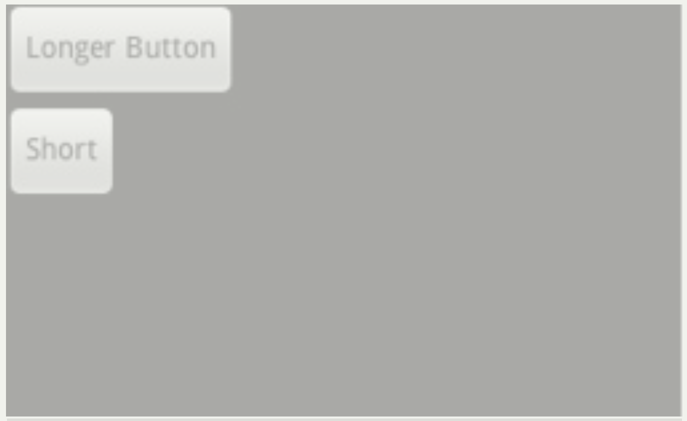
Common Layout Objects

Implementing a User Interface

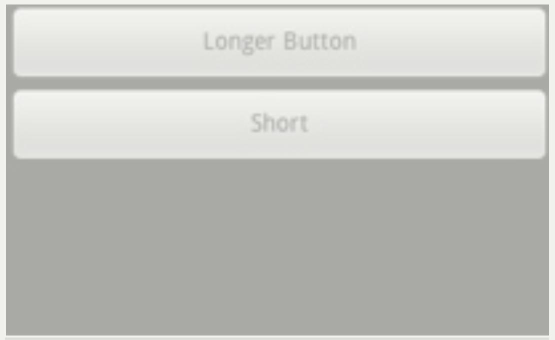
Common Layout Objects



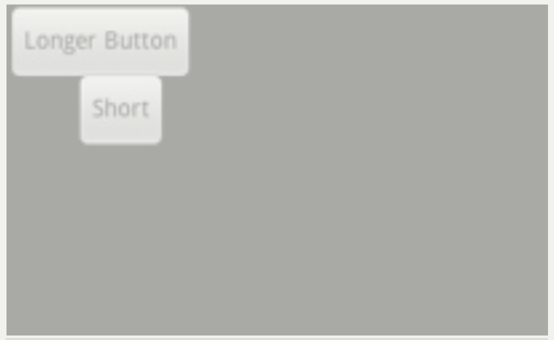
FrameLayout



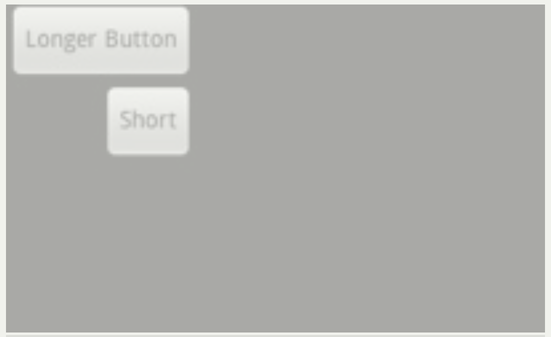
LinearLayout



TableLayout



AbsoluteLayout

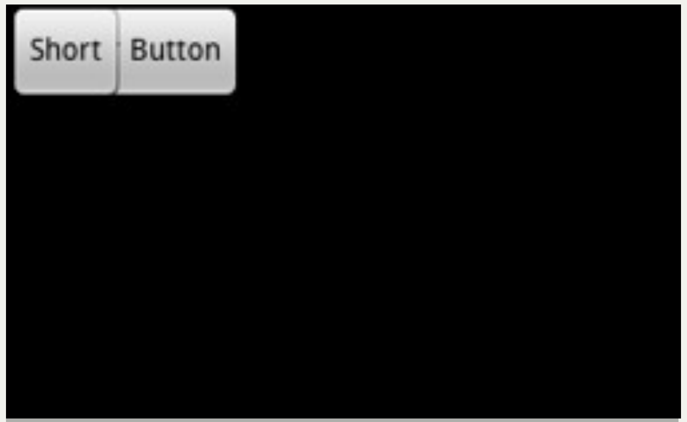


RelativeLayout

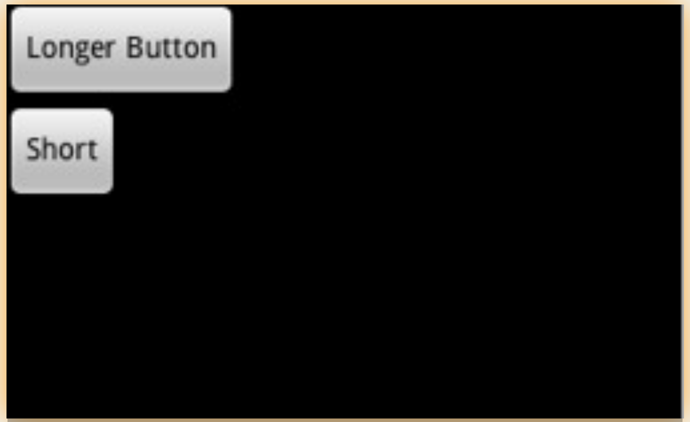
FrameLayout

- Simplest layout object
- Intended as a blank reserved space on your screen that you can later fill with a single object
  - Example: a picture that you'll swap out
- All child elements are pinned to the top left corner of the screen
- Cannot specify a location for a child element

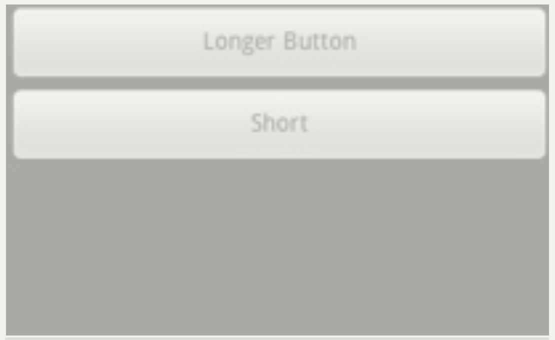
Common Layout Objects



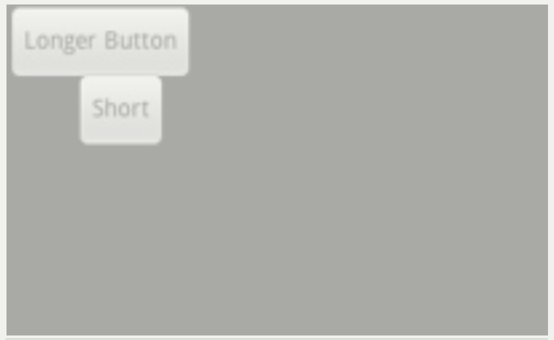
FrameLayout



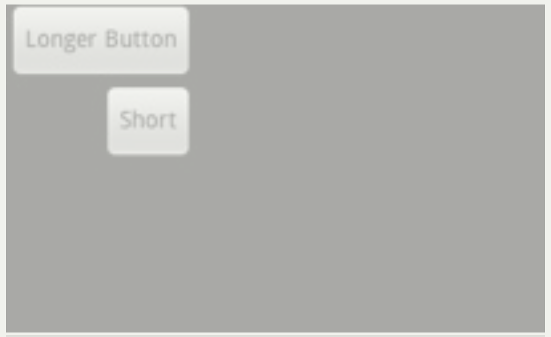
LinearLayout



TableLayout



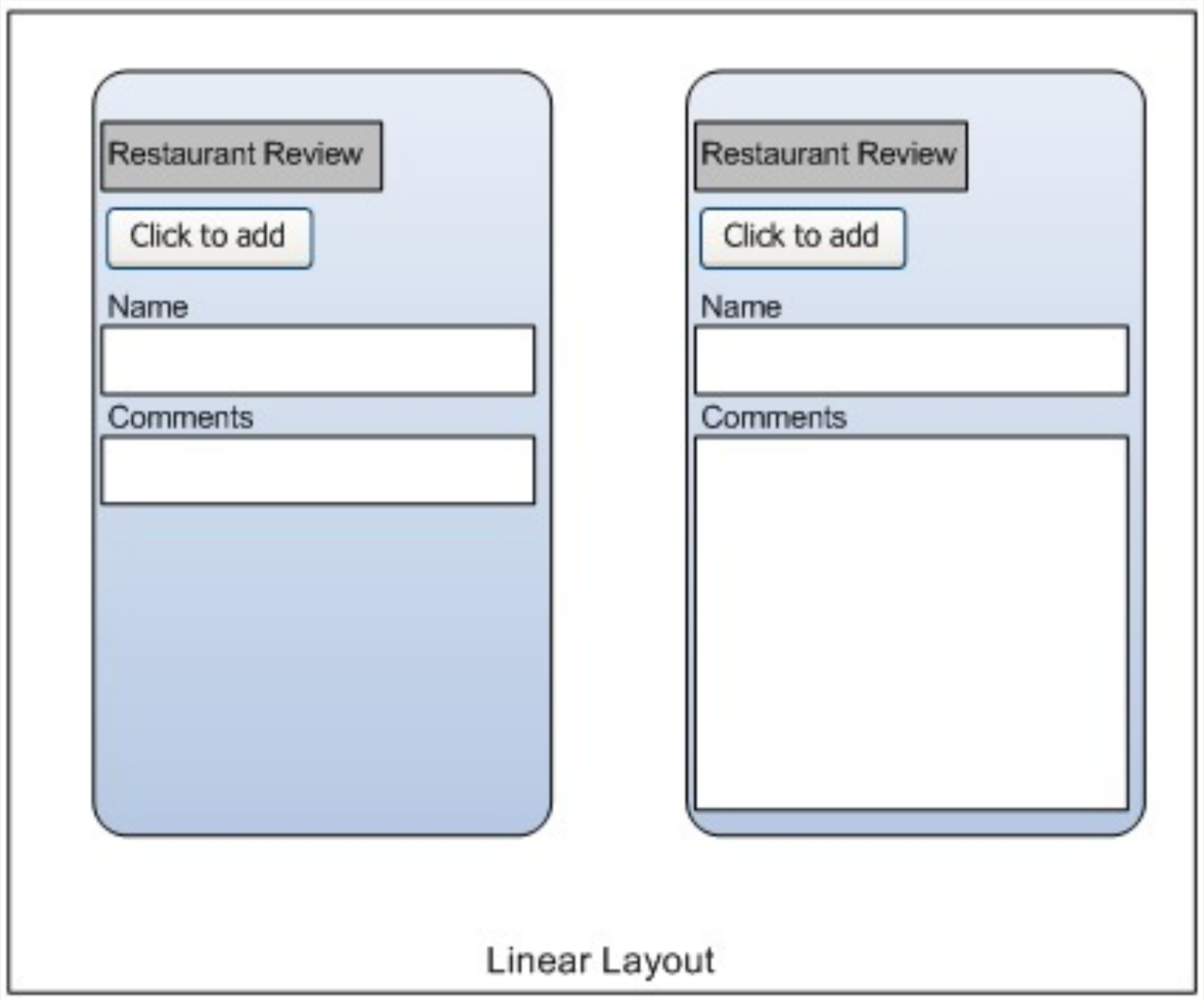
AbsoluteLayout



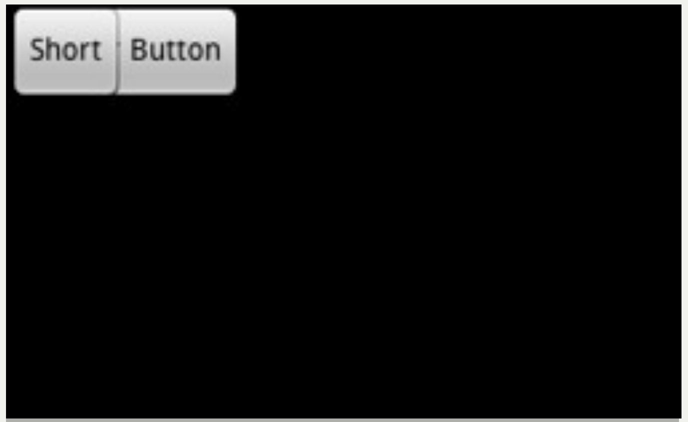
RelativeLayout

LinearLayout

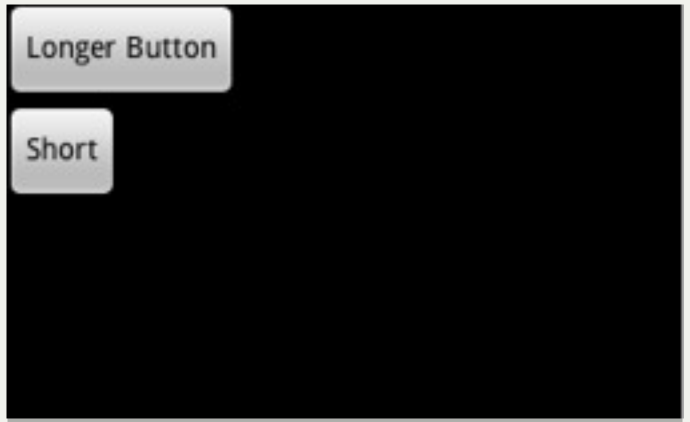
- Aligns all children in a single direction — vertically or horizontally
  - All children are stacked one after the other
    - a vertical list will only have one child per row (no matter how wide they are)
    - a horizontal list will only be one row high (the height of the tallest child, plus padding)



