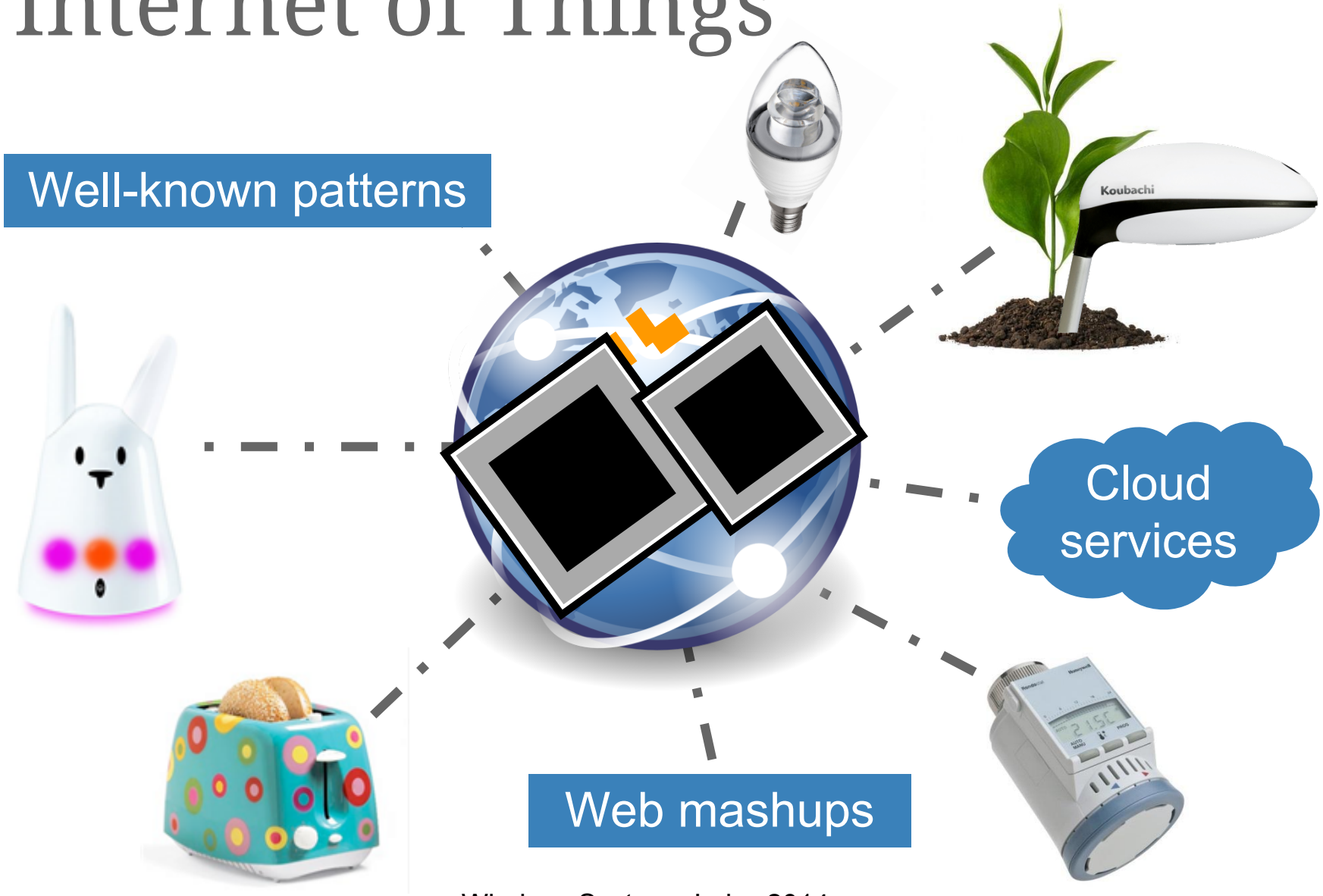
Wireless Systems Lab

First Lesson

About this course

- Internet of Things
- Android and sensors
- Mobile sensing
  - Indoor localization
  - Activity recognition
  - others..
- Exercises
- Projects :)

Internet of Things



What is IoT?

- New technology
  - Wireless, communication, low-power, large-scale, big data, Internet-connectivity
- New concepts
  - New ways of interaction, new ways of thinking
- New business opportunities
- Emerging products and systems

Android Introduction

First Part: Application Fundamentals

Wireless Systems Lab - 2014

Goal

- Understand applications and their components
- Concepts:
  - activity
  - service
  - broadcast receiver
  - content provider
  - intent
  - AndroidManifest

What is Google Android?

What is Google Android?

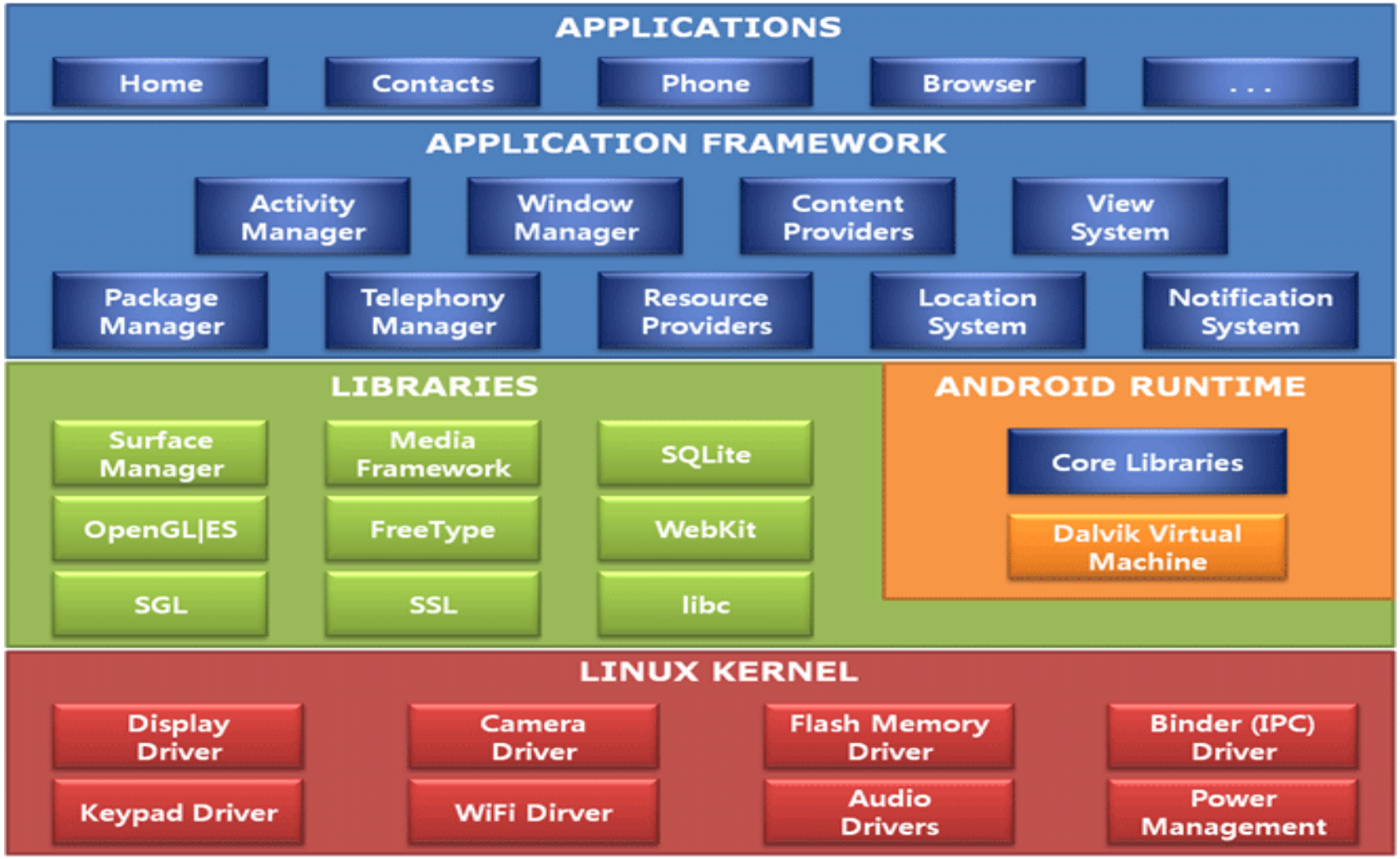
- A software stack for mobile devices that includes
  - An operating system
  - Middleware
  - Key Applications

