Android Development

Session 1

Javier Poncela

Contents

Android Phones

HTC G1

Samsung i7500

HTC Hero

Motorola Cliq

Sony X10

HTC Magic

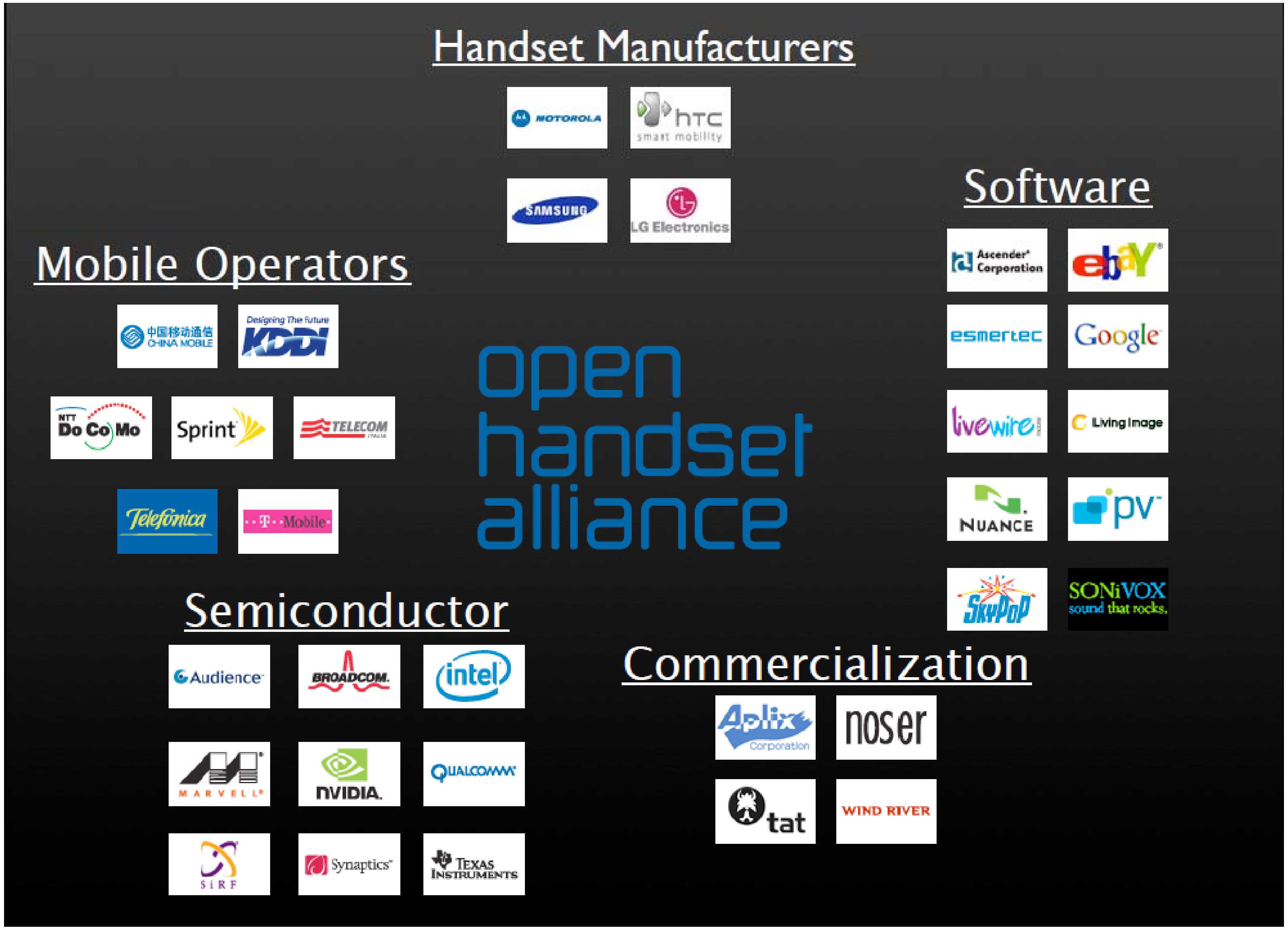
Samsung Moment

Motorola Droid

HTC Tattoo

nexus one

Open Handset Alliance



Open Handset Alliance

- OHA(Open Handset Alliance) is a group of 71 technology and mobile companies, including Google, Intel, Dell, HTC and China Mobile...
- Google announced the Open Handset Alliance and the Android platform in November of 2007, releasing the first beta version Android Software Development Kit (SDK) at the same time
  - Within a matter of a few months, over 1 million people had downloaded versions of the SDK
- OHA's aim
  - Accelerate innovation in mobile phones
  - Offer consumers a richer, less expensive, and better mobile experience
- Google is the 'captain'

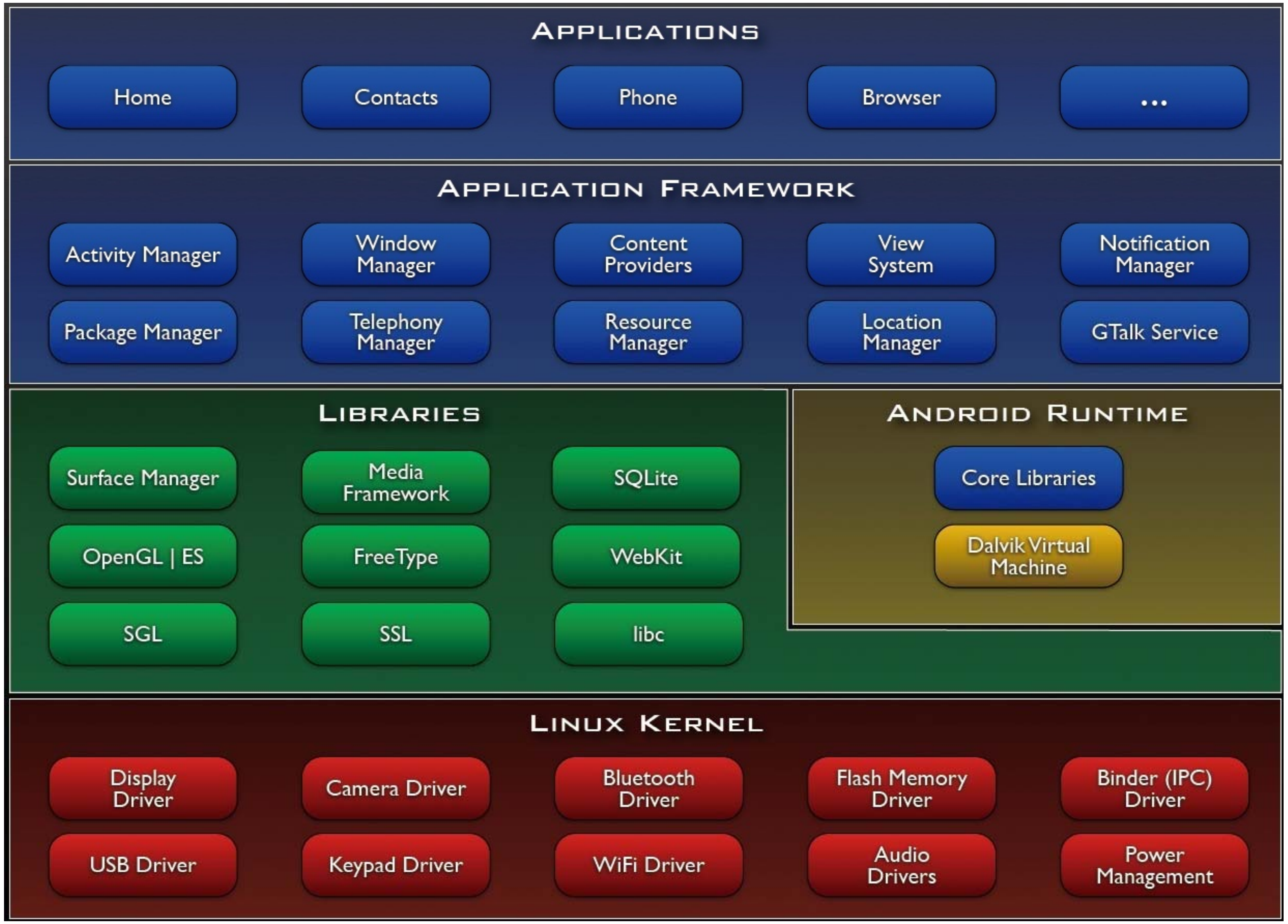
What's Android

- Android is a software stack for mobile devices that includes an operating system, middleware and key applications
- Android is based on JAVA and all its applications are developed in JAVA
- The JAVA VM, known as Dalvik, is highly customized and optimized for mobile devices
- Android SDK offers rich tools for android application development and many useful APIs

Android Features

- Application framework enabling reuse and replacement of components
- Optimized Java virtual machine: Dalvik
- Optimized Graphics Processing, supporting 2D and 3D graphics(OpenGL ES 1.0)
- Integrated open source web browser: WebKit
- SQLite for structured data storage
- Multimedia capability, supporting varieties of audio, video and still image formats
- Bluetooth, GSM, EDGE, 3G and Wi-Fi support
- Camera, GPS, compass, accelerometer and other sensors
- Rich development environment, including an emulator, debugging tools, memory probe tools, log tools and eclipse plugins

Android Architecture



Libraries

- **System C library,** the standard C system library, tuned for embedded Linux-based devices
- **Media Libraries,** support playback and recording of many popular audio and video formats, as well as image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager,** manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- **WebKit,** a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL**, the underlying 2D graphics engine
- **3D libraries,** an implementation based on OpenGL ES 1.0 APIs
- **FreeType** , bitmap and vector font rendering
- **SQLite** , a powerful and lightweight relational database engine

Android Runtime

- The core of Android platform
- Dalvik Virtual Machine (VM)
  - Executes files in Dalvik Executable (.dex) format
- Java core Libraries
  - Provides most of the functionality of the Java language
- How many Dalvik VMs?
  - Multiple Dalvik VMs may run at the same time
  - Every Android application runs in its own process, with its own instance of the Dalvik virtual machine

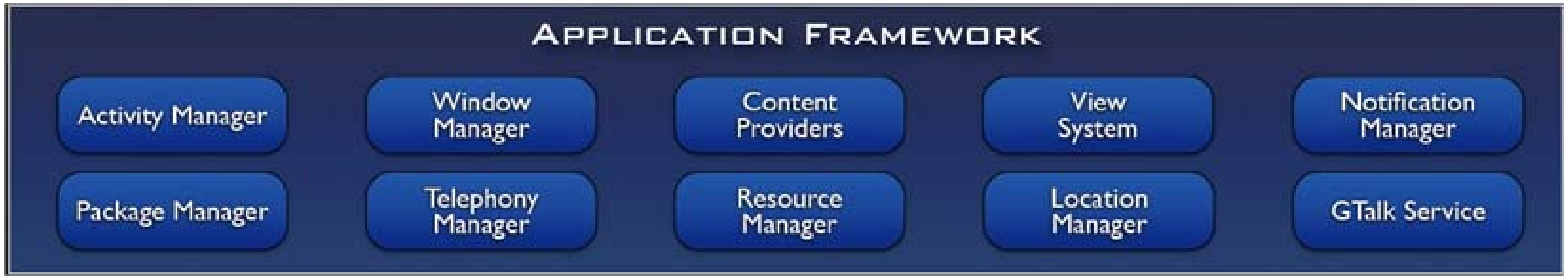
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine

DVM	JVM
Google	Sun
Dalvik Executable	Java Bytecode
Subset of standard Java Library	

- Slow CPU
- Little RAM: 64Mb total, ~10Mb available at runtime
- No swap space

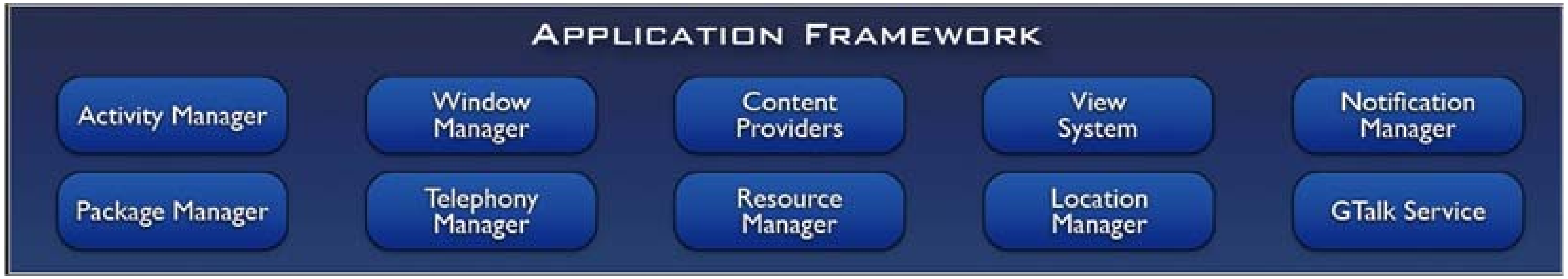
Application Framework

- Simplify the reuse of components
  - Applications can publish their capabilities and any other application may then make use of those capabilities
- Applications is a set of services and systems, include
  - **Activity Manager,** manages the lifecycle of applications and provides a common navigation back stack
  - **Notification Manager,** enables all applications to display custom alerts in the status bar
  - **…**



Application Framework

- **…**
- **Resource Manager**, providing access to non-code resources such as localized strings, graphics, and layout files
- **Content Providers,** access data from other applications (such as Contacts), or to share their own data
- **Views,** used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser



Applications

- A set of core applications shipped with Android platform
  - E-mail client, SMS program, calendar, maps, browser, contacts, and others
- All written in **Java** using Google SDK

- There is no difference between the built-in applications and applications created with SDK
  - Application framework encourages reuse of application components

APPLICATIONS
Home
Contacts
Phone
Browser
...

Applications

- Applications are bundled into an Android PacKage (.apk files) which are archives containing the compiled code, data and resources for the application, so applications are completely self-contained

- You can install applications either through a market (Google Play Store, Amazon Appstore, F-Droid, etc), manually (through ADB or a file manager), from a web server, …

APPLICATIONS
Home
Contacts
Phone
Browser
...

Application Development

- New way of thinking…
  - Limited processing power
  - Limited RAM
  - Limited permanent storage capacity
  - Small screen and low resolution (5”, 10”)
  - High cost of data transfer
  - Slow data transfer rates with high latency
  - Unreliable data connections

- OS Manages process lifetime (app assassin)
  - App responsiveness
  - Setting priority to interaction

Application Development

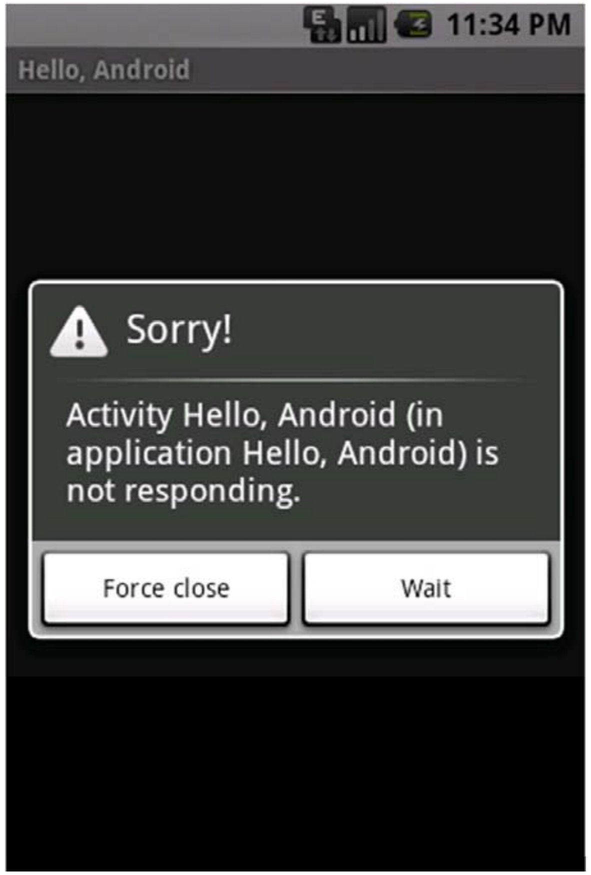
- Behaviour Police

  5s: Application must respond to any user action (e.g., key press) within 5s

  10s: A BroadcastReceiver must return from its OnReceive handler within 10s

  ** Worst case, not goal!:

  <u>Users notice .5s</u>



Activity Hello, Android (in application Hello, Android) is not responding.

Application Development

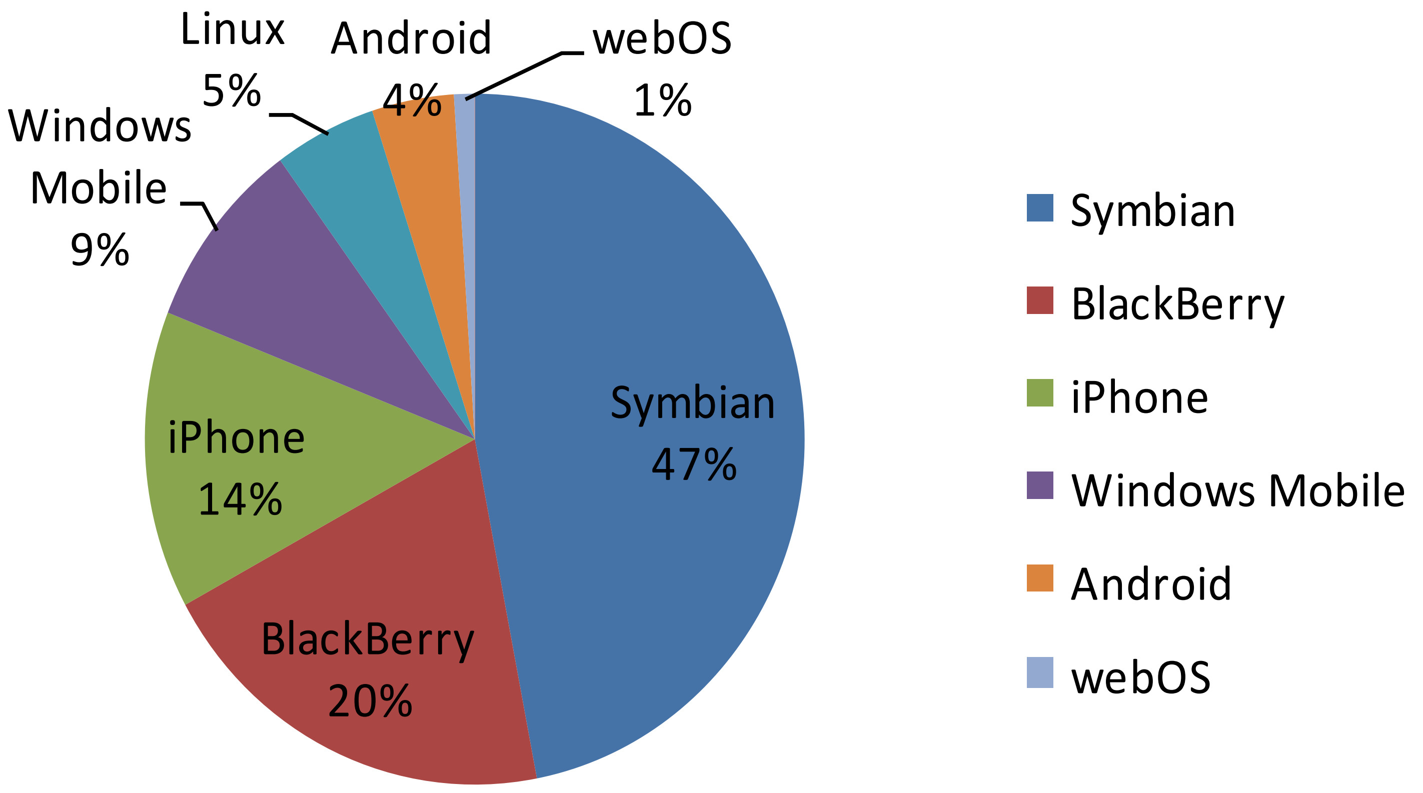
- You MUST
  - Ensure that your app is ready for swift death
  - Yet, it must remain response and/or restart in the background
  - Must come to the foreground quickly

- Good behavihour is expected
  - Is well behaved
  - Switches seamlessly from background to foreground
  - Is polite (e.g., stealing focus)
  - Presents a consistent user interface
  - Is responsive

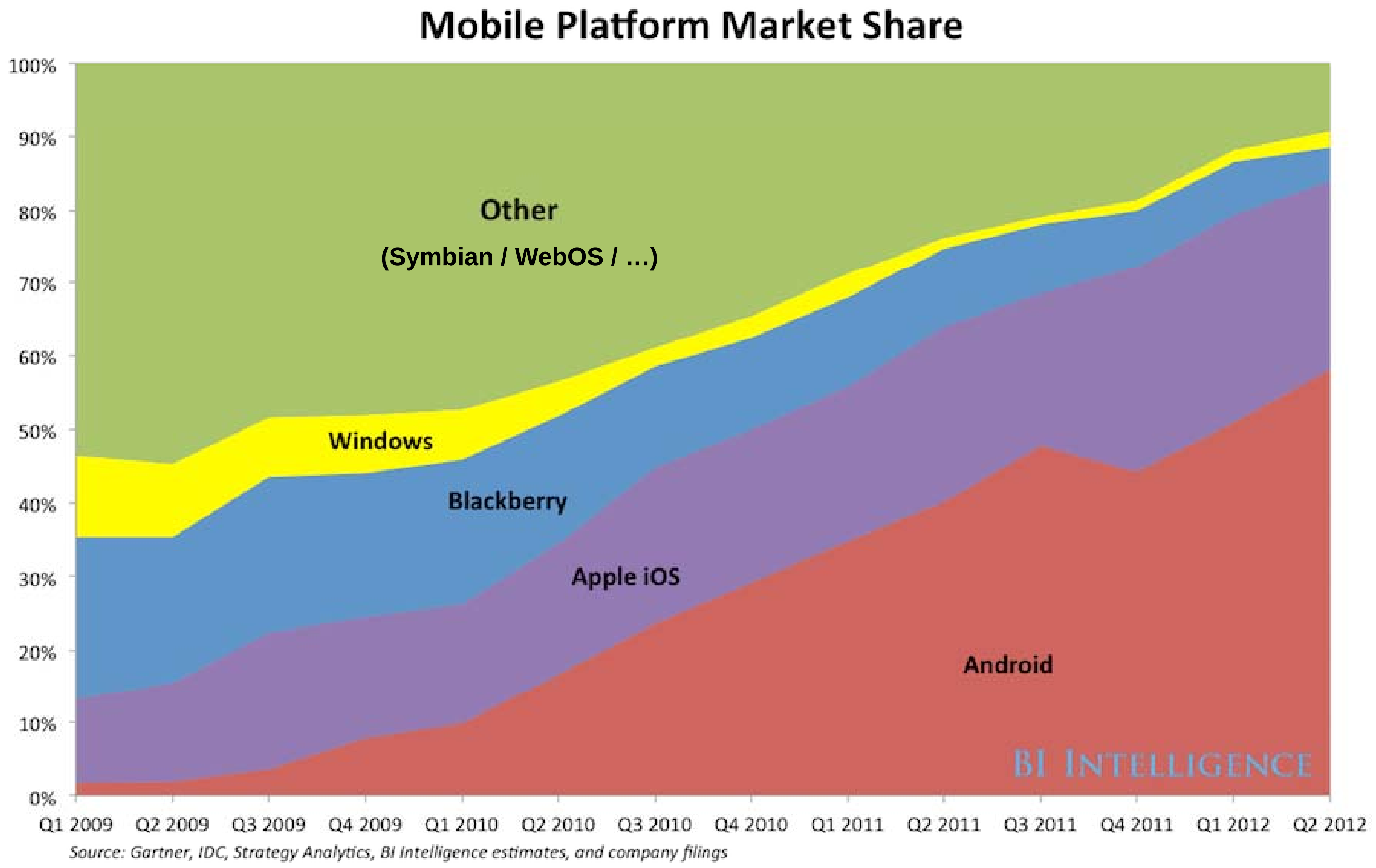
Types of Applications

- Foreground
  - Useful when being used
  - Suspended otherwise
- Background
  - Apart from when being configured, spends most of lifetime hidden (e.g., call screening app)
- Intermittent
  - Some interaction but mostly in the background (e.g., media player)
- Widget
  - Home screen status update

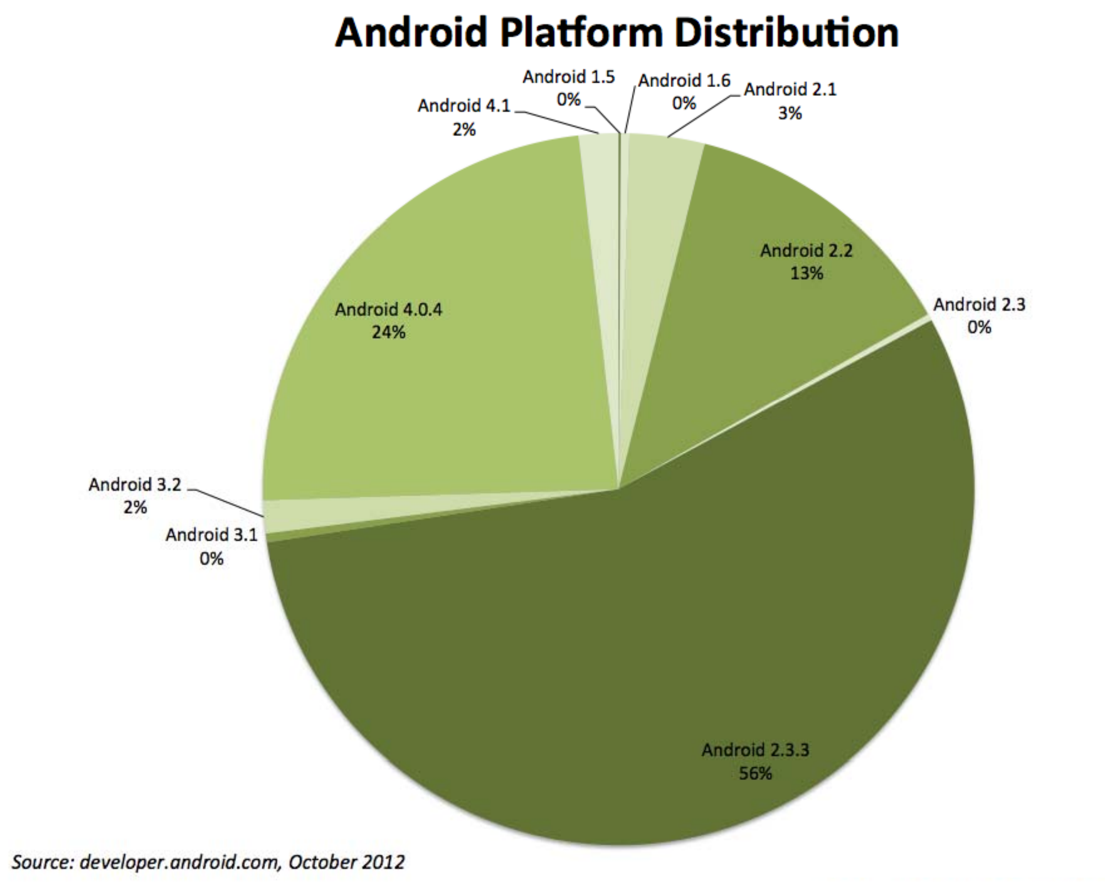
World Market Share - 2009



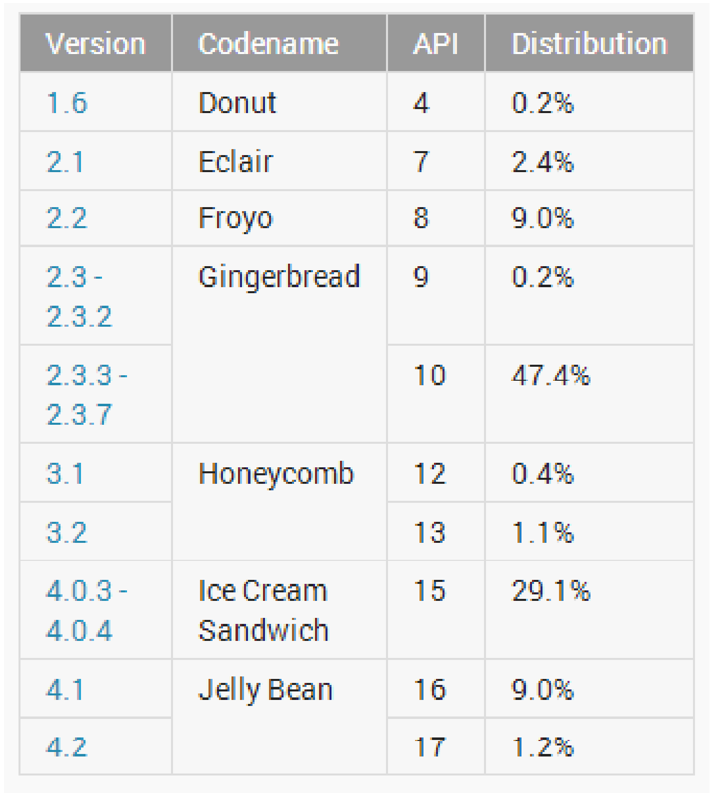
World Market Share – Progression 2009-2012