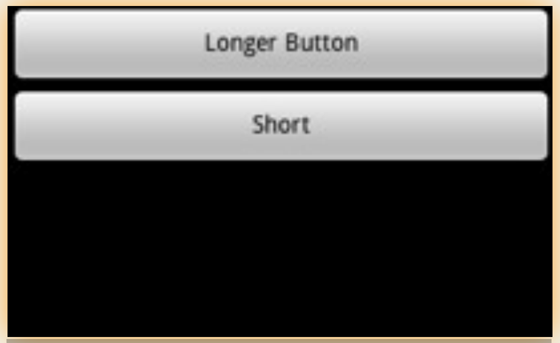
Common Layout Objects



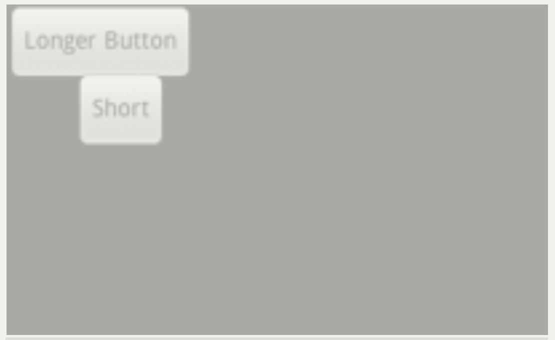
FrameLayout



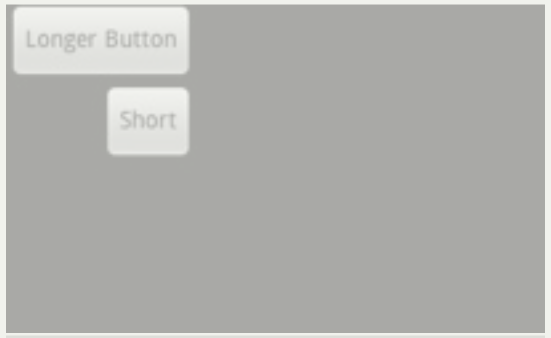
LinearLayout



TableLayout



AbsoluteLayout



RelativeLayout

TableLayout

- Positions its children into rows and columns
- Does not display border lines for their rows, columns, or cells
- Cells cannot span columns, as they can in HTML

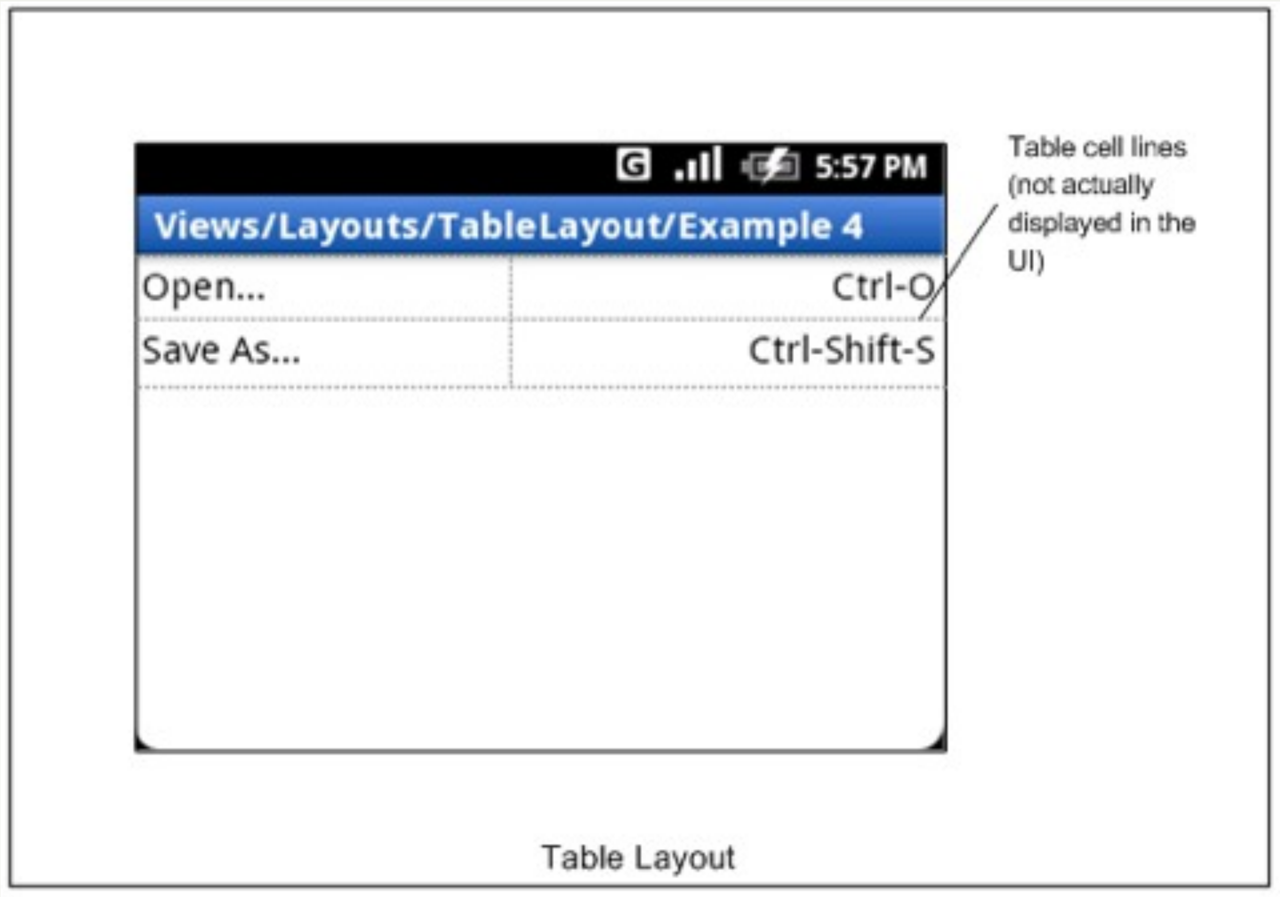
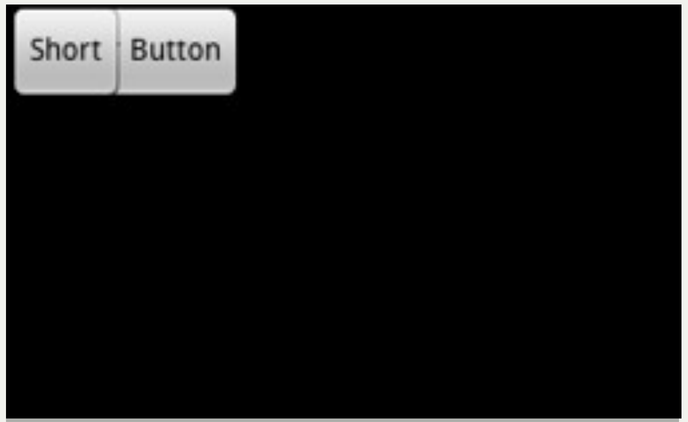
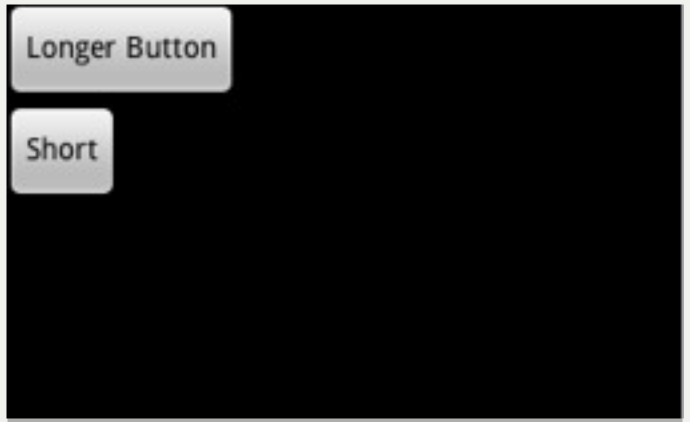

5:57 PM
Views/Layouts/TableLayout/Example 4
Open... Ctrl-O
Save As... Ctrl-Shift-S
Table cell lines (not actually displayed in the UI)

Table Layout

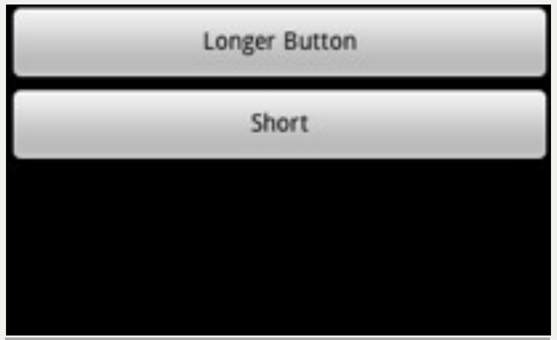
Common Layout Objects



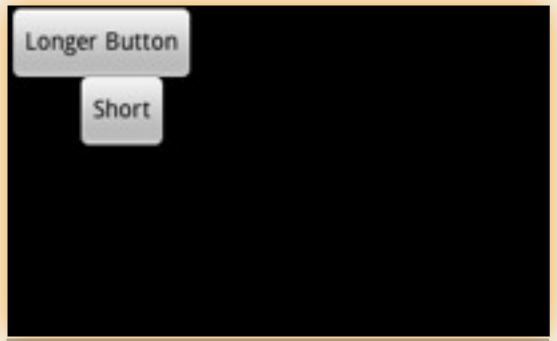
FrameLayout



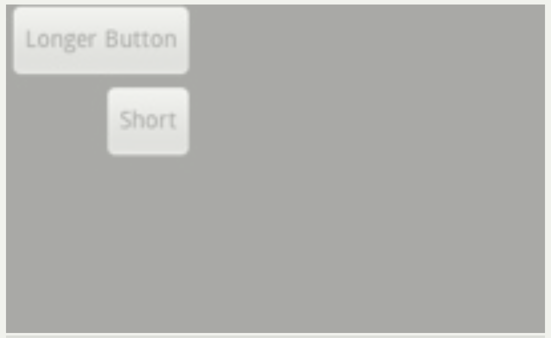
LinearLayout



TableLayout



AbsoluteLayout

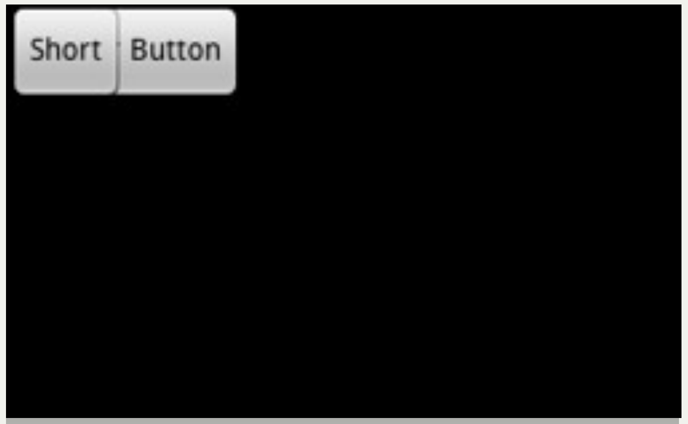


RelativeLayout

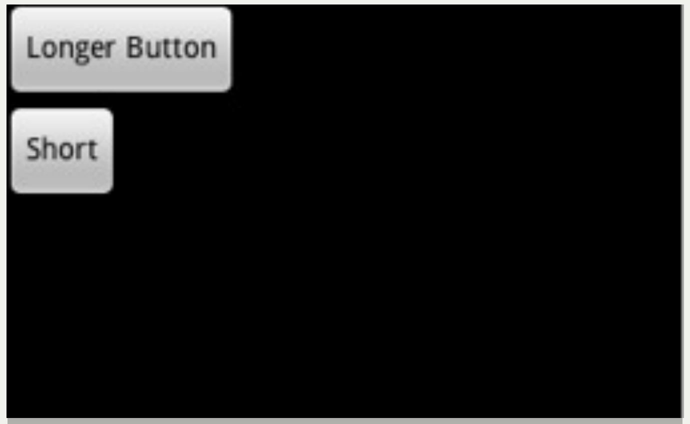
Absolute Layout

- Enables children to specify exact x/y coordinates to display on the screen
  - (0,0) is the upper left corner
  - Values increase as you move down or to the right
- Overlapping elements are allowed (although not recommended)
- NOTE:
  - It is generally recommended NOT to use AbsoluteLayout UNLESS you have good reasons to use it
  - It is because it is fairly rigid and does not work well with different device displays