- Uses Linux to provide core system services
  - Security
  - Memory management
  - Process management
  - Power management
  - Hardware drivers

Applications

- Written in Java (it's possible to write native code – will not cover that here)
- Good separation (and corresponding security) from other applications:
  - Each application runs in its own process
  - Each process has its own separate VM
  - Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported)

Android Components



Android History

Building Blocks

Application Components

- Activities – visual user interface focused on a single thing a user can do
- Services – no visual interface – they run in the background
- Broadcast Receivers – receive and react to broadcast announcements
- Content Providers – allow data exchange between applications

Activities

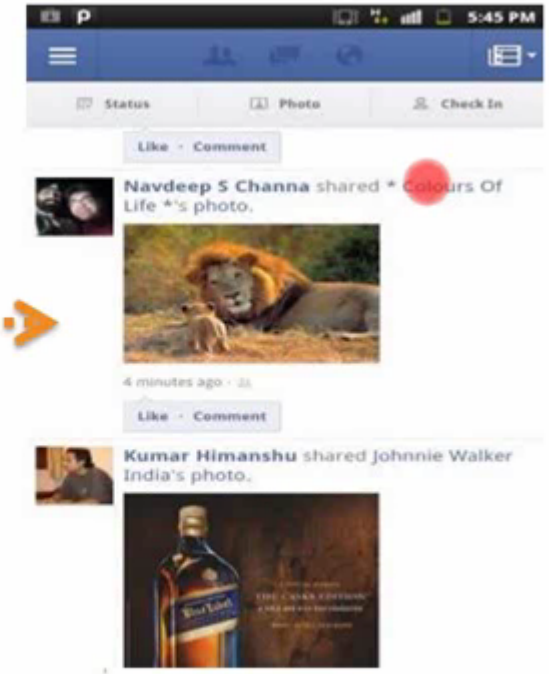
- Basic component of most applications
- Most applications have several activities that start each other as needed
- Each is implemented as a subclass of the base Activity class

Example of activity

Login Activity

Example of activity

Login Activity



News Feed Activity

Example of activity

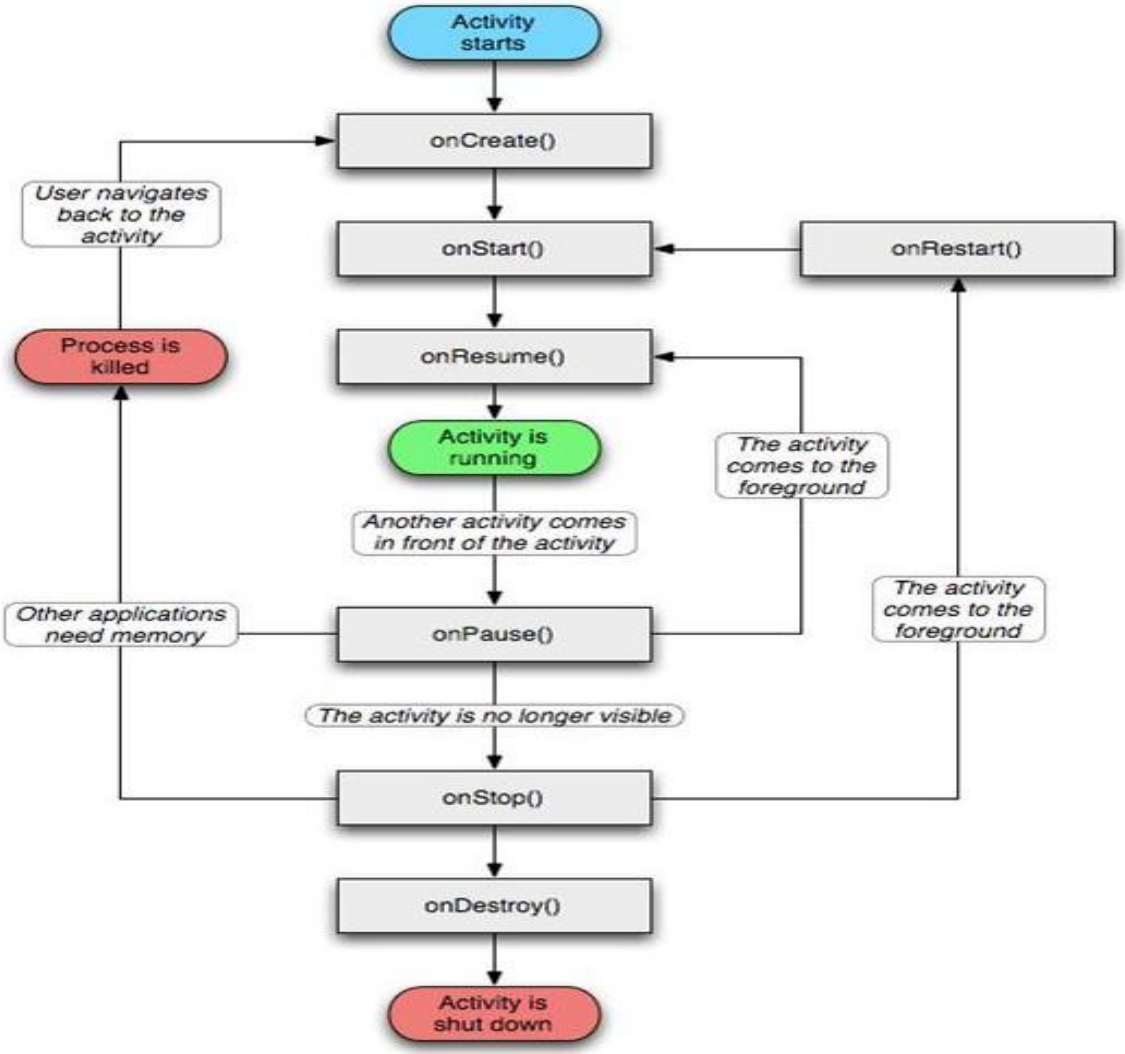
```
/* Example.java */
package uk.ac.ic.doc;
import android.app.Activity;
import android.os.Bundle;

public class Example extends Activity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.interface);
    }
}
```