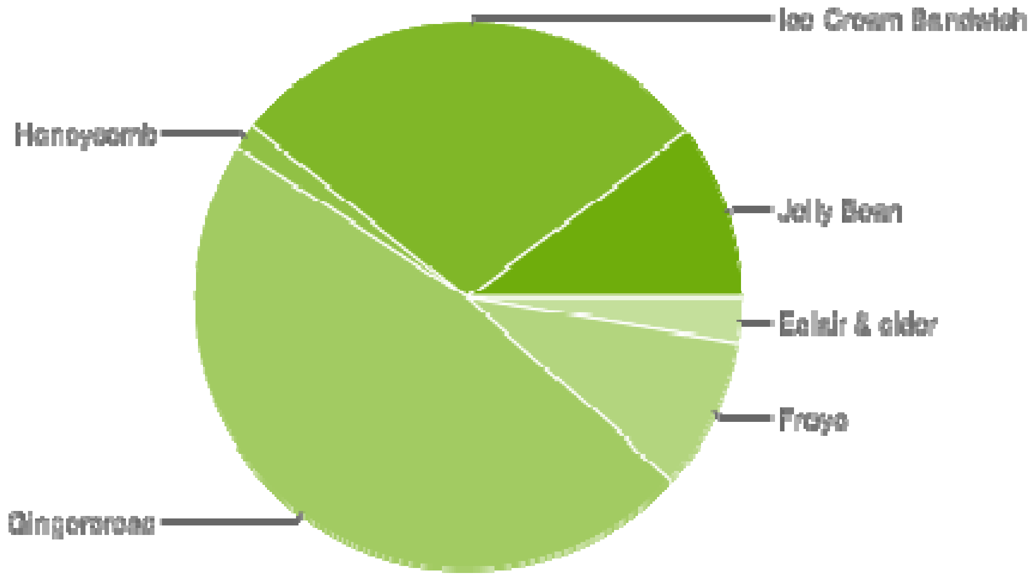
Android Versions 3Q2012



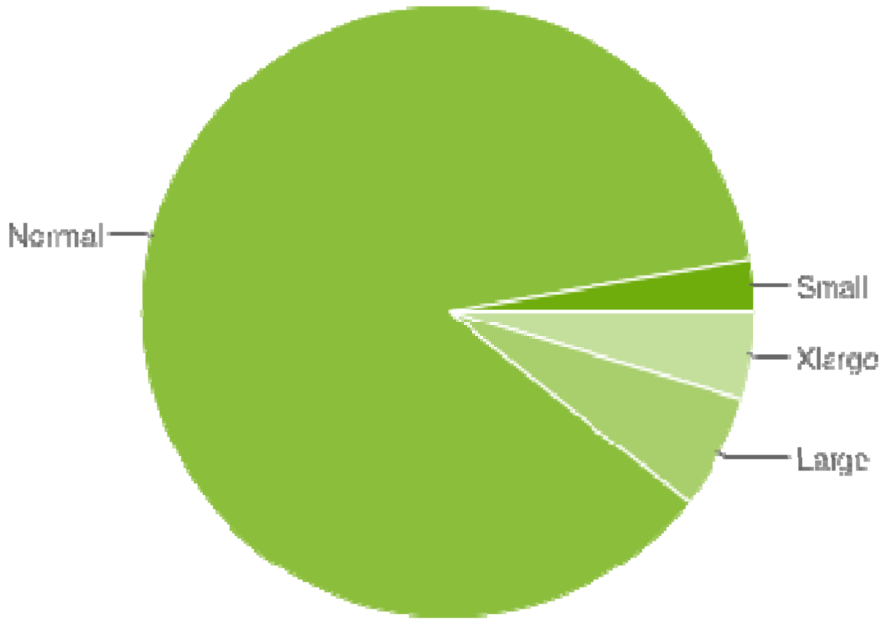
Android Versions 3/Jan/2013

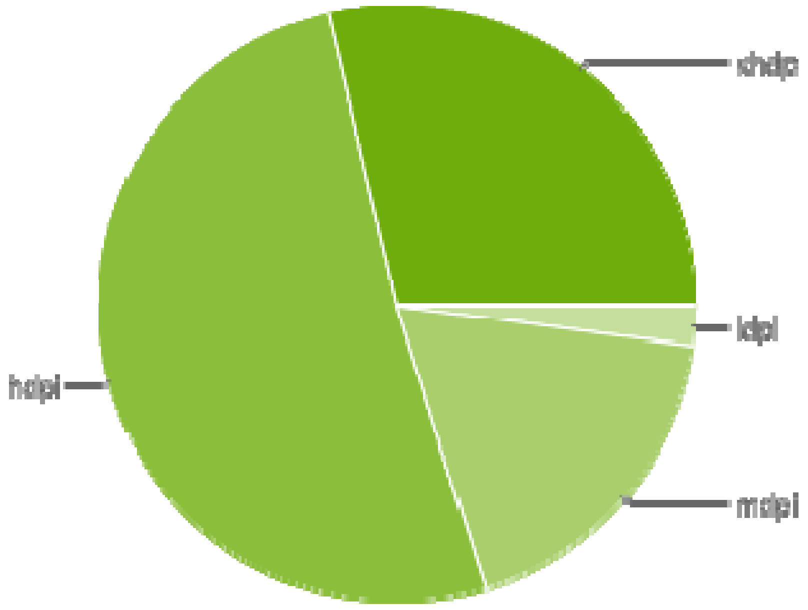


Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	47.4%
3.1	Honeycomb	12	0.4%
3.2		13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2		17	1.2%



Screen Size & Densities





Tools

Software to Download & Install

Java Development Kit (JDK)
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Eclipse IDE for Java
http://www.eclipse.org/downloads/

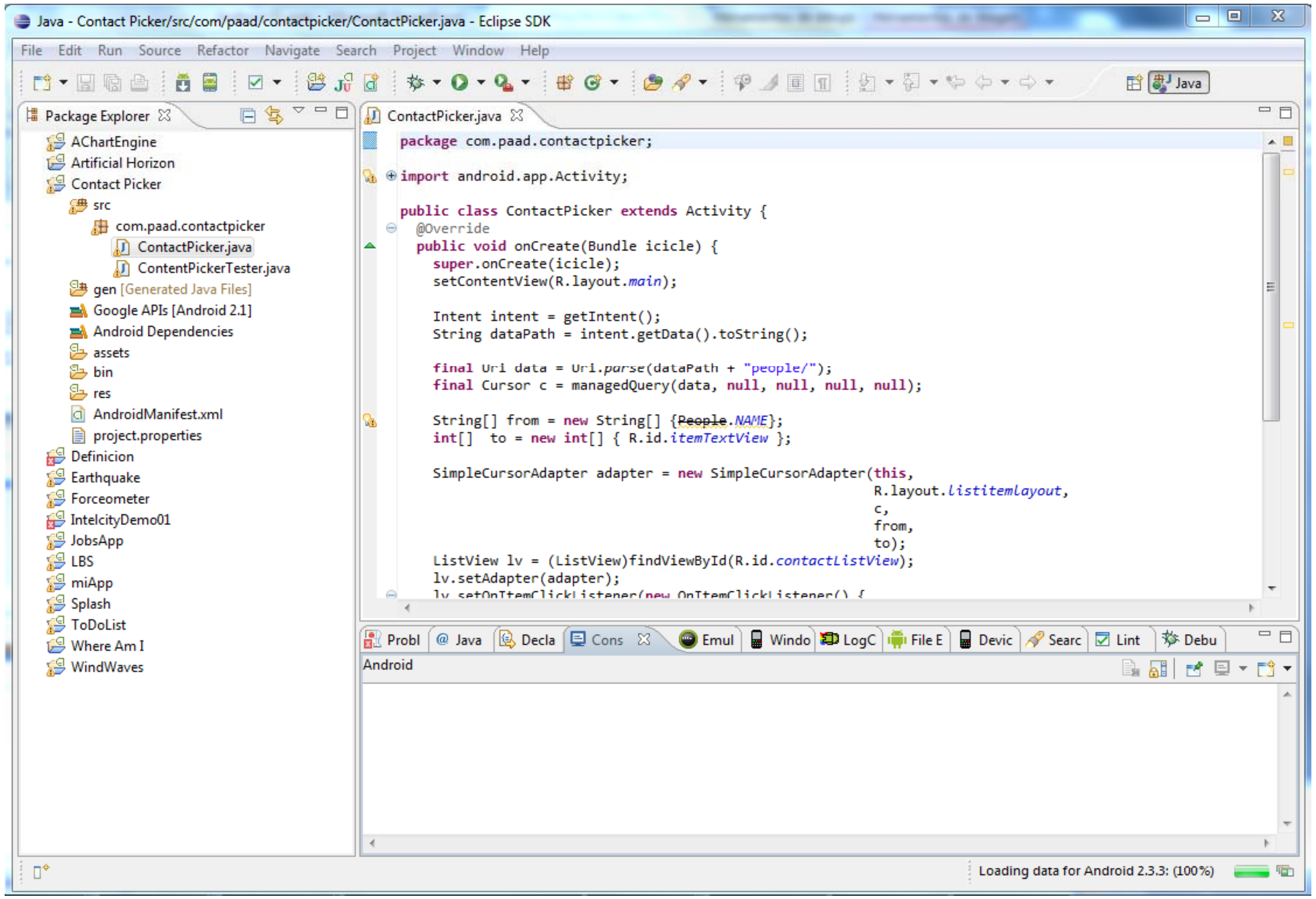
3. Eclipse Android Development Tools (ADT) plugin & Android Software Development Kit (SDK)
   http://developer.android.com/sdk/installing/installing-adt.html

4. Add Android platform & other comps to SDK
   http://developer.android.com/sdk/installing/adding-packages.html

5. ... Plus a Google account

Let’s Try…

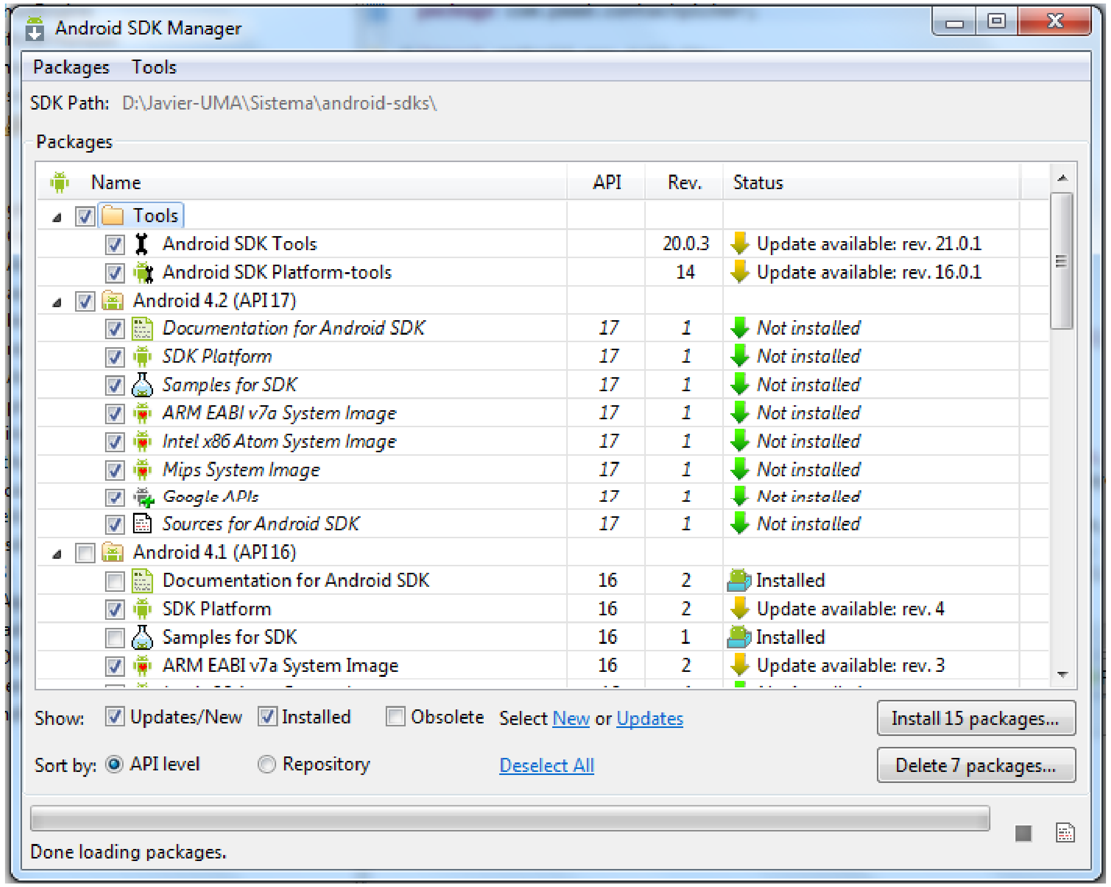
- Start Eclipse



```
package com.paad.contactpicker;

import android.app.Activity;

public class ContactPicker extends Activity {
  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    Intent intent = getIntent();
    String dataPath = intent.getData().toString();

    final Uri data = Uri.parse(dataPath + "people/");
    final Cursor c = managedQuery(data, null, null, null, null);

    String[] from = new String[] {People.NAME};
    int[]  to = new int[] { R.id.itemTextView };

    SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                                                          R.layout.listitemlayout,
                                                          c,
                                                          from,
                                                          to);
    ListView lv = (ListView)findViewById(R.id.contactListView);
    lv.setAdapter(adapter);
    lv.setOnItemClickListener(new OnItemClickListener() {
```

Let’s Try…

- Open Window -> Android SDK Manager
  - Check installed packages



SDK Path: D:\Javier-UMA\Sistema\android-sdks\

Packages

	Name	API	Rev.	Status
✓	Tools			
✓	Android SDK Tools		20.0.3	Update available: rev. 21.0.1
✓	Android SDK Platform-tools		14	Update available: rev. 16.0.1
✓	Android 4.2 (API 17)			
✓	*Documentation for Android SDK*	*17*	*1*	*Not installed*
✓	*SDK Platform*	*17*	*1*	*Not installed*
✓	*Samples for SDK*	*17*	*1*	*Not installed*
✓	*ARM EABI v7a System Image*	*17*	*1*	*Not installed*
✓	*Intel x86 Atom System Image*	*17*	*1*	*Not installed*
✓	*Mips System Image*	*17*	*1*	*Not installed*
✓	*Google APIs*	*17*	*1*	*Not installed*
✓	*Sources for Android SDK*	*17*	*1*	*Not installed*
	Android 4.1 (API 16)			
	Documentation for Android SDK	16	2	Installed
✓	SDK Platform	16	2	Update available: rev. 4
	Samples for SDK	16	1	Installed
✓	ARM EABI v7a System Image	16	2	Update available: rev. 3

Show: [✓ Updates/New] [✓ Installed] [ ] Obsolete Select New or Updates

Sort by: [● API level] ○ Repository Deselect All

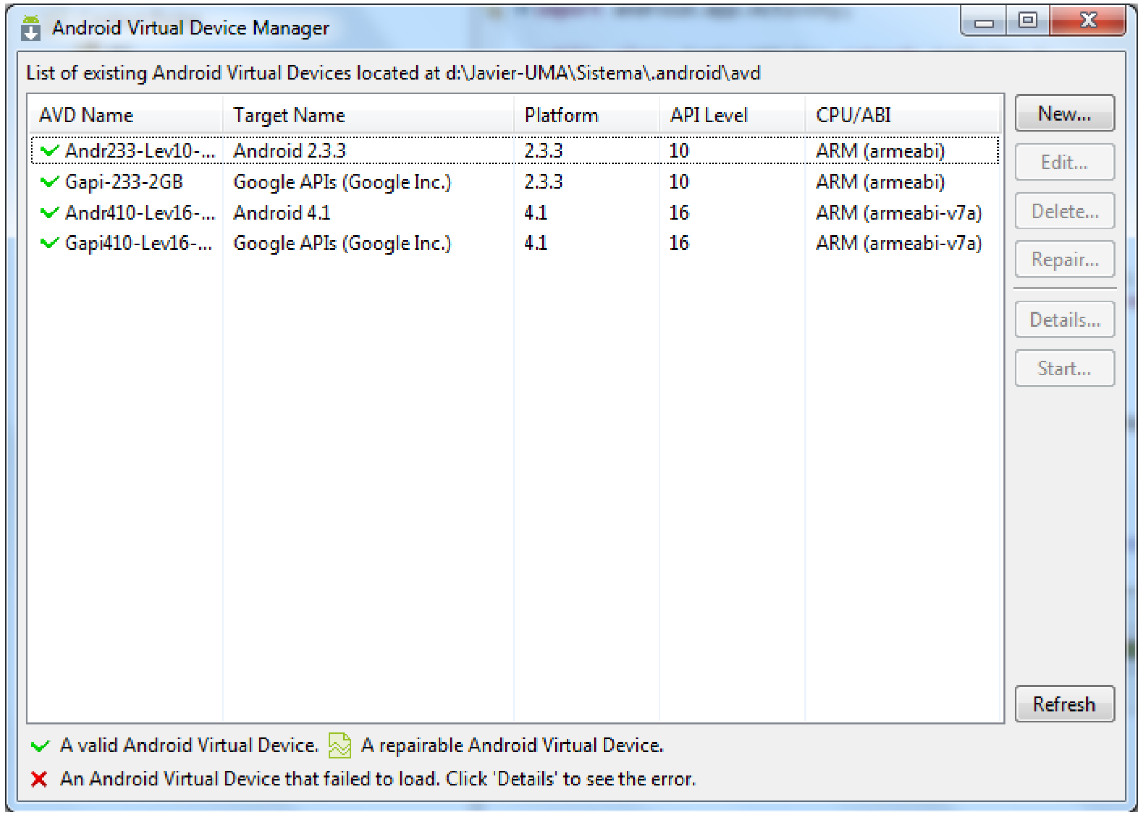
Install 15 packages...

Delete 7 packages...

Done loading packages.

Let’s Try…

- Open Window -> AVD Manager
  - Shows the list of Virtual Devices
  - Allows creating new Virtual Devices



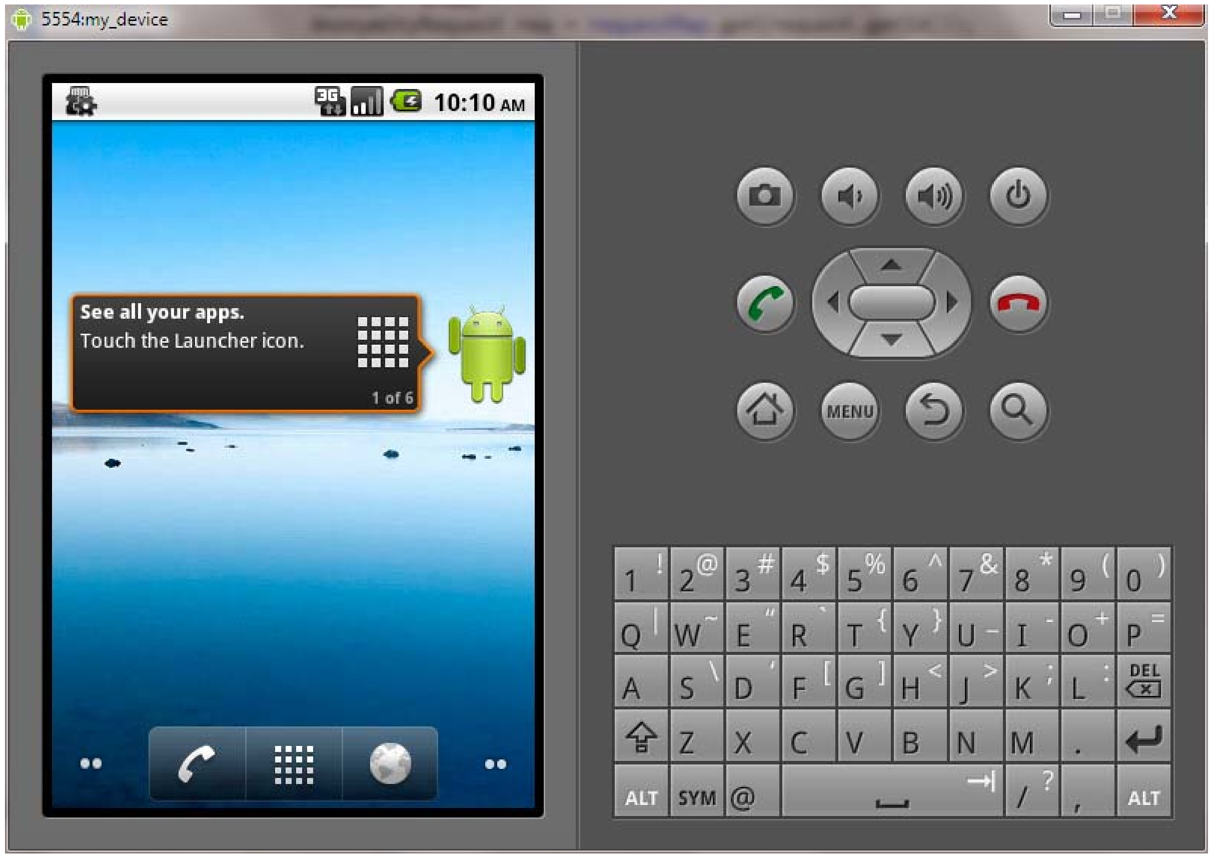
List of existing Android Virtual Devices located at d:\Javier-UMA\Sistema\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Andr233-Lev10-...	Android 2.3.3	2.3.3	10	ARM (armeabi)
✓ Gapi-233-2GB	Google APIs (Google Inc.)	2.3.3	10	ARM (armeabi)
✓ Andr410-Lev16-...	Android 4.1	4.1	16	ARM (armeabi-v7a)
✓ Gapi410-Lev16-...	Google APIs (Google Inc.)	4.1	16	ARM (armeabi-v7a)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Let's Try…

- Select one AVD and Start it!
  - First start-up may take a long time

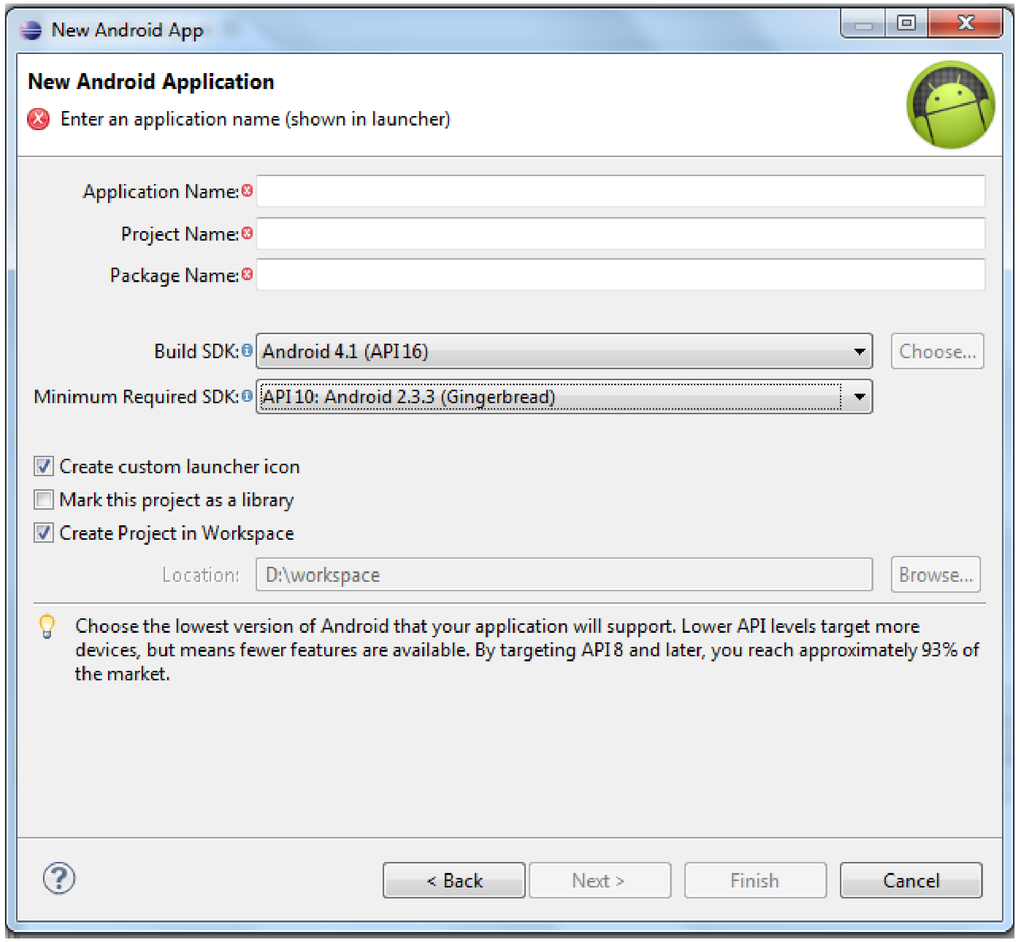


First App: ‘Hello, World’

Create a new Android Project

- Select File->New->Other… -> Android -> Android Application Project
  - Application name
  - Project name
  - Package name

    iobm.tutorial.xxxxx
  - Build SDK
  - Minimum SDK
  - Create Activity



New Android App

New Android Application

Enter an application name (shown in launcher)

Application Name:

Project Name:

Package Name:

Build SDK: Android 4.1 (API 16)

Minimum Required SDK: API 10: Android 2.3.3 (Gingerbread)

☑ Create custom launcher icon

☐ Mark this project as a library

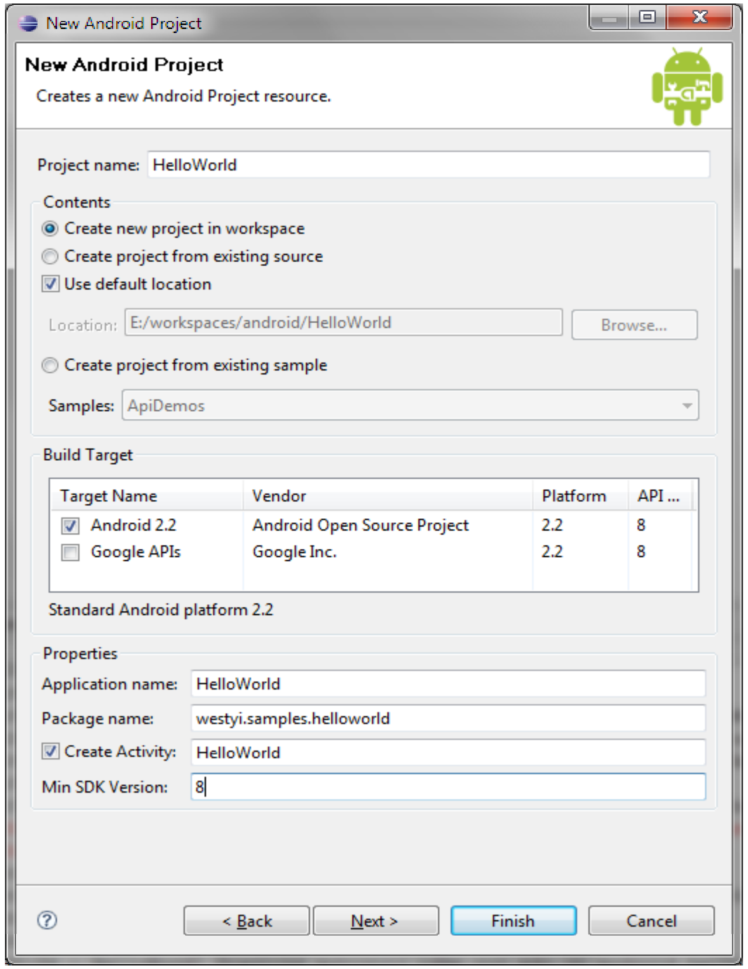
☑ Create Project in Workspace

Location: D:\workspace

Choose the lowest version of Android that your application will support. Lower API levels target more devices, but means fewer features are available. By targeting API 8 and later, you reach approximately 93% of the market.