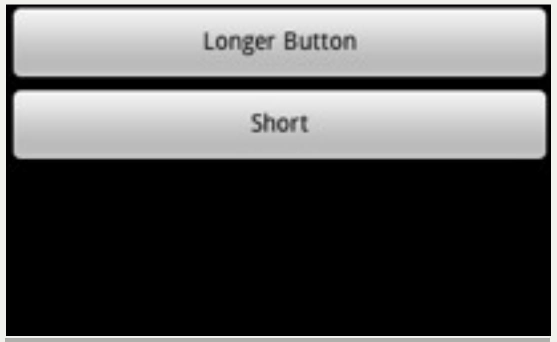
Common Layout Objects



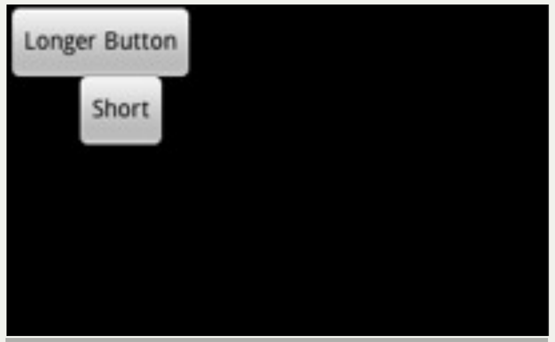
FrameLayout



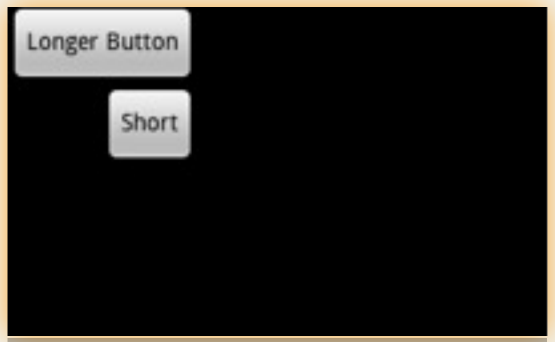
LinearLayout



TableLayout



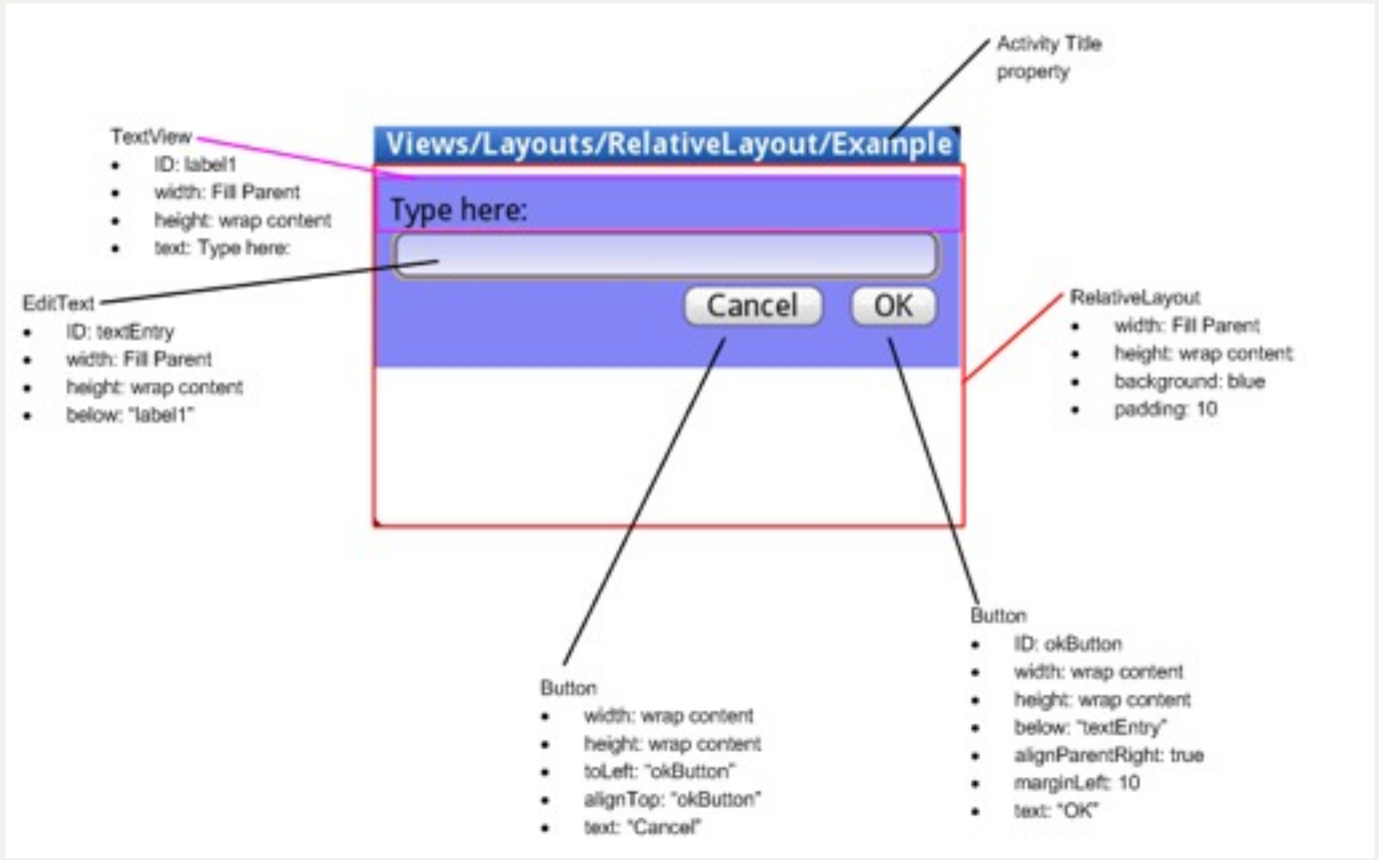
AbsoluteLayout



RelativeLayout

RelativeLayout

- Lets children specify their position relative to each other (specified by ID), or to the parent



Activity Title property

TextView
- ID: label1
- width: Fill Parent
- height: wrap content
- text: Type here:

EditText
- ID: textEntry
- width: Fill Parent
- height: wrap content
- below: "label1"

RelativeLayout
- width: Fill Parent
- height: wrap content
- background: blue
- padding: 10

Button
- width: wrap content
- height: wrap content
- toLeft: "okButton"
- alignTop: "okButton"
- text: "Cancel"

Button
- ID: okButton
- width: wrap content
- height: wrap content
- below: "textEntry"
- alignParentRight: true
- marginLeft: 10
- text: "OK"

Important Layout Paramters

Allgemein:

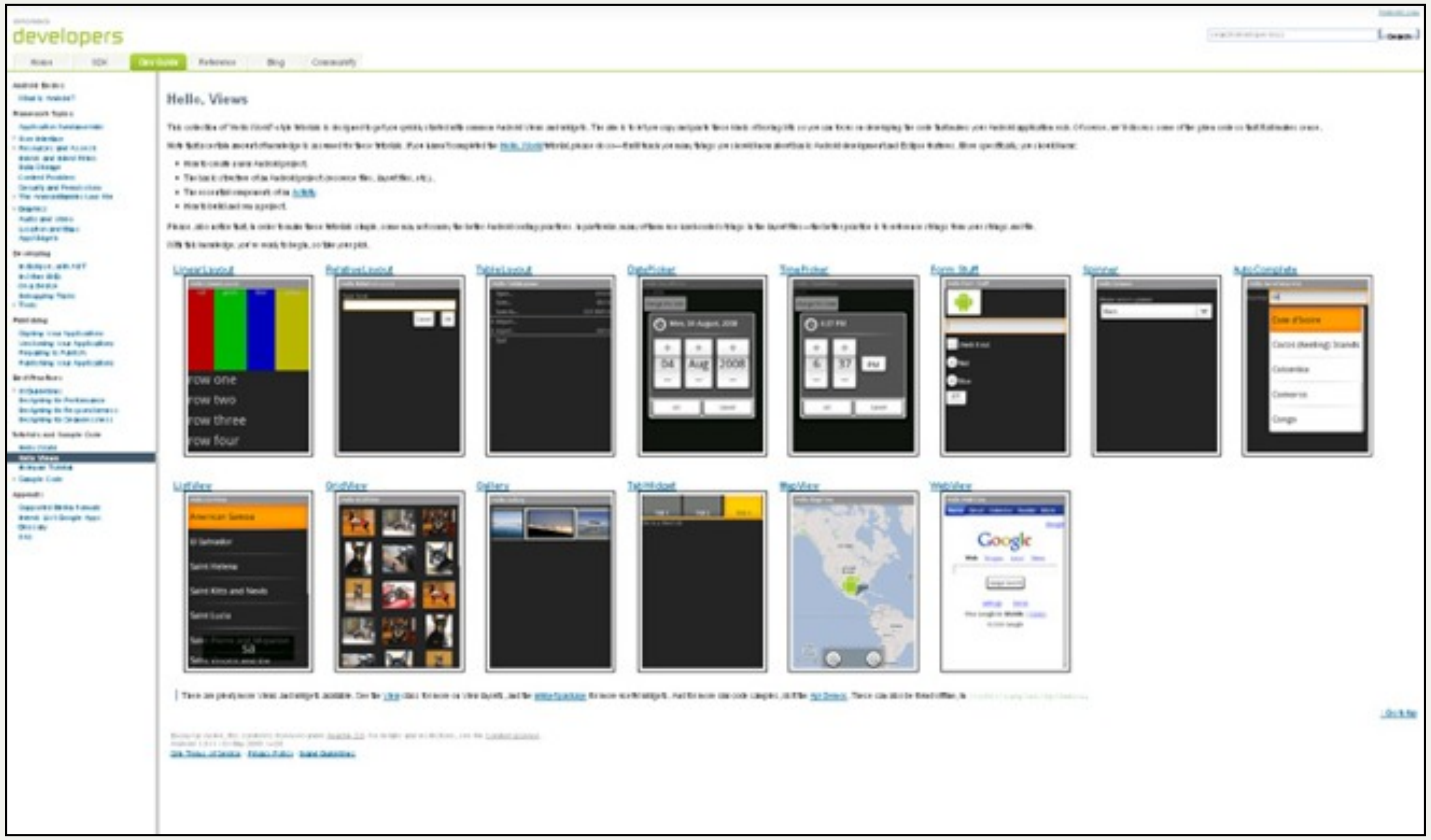
Layout-Height:	fill_parent, wrap_content,
Layout-Width:	fill_parent, wrap_content,
Id:	@+id/my_variable
Min-Height, Max-Height...	
Min-Width, Max-Width	

Speziell:

EditText	Input type	text, textEmailAddress, number, numberDecimal
TextView, Button, EditText	Text	@string/resource_id
TextView	Text color, Text size	

Online Reference

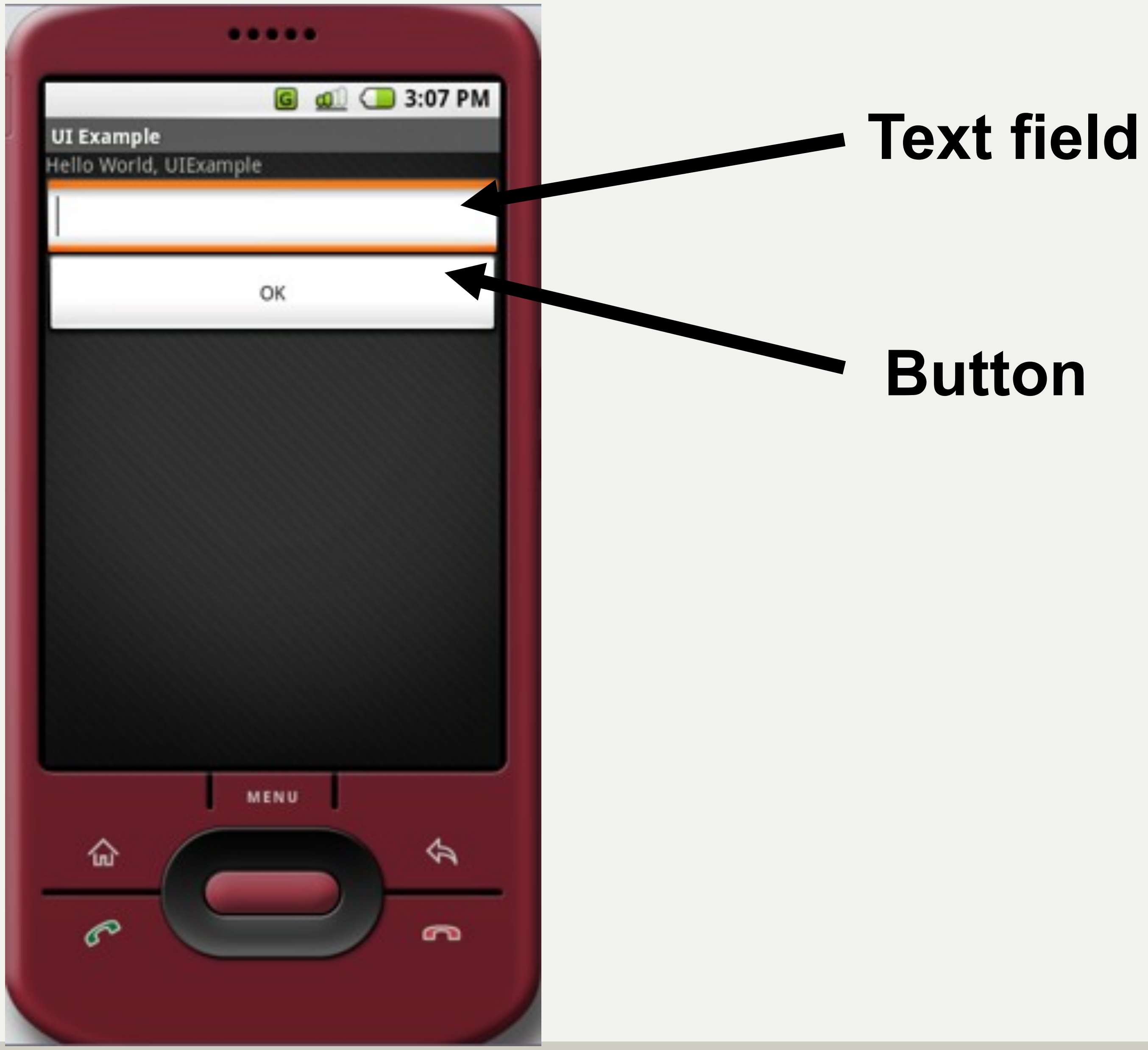
http://developer.android.com/guide/tutorials/views/index.html