Activities – The View

- Each activity has a default window to draw in (although it may prompt for dialogs or notifications)
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

Activity lifecycle



Services

- Does not have a visual interface
- Runs in the background indefinitely
- Examples
  - Network Downloads
  - Playing Music
  - TCP/UDP Server
- You can bind to a an existing service and control its operation

Broadcast Receivers

- Receive and react to broadcast announcements
- Extend the class BroadcastReceiver
- Examples of broadcasts:
  - Low battery, power connected, shutdown, timezone changed, etc.
  - Other applications can initiate broadcasts

Content Providers

- Makes some of the application data available to other applications
- It's the only way to transfer data between applications in Android (no shared files, shared memory, pipes, etc.)
- Extends the class ContentProvider;
- Other applications use a ContentResolver object to access the data provided via a ContentProvider

Intents

- An intent is an Intent object with a message content.
- Activities, services and broadcast receivers are started by intents. ContentProviders are started by ContentResolvers:
  - An activity is started by Context.startActivity(Intent intent) or Activity.startActivityForResult(Intent intent, int RequestCode)
  - A service is started by Context.startService(Intent service)
  - An application can initiate a broadcast by using an Intent in any of Context.sendBroadcast(Intent intent), Context.sendOrderedBroadcast(), and Context. sendStickyBroadcast()

Intents

- Allows communication between components
  - Message passing
  - Bundle

```
Intent intent = new Intent(CurrentActivity.this, OtherActivity.class);
startActivity(intent);
```

Intents

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // Button listener
    Button btnStart = (Button) findViewById(R.id.btn_start);
    btnStart.setOnClickListener(new View.OnClickListener() {
       public void onClick(View view) {
         Intent intent =
            new Intent(CurrentActivity.this, OtherActivity.class);
        startActivity(intent);
       }
    });
}
```

Shutting down components

- Activities
  - Can terminate itself via finish();
  - Can terminate other activities it started via finishActivity();
- Services
  - Can terminate via stopSelf(); or Context.stopService();
- Content Providers
  - Are only active when responding to ContentResolvers
- Broadcast Receivers
  - Are only active when responding to broadcasts

Android Manifest

- Its main purpose in life is to declare the components to the system:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="uk.ac.ic.doc" android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
       android:label="@string/app_name">
       <activity android:name=".SampleActivity"
          android:label="@string/activity_title_text_ref">
         <intent-filter>
            /* ... */
         </intent-filter>
       </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

Intent Filters

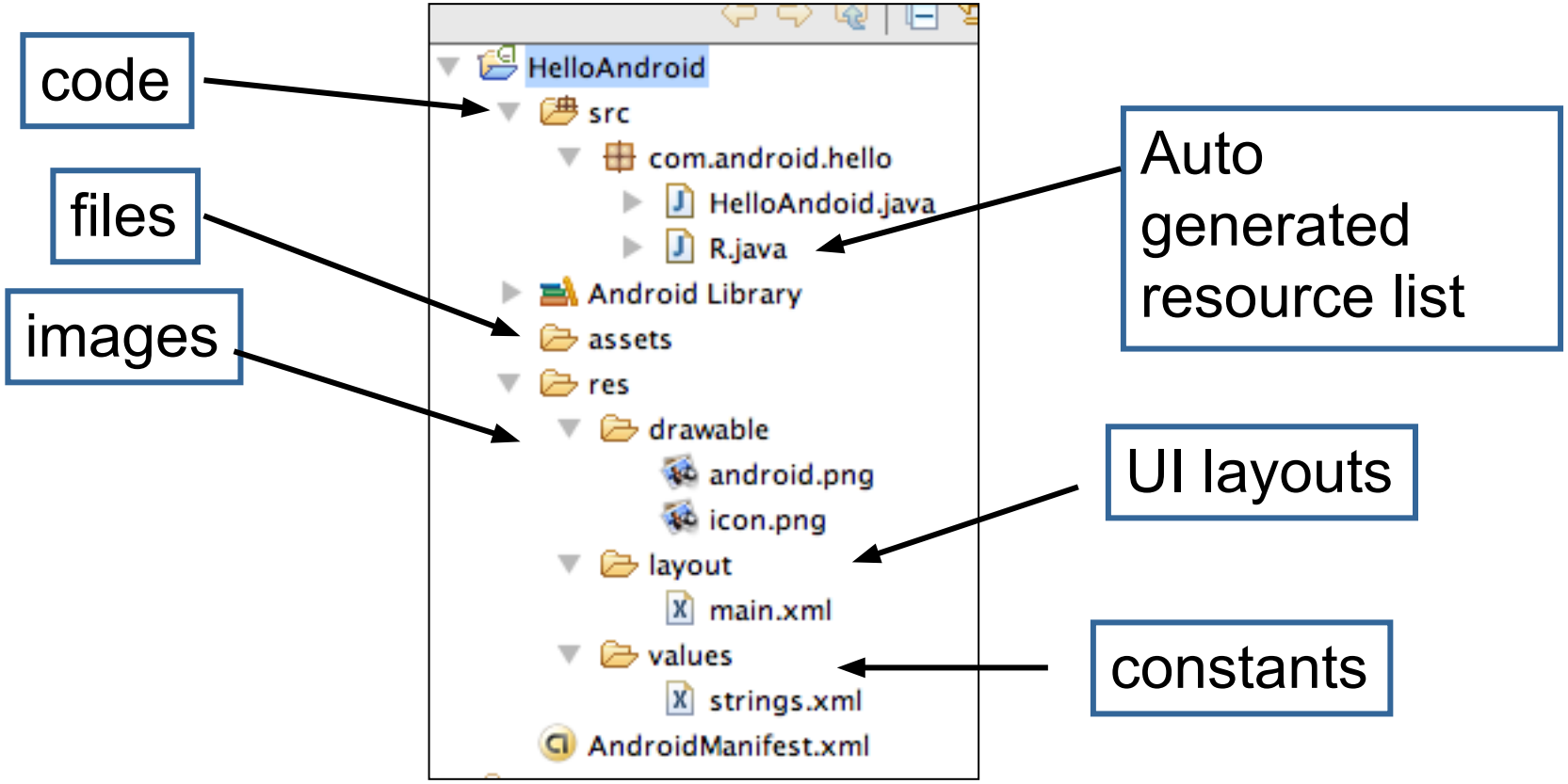
- Declare Intents handled by the current application (in the AndroidManifest):