Create a new Android Project

- Select File->New->Other... -> Android -> Android Application Project
  - Application name
  - Project name
  - Package name
    iobm.tutorial.xxxxx
  - Build SDK
  - Minimum SDK
  - Create Activity


New Android Project

New Android Project

Creates a new Android Project resource.

Project name: HelloWorld

Contents
- (●) Create new project in workspace
- ( ) Create project from existing source
- [✓] Use default location

Location: E:/workspaces/android/HelloWorld Browse...

- ( ) Create project from existing sample

Samples: ApiDemos

Build Target

Target Name	Vendor	Platform	API ...
✓ Android 2.2	Android Open Source Project	2.2	8
☐ Google APIs	Google Inc.	2.2	8

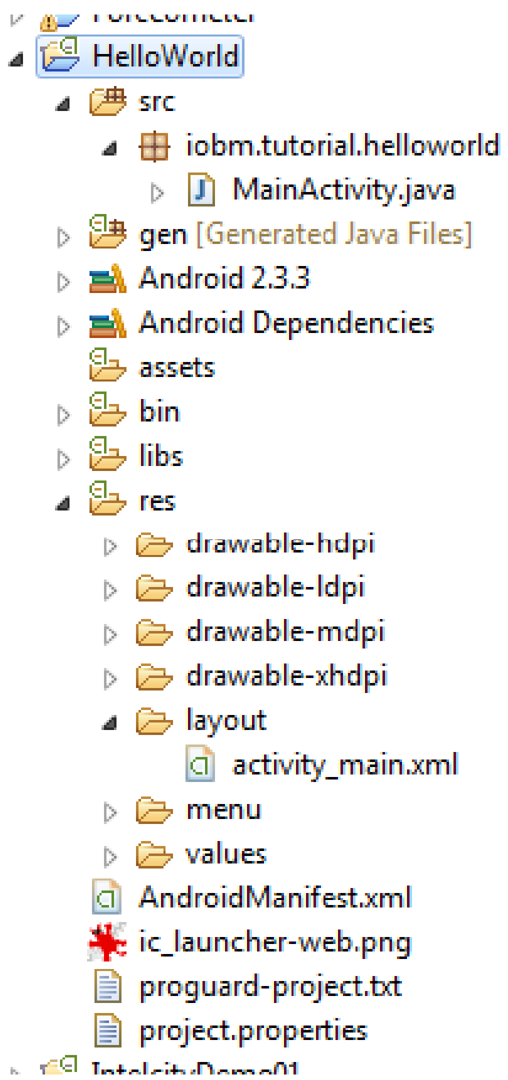
Standard Android platform 2.2

Properties

Application name:	HelloWorld
Package name:	westyi.samples.helloworld
✓ Create Activity:	HelloWorld
Min SDK Version:	8

Hello World Project

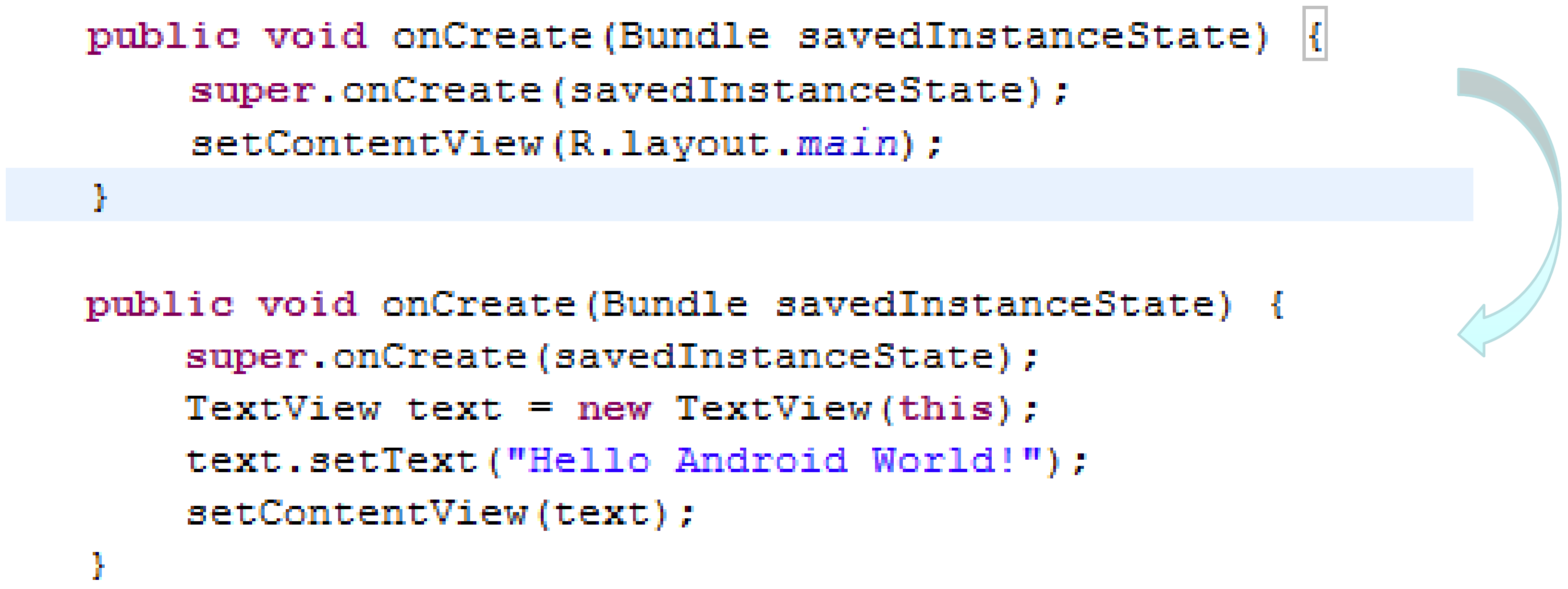
- src: source folder
- gen: SDK generated file
- android 2.3.3: reference lib
- assets: binary resources
- res: resource files and resource description files
- AndroidManifest.xml: application description file
- default.properties: project properties file
- Apk: zip-format



```
HelloWorld
  src
    iobm.tutorial.helloworld
      MainActivity.java
  gen [Generated Java Files]
  Android 2.3.3
  Android Dependencies
  assets
  bin
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    layout
      activity_main.xml
    menu
    values
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

Say Hello World

- Modify HelloWorld.java

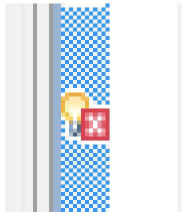


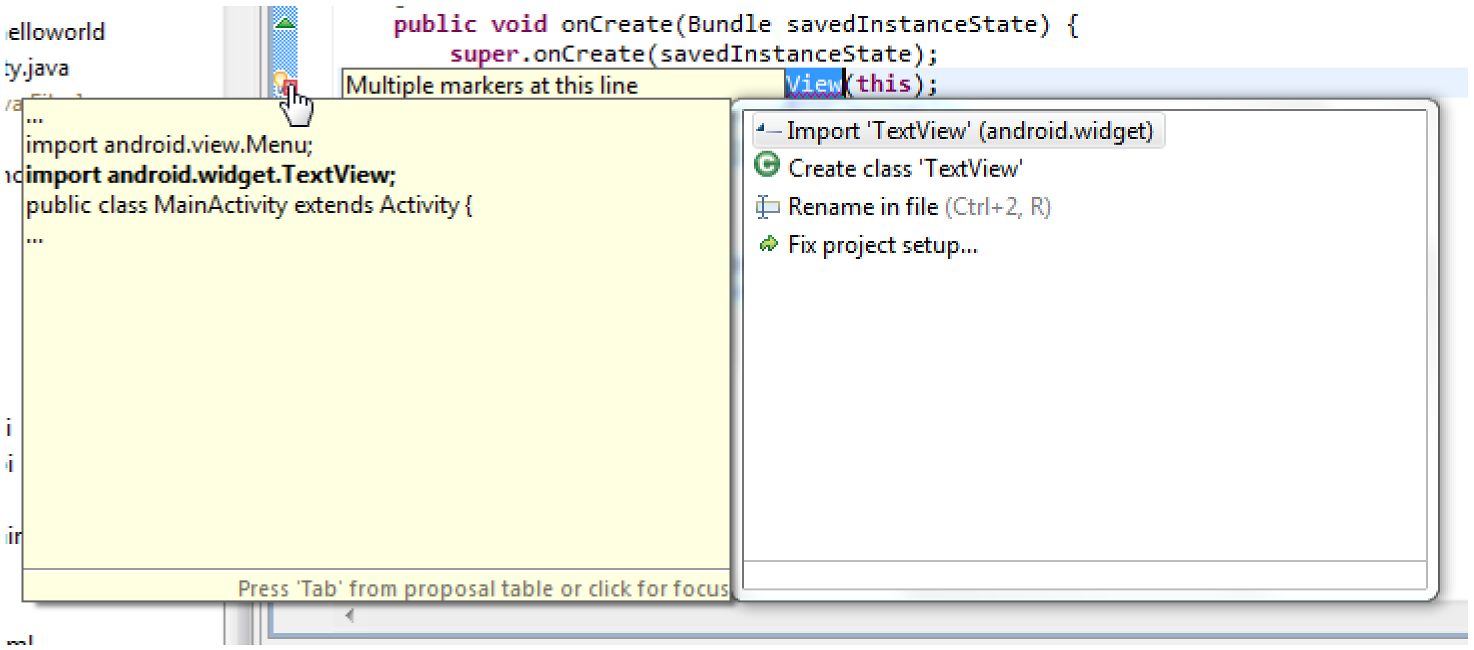
```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```

```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView text = new TextView(this);
    text.setText("Hello Android World!");
    setContentView(text);
}
```

Import Widget Library

- Click left button on error indication
- Import 'TextView' widget





```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
                                View(this);
```

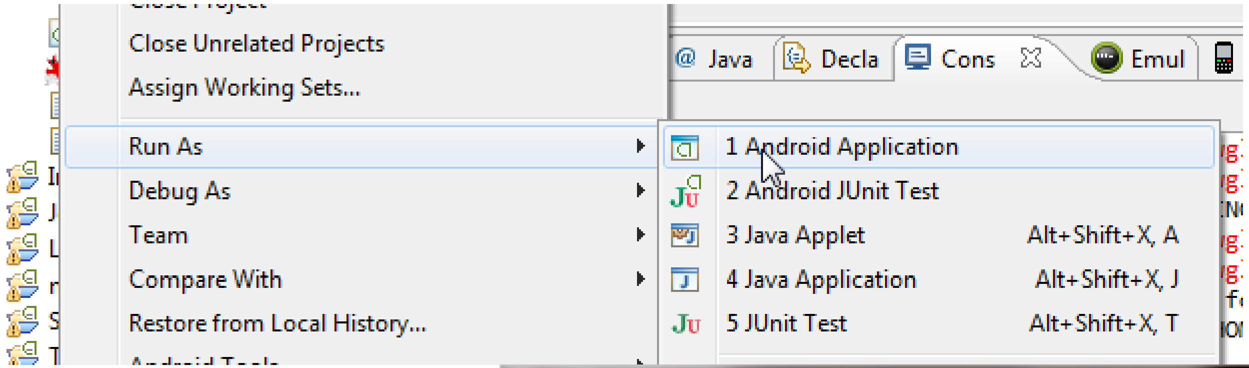
Multiple markers at this line

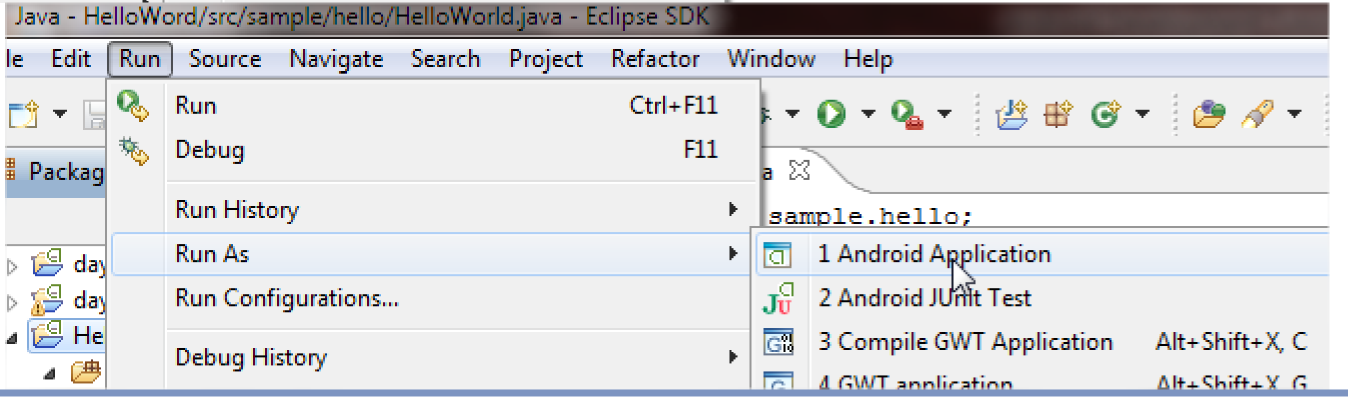
```
...
import android.view.Menu;
import android.widget.TextView;
public class MainActivity extends Activity {
...
```

elloworld
y.java
Press 'Tab' from proposal table or click for focus
Import 'TextView' (android.widget)
Create class 'TextView'
Rename in file (Ctrl+2, R)
Fix project setup...

Run Hello World

- Select *HelloWorld* Project
  a) Right-click, Run as -> Android Application
  b) Run->Run as->Android Application
- ADT will start a proper AVD and run application on it




Java - HelloWord/src/sample/hello/HelloWorld.java - Eclipse SDK
le Edit Run Source Navigate Search Project Refactor Window Help
Run Ctrl+F11
Debug F11
Run History
Run As
Run Configurations...
Debug History
Packag
sample.hello;
1 Android Application
2 Android JUnit Test
3 Compile GWT Application Alt+Shift+X, C
4 GWT application Alt+Shift+X, G

Behind HelloWorld

- R.java, generated by Android SDK, represents all the resources of the app
  - Resources are all in *res* folder
  - Resources are pre-compiled into binary format

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */
package sample.hello;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
```

Behind HelloWorld

- res/layout, contains layout declarations of the app in XML format
  - UIs are built according to the layout file

main.xml

```xml
<?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="@string/hello" />
</LinearLayout>
```

Behind HelloWorld

- res/values, contains string declarations or other values(e.g.:colors) of the app
  - string.xml, contains string resources

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloWorld!</string>
    <string name="app_name">HelloWorld</string>
</resources>
```

referenced in res/layout/main.xml

referenced in AndroidManifest.xml

Behind HelloWorld

- res/drawable, contains all image resources
  - Folders may have suffixes, app will choose the most suitable one, so do the other resources
  - Three folders: drawable-ldpi, drawable-hdpi, drawable-mdpi, each contains an icon.png file
  - App will choose the proper icon according to the device DPI
  - Reference name:@drawable/icon
- Other folders we may use in future
  - menu, anim (animation), xml (preference and searchable)

Behind HelloWorld

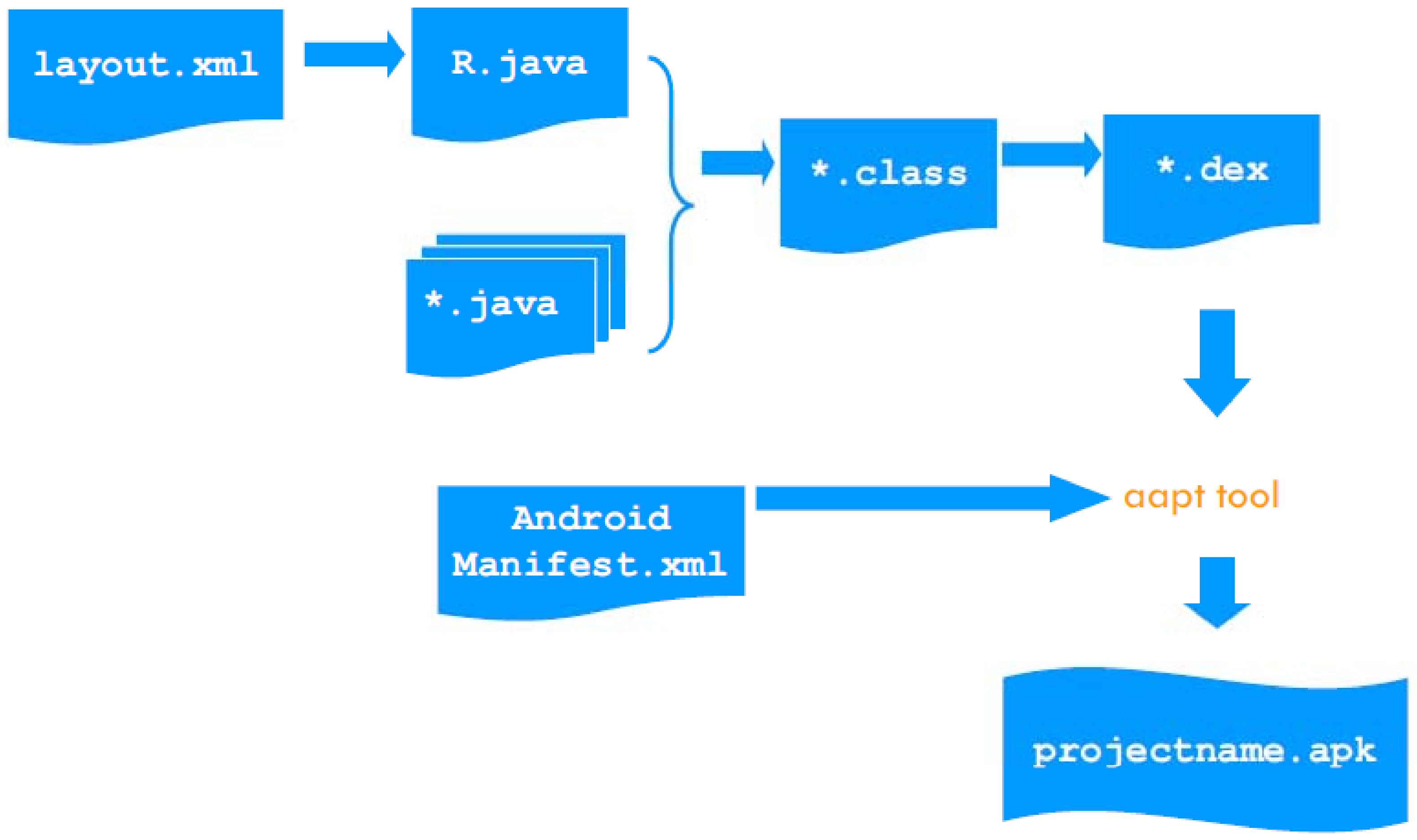
- AndroidManifest.xml describes the application
  - Declare app’s name, version, icon, permission, etc…
  - Declare the application's components: activity, service, receiver or provider

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="sample.hello" android:versionCode="1" android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".HelloWorld" 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>
<uses-sdk android:minSdkVersion="8" />
</manifest>
```

Behind HelloWorld

- Basically, an *Activity* presents a **visual user interface** for one focused endeavor the user can undertake
- An application might consist of just one activity or several, each Activity is derived from *android.app.Activity* and should be declared in AndroidManifest.xml file
- Each activity is given a default window to draw in, the window may be full screen or smaller and on top of other window
- The visual content of the window is provided by a hierarchy of views — objects derived from the base View class

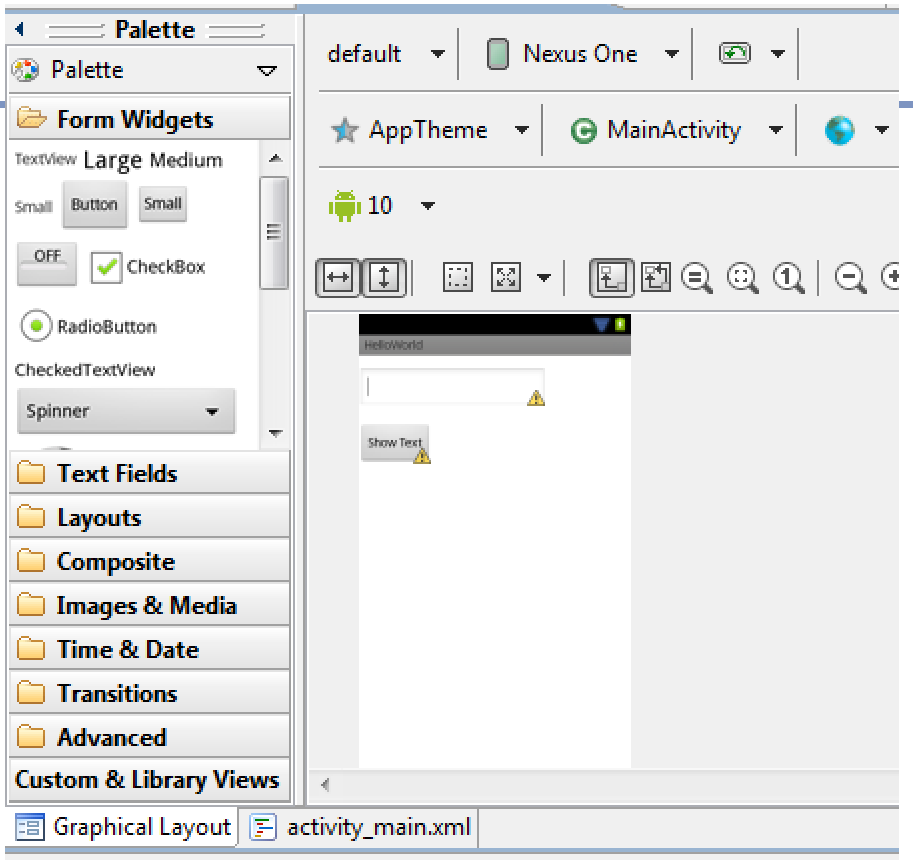
APK Generation Process



.apk files have the .zip format

Beyond HelloWorld

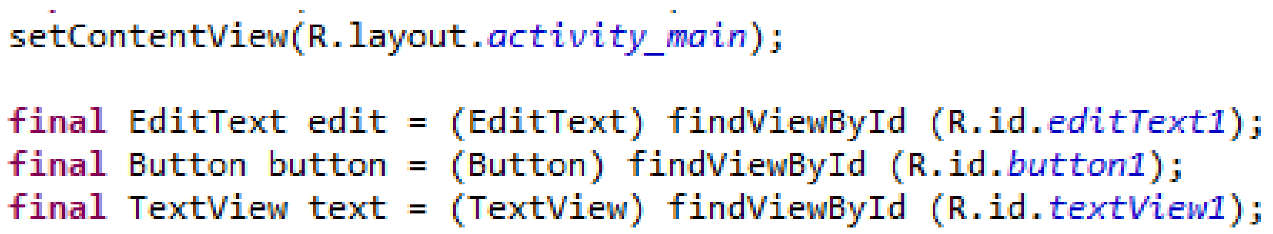
- Build up an app that you can input your greetings and display your greetings
  - Input: EditText
  - Display: TextView
  - Of course, we have to add a button
- Edit res/layout/main.xml file to add these components
  - Each has an android:id property, used to reference it in code



```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="14dp"
    android:ems="10" >
    <requestFocus />
</EditText>
```

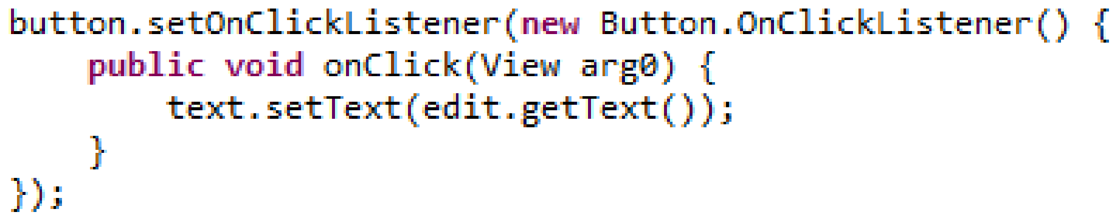
Beyond HelloWorld

- Modify HelloWorld.java
  - Firstly get the references declared in main.xml



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

final EditText edit = (EditText) findViewById (R.id.editText1);
final Button button = (Button) findViewById (R.id.button1);
final TextView text = (TextView) findViewById (R.id.textView1);
```

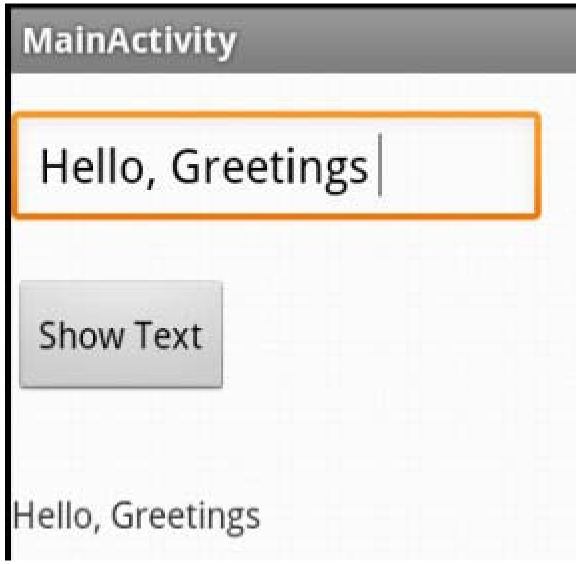
  - Then add event response for Button



```
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View arg0) {
        text.setText(edit.getText());
    }
});
```

Beyond HelloWorld

- Finished!
- Run->Run as->Android Application



Android Development

Session 2

Javier Poncela

Contents

Applications

- By default, each application:
  - Assigned a unique Linux user ID
  - Executes in its own Linux process
- By default, each process runs its own virtual machine

- Android manages process creation & shutdown
  - Starts process when any of the application's code needs to be executed
  - Shuts down when process is no longer needed and/or system resources are required by other applications

- Apps can have multiple entry points
  - i.e., not just main() method
- App comprise components that the system can instantiate and run as needed

Application Components

- Activity
  - Present a visual user interface for one focused endeavor the user can undertake
    - Example: a list of menu items users can choose from
- Services
  - Run in the background for an indefinite period of time, no visual interface
    - Example: calculate the result to activities that need it
- Broadcast Receivers
  - Receive and react to broadcast announcements
    - Example: announcements that the time zone has changed
- Content Providers
  - Store and retrieve data and make it accessible to all applications
    - Example: E-mail contacts, Music list

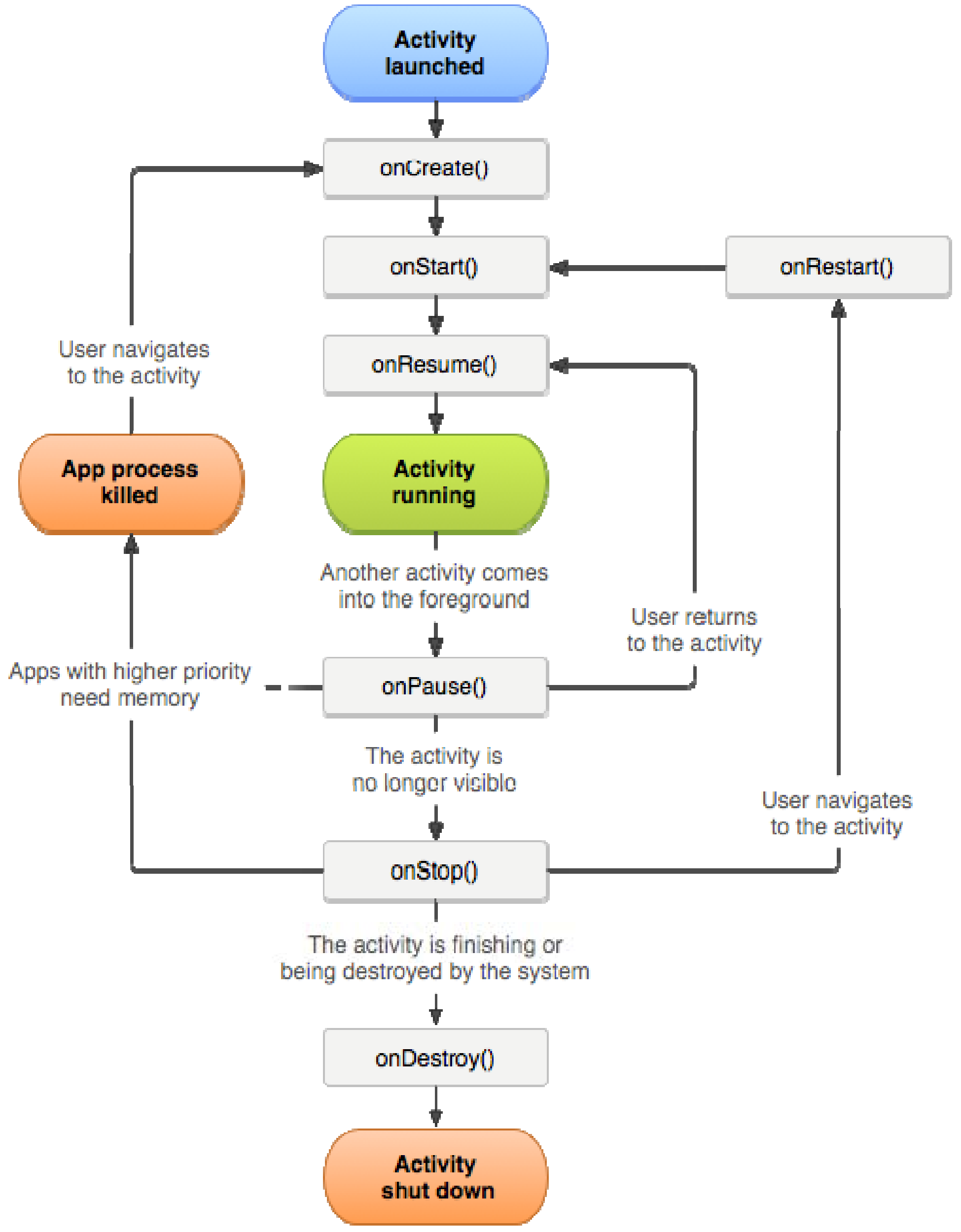
Activities

Activities

- Primary class for interacting with user
  - Usually implements a focused task
  - Usually Involves one screen full of data
- A single Activity defines a single viewable screen
  - Defines the actions, not the layout
  - Example: Calculator
- There may be multiple Activities (“Screens”) per application
  - Each is a separate entity
- Activities have a structured life cycle
  - Different events in their life happen either via the user touching buttons or programmatically