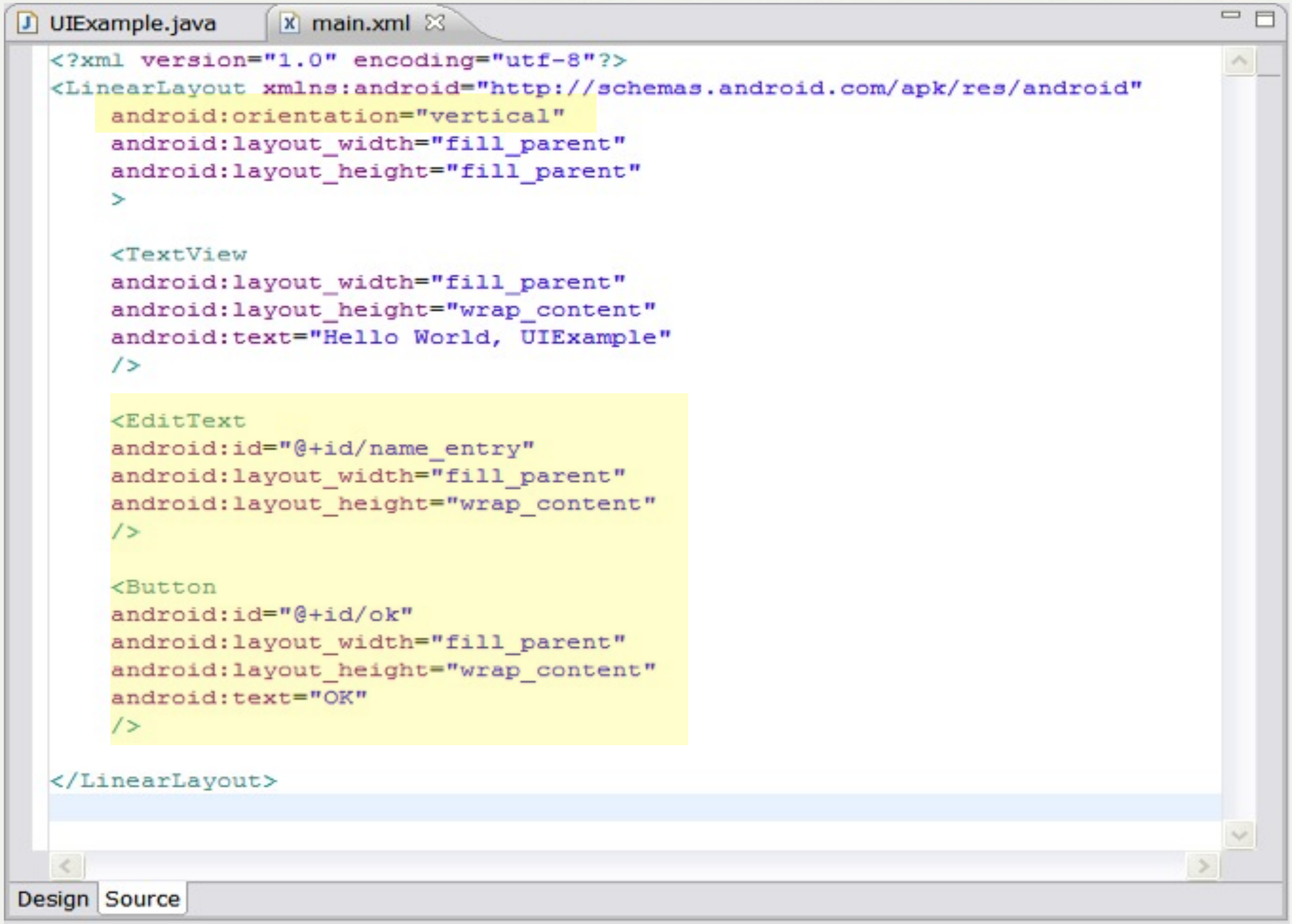
Hooking into a Screen Element

Implementing a User Interface

Hooking into a Screen Element

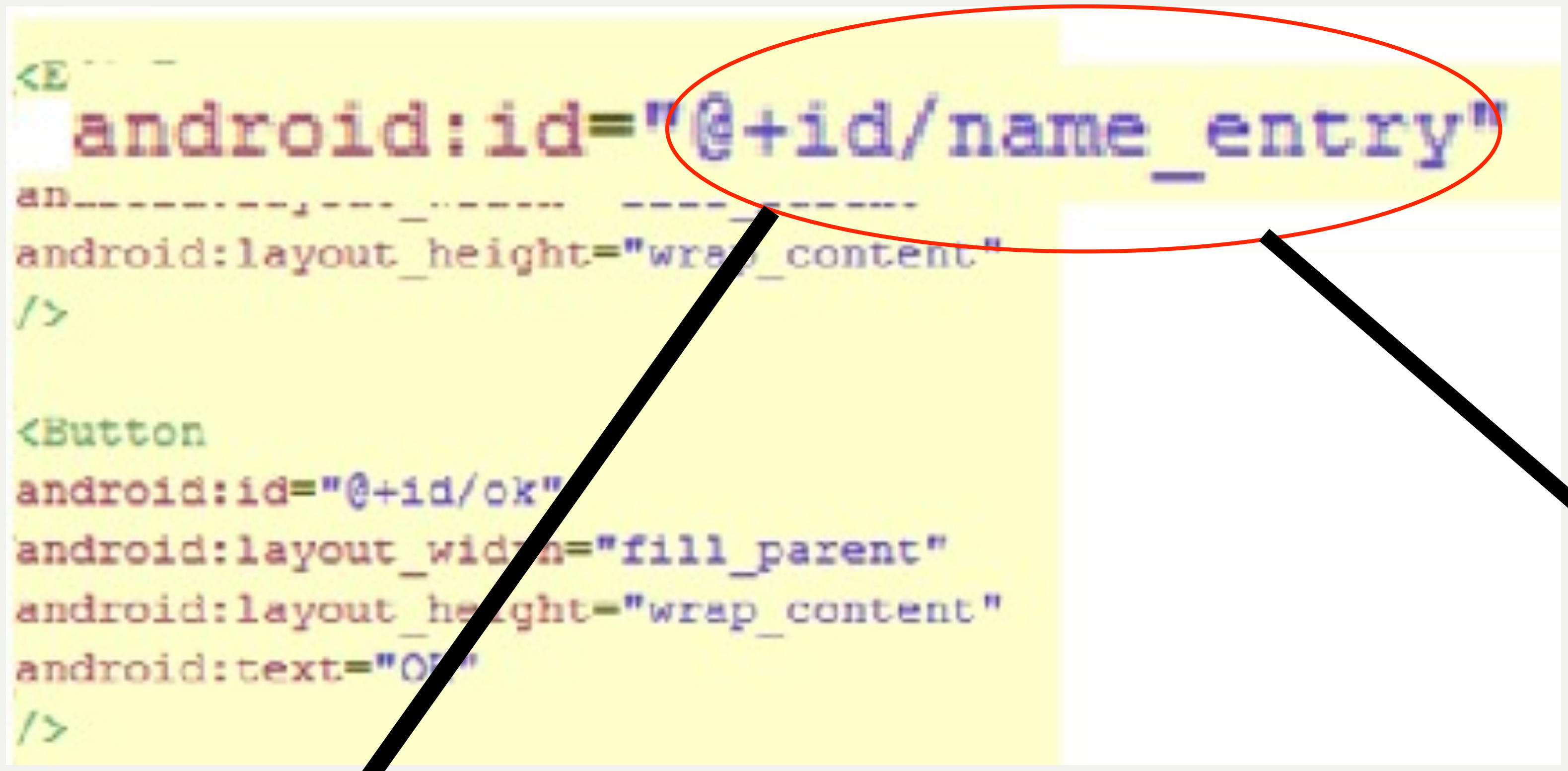


Hooking into a Screen Element



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

    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello World, UIExample"
    />

    <EditText
    android:id="@+id/name_entry"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />

    <Button
    android:id="@+id/ok"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="OK"
    />

</LinearLayout>
```

Hooking into a Screen Element

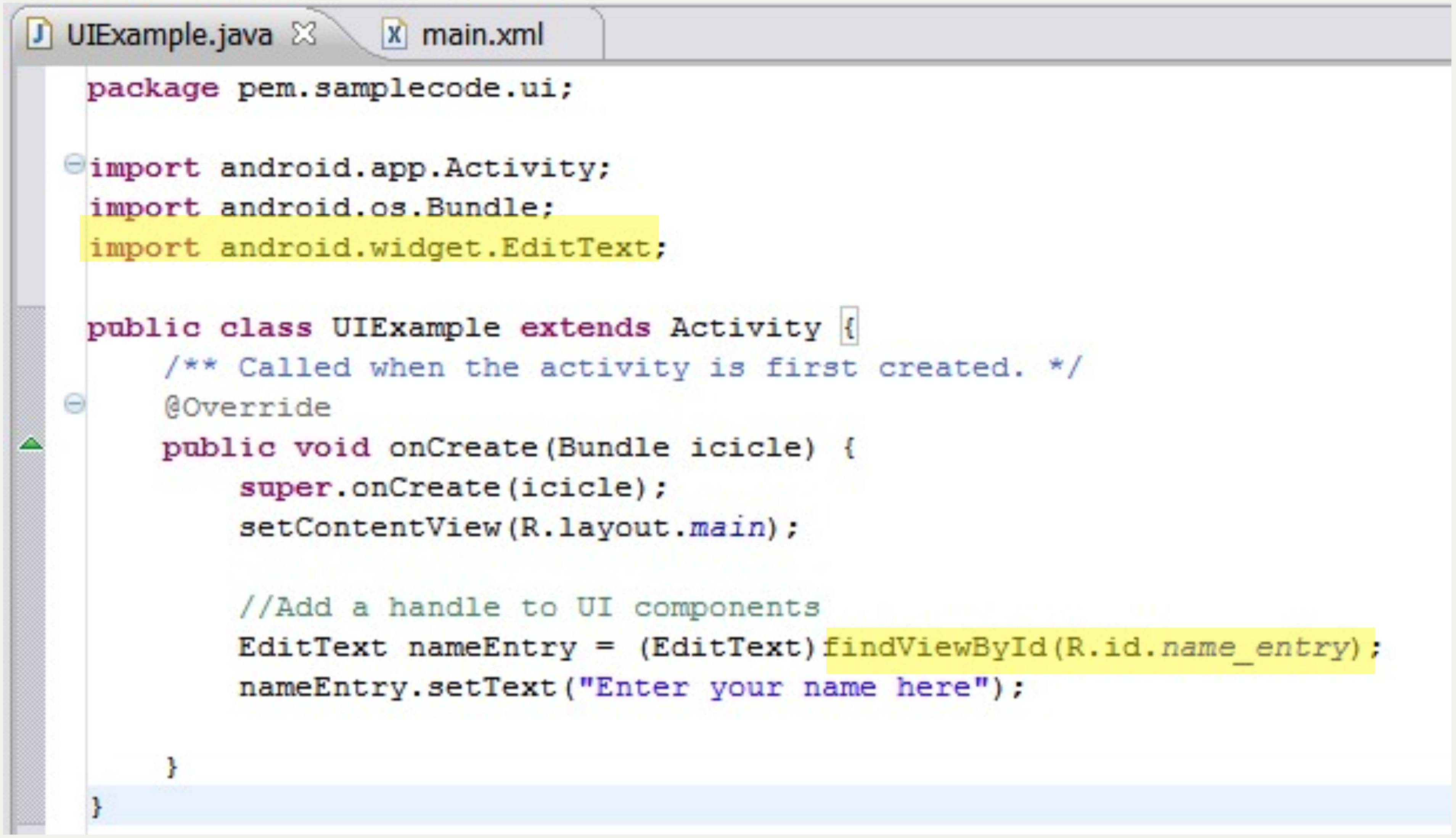


```
<E
android:id="@+id/name_entry"
an...
android:layout_height="wrap_content"
/>