```
<?xml version="1.0" encoding="utf-8"?>
 <manifest . . . >
    <application . . . >
       <activity android:name="com.example.project.FreneticActivity"
              android:icon="@drawable/small_pic.png"
              android:label="@string/freneticLabel"
              . . .  >
          <intent-filter . . . >
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
          <intent-filter . . . >
             <action android:name="com.example.project.BOUNCE" />
             <data android:mimeType="image/jpeg" />
             <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
       </activity>
       . . .
    </application>
 </manifest>
```

Shows in the Launcher and is the main activity to start

Handles JPEG images in some way

There is a common file structure for applications



Android Design Philosophy

- Applications should be:
  - Fast
    - Resource constraints: <200MB RAM, slow processor
  - Responsive
    - Apps must respond to user actions within 5 seconds
  - Secure
    - Apps declare permissions in manifest
  - Seamless
    - Usability is key, persist data, suspend services
    - Android kills processes in background as needed

Other design principles

- http://developer.android.com/design/index.html

- Great reference!

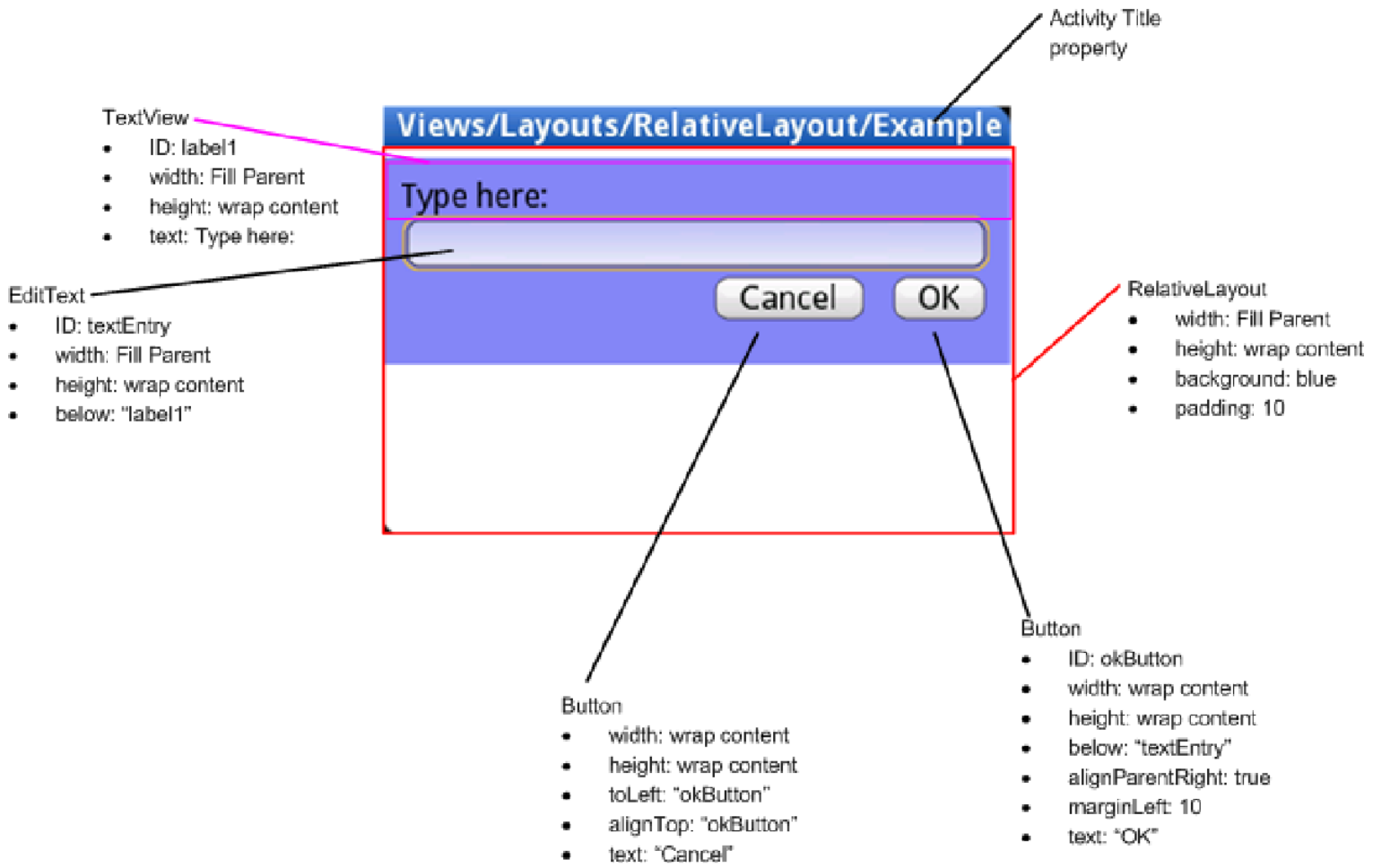
Android Introduction

Second Part: User Interface

Wireless Systems Lab - 2014

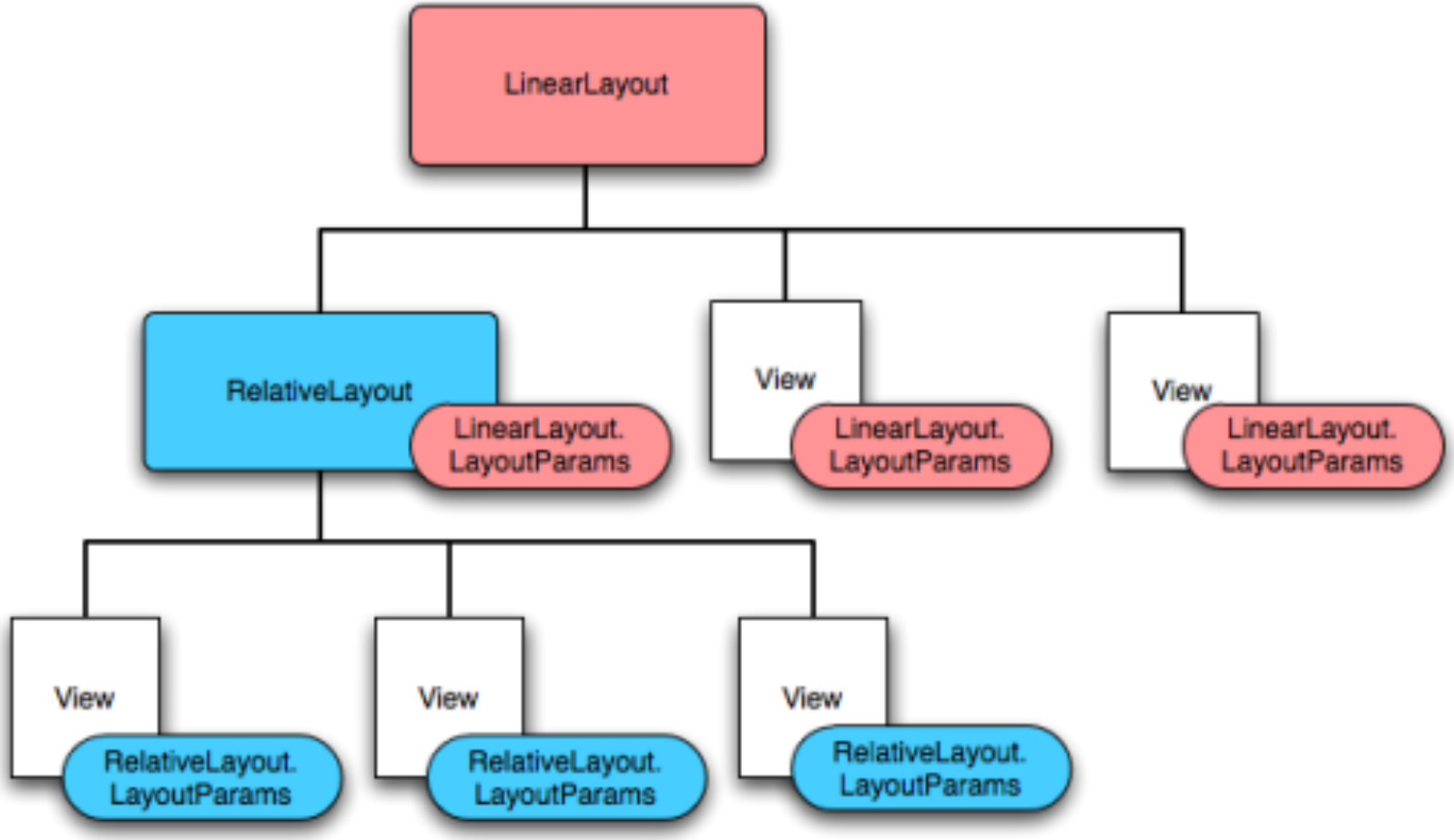
There are many types of UI components in Android

UI layouts are in Java and XML

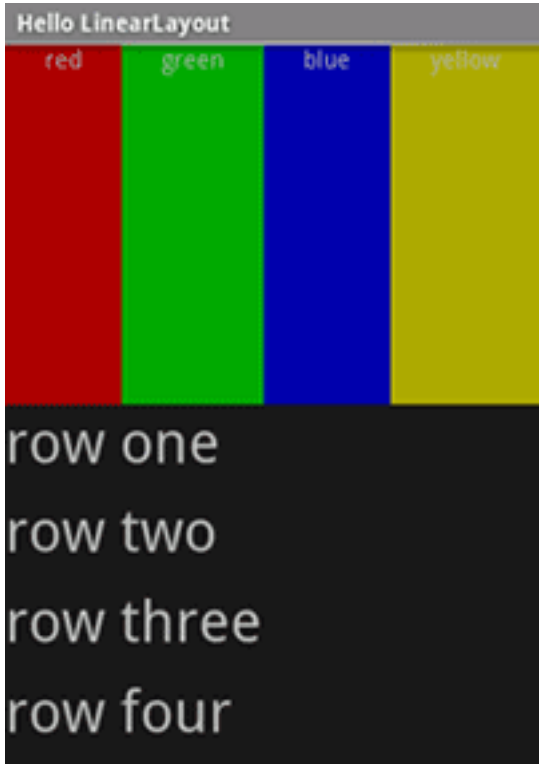


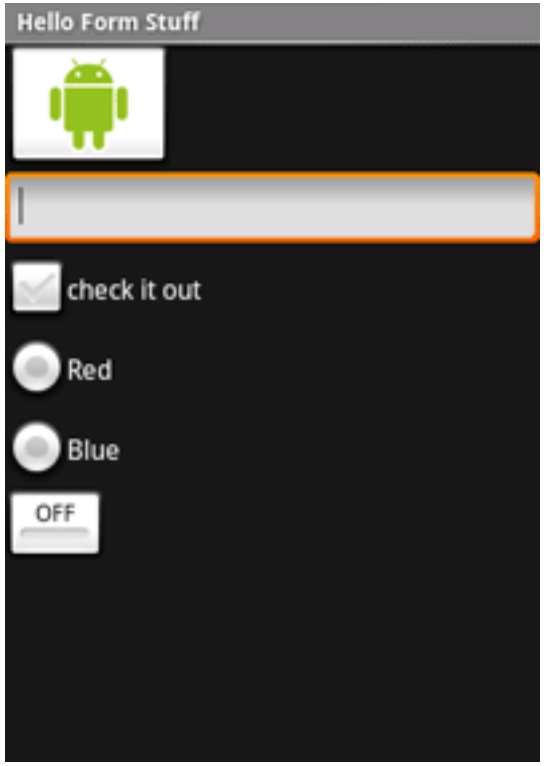