Activity Lifecycle

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()
- onRestart()

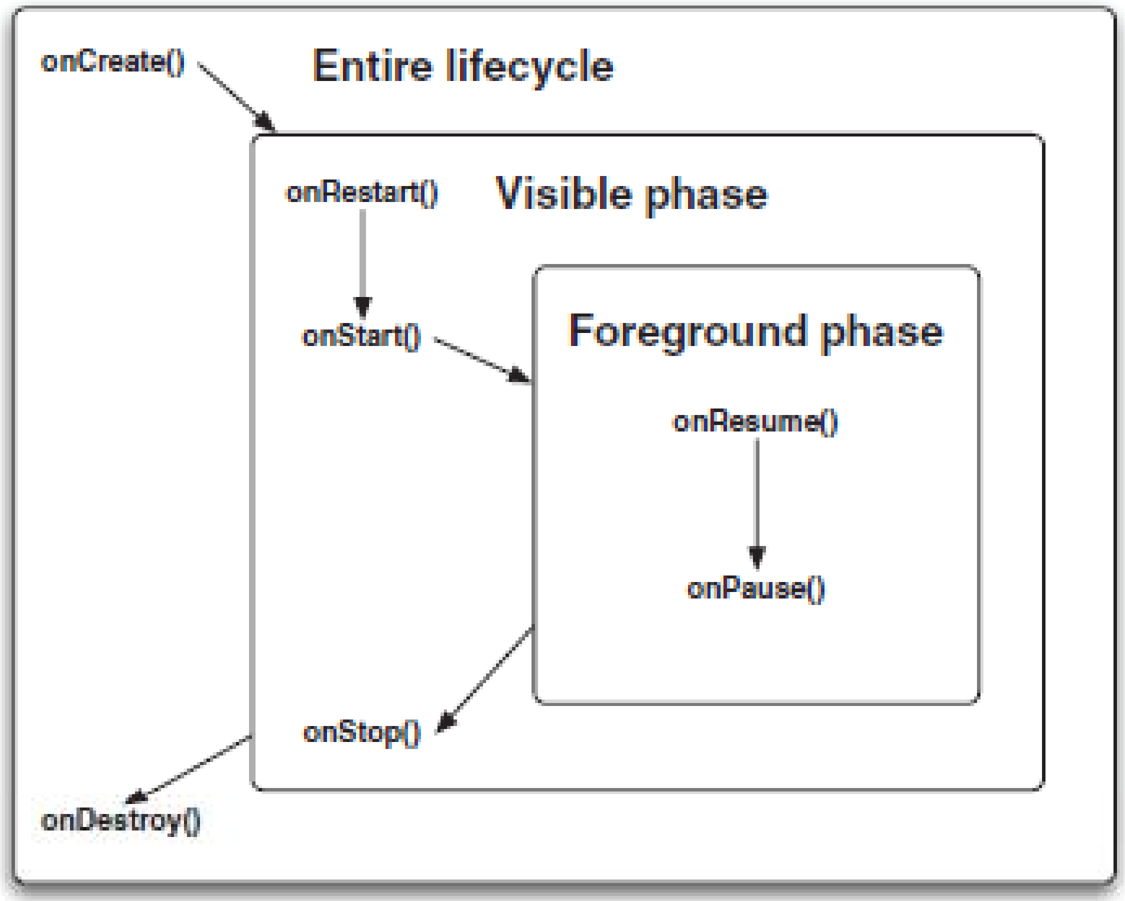


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity Lifecycle

- **onCreate()** — Called when the activity is first created
- **onStart()** — Called when the activity becomes visible to the user
- **onResume()** — Called when the activity starts interacting with the user
- **onPause()** — Called when the current activity is being paused and the previous activity is being resumed
- **onStop()** — Called when the activity is no longer visible to the user
- **onDestroy()** — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- **onRestart()** — Called when the activity has been stopped and is restarting again

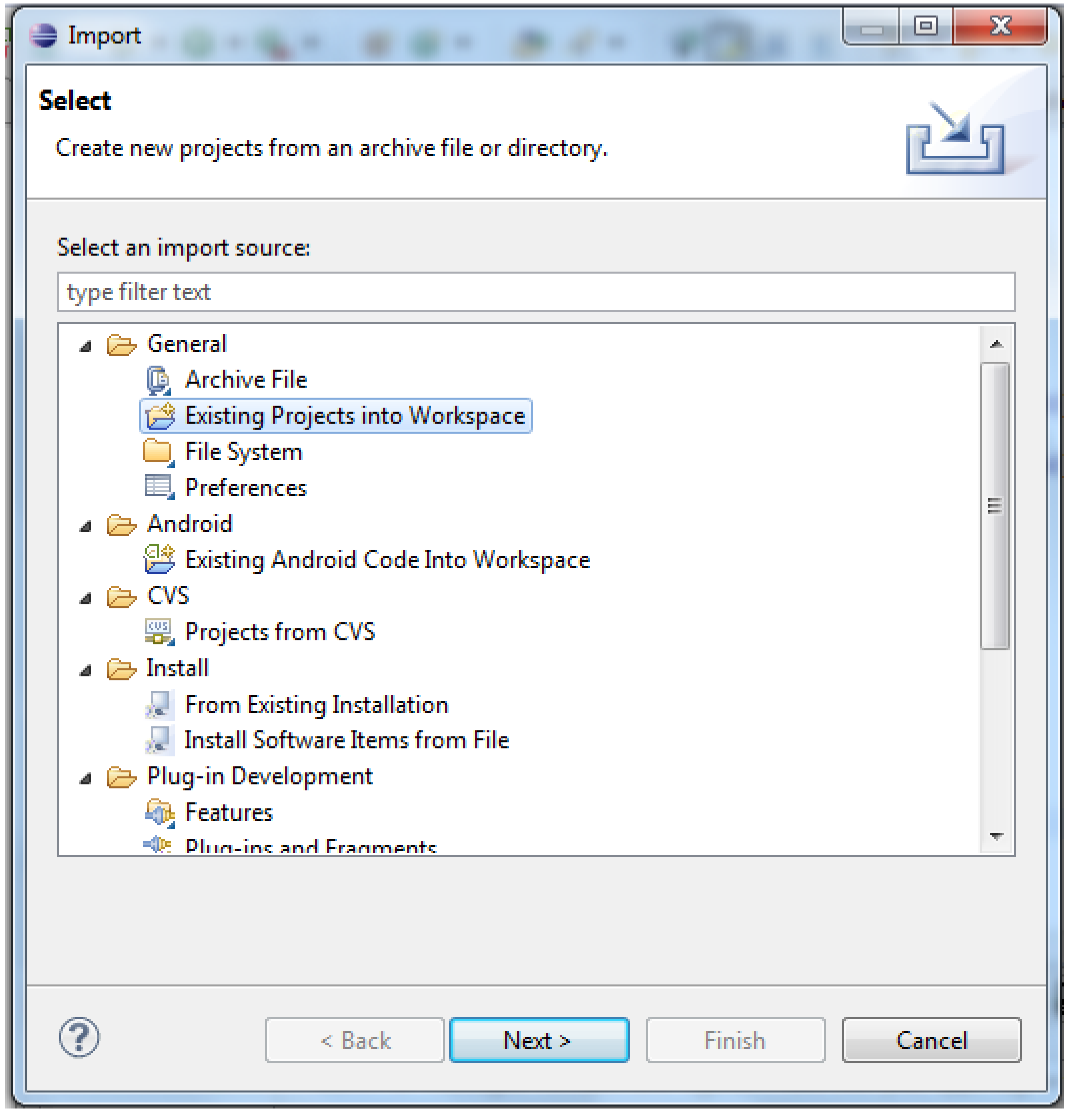
Activity Lifecycle



Let’s Try… Project Activity101

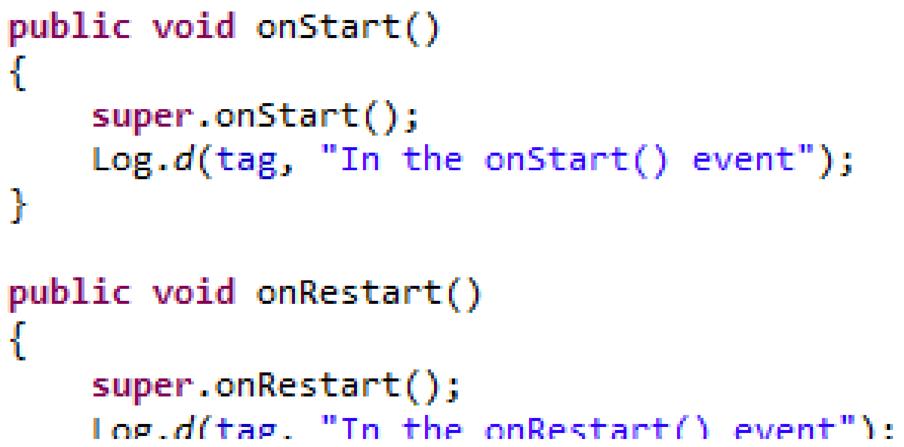
- Import project
  File -> Import

- Have a look at the code

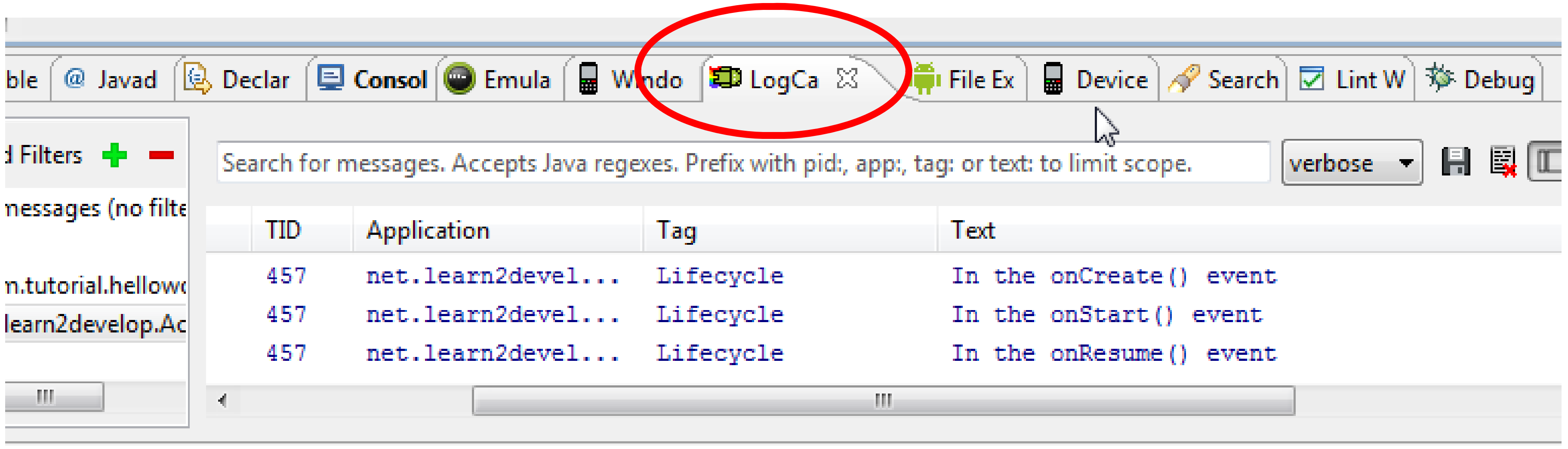


Let’s Try… Project Activity101

- State changes are logged



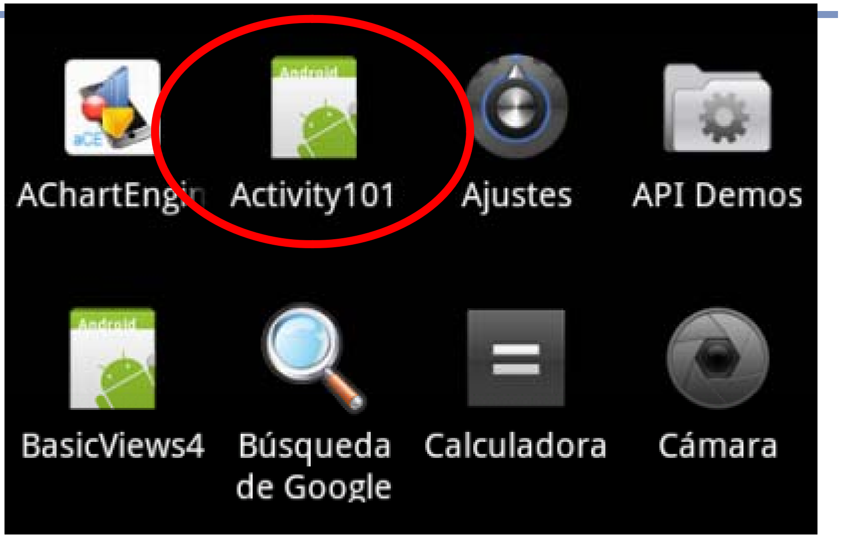
```
public void onStart()
{
    super.onStart();
    Log.d(tag, "In the onStart() event");
}

public void onRestart()
{
    super.onRestart();
    Log.d(tag, "In the onRestart() event");
```

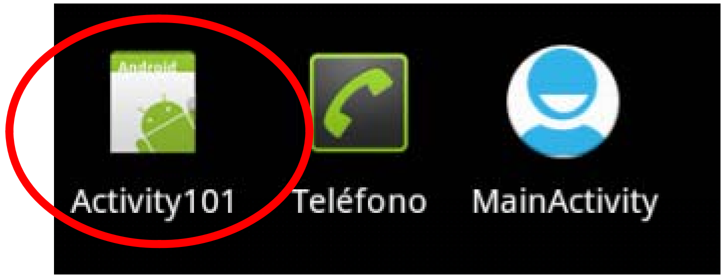


```
TID   Application          Tag         Text
457   net.learn2devel...   Lifecycle   In the onCreate() event
457   net.learn2devel...   Lifecycle   In the onStart() event
457   net.learn2devel...   Lifecycle   In the onResume() event
```

Let's Try… Project Activity101

- Do the following (observe log after each step):
- A)
  - Start the application
  - Press Back button
- B)
  - Start the application
  - Press Home button
    - Notice onDestroy is not called
  - Start the Phone application
  - Keep Home button pressed
    - List of recent apps appears
  - Choose Activity application
  - Press Back button





Lifecycle - Some Comments

- As you can see, an activity is destroyed when you press the Back button
  - This is crucial, as whatever state the activity is currently in will be lost

    ➔ Hence, you need to write additional code in your activity to preserve its state when it is destroyed

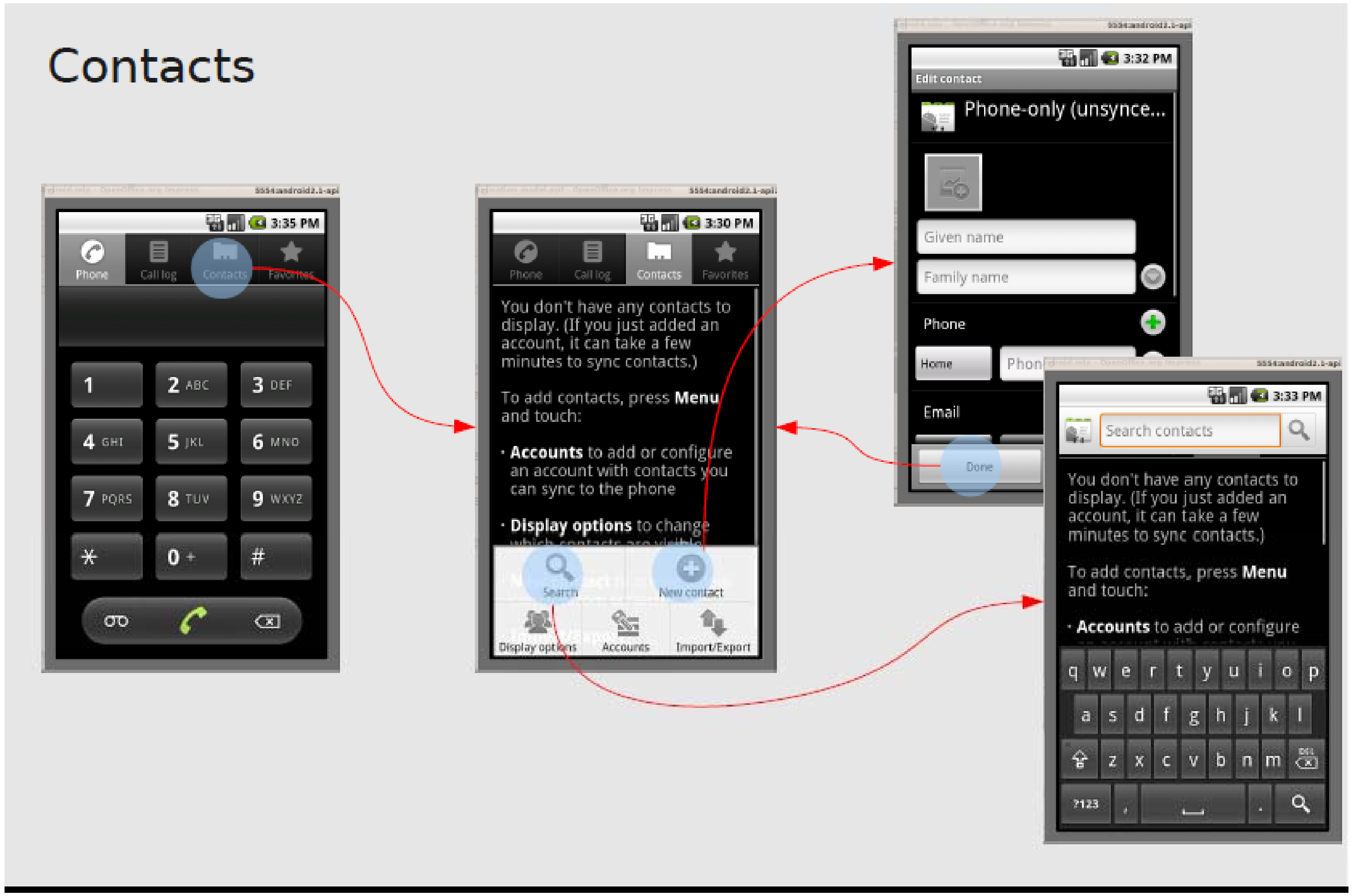
- Note that the onPause() event is called both when an activity is sent to the background, as well as when it is killed when the user presses the Back button

- What happens if device configuration changes?
  - Orientation, locale, etc

  On configuration changes, Android usually kills & restarts the current Activity

Activity Stack

- Applications usually consist of several activities (screens)



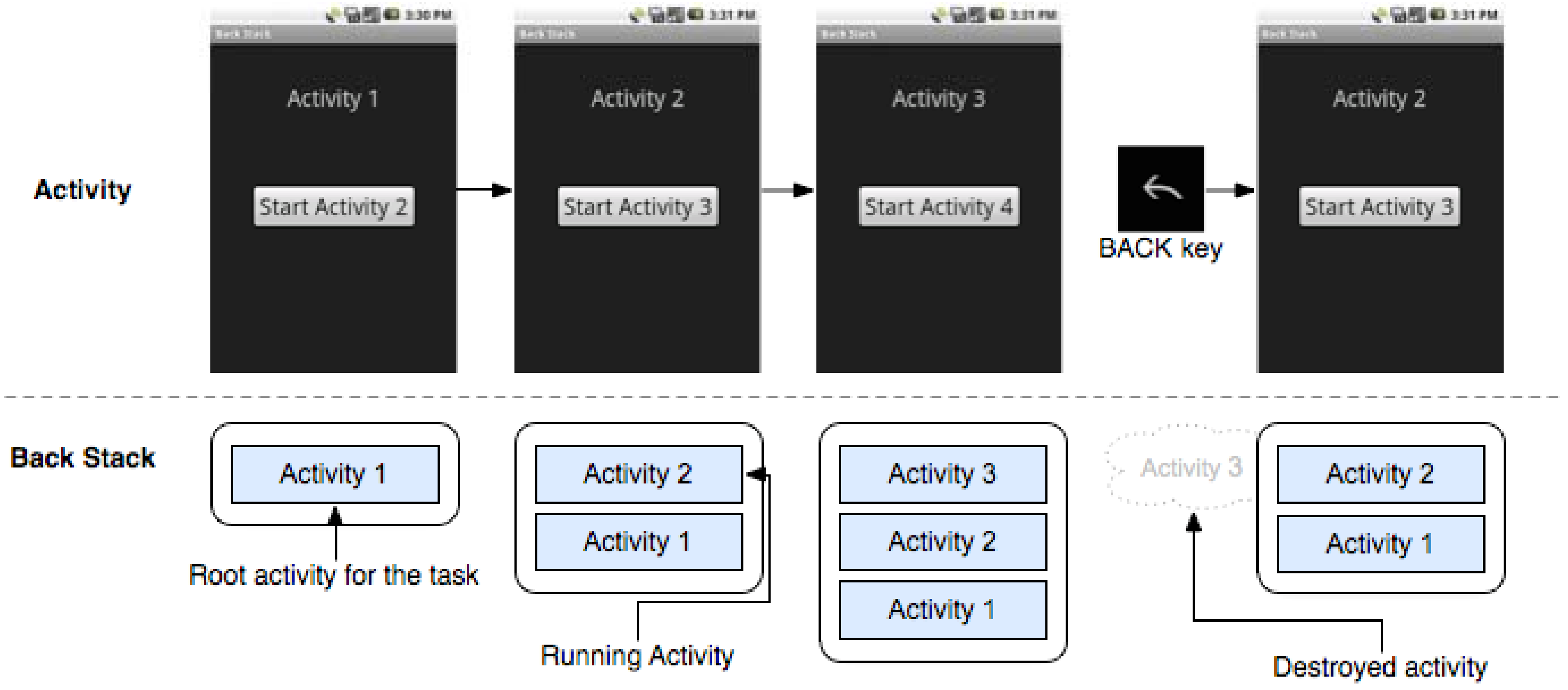
You don't have any contacts to display. (If you just added an account, it can take a few minutes to sync contacts.)

To add contacts, press **Menu** and touch:

- **Accounts** to add or configure an account with contacts you can sync to the phone
- **Display options** to change which contacts are visible

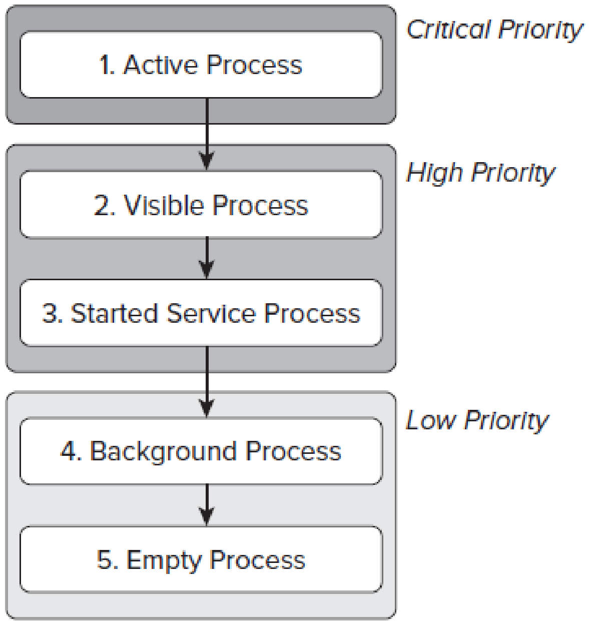
Activity Stack

- Android manages a system Activity stack



Activity Stack Management

- States
  - Not started – not yet created
  - Active
    - Resumed/Running - visible, has focus
    - Paused - visible, does not have focus, can be terminated
    - Stopped - not visible, does not have focus, can be terminated
  - Finished – done
- The OS ranks processes and **kills those with lowest priority**
  - We have no control on when an Activity is destroyed…
    - User action
    - OS decision (low memory)
  - … But changes are notified



Typical Actions on State Change

- onCreate()
  - Activity is being created
  - Setup global state
    - Call super.onCreate()
    - Inflate UI views
    - Configure views as necessary
    - Set the activity’s content view

- onPause()
  - Focus about to switch to another Activity
    - Save persistent state
    - Shutdown unneeded behavior

Saving Information during Changes in Configuration

- Changing screen orientation destroys an activity and re-creates it
  - Desirable to keep state, including already made user inputs (i.e. text written)

- Preserve state of an activity
  - onPause(): use your own mechanisms
  - onSaveInstanceState(): simply preserve the state so that it can be restored later
    - Simpler
    - Provides a Bundle object as an argument
    - Fired whenever an activity is about to be killed or put into the background. However, it is not fired when an activity is being unloaded from the stack

Saving Information during Changes in Configuration

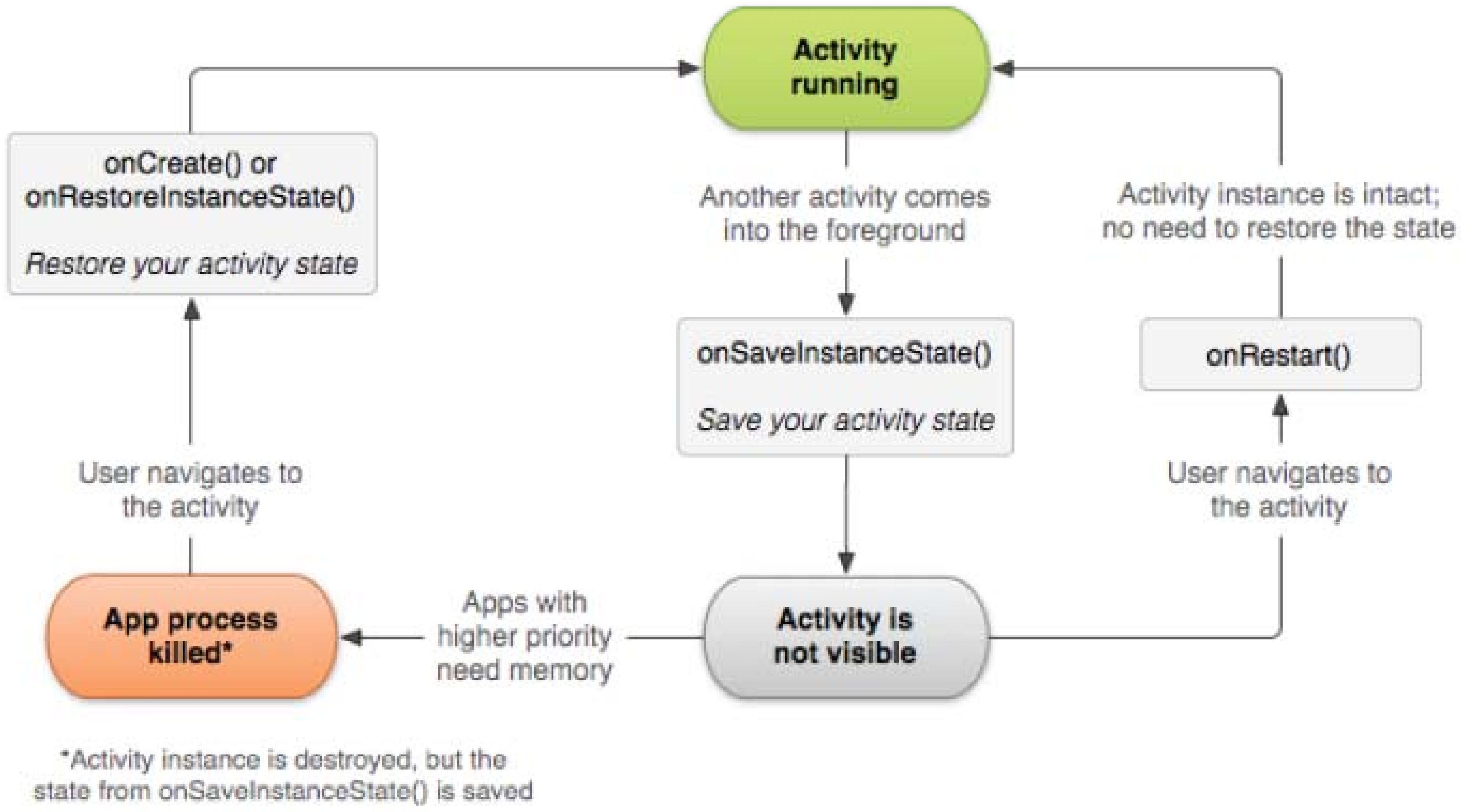
- Saving state

```
@Override
public void onSaveInstanceState(Bundle outState) {
    //---save whatever you need to persist---
    outState.putString("ID", "1234567890");
    super.onSaveInstanceState(outState);
}
```