<Button
android:id="@+id/ok"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="OK"
/>
```

@+id syntax:
Creates a resource number in the R class (R.java file) if one doesn't exist, or uses it if it does exist.

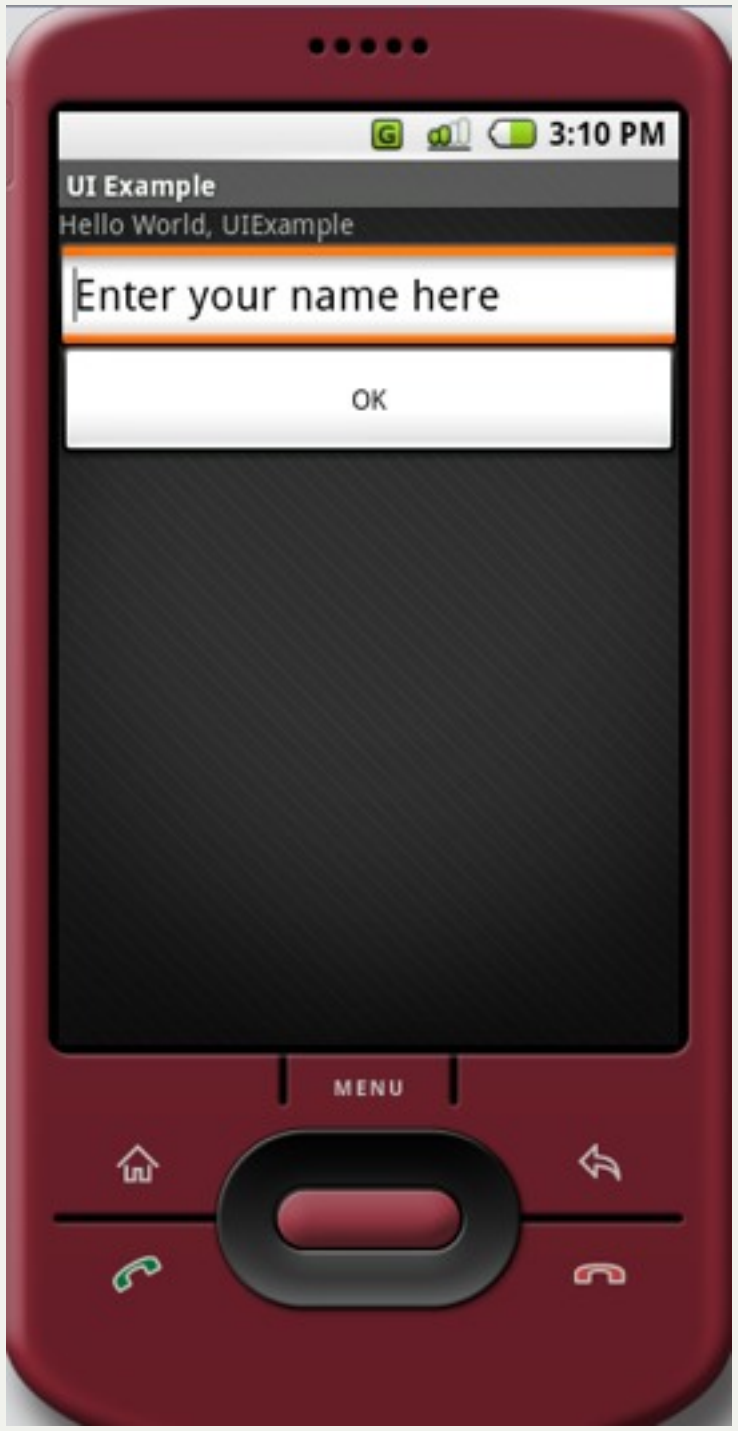
Any String value (no spaces)

Hooking into a Screen Element

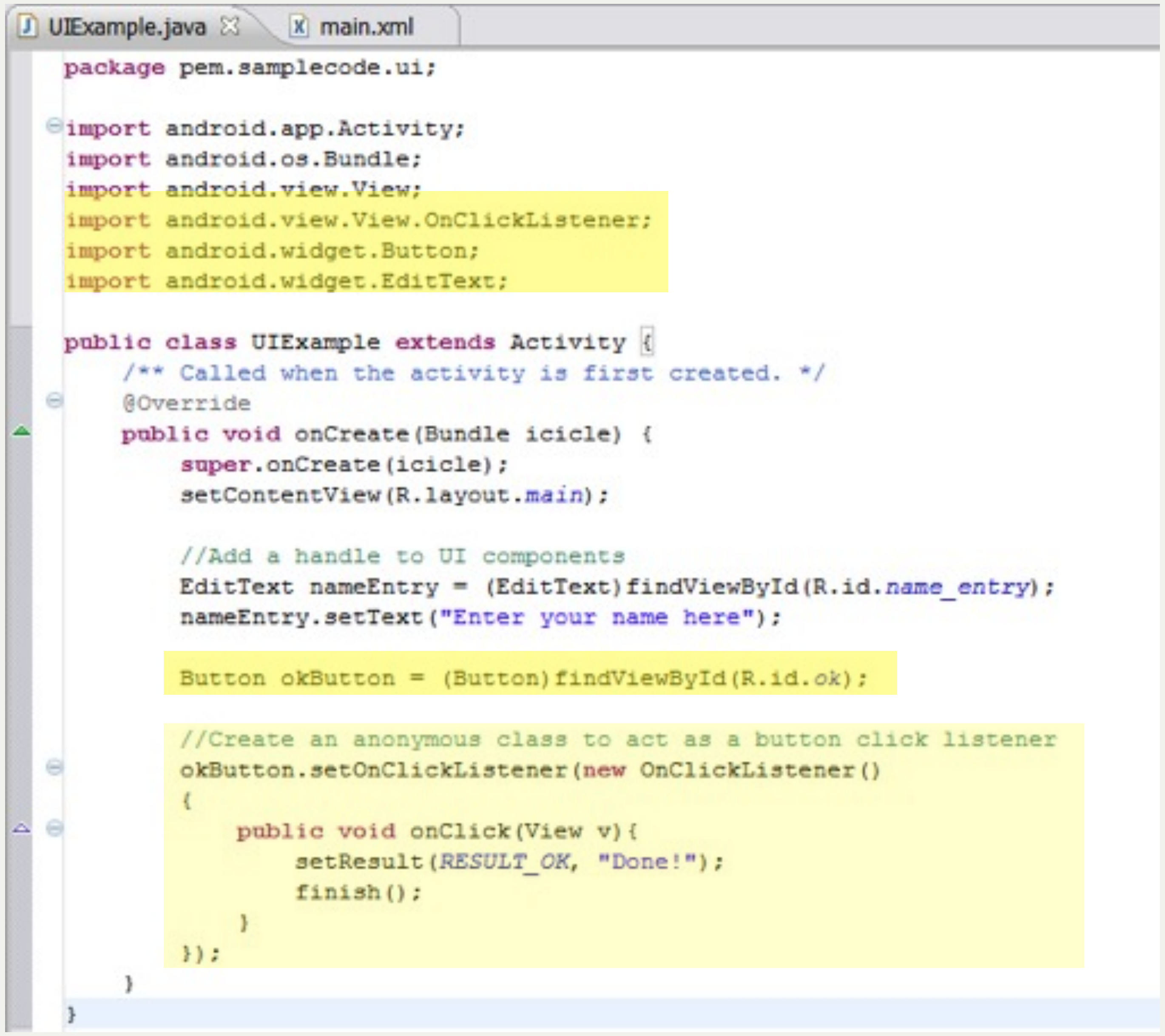

UIExample.java main.xml

```
package pem.samplecode.ui;

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

public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        //Add a handle to UI components
        EditText nameEntry = (EditText)findViewById(R.id.name_entry);
        nameEntry.setText("Enter your name here");

    }
}
```

Hooking into a Screen Element



Listening for UI Notifications

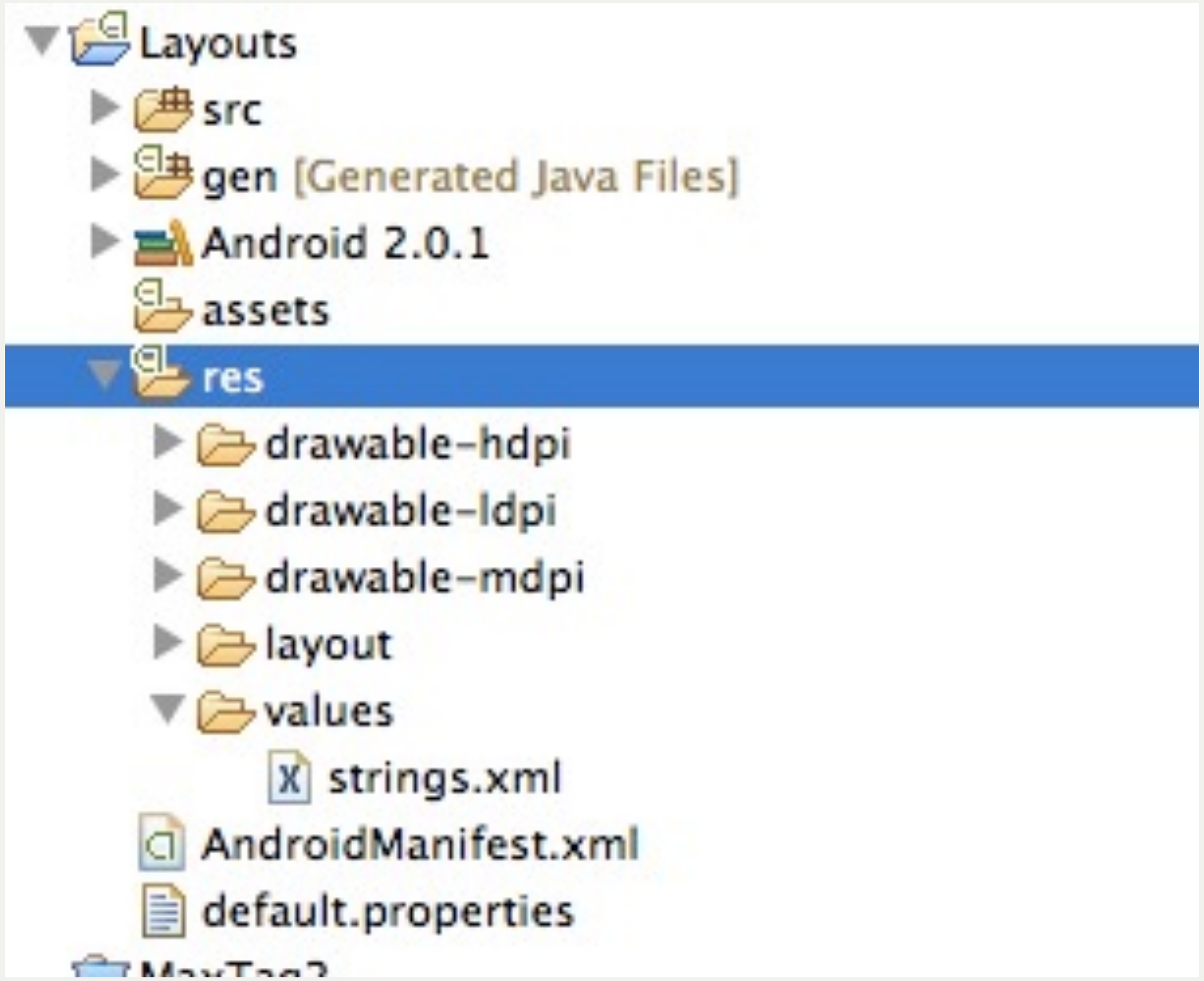


```
package pem.samplecode.ui;

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

public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        //Add a handle to UI components
        EditText nameEntry = (EditText)findViewById(R.id.name_entry);
        nameEntry.setText("Enter your name here");

        Button okButton = (Button)findViewById(R.id.ok);