```
setContentView(R.layout.hello_activity); //will load the XML UI file
```

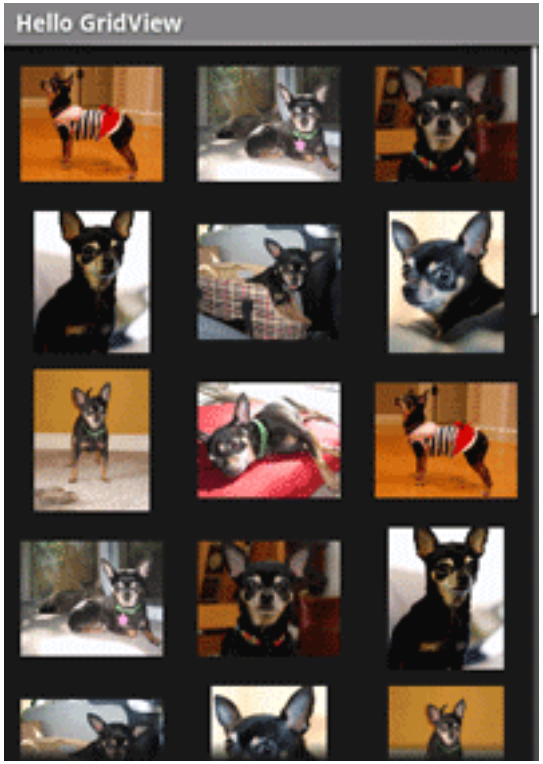
All layouts are hierarchical



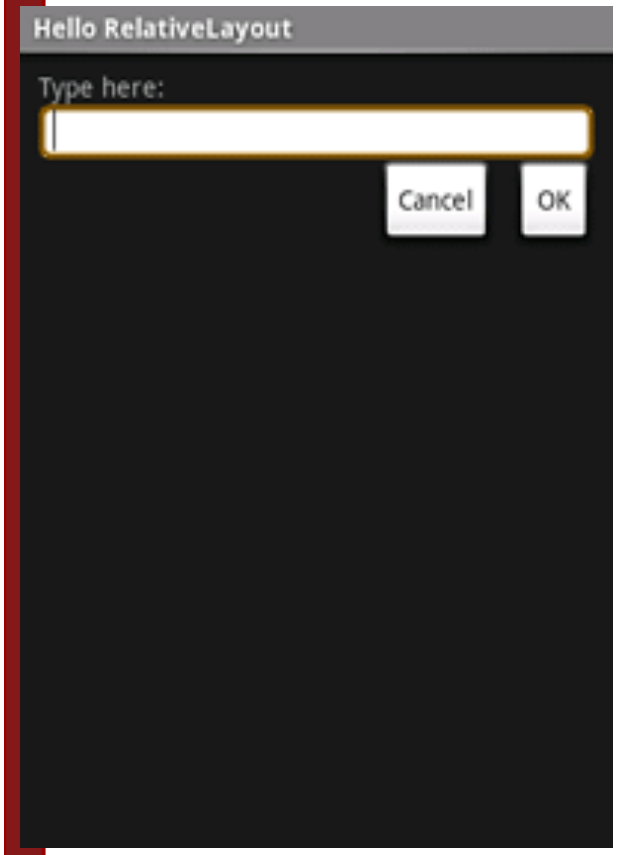
Views

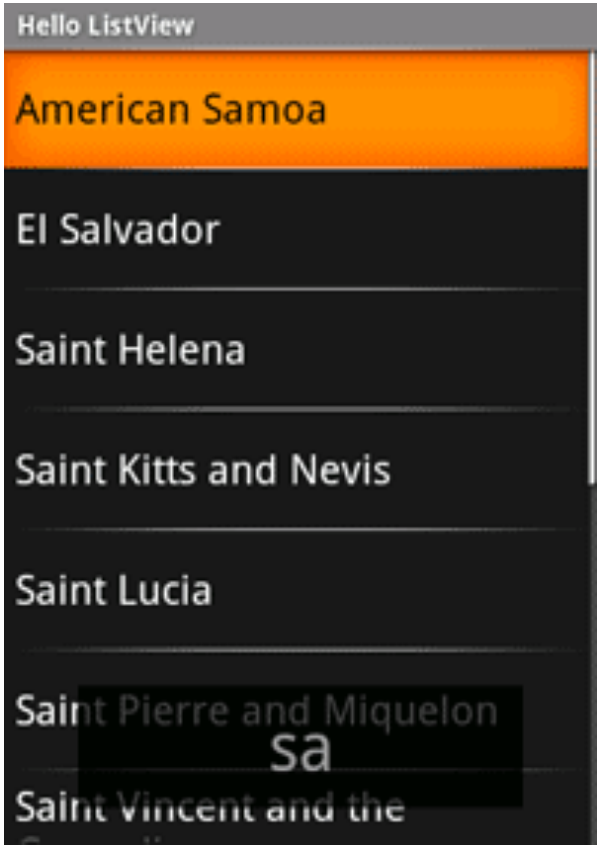

Hello LinearLayout
red
green
blue
yellow
row one
row two
row three
row four

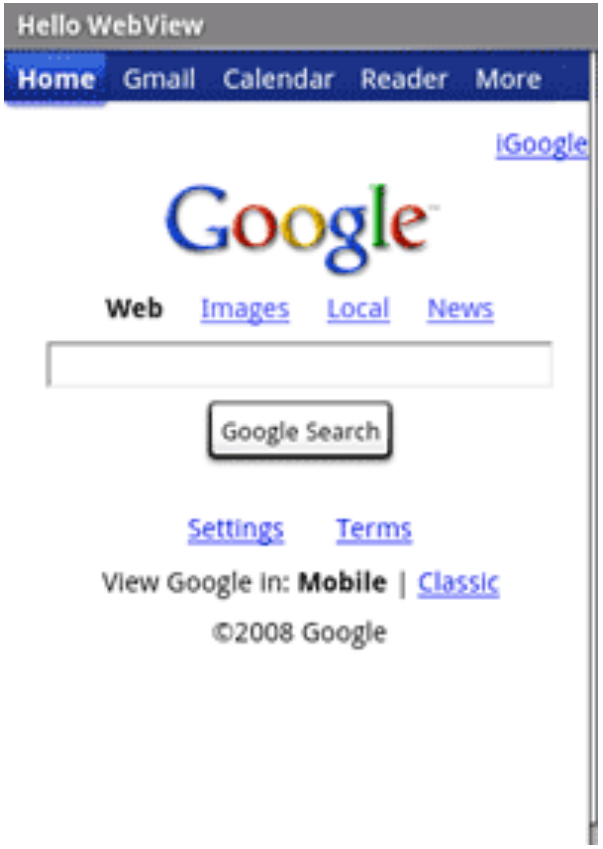

Hello Form Stuff
check it out
Red
Blue
OFF


Hello GridView

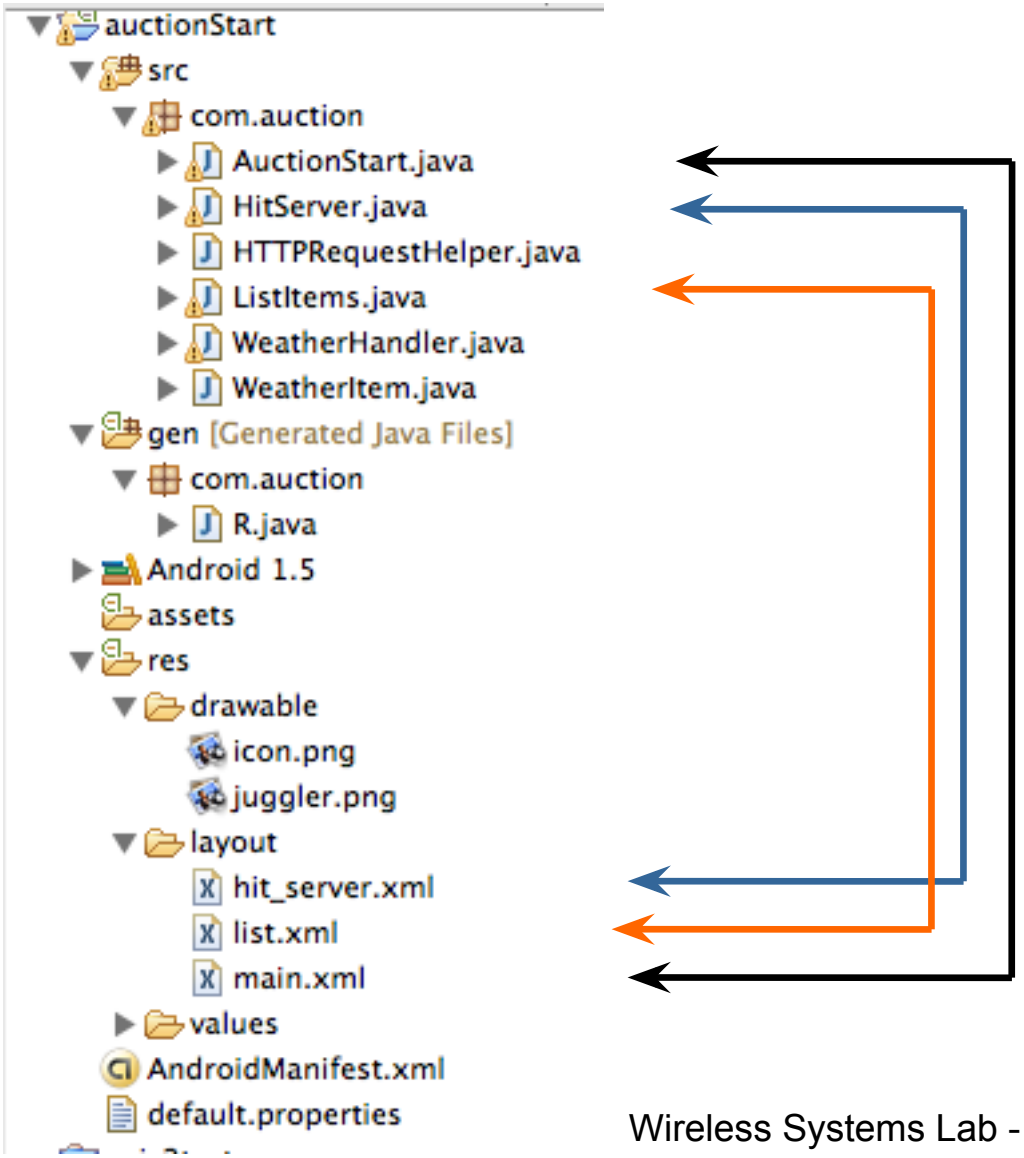
Views


Hello RelativeLayout
Type here:
Cancel
OK