- Restoring state: When an activity is re-created, onCreate() is first fired, followed by the onRestoreInstanceState() event, which enables you to retrieve the state through the Bundle

```
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    //---retrieve the information persisted earlier---
    String ID = savedInstanceState.getString("ID");
}
```

Saving Information during Changes in Configuration


Activity running
onCreate() or onRestoreInstanceState()
Restore your activity state
Another activity comes into the foreground
Activity instance is intact; no need to restore the state
onSaveInstanceState()
Save your activity state
onRestart()
User navigates to the activity
User navigates to the activity
App process killed*
Apps with higher priority need memory
Activity is not visible
*Activity instance is destroyed, but the state from onSaveInstanceState() is saved

Finishing an Activity

- Call `finish()`
- Call `setResult(code)` with the result code
  - Usually RESULT_OK or RESULT_CANCELED and the Intent
- Call `finishActivity(requestCode)` from caller

Intents

Intents

- If an application has more than one activity, you may need to navigate from one activity to another
- This navigation is accomplished through Intents

- An Intent, is a passive data structure holding an abstract description of an operation to be performed
  - An 'simple' intent consists of two parts
    - An action and
    - The data that that action is supposed to use

```
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);
      }
   });
}
```

Intents

- Intents can be used to:
  - Declare your intention that an Activity or Service be started to perform an action, usually with (or on) a particular piece of data
  - Broadcast that an event (or action) has occurred
  - Explicitly start a particular Service or Activity

- Most commonly used to start new Activities
  - Explicitly: specifying the class to load

    ```
    new Intent(CurrentActivity.this, OtherActivity.class);
    ```
  - Implicitly: requesting that an action be performed on a piece of data

    ```
    Intent intent = new Intent(Intent.ACTION_DIAL,
                               Uri.parse("tel:555-2368"));
    ```

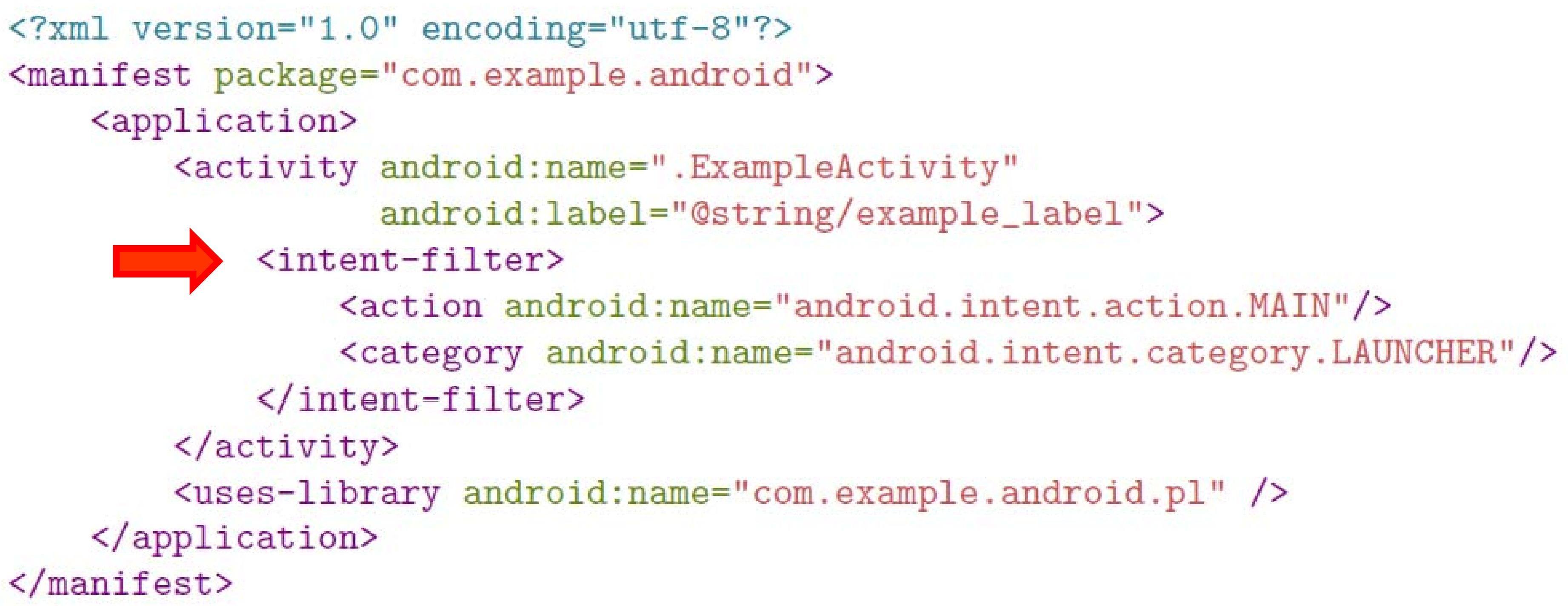
Intents

- Intent classes are *late binding*
- This means they are mapped and routed to a component that can handle a specified task at runtime, rather than at build or compile time.

- One Activity tells the platform, “I need a map to Granada, GR, ES,” and another component, one the platform determines is capable, handles the request and returns the result.

Implicit Intents – Resolution Process

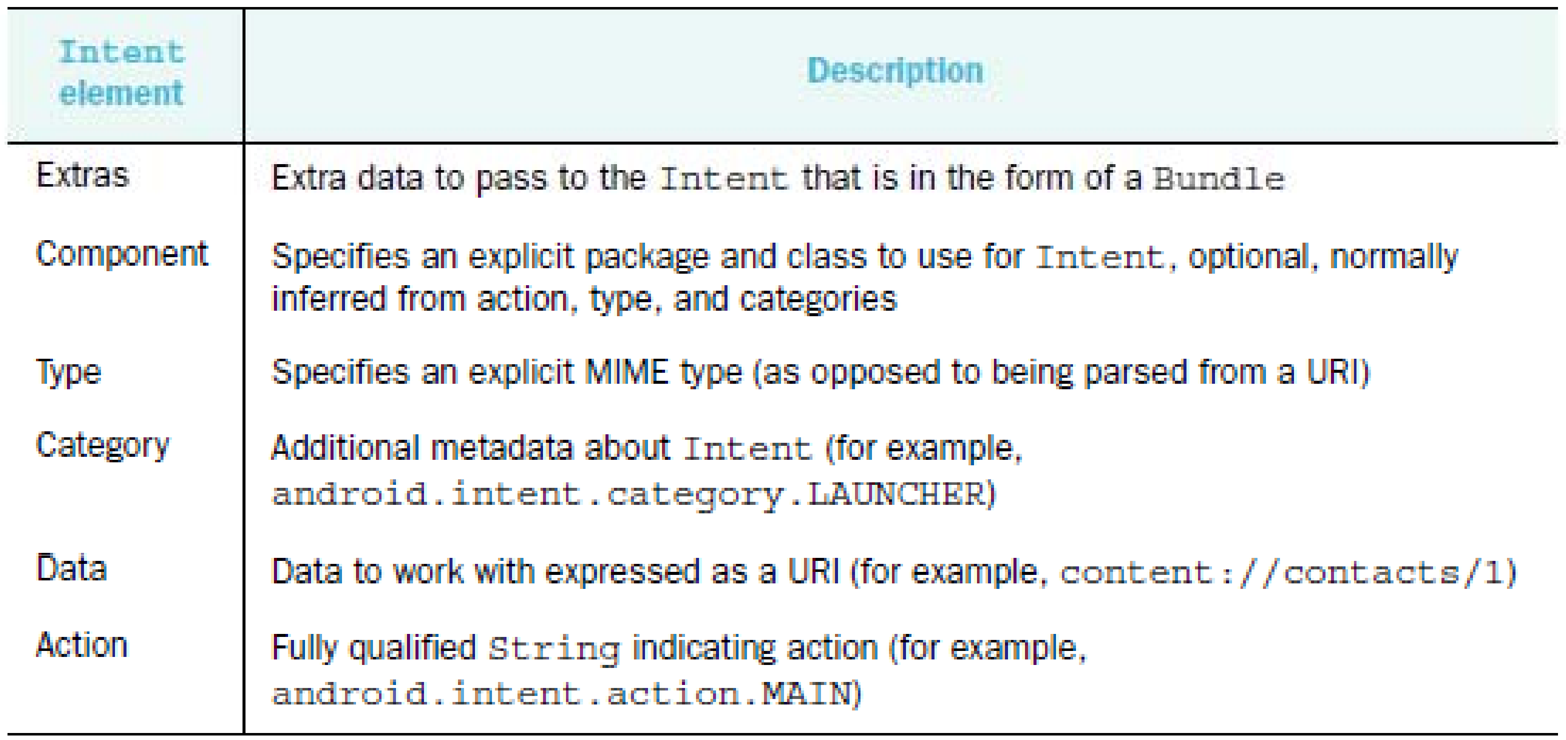
- Components register with the platform to be the destination for particular intent types using the <intent-filter> element in the AndroidManifest.xml
- When an implicit intent is used, the platform chooses the best suitable available component



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest package="com.example.android">
    <application>
        <activity android:name=".ExampleActivity"
                  android:label="@string/example_label">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <uses-library android:name="com.example.android.pl" />
    </application>
</manifest>
```

Intent Definition

- Intents are made up of three primary pieces of information —action, categories, and data— and include an additional set of optional elements.



Intent element	Description
Extras	Extra data to pass to the `Intent` that is in the form of a `Bundle`
Component	Specifies an explicit package and class to use for `Intent`, optional, normally inferred from action, type, and categories
Type	Specifies an explicit MIME type (as opposed to being parsed from a URI)
Category	Additional metadata about `Intent` (for example, `android.intent.category.LAUNCHER`)
Data	Data to work with expressed as a URI (for example, `content://contacts/1`)
Action	Fully qualified `String` indicating action (for example, `android.intent.action.MAIN`)

Implicit Intents – Resolution Process

- When an Intent is requested, resolution takes place through the registered filters, using the action, data, and categories of the Intent

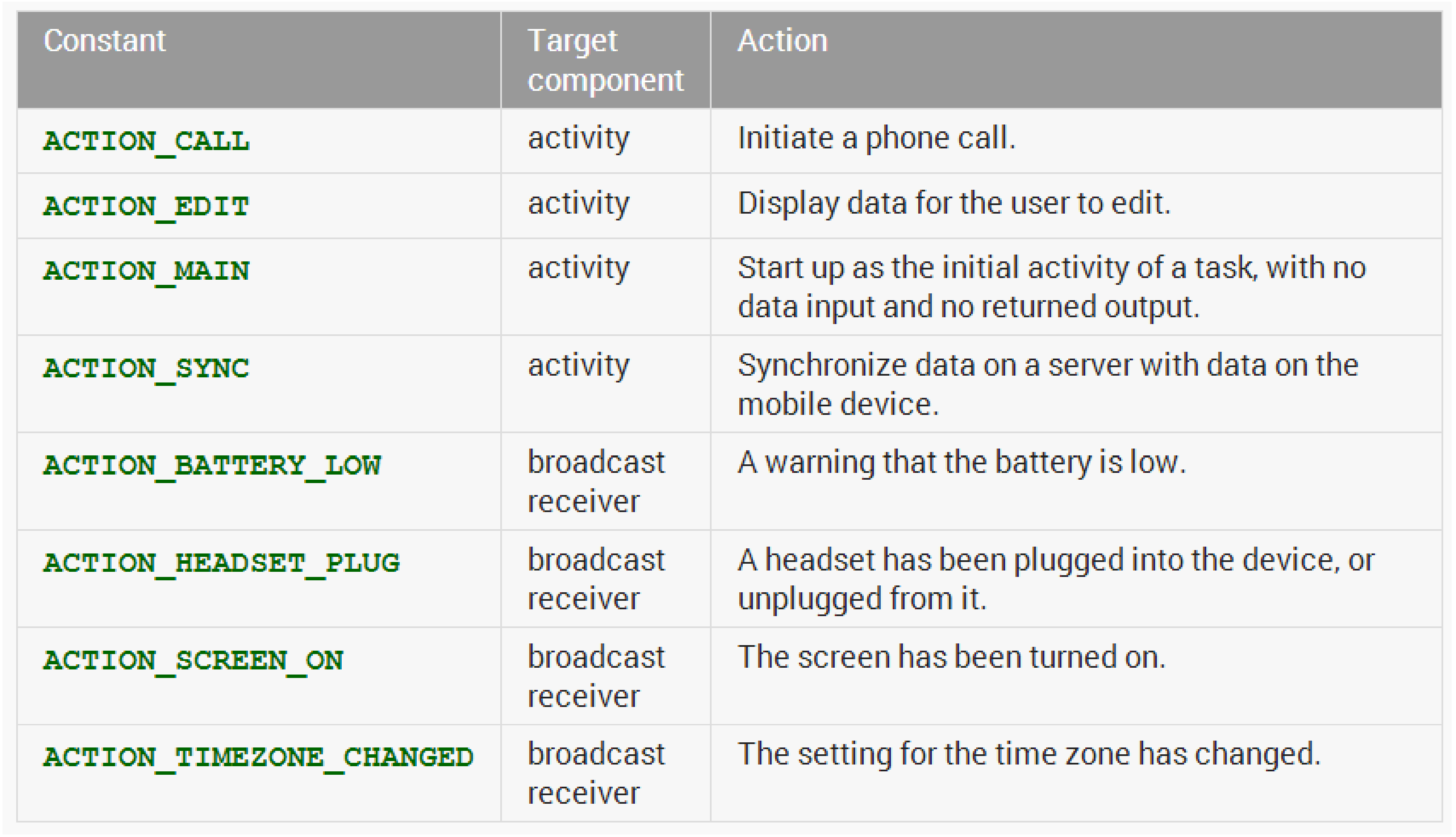
- There are three basic rules
  - The action must match
  - The category must match
  - If specified, the data type must match, or the combination of data scheme and authority and path must match

Implicit Intents – Resolution Process

- Action matching
  - The action specified in the Intent object must match one of the actions listed in the filter
  - An Intent object that doesn't specify an action automatically passes the test
  - If the filter does not list any actions, there is nothing for an intent to match, so all intents with action(s) fail

```
<intent-filter . . . >
    <action android:name="com.example.project.SHOW_CURRENT" />
    <action android:name="com.example.project.SHOW_RECENT" />
    <action android:name="com.example.project.SHOW_PENDING" />
    . . .
</intent-filter>
```

Intents – Android Actions



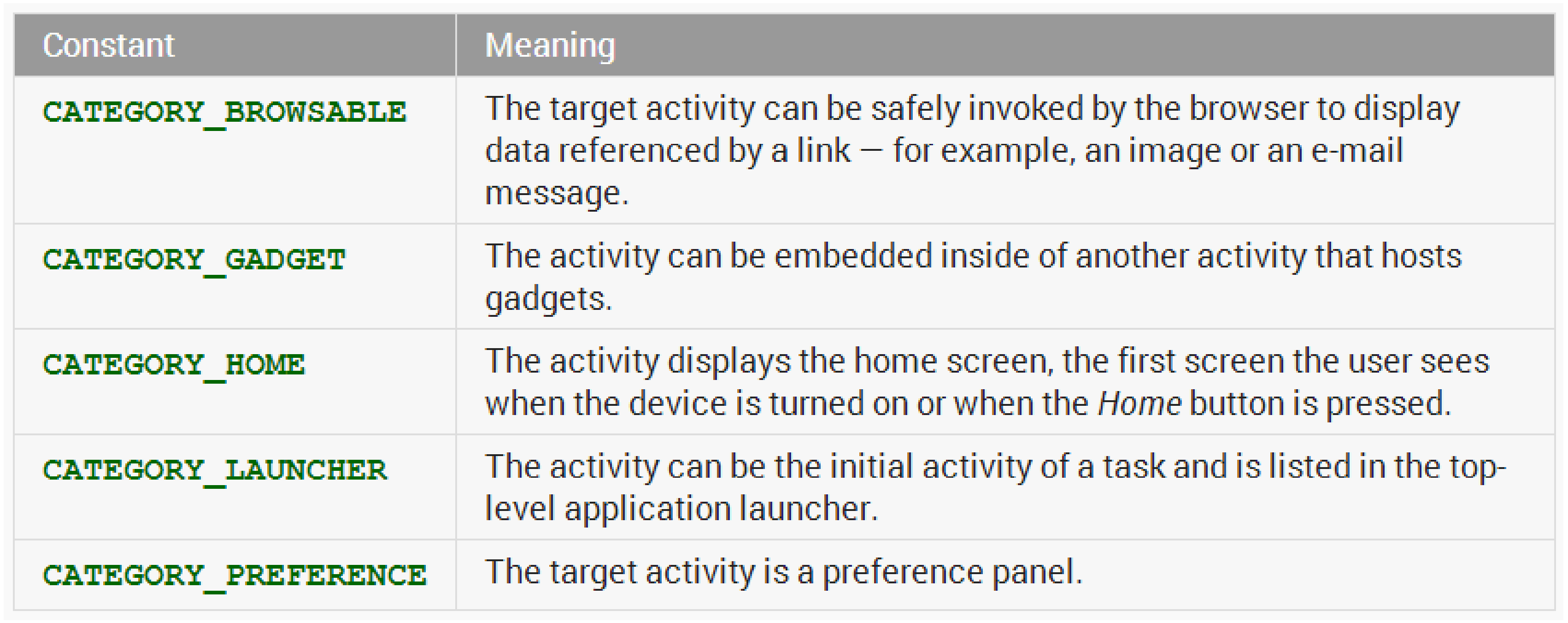
Constant	Target component	Action
ACTION_CALL	activity	Initiate a phone call.
ACTION_EDIT	activity	Display data for the user to edit.
ACTION_MAIN	activity	Start up as the initial activity of a task, with no data input and no returned output.
ACTION_SYNC	activity	Synchronize data on a server with data on the mobile device.
ACTION_BATTERY_LOW	broadcast receiver	A warning that the battery is low.
ACTION_HEADSET_PLUG	broadcast receiver	A headset has been plugged into the device, or unplugged from it.
ACTION_SCREEN_ON	broadcast receiver	The screen has been turned on.
ACTION_TIMEZONE_CHANGED	broadcast receiver	The setting for the time zone has changed.

Implicit Intents – Resolution Process

- Category matching
  - Every category in the Intent object must match a category in the filter (superset)
  - Unlike with actions, an <intent-filter> with no categories will match *only an Intent with no categories (it is not treated as a wildcard).*

```
<intent-filter . . . >
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    . . .
</intent-filter>
```

Intents – Android Categories



Constant	Meaning
`CATEGORY_BROWSABLE`	The target activity can be safely invoked by the browser to display data referenced by a link — for example, an image or an e-mail message.
`CATEGORY_GADGET`	The activity can be embedded inside of another activity that hosts gadgets.
`CATEGORY_HOME`	The activity displays the home screen, the first screen the user sees when the device is turned on or when the *Home* button is pressed.
`CATEGORY_LAUNCHER`	The activity can be the initial activity of a task and is listed in the top-level application launcher.
`CATEGORY_PREFERENCE`	The target activity is a preference panel.

Will Some App Receive the Intent?

- Call `queryIntentActivities()` to get a list of activities capable of handling the Intent

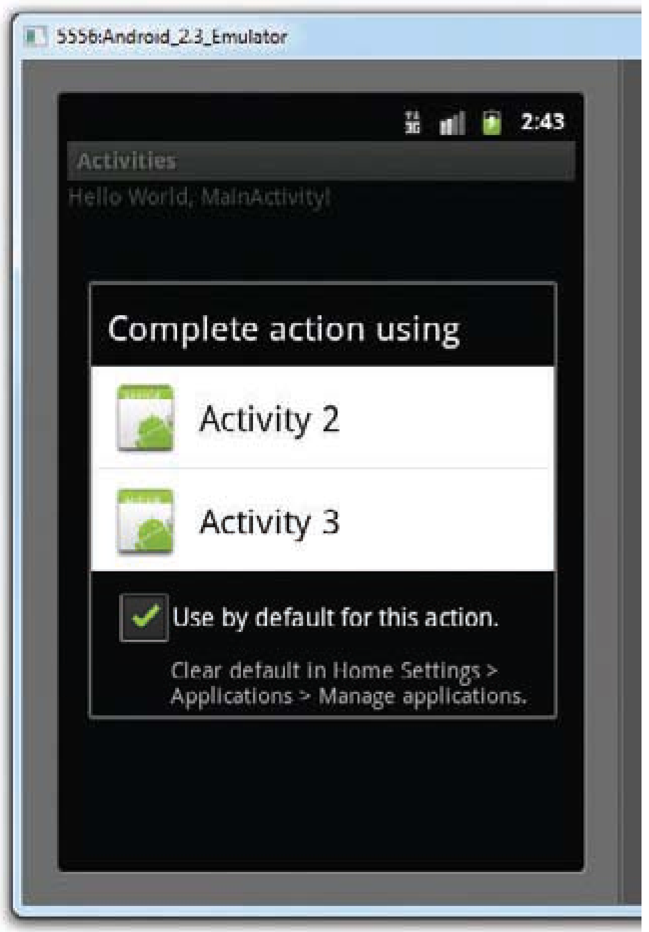
```
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities =
                packageManager.queryIntentActivities(intent, 0);
boolean isIntentSafe = activities.size() > 0;
```

Resolving Intent Colision

- What happens if another activity (in either the same or a separate application) has the same filter name?

- Then the Android OS will display a selection dialog

- So, we don't have to worry, the user will choose its preference



Hello World, MainActivity!

Complete action using

Activity 2

Activity 3

Use by default for this action.

Clear default in Home Settings > Applications > Manage applications.

Returning Results from an Intent

- `startActivity()` invokes another activity but does not return a result to the current activity
- Use `startActivityForResult()` when we want a result back

```
startActivityForResult(new Intent(ACTION_XXXX),
                                         request_Code);
```

- The returned result must be handled

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == request_Code) {
        if (resultCode == RESULT_OK) {
            Toast.makeText(this,data.getData().toString(),
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Passing Data via Intents

- ‘Bundling’ the data

```
Intent i = new Intent("net.learn2develop.ACTIVITY2");
Bundle extras = new Bundle();
extras.putString("Name", "Your name here");
i.putExtras(extras);
startActivityForResult(i, 1);
```

- ‘Unbundle’ the data

```
Bundle extras = getIntent().getExtras();
if (extras!=null)
{
    defaultName = extras.getString("Name");
}
```

Specificy How to Handle an Intent

- Intent calls may carry flags to modify default OS handling of the Intent

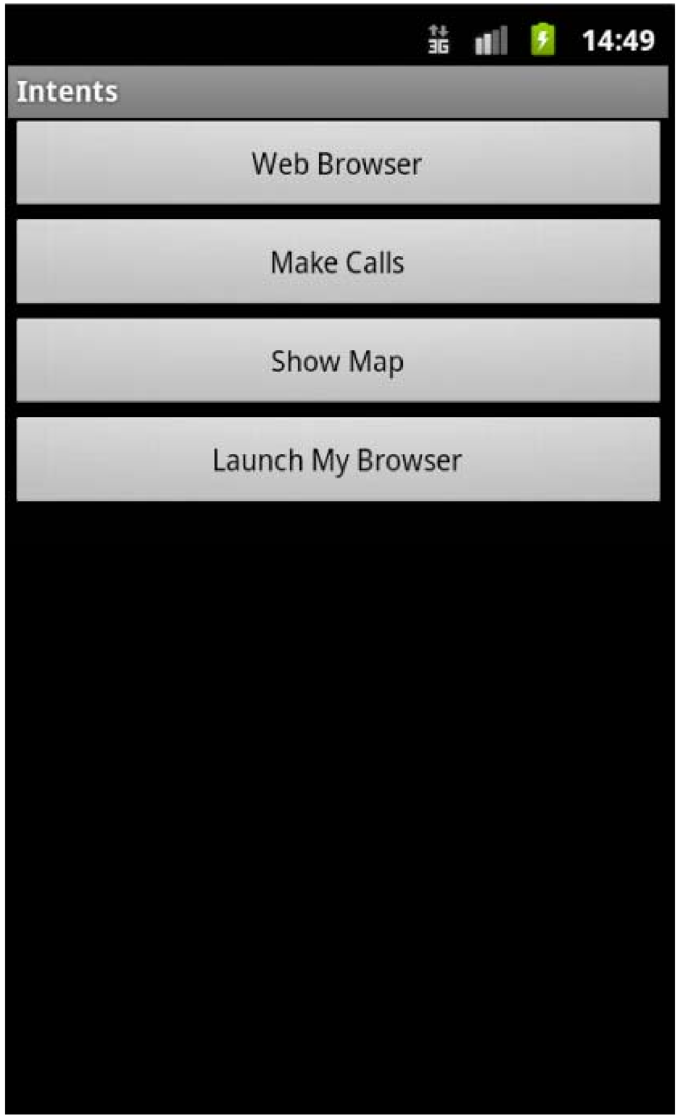
- Some examples
  - FLAG_ACTIVITY_NO_HISTORY
    - Don’t put this Activity in the History stack
  - FLAG_DEBUG_LOG_RESOLUTION
    - Causes extra logging information to be printed when this Intent is processed

- How?