        //Create an anonymous class to act as a button click listener
        okButton.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v){
                setResult(RESULT_OK, "Done!");
                finish();
            }
        });
    }
}
```

Resource Folders and Localization

Implementing a User Interface

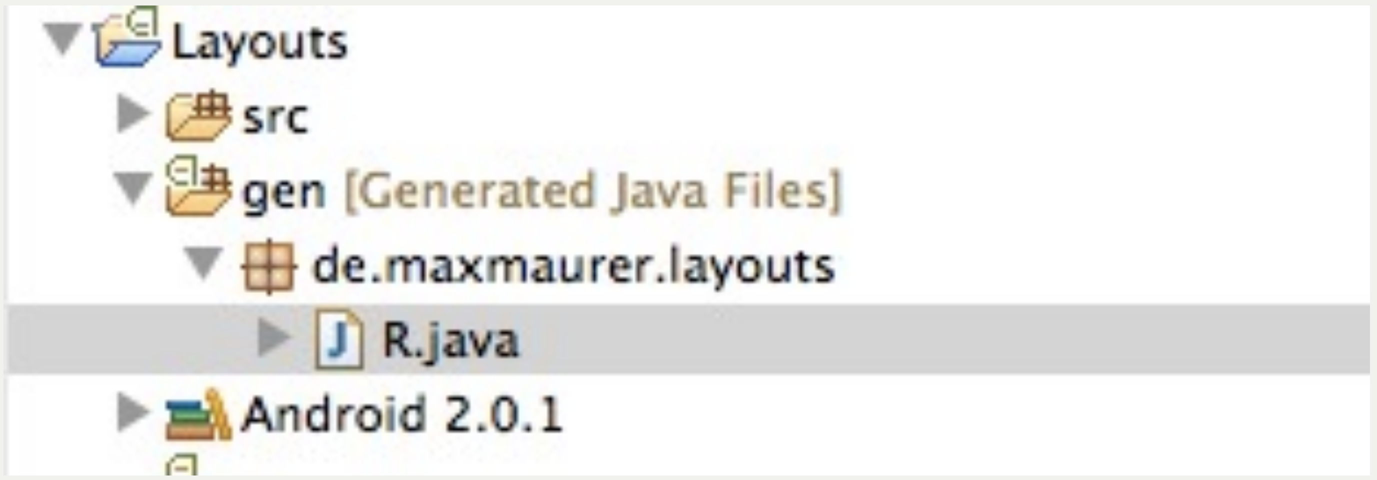
Resource Folders

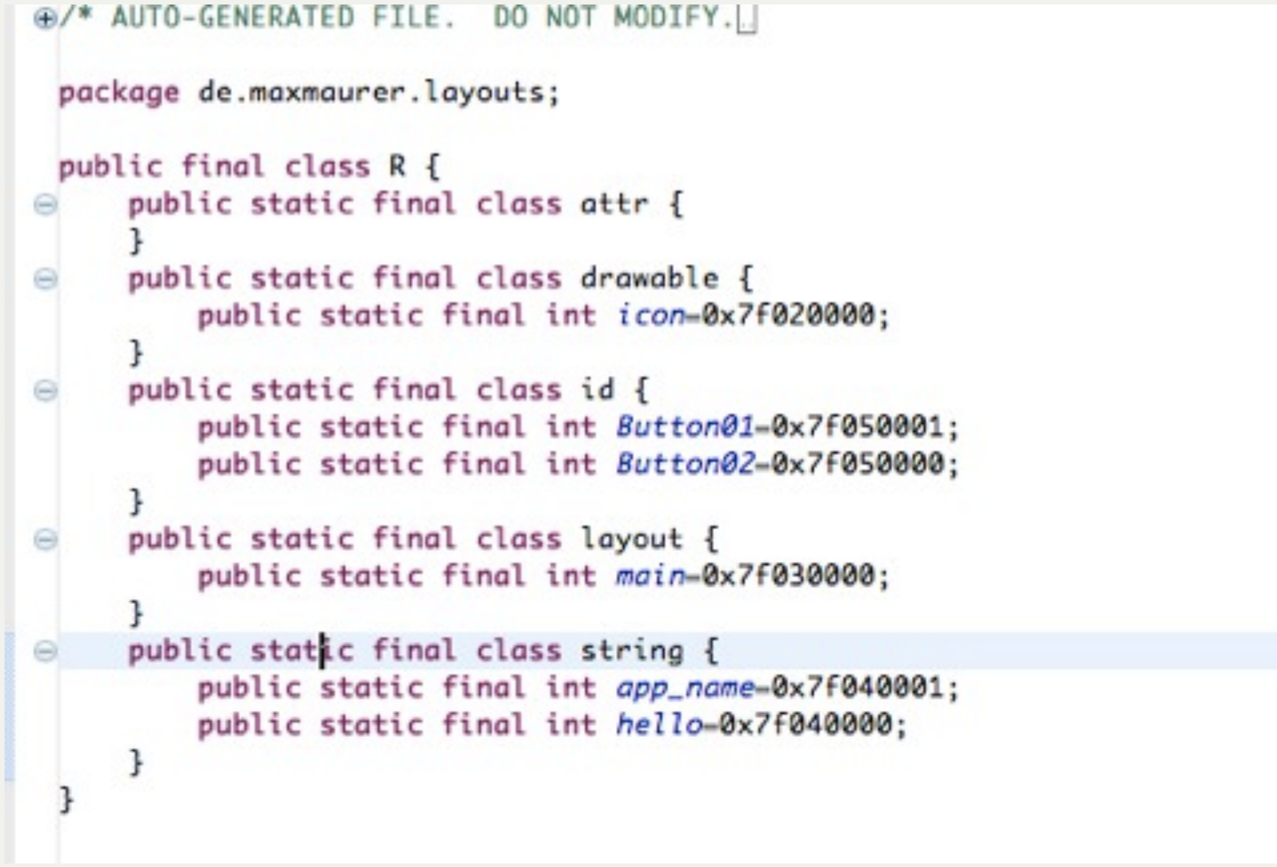


```
▼ Layouts
    ▶ src
    ▶ gen [Generated Java Files]
    ▶ Android 2.0.1
      assets
    ▼ res
        ▶ drawable-hdpi
        ▶ drawable-ldpi
        ▶ drawable-mdpi
        ▶ layout
        ▼ values
              strings.xml
      AndroidManifest.xml
      default.properties
```

Resource Folders

- Folder structure is automatically parsed into Resource-File
- Do not modify this file!

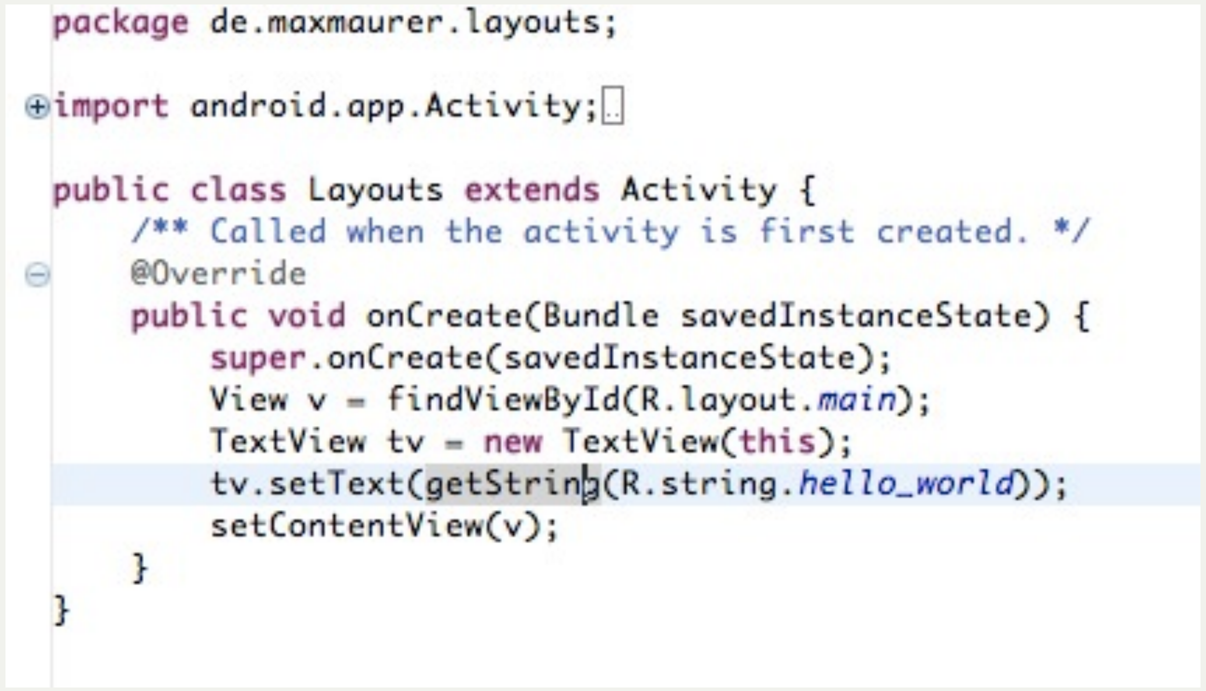




```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package de.maxmaurer.layouts;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int Button01=0x7f050001;
        public static final int Button02=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

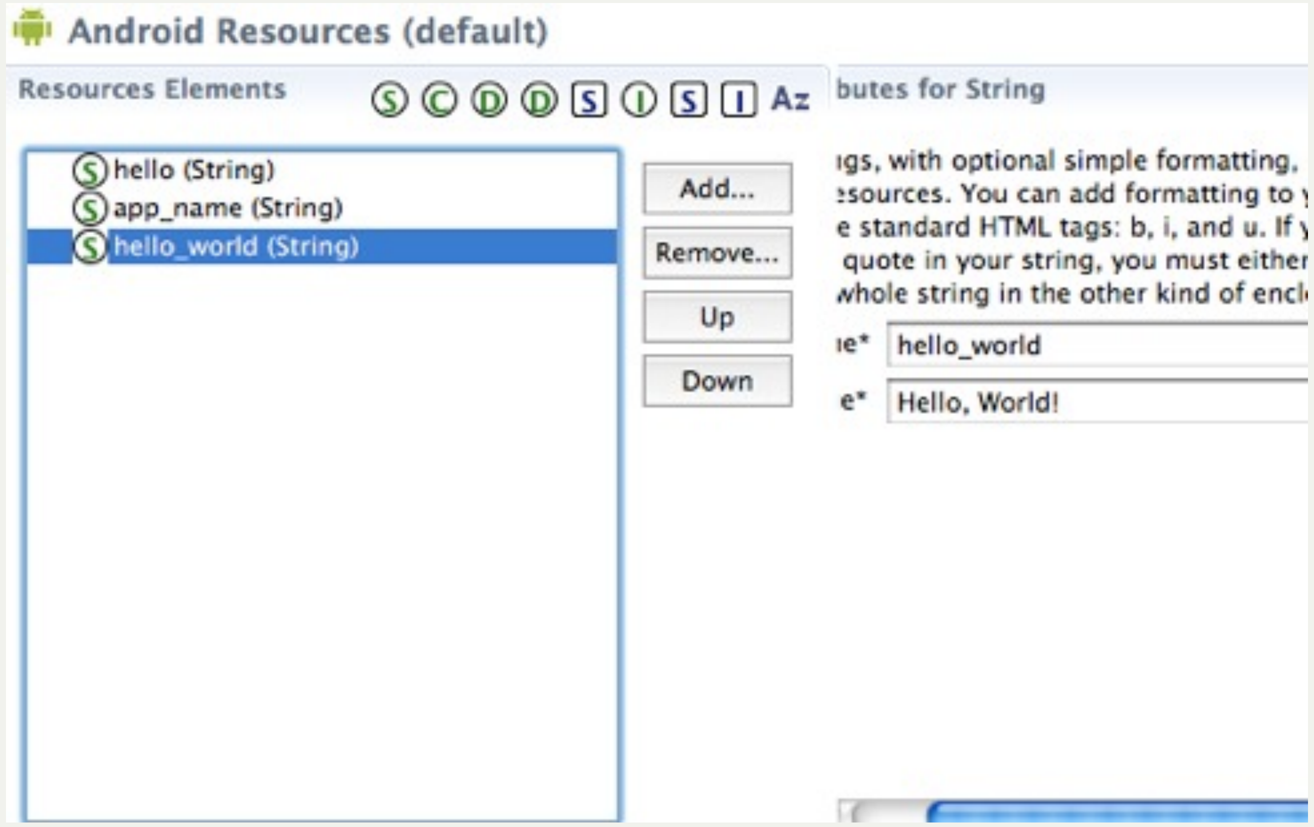
Resource Folders

- Separate storage of Strings and Graphics
- Makes it easier to modify software parts
- Resources are accessed via „R.java“



```
package de.maxmaurer.layouts;

import android.app.Activity;

public class Layouts extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        View v = findViewById(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText(getString(R.string.hello_world));
        setContentView(v);
    }
}
```

Resource Folders


Android Resources (default)

Resources Elements

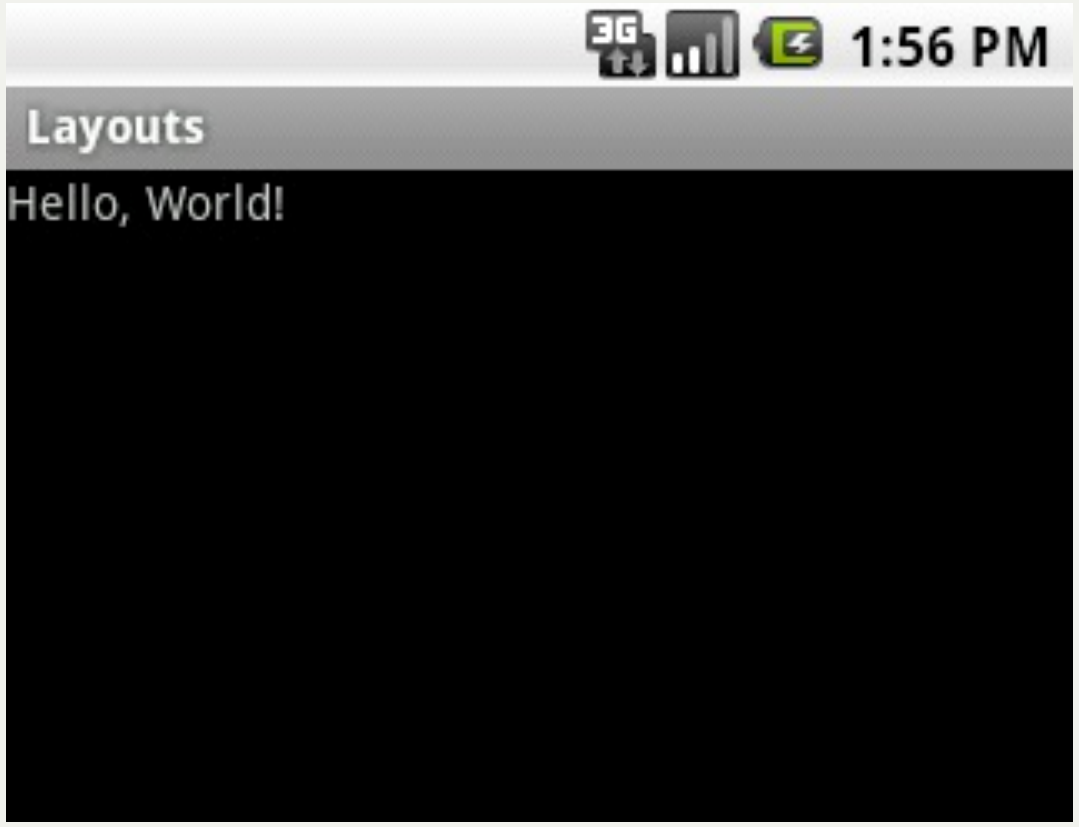
- hello (String)
- app_name (String)
- hello_world (String)

butes for String

ıgs, with optional simple formatting,
ɛsources. You can add formatting to
e standard HTML tags: b, i, and u. If y
quote in your string, you must either
whole string in the other kind of encl