Hello ListView
American Samoa
El Salvador
Saint Helena
Saint Kitts and Nevis
Saint Lucia
Saint Pierre and Miquelon
sa
Saint Vincent and the


Hello WebView
Home
Gmail
Calendar
Reader
More
iGoogle
Google
Web
Images
Local
News
Google Search
Settings
Terms
View Google in: Mobile | Classic
©2008 Google

One layout per activity (class)



main.xml goes with AuctionStart

list.xml goes with ListItems

hit_server.xml goes
with HitServer

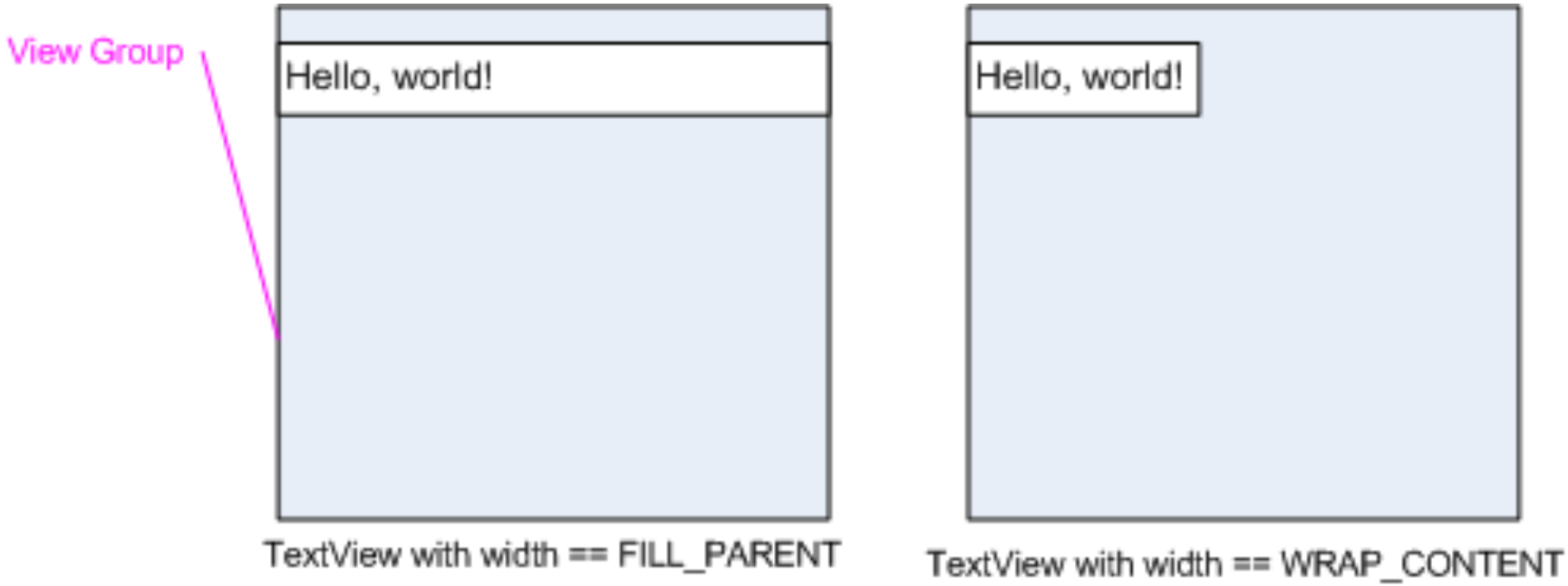
Xml layout file details components

```xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content" android:layout_height="wrap_content">
    <TableRow>
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/BackButton"
        android:text="Back"></Button>

    <EditText android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="server says: "
        android:id="@+id/welcomeMessage"></EditText>
        </TableRow>
        <TableRow>
    <EditText android:layout_height="wrap_content" android:id="@+id/serverResponse"
        android:text="response" android:layout_width="fill_parent"></EditText>
    </TableRow>
</TableLayout>
```

Hierarchy of views as noted earlier

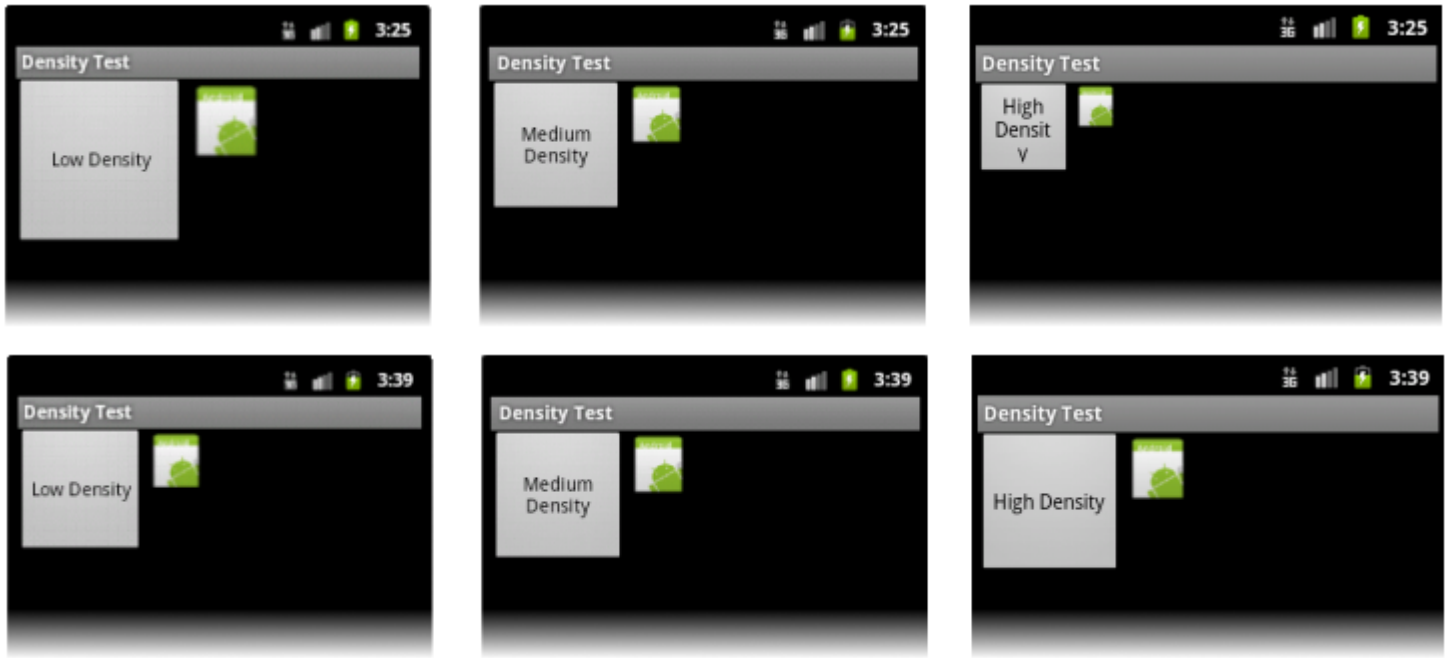
Children do as told in Android



TextView is child of parent viewgroup and fills, or wraps content

Elements and layouts

- dip vs. px
- Component dimensions
  - wrap_content
  - match_parent



Linear Layout