```
Intent newInt= new Intent(Intent.ACTION_SEND);
newInt.setFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
```

Let’s Try… Project Intents

- Import project
- Web Browser
  - User selection dialog
- Make Calls
  - Phone number preset
  - Change ACTION_DIAL for ACTION_CALL
- Show Map (Do not Press!!)
  - API for maps not installed
- Launch My Browser
  - No intent-filter matches

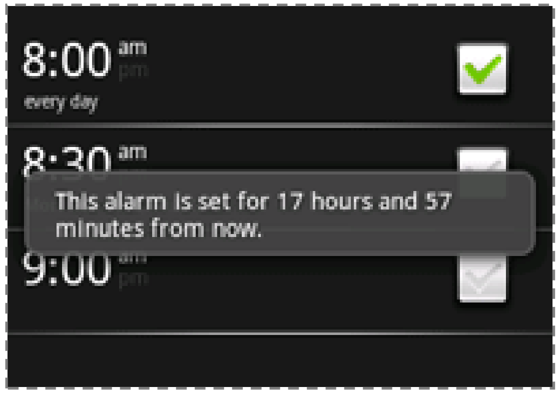


Notifications

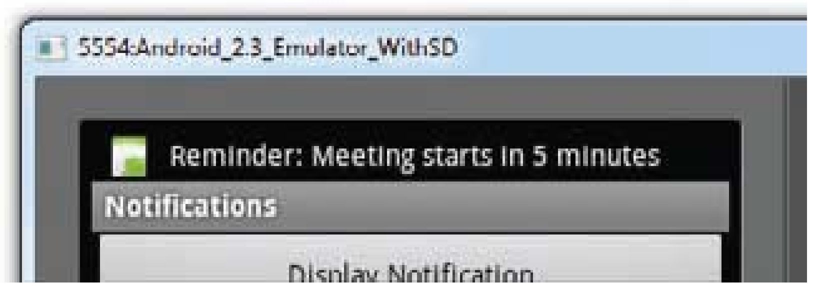
Displaying Notifications

- Toast: Short persistence notification on screen

```
Toast.
      makeText(this, “Notification text”,
               Toast.LENGTH_SHORT)
      .show();
```

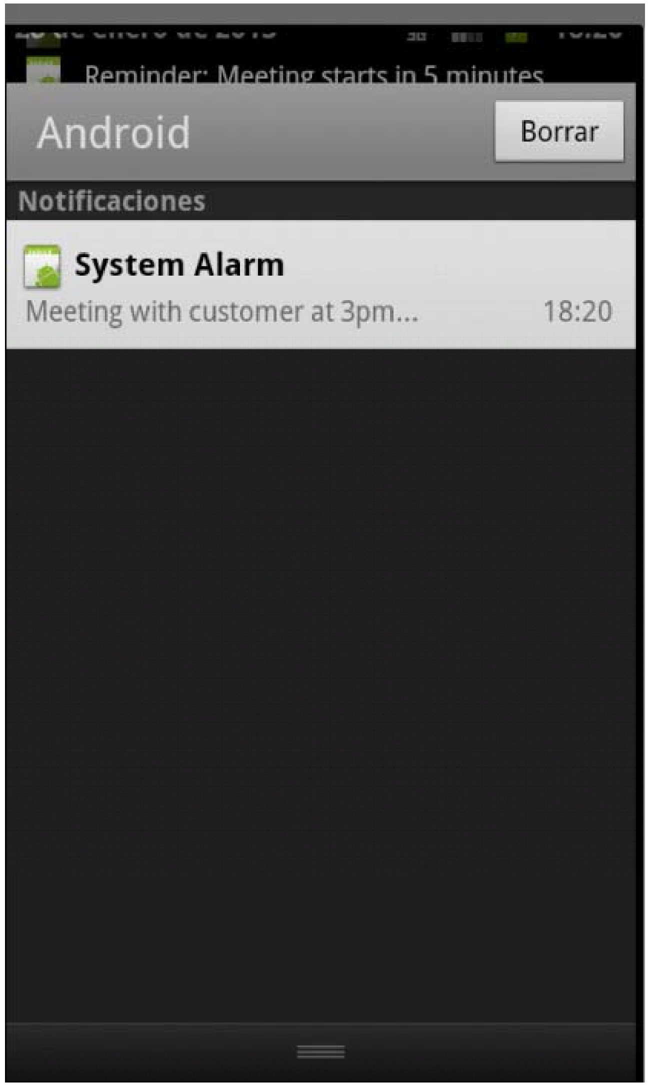

8:00 am pm
every day
8:30 am
This alarm is set for 17 hours and 57 minutes from now.
9:00 am pm

- Notifications: display a persistent message at the top of the device


5554:Android_2.3_Emulator_WithSD
Reminder: Meeting starts in 5 minutes

Let's Try… Project Notifications

- Import Project
- Press button
- Press notification icon and drag down
- Have a look at the code
- Use a toast



Reminder: Meeting starts in 5 minutes

System Alarm
Meeting with customer at 3pm… 18:20

Let's Try… Project Notifications

- There are now two activities in the manifest

- PendingIntent
  - An Intent that it is not yet launched, but it is prepared to be launched in the future

```
Intent i = new Intent(this, NotificationView.class);
i.putExtra("notificationID", notificationID);
PendingIntent pendingIntent =
          PendingIntent.getActivity(this, 0, i, 0);
// getActivity (context, request code, intent, flags)
```

Let's Try… Project Notifications

- Obtain an instance of the NotificationManager class

```
NotificationManager nm = (NotificationManager)
    getSystemService(NOTIFICATION_SERVICE);

Notification notif = new Notification(
    R.drawable.icon,
    "Reminder: Meeting starts in 5 minutes",
    System.currentTimeMillis());
```

- Set the details of the notification

```
CharSequence from = "System Alarm";
CharSequence message = "Meeting with customer at 3pm...";
notif.setLatestEventInfo(this, from, message, pendingIntent);
//---100ms delay, vibrate for 250ms, pause for 100 ms and
// then vibrate for 500ms---
notif.vibrate = new long[] { 100, 250, 100, 500};
```

Let’s Try… Project Notifications

- Display the notification when the button is clicked

```
nm.notify(notificationID, notif);
```

- When the user clicks on the notification, the NotificationView activity is launched, and the notification canceled

```
//---look up the notification manager service---
NotificationManager nm = (NotificationManager)
    getSystemService(NOTIFICATION_SERVICE);

//---cancel the notification that we started
nm.cancel(getIntent().getExtras().getInt("notificationID"));
```

Android Development

Session 3

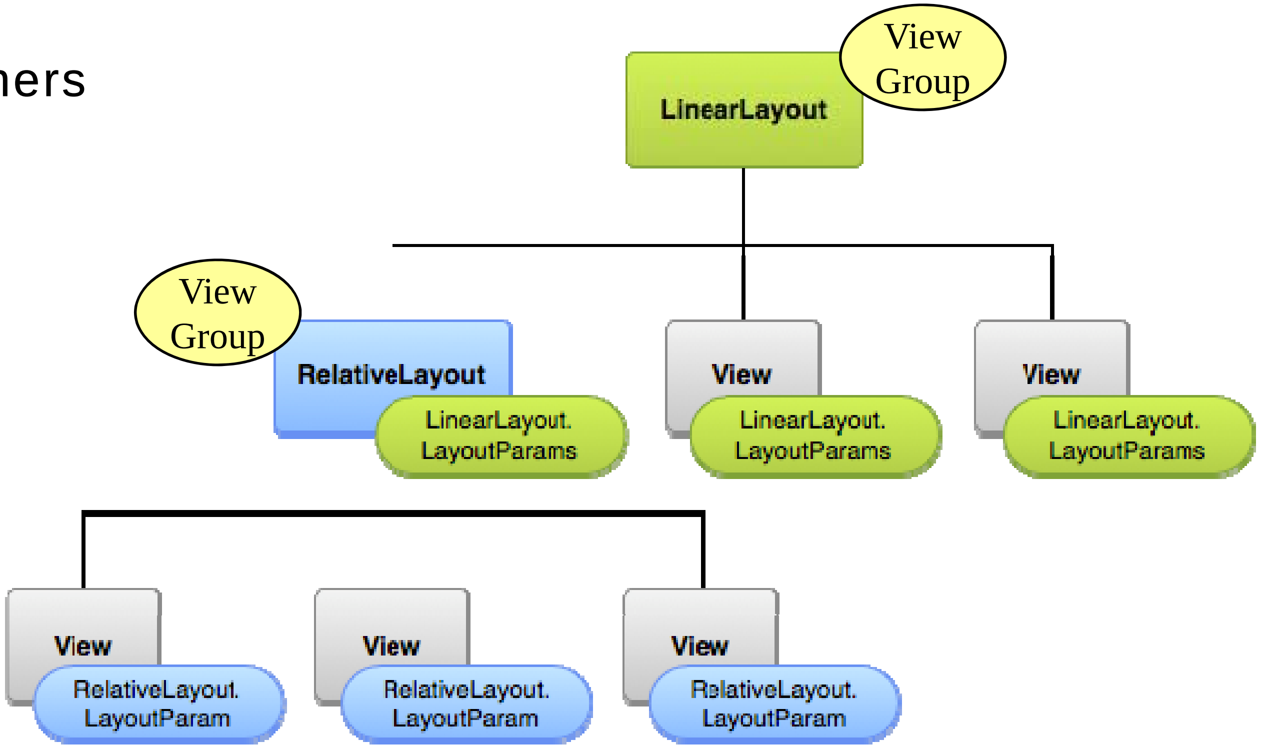
Javier Poncela

Contents

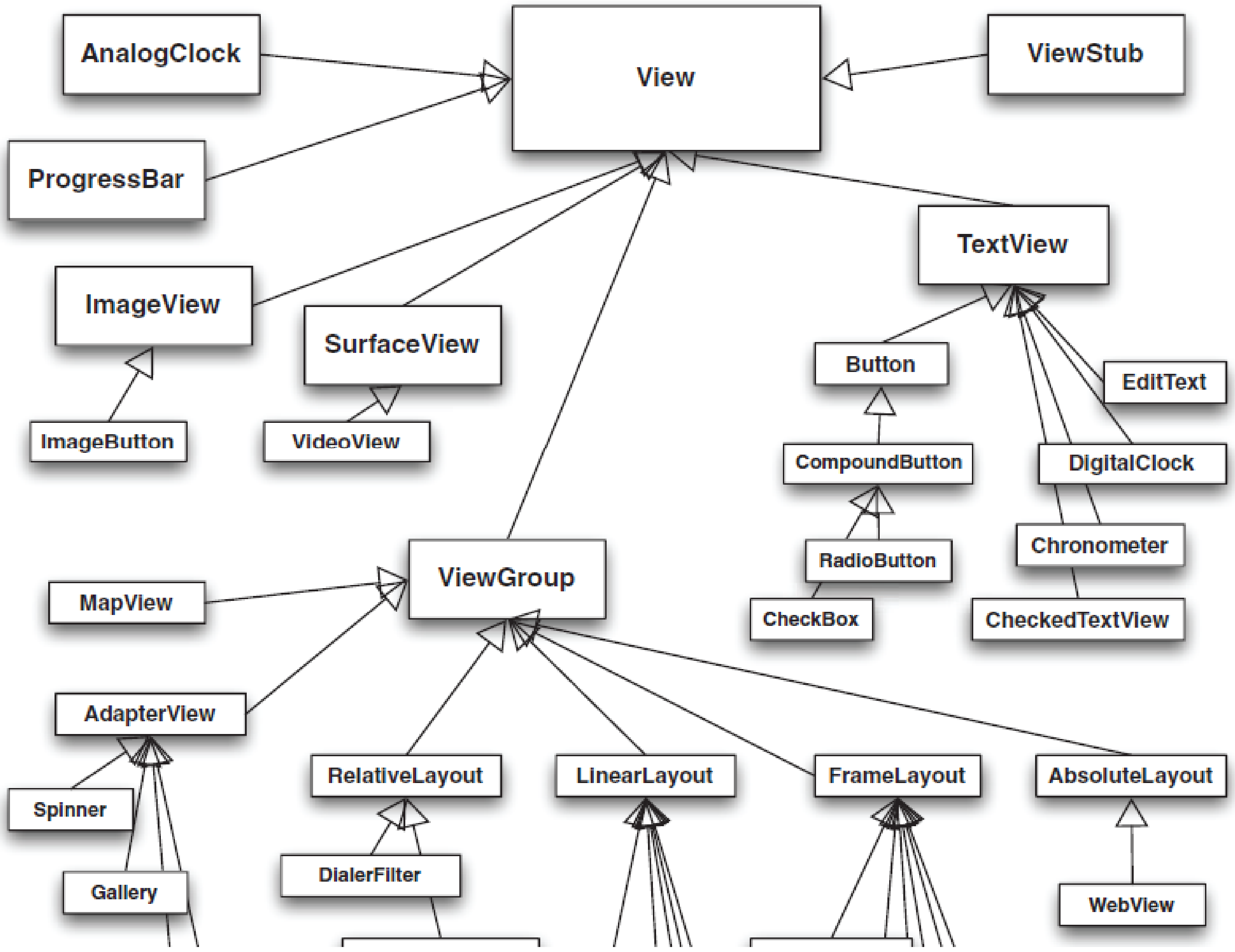
User Interface

Views

- *Views*: Base class for all visual interface elements
  - Common operations
    - Set properties
    - Set focus
    - Attach Listeners
    - Set visibility
- *View Groups*: Extensions of View class that can contain multiple child Views
  - Layouts are View Groups


View Group
LinearLayout
View Group
RelativeLayout
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
RelativeLayout.LayoutParam
View
RelativeLayout.LayoutParam
View
RelativeLayout.LayoutParam

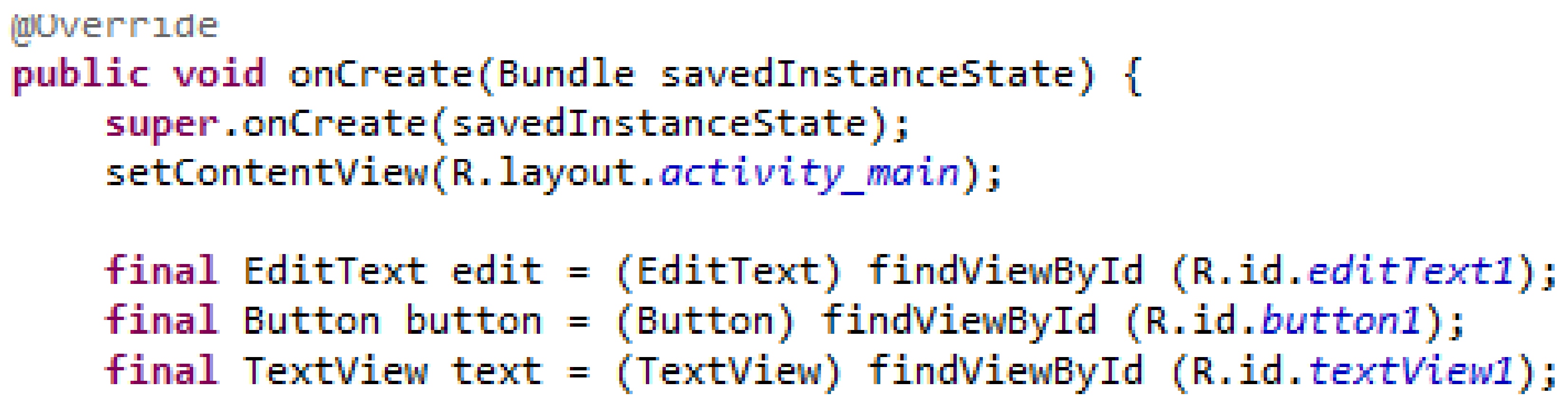
Views


AnalogClock
View
ViewStub
ProgressBar
TextView
ImageView
SurfaceView
Button
EditText
ImageButton
VideoView
CompoundButton
DigitalClock
Chronometer
RadioButton
ViewGroup
MapView
CheckBox
CheckedTextView
AdapterView
RelativeLayout
LinearLayout
FrameLayout
AbsoluteLayout
Spinner
Gallery
DialerFilter
WebView

Views

- Views are usually defined in XML

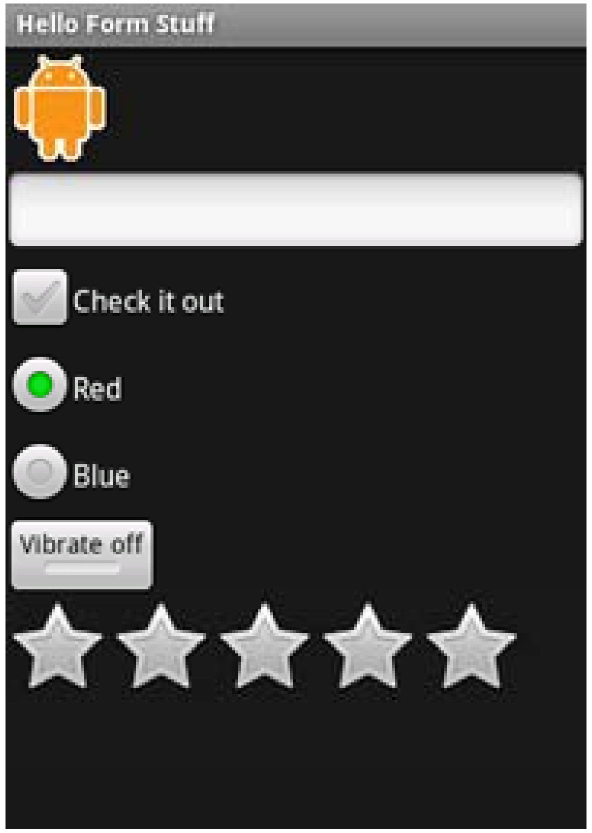
- … and then, they are Inflated in the code
    - Component is looked for in the predefined elements
    - Properties are read
    - Component is created



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

    final EditText edit = (EditText) findViewById (R.id.editText1);
    final Button button = (Button) findViewById (R.id.button1);
    final TextView text = (TextView) findViewById (R.id.textView1);
```

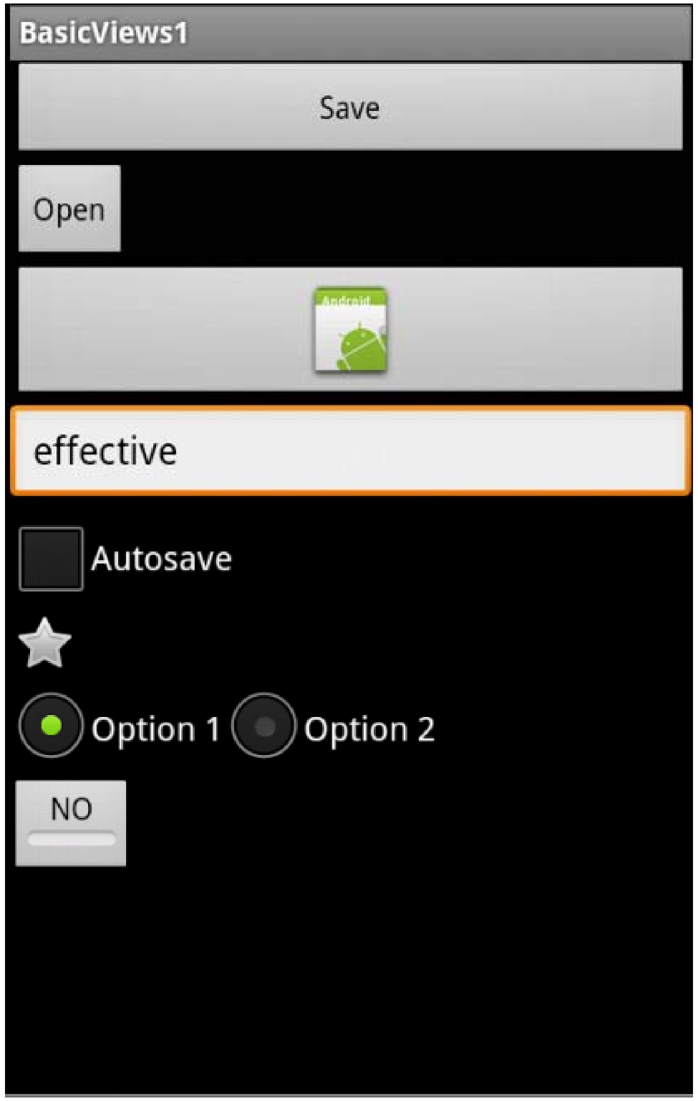
Views – Widget Toolbox

- Many predefined widgets
  - Buttons
  - Text field
  - Editable text field
  - Check box
  - Radio buttons
  - Toggle Button
  - DatePicker
  - TimePicker
  - Spinner
  - AutoComplete
  - Gallery
  - MapView
  - WebView
  - ...



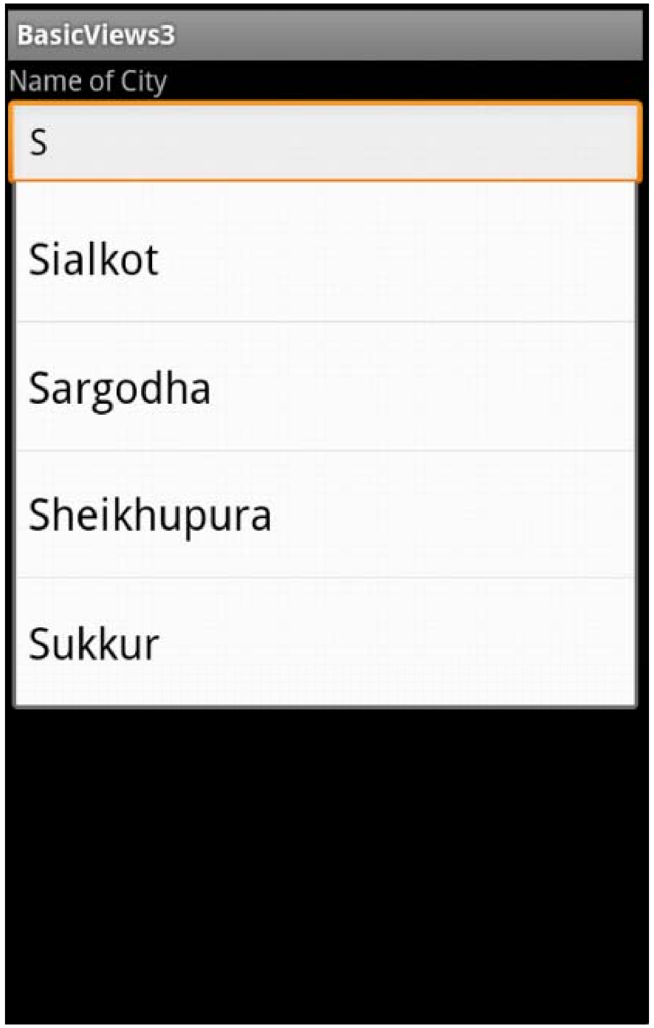
Let's Try… Project BasicViews1

- Import project
- Look at res/layout/main.xml
- Look at code .java
  - See response to events
  - See response to save button clicks
    - xml: android:onClick
    - java: btnSaved_clicked



Let’s Try… Project BasicViews3

- AutoCompleteTextView
  - shows a list of completion suggestions automatically while the user is typing
- Method setThreshold

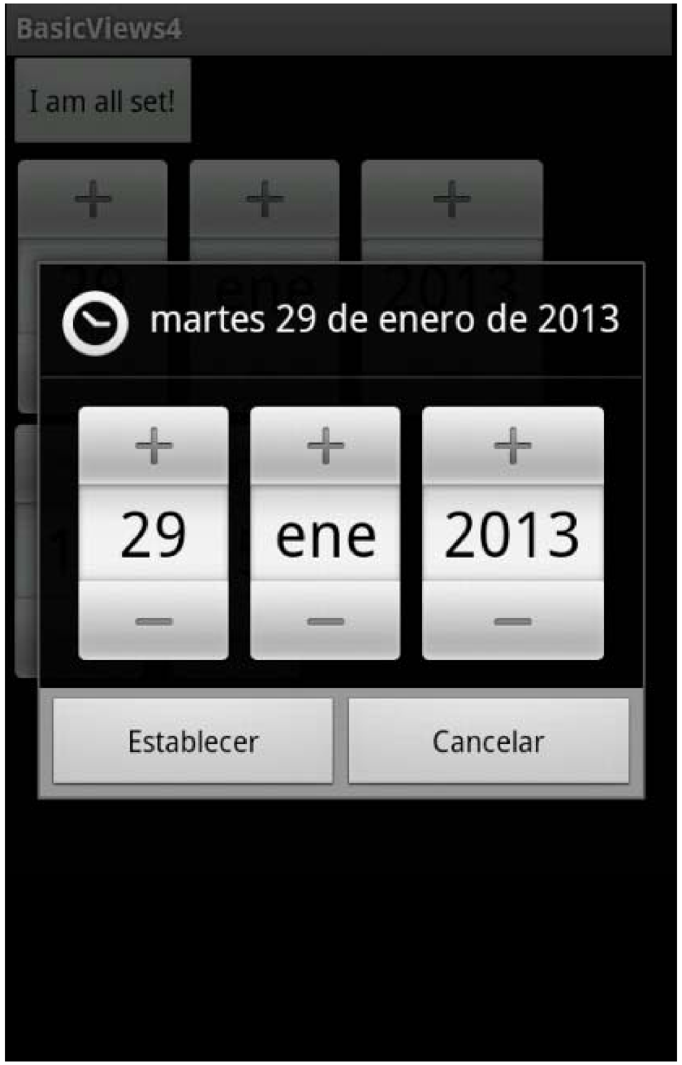


Let's Try… Project BasicViews4

- Time and date selection
  - TimePicker
  - DatePicker

- Have a look at
  - xml file
  - java dialog management

- Improvement
  - Create two additional buttons for calling
    - Btn1: TimePicker Dialog
    - Btn2: DatePicker Dialog

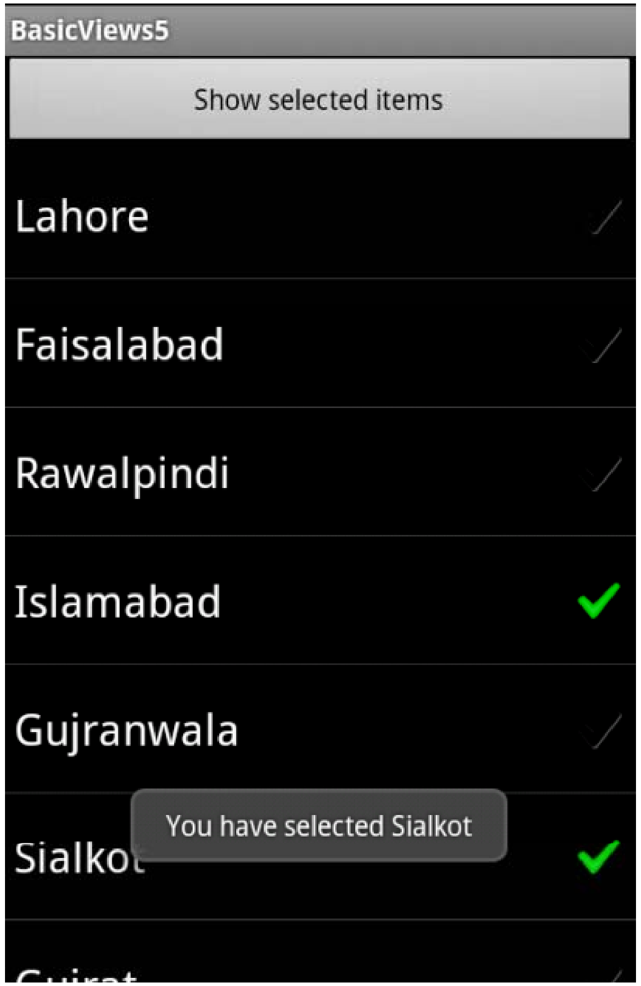


Let's Try… Project BasicViews5

- ListView: Displays a list of items in a vertically scrolling list
- Check behaviour with different Choice modes (lines 23-25)
- Items are obtained from resource in strings.xml

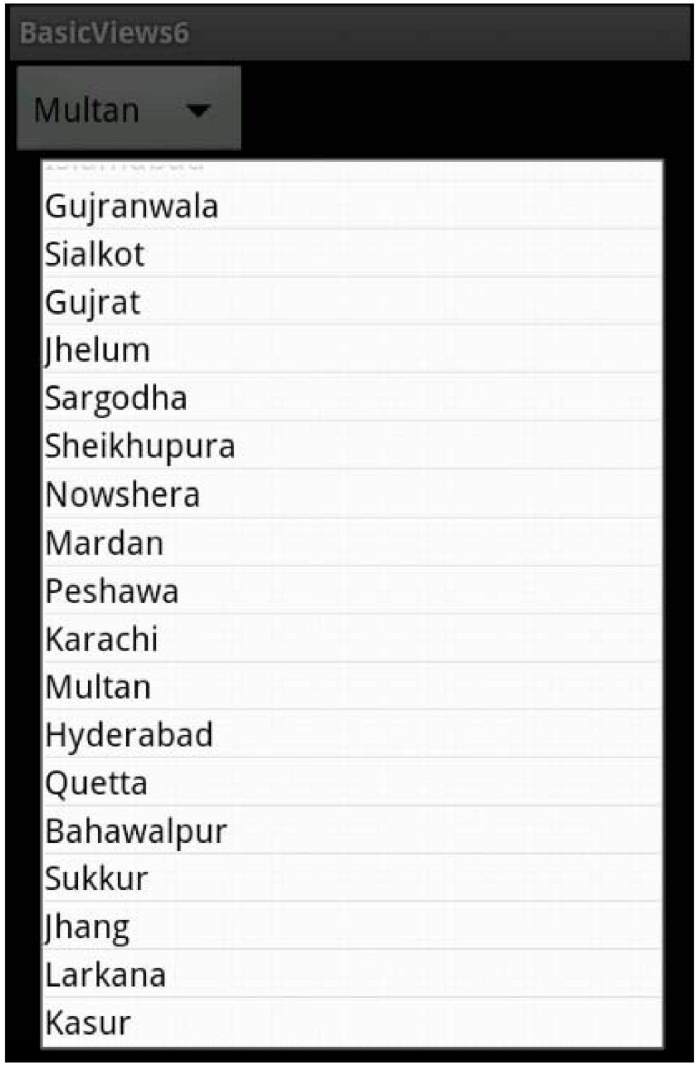
  `getResources`
- Press 's' in the emulator