ıe* hello_world

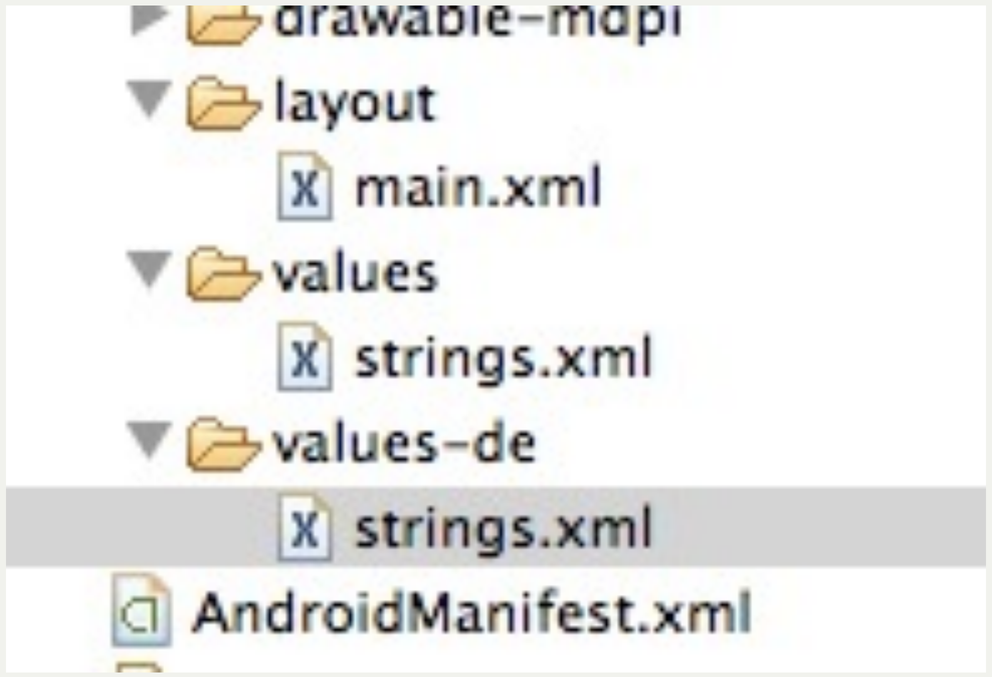
e* Hello, World!



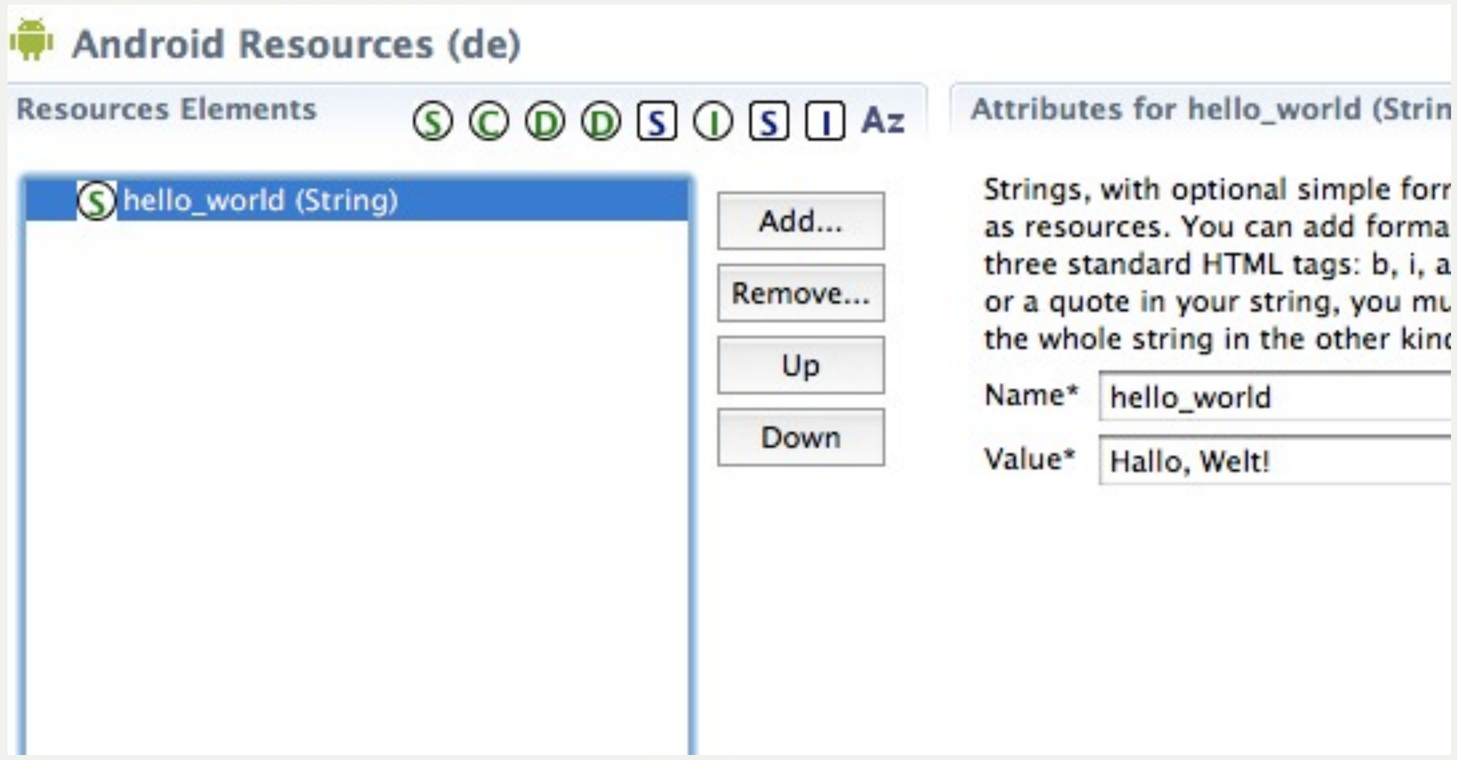
Hello, World!

Localization

- Creating folders for other languages does not need any code change
- Watch the application size!



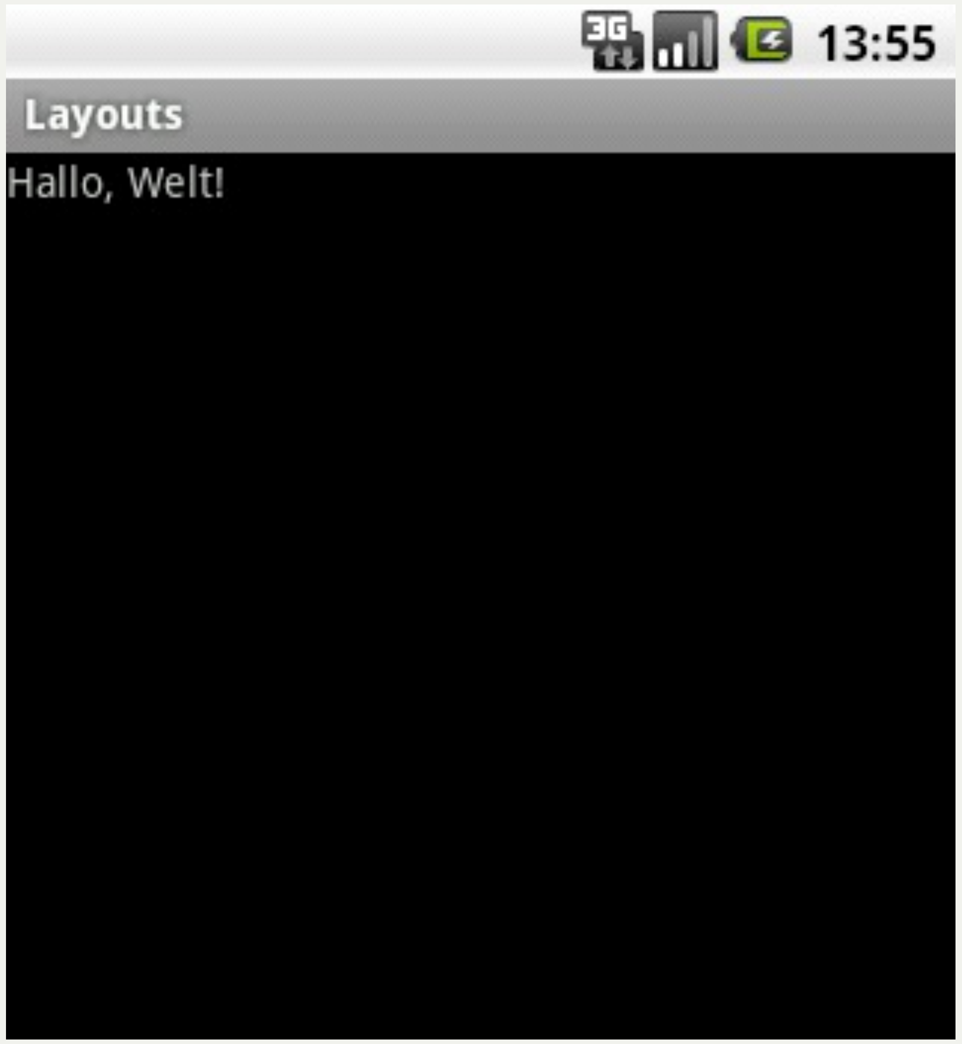
Localization



Strings, with optional simple forr
as resources. You can add forma
three standard HTML tags: b, i, a
or a quote in your string, you mu
the whole string in the other kinc

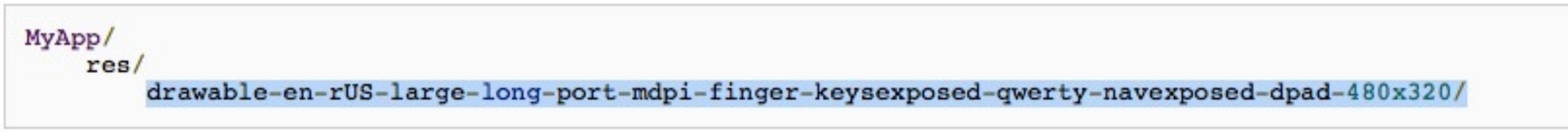
Name* hello_world

Value* Hallo, Welt!



Localization

- May be used for other device specific things as well
  - Country
  - Screen dimensions
  - Screen orientation
  - Touchscreen type (finger, stylus)
  - and many more



```
MyApp/
    res/
        drawable-en-rUS-large-long-port-mdpi-finger-keysexposed-qwerty-navexposed-dpad-480x320/
```

Application Themes

Implementing a User Interface

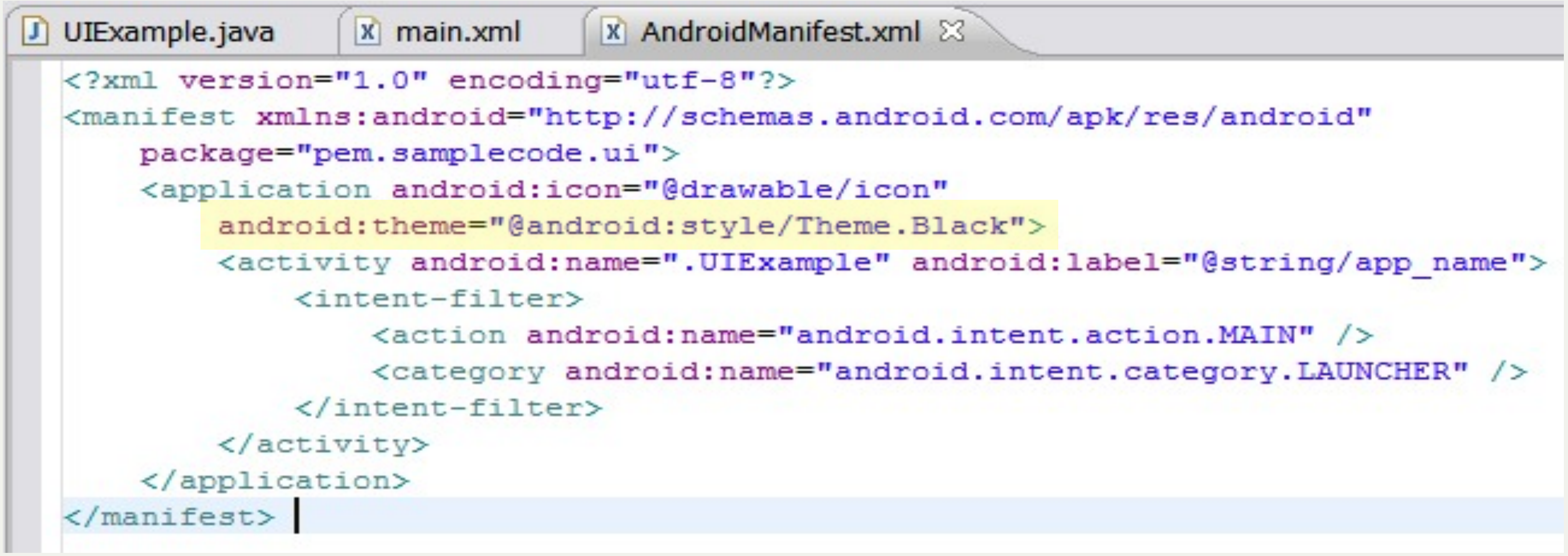
Applying a Theme to Your Application

- Default theme: android.R.style.Theme
  - http://developer.android.com/reference/android/R.style.html