```
/* linear.xml */
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
    <TextView android:text="red" />
    <TextView android:text="green" />
</LinearLayout>
<LinearLayout android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
    <TextView android:text="row one" />
</LinearLayout>
```

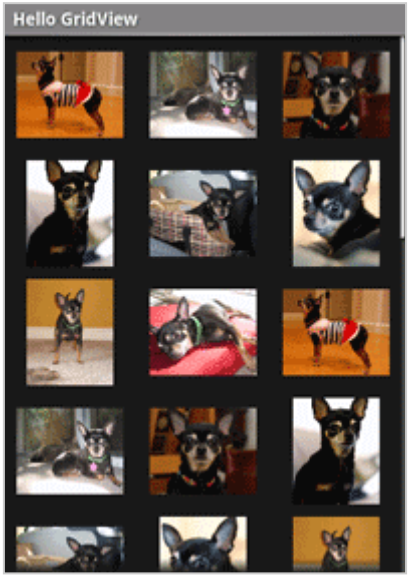
Table Layout

- Like the HTML **div** tag

```
/* table.xml */
<?xml version="1.0" encoding="utf-8"?>
<TableLayout android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
   <TableRow>
      <TextView android:layout_column="1"
          android:text="Open..."
         android:padding="3dip" />
      <TextView android:text="Ctrl-O"
         android:gravity="right"
         android:padding="3dip" />
    </TableRow>
</TableLayout>
```

Grid View Layout

```
/* grid.xml */
<?xml version="1.0" encoding="utf-8"?>
<GridView
    android:id="@+id/gridview"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:columnWidth="90dp"
   android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
   android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
   android:gravity="center"
/>
```

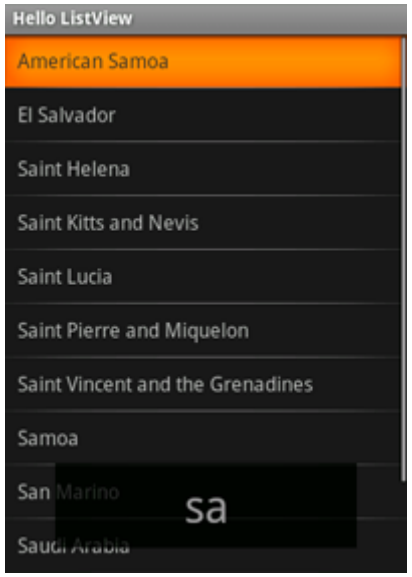


Tab Layout

```
/* tab.xml */
<?xml version="1.0" encoding="utf-8"?>
<TabHost android:id="@android:id/tabhost"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">
        <TabWidget android:id="@android:id/tabs"
           android:layout_width="match_parent"
           android:layout_height="wrap_content"/>
        <FrameLayout
           android:layout_width="match_parent"
           android:layout_height="match_parent"/>
    </LinearLayout>
</TabHost>
```

List View Layout

```
/* list_item.xml */
<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    android:textSize="16sp" />
```



List View Layout

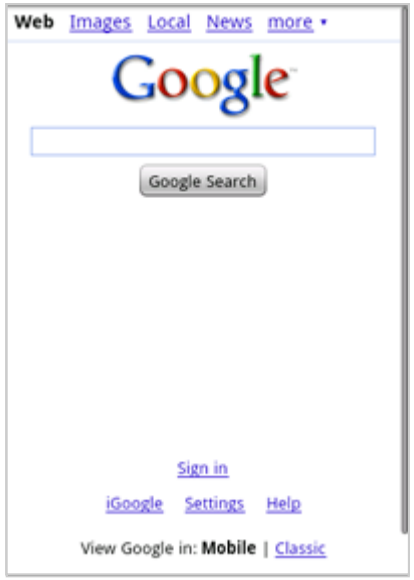
- List View in Java

```
/* ListViewExample.java */
public class ListViewExample extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
       setListAdapter(new ArrayAdapter<String>(this,
              R.layout.list_item, COUNTRIES));
       ListView lv = getListView();//
      lv.setTextFilterEnabled(true);
      lv.setOnItemClickListener(new OnItemClickListener() {
         public void onItemClick(AdapterView<?> parent, View view,
                                                int position, long id) {
            Toast.makeText(getApplicationContext(),
             ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
       }});
}
```

Lists can be handled via adapters and filled from xml file of values

More Element and layouts...

- DatePicker
- TimePicker
- Spinner
- AutoComplete
- Gallery
- MapView
- WebView



Lets build a simple App

- Form Activity with Display Activity