  `setTextFilterEnabled`



Let’s Try… Project BasicViews6

- SpinnerView: Displays one item at a time from a list and enables users to choose among them



BasicViews6

Multan

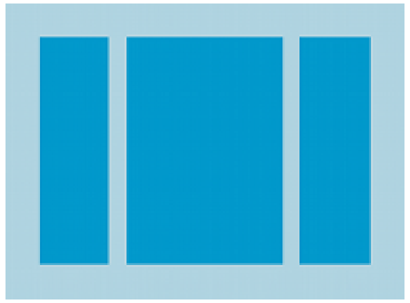
Gujranwala
Sialkot
Gujrat
Jhelum
Sargodha
Sheikhupura
Nowshera
Mardan
Peshawa
Karachi
Multan
Hyderabad
Quetta
Bahawalpur
Sukkur
Jhang
Larkana
Kasur

Layouts

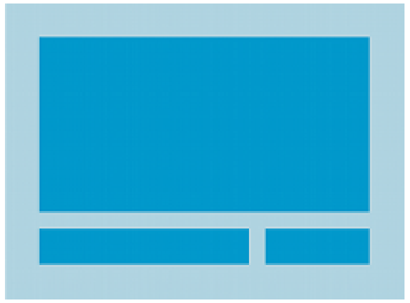
Layouts

- Visual structure for a user interface
  - Declare UI elements in XML
  - Instantiate layout elements at runtime (not recommended)

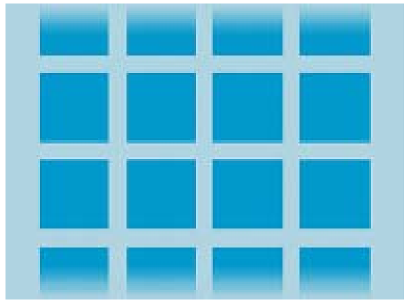
Linear Layout



Relative Layout



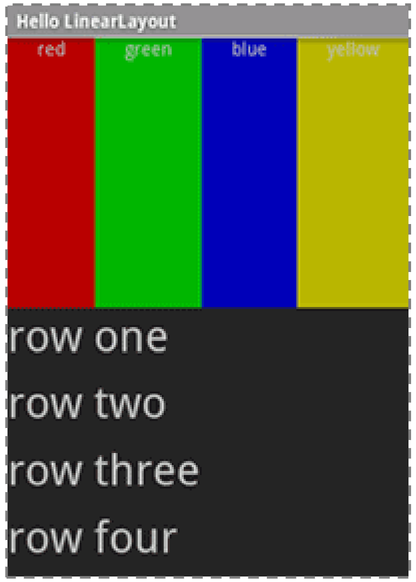
Grid/Table Layout



Linear Layout

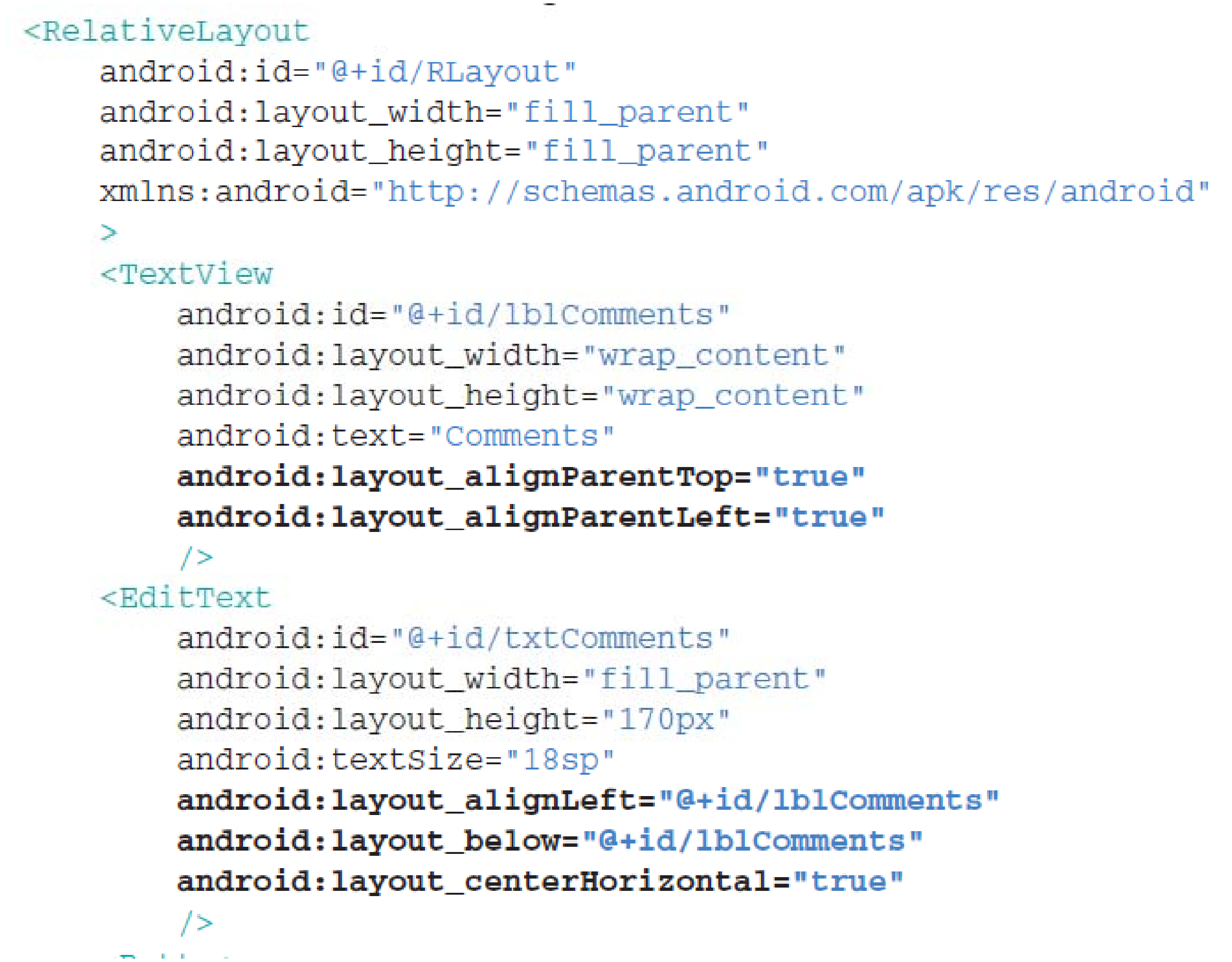
- Elements shown in sequence ‘horizontal’ or ‘vertical’
  - Android takes care of the arrangement

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



Relative Layout

- Elements placed in relation to the position of others

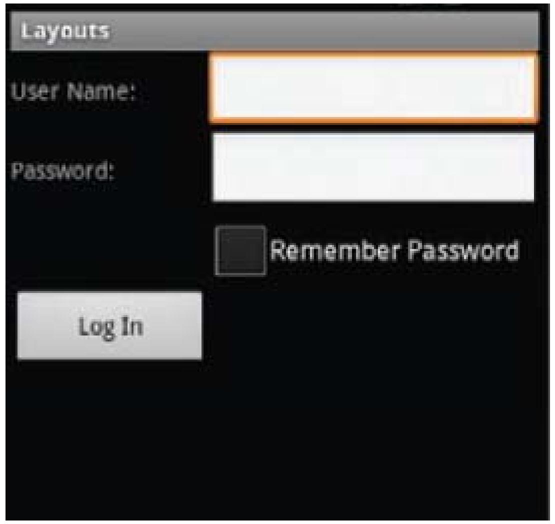


```
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        />
```

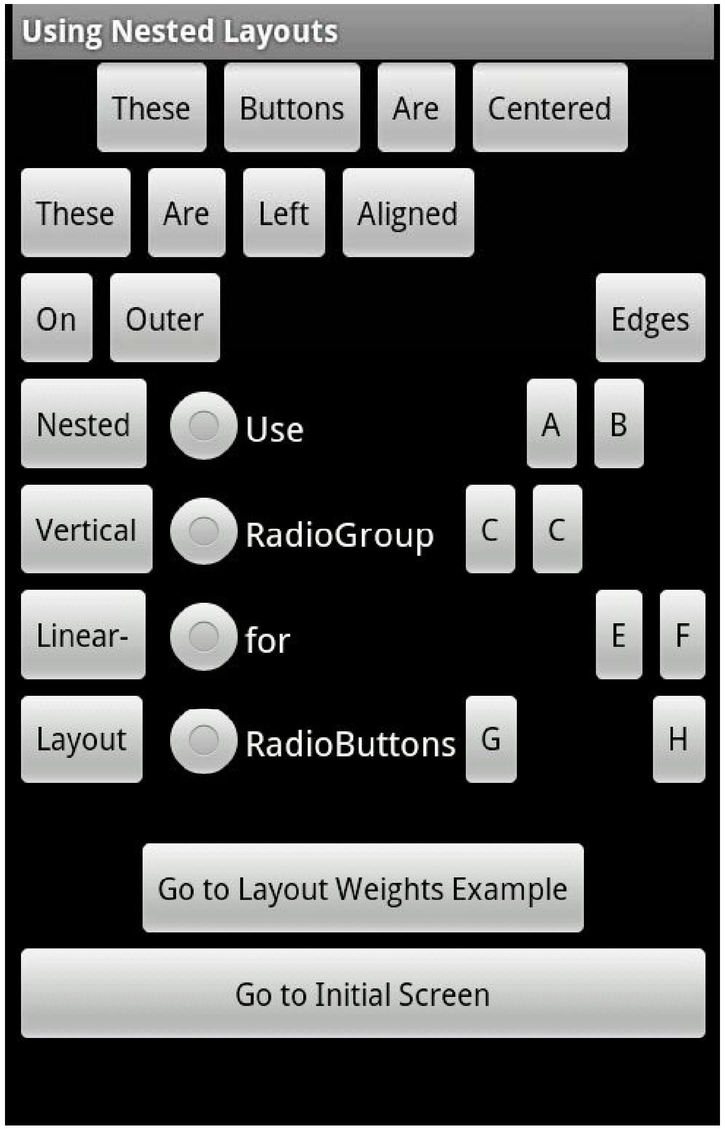
Table Layout

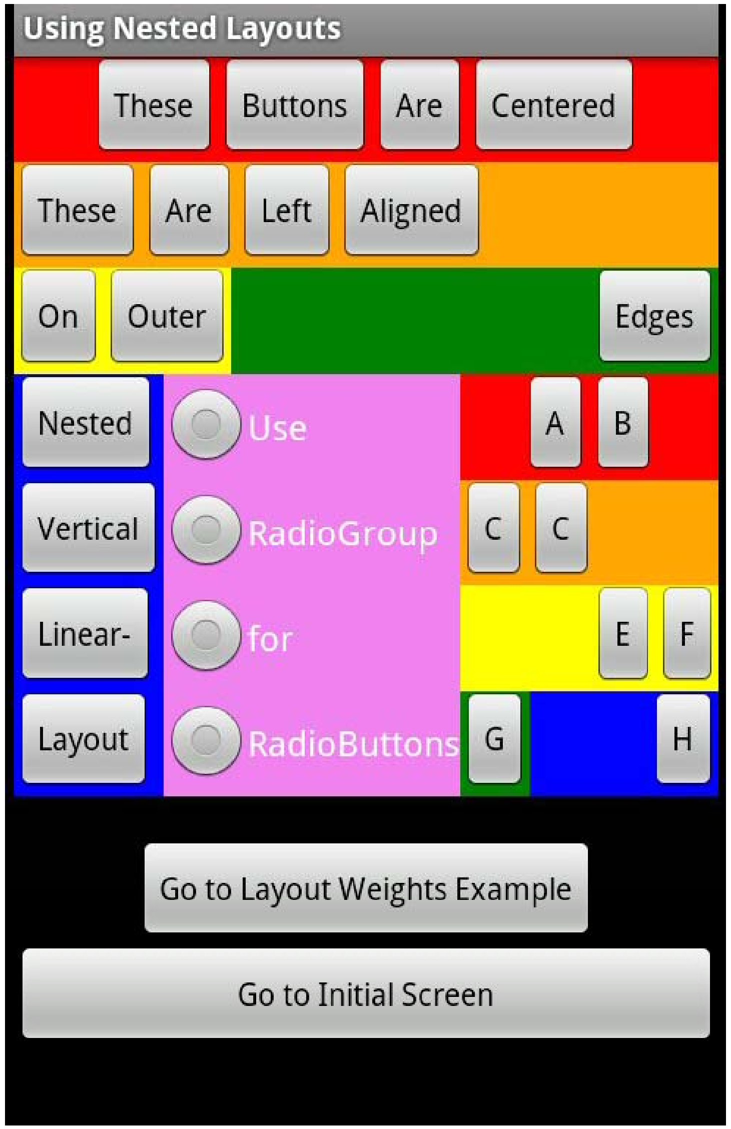
- Groups views into rows and columns

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    >
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120px"
            />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
            />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
            />
    </TableRow>
```

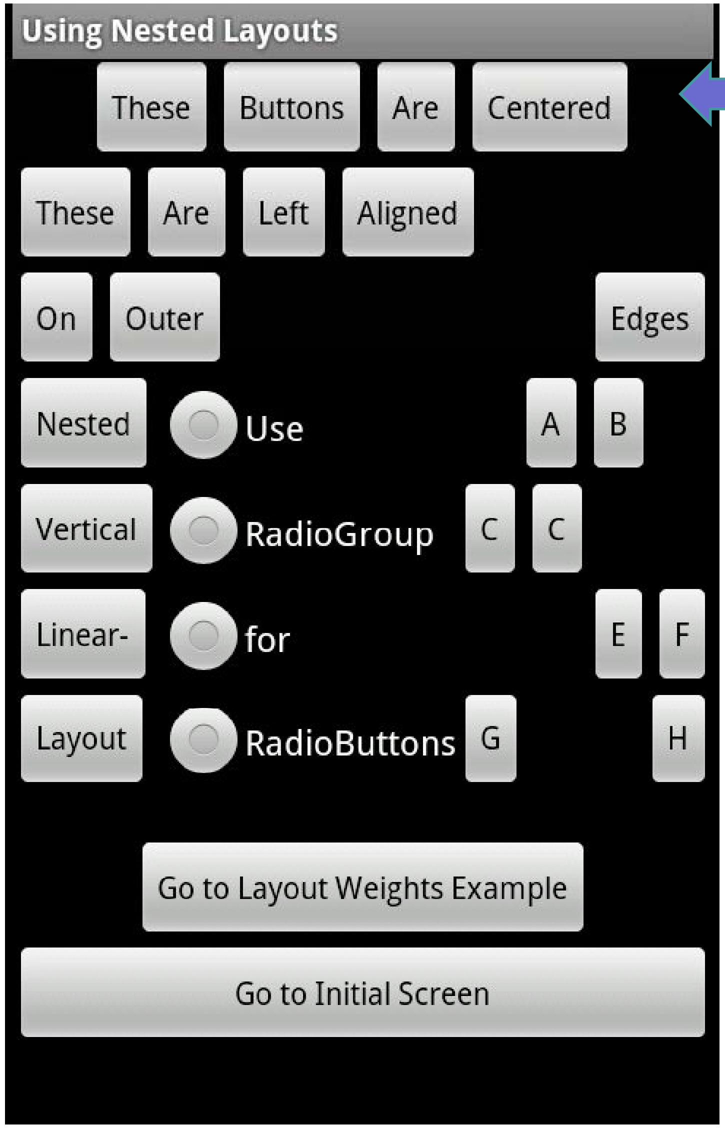


Layouts can be nested…





Layouts can be nested…



```
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:gravity="center_horizontal"
  android:background="@color/color_1">

  <Button android:text="These"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

  <Button android:text="Buttons"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

  <Button android:text="Are"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>

  <Button android:text="Centered"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

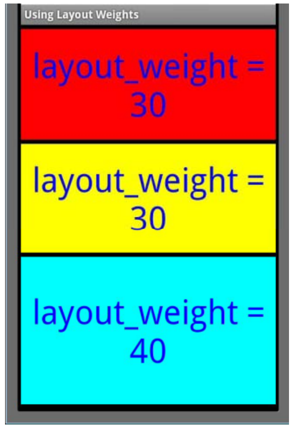
Measurement Units

- dp — Density-independent pixel. 160dp is equivalent to one inch of physical screen size. This is the recommended unit of measurement when specifying the dimension of views in your layout. You can specify either “dp” or “dip” when referring to a density-independent pixel.
- sp — Scale-independent pixel. This is similar to dp and is recommended for specifying font sizes.
- pt — Point. A point is defined to be 1/72 of an inch, based on the physical screen size.
- px — Pixel. Corresponds to actual pixels on the screen. Using this unit is not recommended, as your UI may not render correctly on devices with different screen sizes.

HVGA-L : 480x320	QVGA-L : 320x240	WVGA-L : 800x480
HVGA-P : 320x480	QVGA-P : 240x320	

Common Attributes

- Size
  - android:layout_height, android:layout_width
    - match_parent/ fill_parent : fill the parent space (minus padding)
    - wrap_content: use natural size (plus padding)
    - An explicit size with a number and a dimension
  - android:layout_weight
    - A number that gives proportional sizes



- Alignment
  - android:layout_gravity
    - How the View is aligned within containing View.
  - android:gravity
    - How the text or components inside the View are aligned.
  - Possible values
    - top, bottom, left, right, center_vertical, center_horizontal, center (i.e., center both ways), fill_vertical, fill_horizontal, fill (i.e., fill both directions)

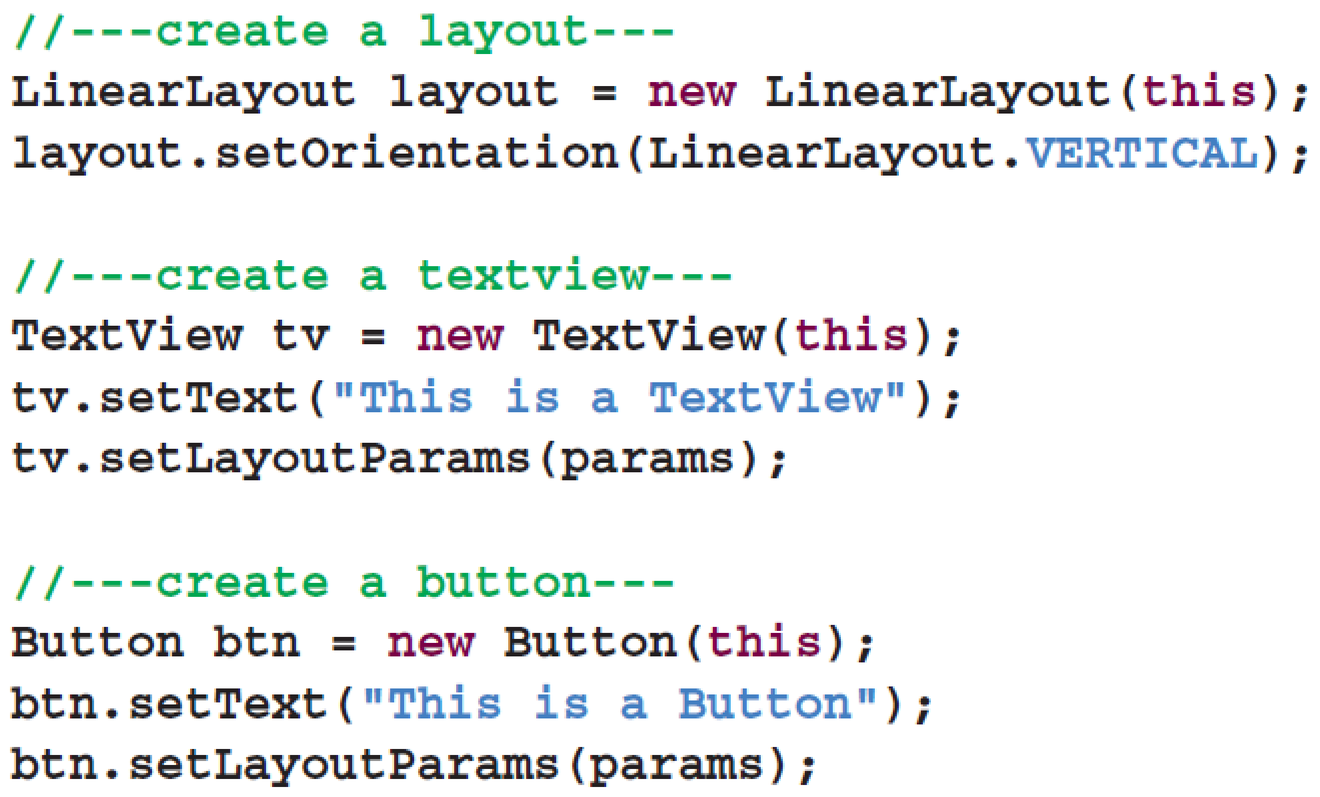
Common Attributes

- Margins (blank space outside)
  - android:layout_marginBottom, android:layout_marginTop, android:layout_marginLeft, android:layout_marginRight
    - Units: dp, sp, ...

- Padding (blank space inside)
  - android:paddingBottom, android:paddingTop, android:paddingLeft, android:paddingRight
    - Units: dp, sp, ...

Creating the User Interface Programmatically

- So far, all the UIs have been created using XML
- But UIs can also be created in code



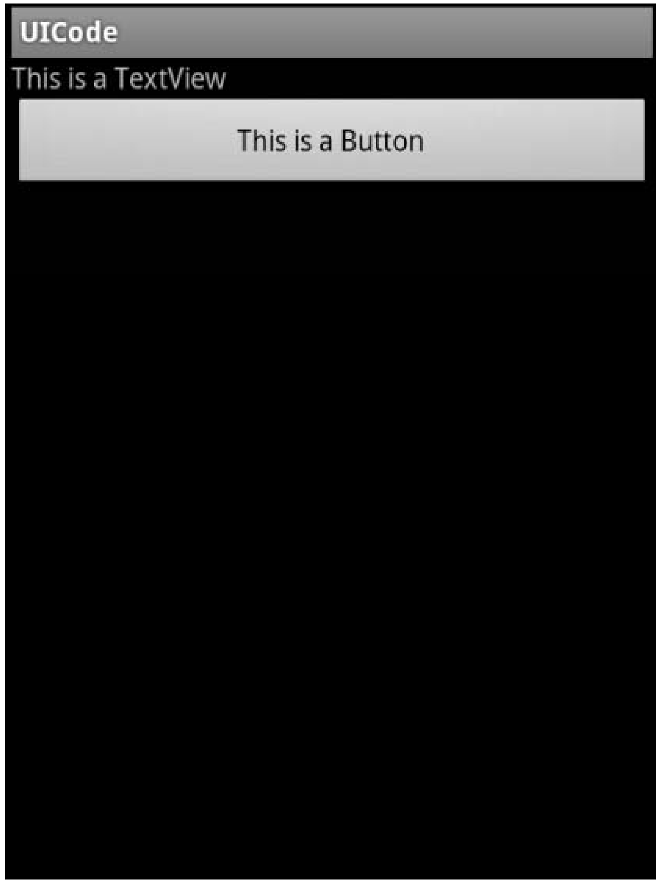
```
//---create a layout---
LinearLayout layout = new LinearLayout(this);
layout.setOrientation(LinearLayout.VERTICAL);

//---create a textview---
TextView tv = new TextView(this);
tv.setText("This is a TextView");
tv.setLayoutParams(params);

//---create a button---
Button btn = new Button(this);
btn.setText("This is a Button");
btn.setLayoutParams(params);
```

Let’s Try… Project UICode

- See main.xml

- See .java code
  - Layout is not inflated

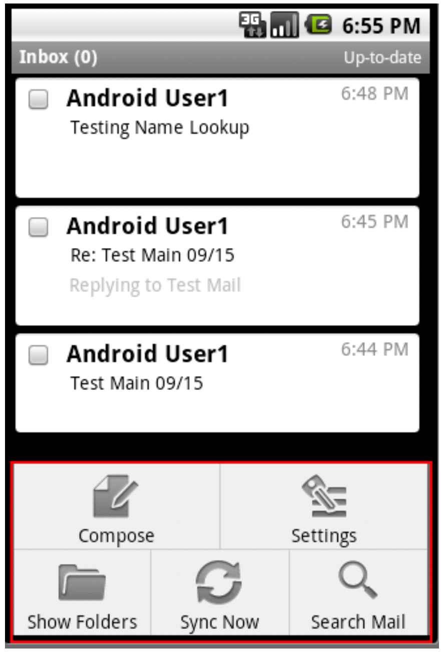
  - Each element is added in the code

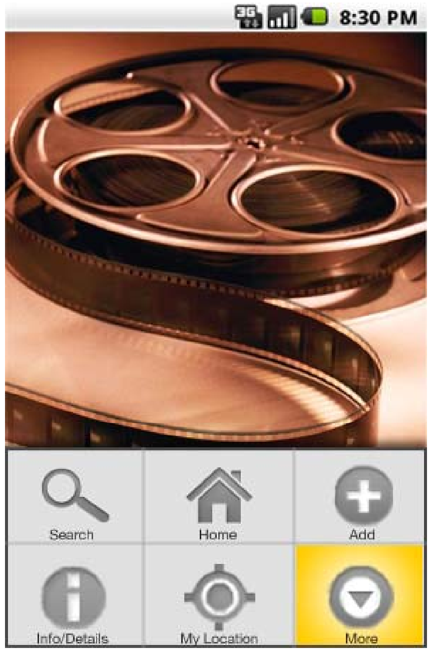


Menus

Menus

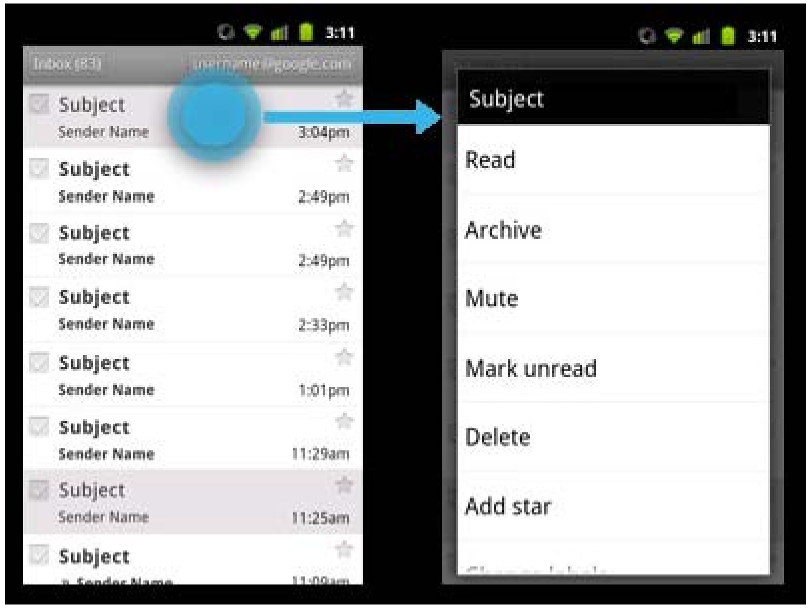
- Options – Primary menu shown when user presses the MENU button of the device
  - Displayed at the bottom of screen
  - May include icons and text
  - If there are more than 6 items, a More item is displayed that opens an Expanded menu





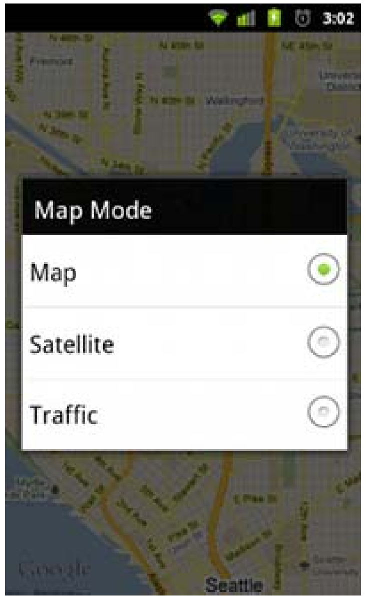
Menus

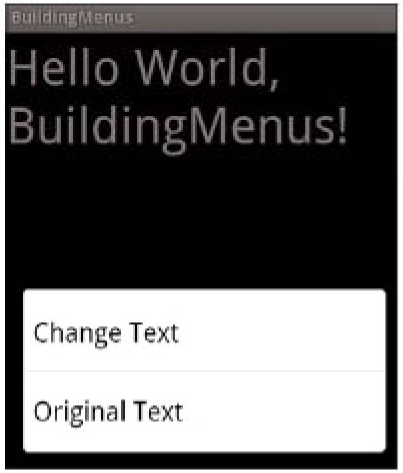
- Context – View-specific menu to be shown when user touches and holds the view
  - Floating menu that appears when the user performs a long-click on an element
  - Similar to “right-click” menu