- Two ways to set the theme
  - Adding the theme attribute in `AndroidManifest.xml`
  - Calling `setTheme()` inside the `onCreate()` method

Editing AndroidManifest.xml

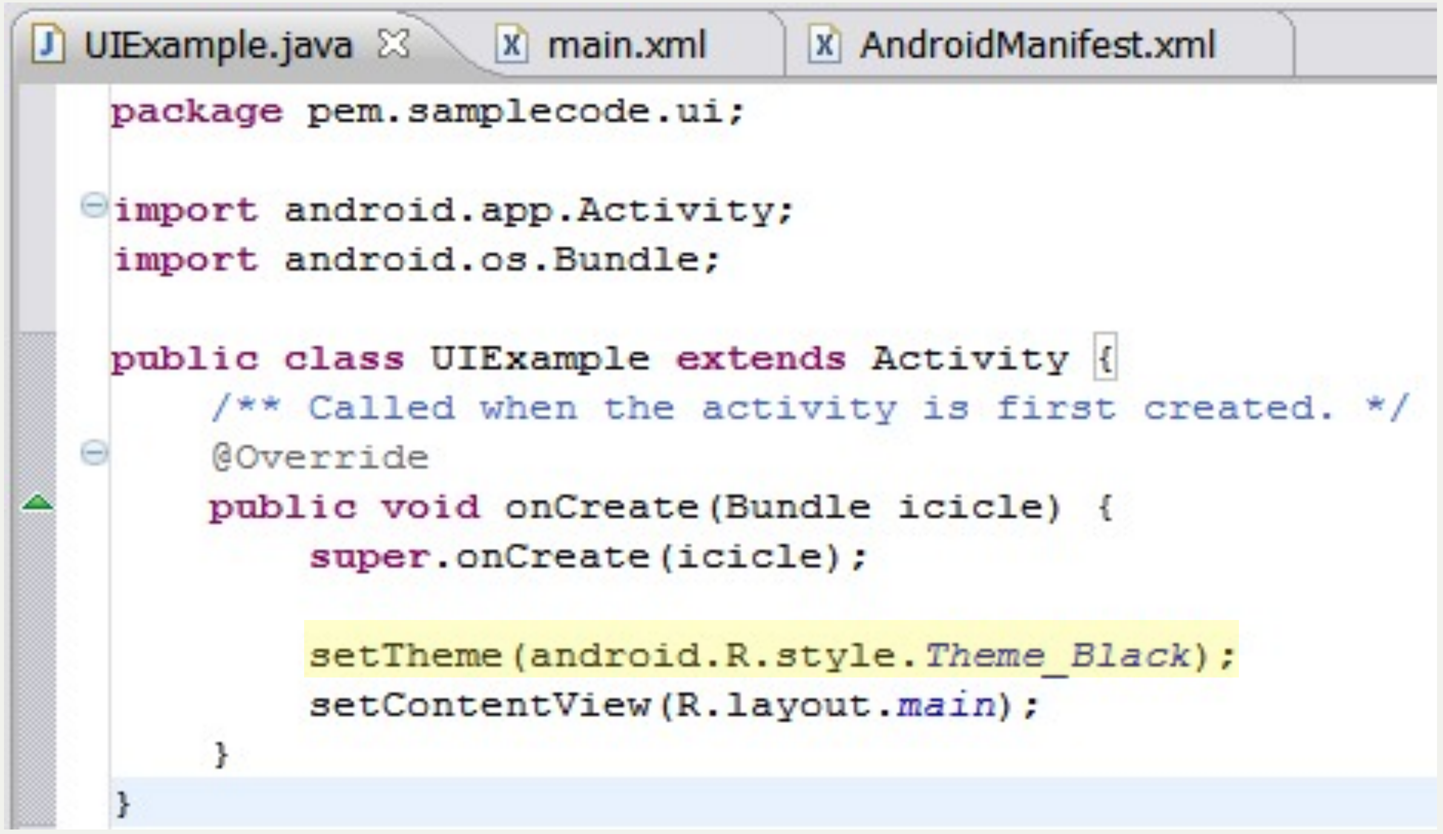
- Adding the theme attribute in AndroidManifest.xml



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="pem.samplecode.ui">
    <application android:icon="@drawable/icon"
        android:theme="@android:style/Theme.Black">
        <activity android:name=".UIExample" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

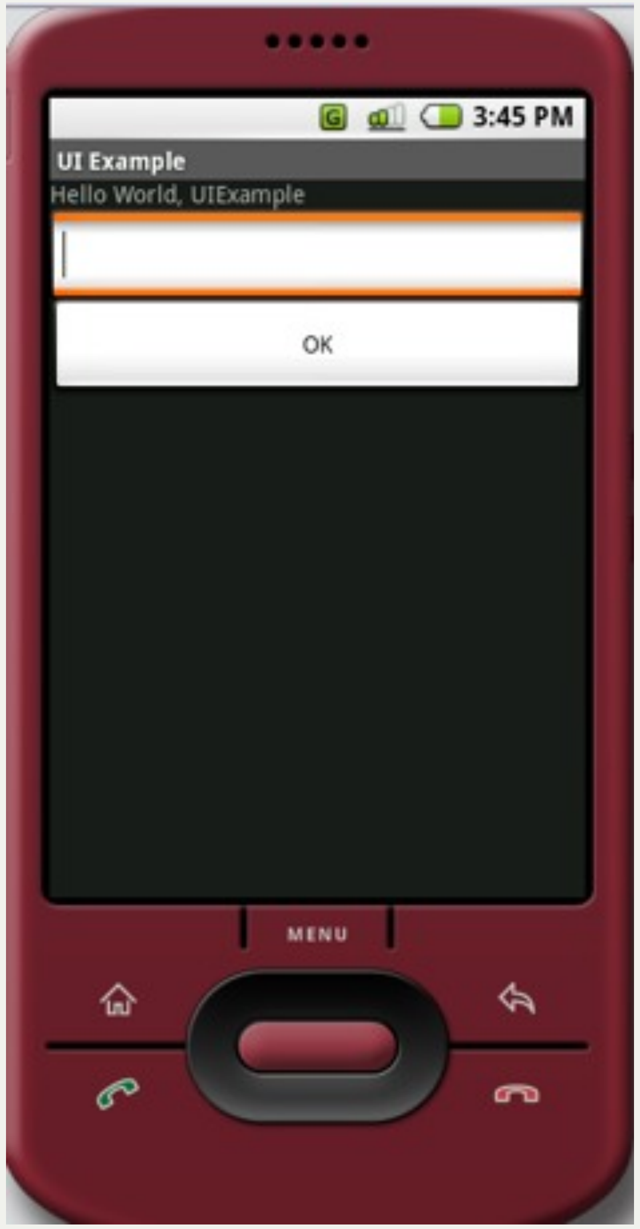
Applying a Theme using Code

- Calling setTheme() inside the onCreate() method

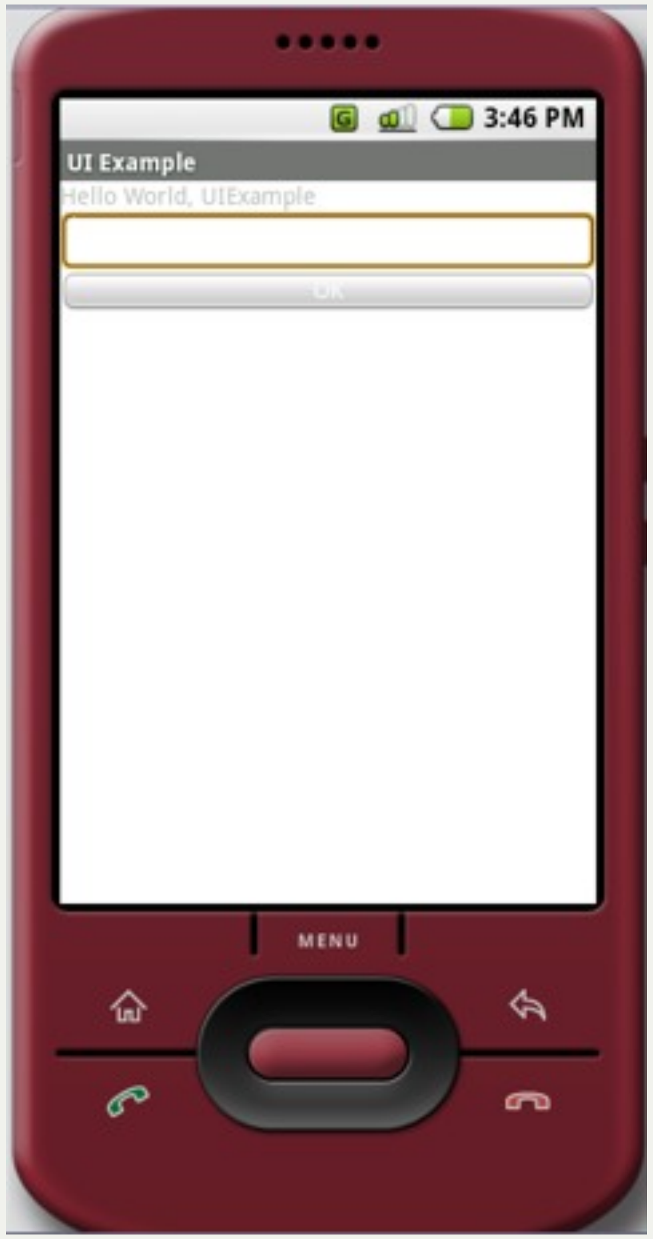


```
package pem.samplecode.ui;

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

public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        setTheme(android.R.style.Theme_Black);
        setContentView(R.layout.main);
    }
}
```

Black



Light Weight

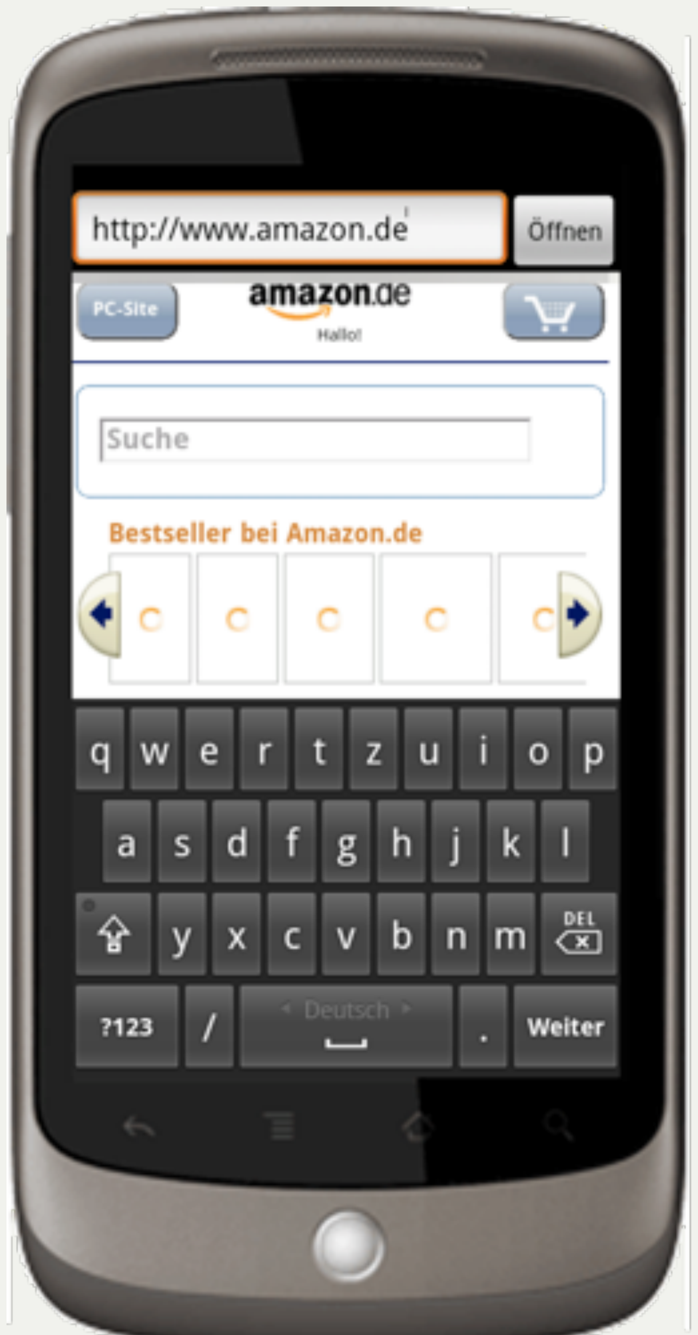


Exercise 2

Implementing a User Interface

Exercise 2

- Fortführung der bisherigen Aufgabe
- In neues Projekt kopieren
- Browser um Adresszeile und „GoTo"-Button ergänzen
- Zurück und Vorwärts-Button ergänzen
- **Vorsicht bei Seiten mit Redirect!**



Fragen?
Viel Spaß!