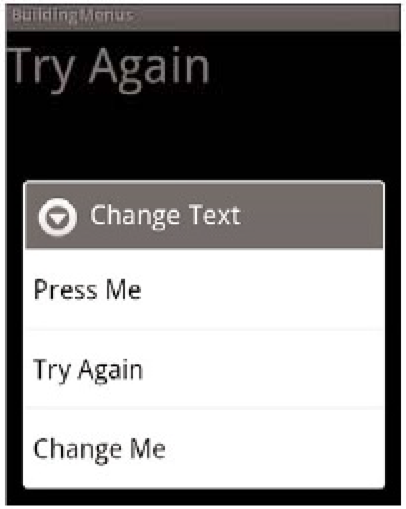


Menus

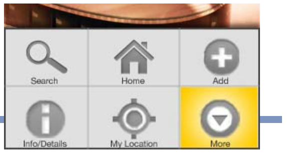
- Submenu or Popup – A menu activated when user touches a visible menu item
  - Good to provide options for a second part of a command/menu item
  - Can't be nested: a submenu item can not expand another submenu







Defining Options Menus



- Override method onCreateOptionsMenu()
  - Triggered the first time an Activity’s menu is displayed

```
menu.add (groupId, itemId, order, text)
```

```
public boolean onCreateOptionsMenu(Menu menu) {
   MenuItem mnu1 = menu.add(0, 0, 0, "Item 1”);
        {
            mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.ic_launcher);
        }
   MenuItem mnu2 = menu.add(0, 1, 1, "Item 2”);
   return super.onCreateOptionsMenu(menu);
}
```

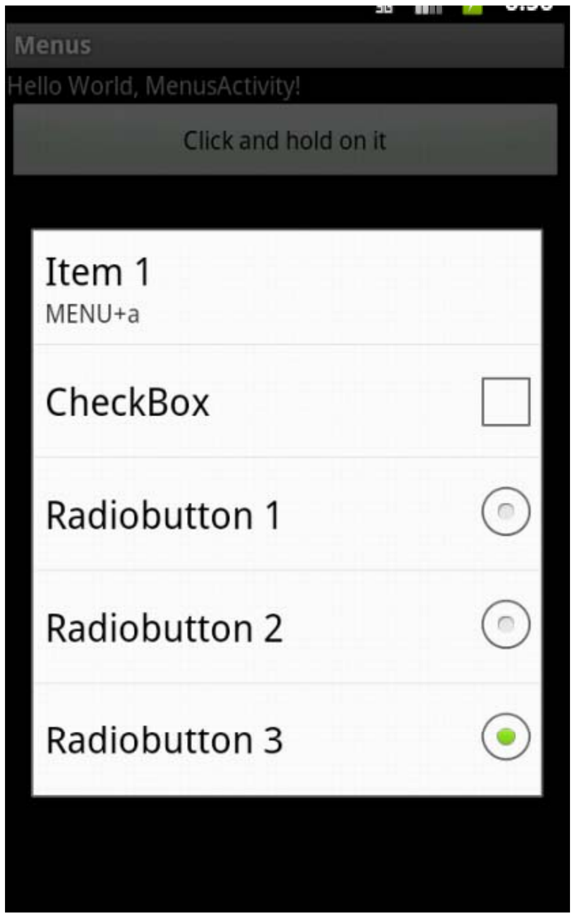
Menu Items

- Shortcut keys

```
menu.setQwertyMode(true);
MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
{
    mnu1.setAlphabeticShortcut('a');
```

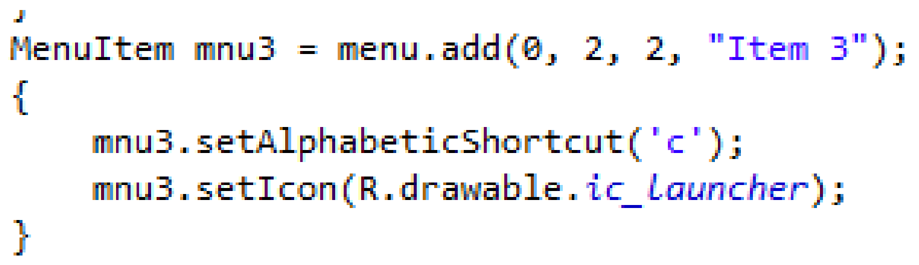
- Checkboxes and radio buttons

```
// Create a new check box item.
menu.add(0, 1, Menu.NONE, "CheckBox").setCheckable(true);
// Create a radio button group.
menu.add(1, 0, Menu.NONE, "Radiobutton 1");
menu.add(1, 1, Menu.NONE, "Radiobutton 2");
menu.add(1, 2, Menu.NONE, "Radiobutton 3").setChecked(true);
menu.setGroupCheckable(1, true, true);
```

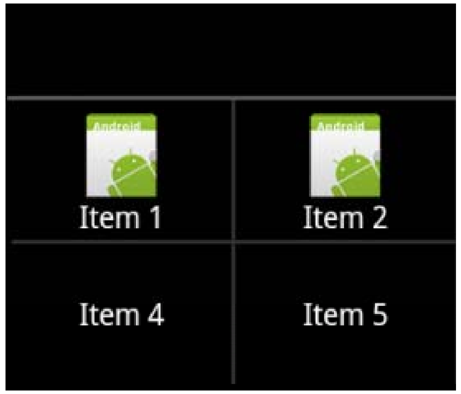


Menu Items

- Icons



```
,
MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
{
    mnu3.setAlphabeticShortcut('c');
    mnu3.setIcon(R.drawable.ic_launcher);
}
```



- Intents
  - An Intent assigned to a Menu Item is triggered when the clicking of a Menu Item isn't handled by Activity's onOptionsItemSelected handler

```
menuItem.setIntent(new Intent(this, MyOtherActivity.class));
```

Updating Menus Dynamically

- Imagine we want to display different menu items in different situations
  - onCreate... is only called the first time

- Method onPrepareOptionsMenu( ) is called immediately before the Menu is displayed

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
  super.onPrepareOptionsMenu(menu);
  MenuItem menuItem = menu.findItem(MENU_ITEM);

  [ ... modify menu items ... ]

  return true;
}
```

Handling Menu Selections

- Method `onOptionsItemSelected()` is called when a menu item is selected
  - Response to menu selection

```
// Return false if you item is not handled
public boolean onOptionsItemSelected(MenuItem item) {
   switch(item.getItemId()) {
      case 0:
            [... Perform action ...]
            return true;
      case 1:
            [... Perform action ...]
            return true;
      default: return false;
   }
}
```

Context Menus

- Define menu

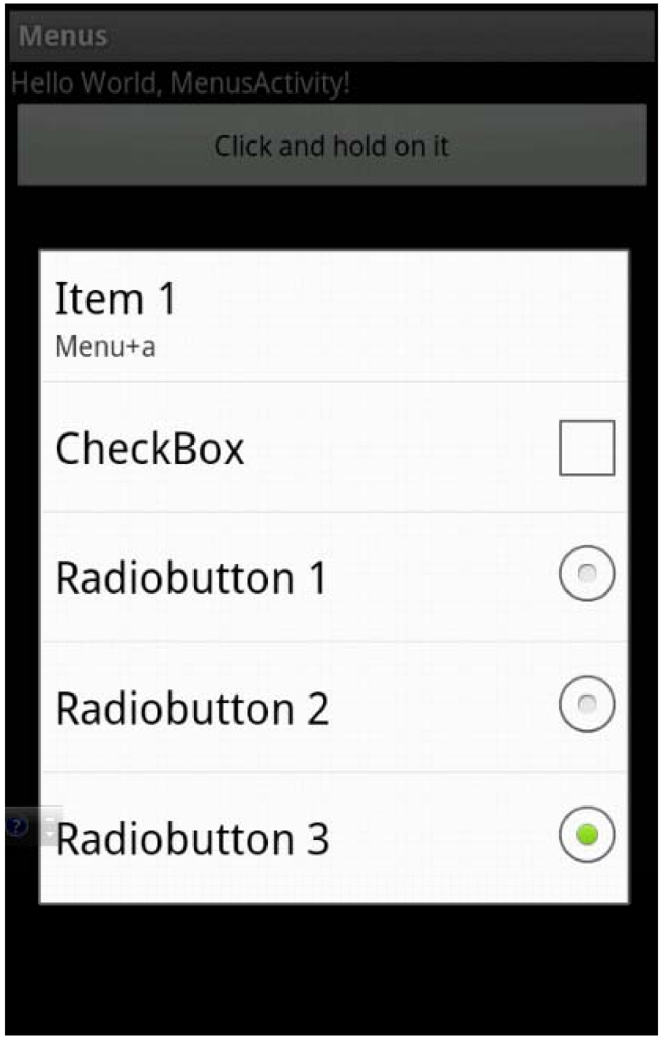
```
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenuInfo menuInfo){
   super.onCreateContextMenu(menu, v, menuInfo);
   menu.setHeaderTitle("Context Menu");
   MenuItem mnu1 = menu.add(0, 0, 0, "Item 1”);
       mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.ic_launcher);
   MenuItem mnu2 = menu.add(0, 1, 1, "Item 2”);
}
```

- Handle menu selections

```
public boolean onContextItemSelected(MenuItem item) {
   switch(item.getItemId()) {
      case 0:
            [... Perform action ...]
            return true;
...
```

Let’s Try… Project Menus

- Import project
- See how
  - Context Menu is created
  - Item selection is handled
- First menu items has a key shortcut (‘a’)
- Radiobuttons belong to a separate Group: Group 1
  - Menu groups may be enabled/disabled:
    `setGroupEnabled()`



Submenus

- Selecting a submenu presents a single floating window that displays all of its Menu Items
  - Android does not support nested submenus

```
SubMenu sub = menu.addSubMenu(0, 0, Menu.NONE, "Submenu");
sub.setHeaderIcon(R.drawable.icon);
sub.setIcon(R.drawable.icon);
MenuItem submenuItem = sub.add(0, 0, Menu.NONE, "Submenu Item");
```

Defining Menus in XML

- Menus can be defined using XML syntax
  - res/menu folder
  - Inflate menu resource using MenuInflater in appropriate onCreate...Menu() methods

```
public boolean onCreateOptionsMenu(Menu menu){
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.my_menu, menu);
    return true;
}
```

  - Handling item selection in appropriate on...ItemsSelected() methods

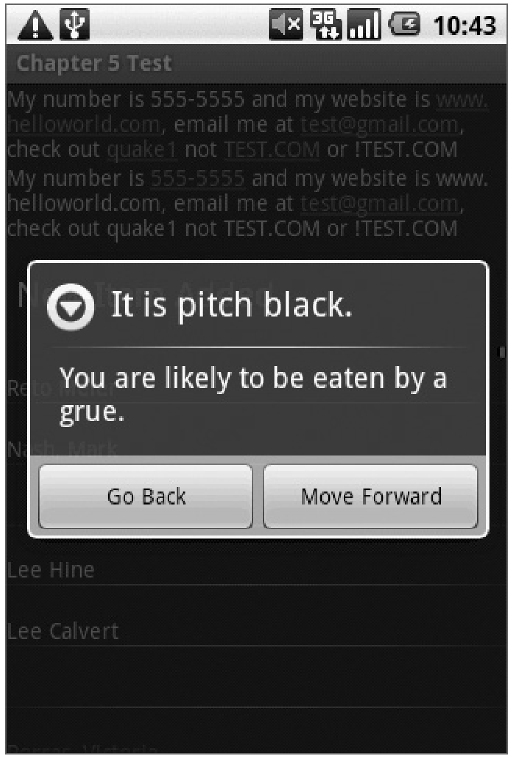
Menu defined in XML

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android"
  android:name="Context Menu">
  <item
    android:id="@+id/item01"
    android:icon="@drawable/menu_item"
    android:title="Item 1">
  </item>
  <item
    android:id="@+id/item02"
    android:checkable="true"
    android:title="Item 2">
  </item>
<item
    android:id="@+id/item04"
    android:title="Submenu">
    <menu>
      <item
        android:id="@+id/item05"
        android:title="Submenu Item">
      </item>
    </menu>
  </item>
</menu>
```

Dialogs

Dialogs

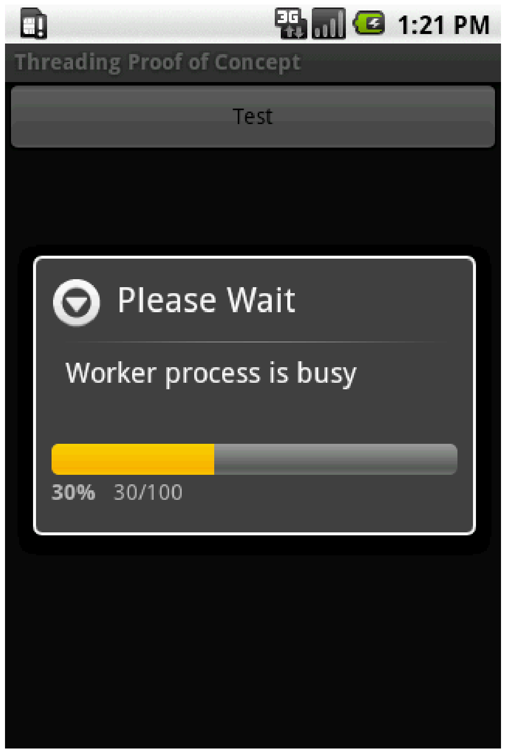
- Dialogs are floating activities
  - Based on Dialog class
- General-purpose dialogs
  - AlertDialog
  - ProgressDialog
  - Toasts


Chapter 5 Test

My number is 555-5555 and my website is www.helloworld.com, email me at test@gmail.com, check out quake1 not TEST.COM or !TEST.COM
My number is 555-5555 and my website is www.helloworld.com, email me at test@gmail.com, check out quake1 not TEST.COM or !TEST.COM

It is pitch black.

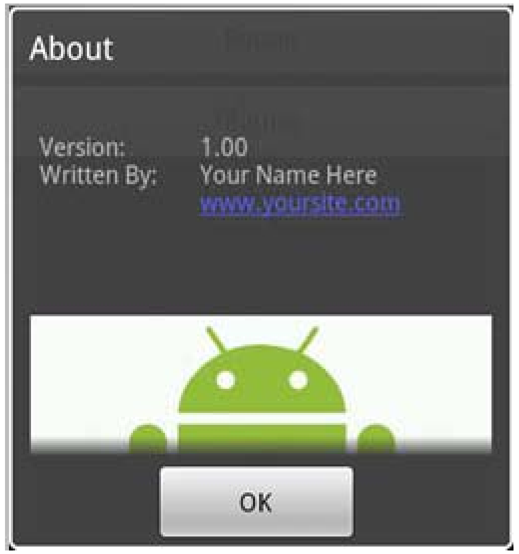
You are likely to be eaten by a grue.


Threading Proof of Concept

Please Wait

Worker process is busy

30% 30/100

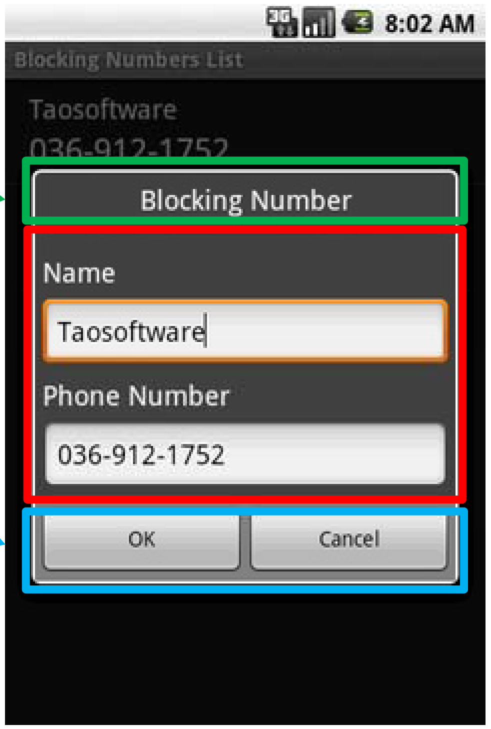

About

Version: 1.00
Written By: Your Name Here
www.yoursite.com

Dialogs

- Dialogs are modal
  ➔ Block all user input and must be dismissed before the user can continue.

- Dialogs share a common structure
  - Optional title region: Introduces content
  - Content area: text, UI elements, text fields, checkboxes, etc…
  - Action Buttons: Typically OK/Cancel, indicating the preferred option.



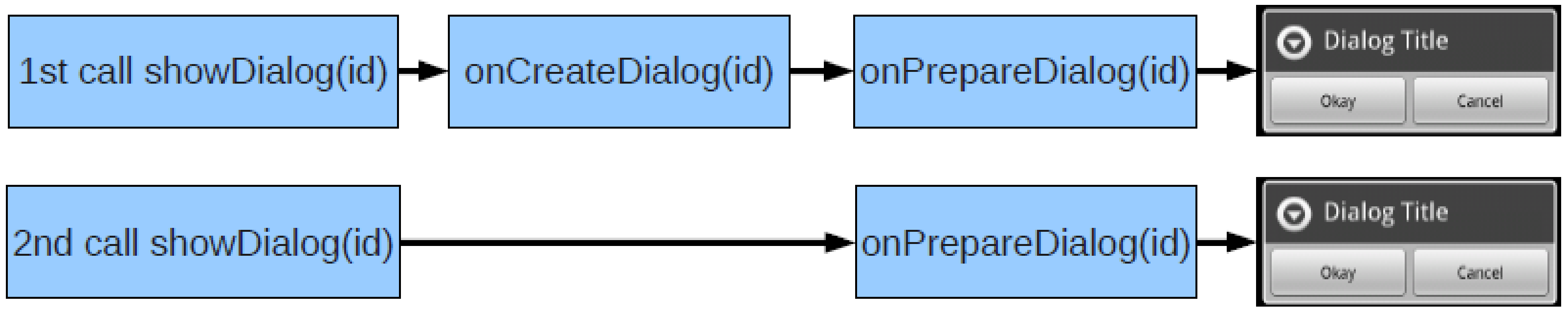
Creating Custom Dialogs

- Create and show… using a reference to a class
  - Dialog view is defined as a layout
  - Straightforward way but…
  - … Dialog is created every time as a new instance

```
Dialog d = new Dialog(MyActivity.this);
// Set the title
d.setTitle("Dialog Title");
// Inflate the layout
d.setContentView(R.layout.dialog_view);
d.show();
```

Creating Custom Dialogs

- To avoid repeated creation and destruction
  - Use showDialog(int)
  - It will call
    - onCreateDialog(int)
    - onPrepareDialog(int)
  - Instance is created once, can be modified whenever is to be shown
  - … same a Menus



Creating Custom Dialogs – onCreateDialog()

- Same method is called for creation of any dialog
  - Switch is used to identify dialog
    - Example: Show time

```
static final private int TIME_DIALOG = 1;
...
showDialog(TIME_DIALOG);
...
@Override
public Dialog onCreateDialog(int id) {
  switch(id) {
    case (TIME_DIALOG) :
      AlertDialog.Builder timeDialog = new AlertDialog.Builder(this);
      timeDialog.setTitle("The Current Time Is...");
      return timeDialog.create();
    }
  return null;
}
```

Creating Custom Dialogs – onPrepareDialog()

- Modify the dialog just before showing it
  - Example: Show time

```
@Override
public void onPrepareDialog(int id, Dialog dialog) {
  switch(id) {
    case (TIME_DIALOG) :
      SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
      Date currentTime = new  Date(java.lang.System.currentTimeMillis());
      String dateString = sdf.format(currentTime);
      AlertDialog timeDialog = (AlertDialog)dialog;
      timeDialog.setMessage(dateString);
      break;
    }
  }
}
```

AlertDialog

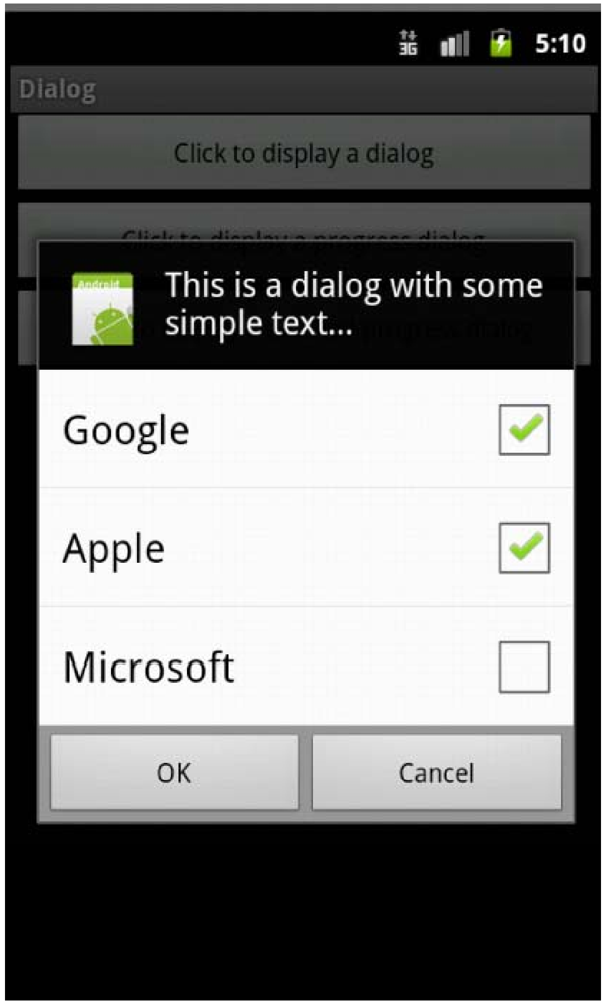
- Allows to
  - Present a message to the user offering them one to three options in the form of buttons (OK/Yes/No/Cancel)
  - List options in the form of checkboxes or radio buttons
  - Provide a text entry box for user input.

```
return new AlertDialog.Builder(this)
.setIcon(R.drawable.icon)
.setTitle("This is a dialog with some simple text...")
.setPositiveButton("OK", new
    DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog,
    int whichButton)
    {
        Toast.makeText(getBaseContext(),
             "OK clicked!", Toast.LENGTH_SHORT).show();
    }
})
.setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
```

Let's Try… Project Dialog

- Import project

- See code
  - See case 0 in onCreateDialog
    - Commented code is equivalent to running code
    - See how function calls may be linked on an object
  - See how dialog buttons are created
  - See how clicks are handled
    - Multioptions
    - Buttons
  - Dialog icon + heading



This is a dialog with some simple text...

Google ✓

Apple ✓

Microsoft

ProgressDialog

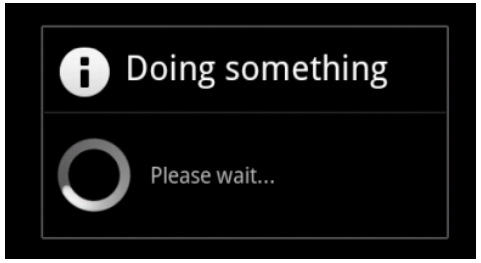
- Show the progress of some activity
  - Example: Status of a download, load a webpage, request information from a server

```
public void onClick2(View v) {
    //---show the dialog---
    final ProgressDialog dialog = ProgressDialog.show(
        this, "Doing something", "Please wait...", true);
    new Thread(new Runnable(){
        public void run(){
            try {
                //---simulate doing something lengthy---
                Thread.sleep(5000);
                //---dismiss the dialog---
                dialog.dismiss();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }).start();
}
```

Start dialog

Sleep for a while

Close dialog

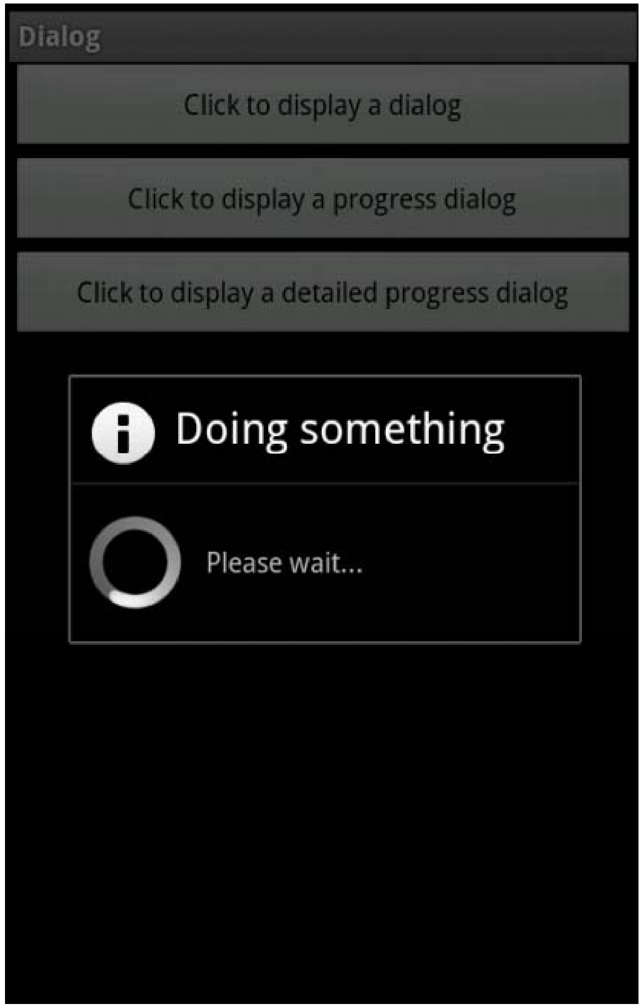

Doing something

Please wait...

Let’s Try… Project Dialog

- Import project

- See code
  - Method `onClick2`
  - Use class `ProgressDialog` and `show`


Dialog
Click to display a dialog
Click to display a progress dialog
Click to display a detailed progress dialog
Doing something
Please wait...

ProgressDialog …. Detailed Progress

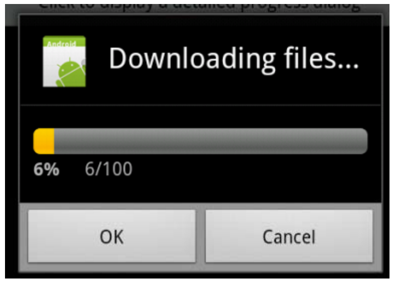
- It is possible to display progress status in the dialog

```
progressDialog.setProgress(0);
```

- Update when required

```
progressDialog.incrementProgressBy((int)(100/15));
```

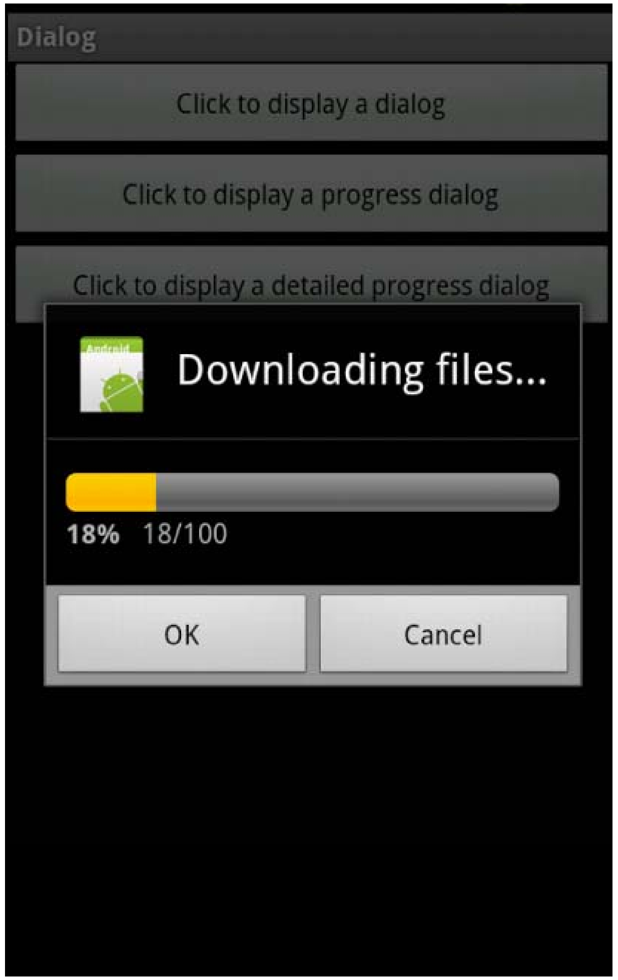
- Note that Dialog is created as before



```
progressDialog = new ProgressDialog(this);
progressDialog.setIcon(R.drawable.ic_launcher);
progressDialog.setTitle("Downloading files...");
progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "OK",
    new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog,
        int whichButton)
```

Let's Try… Project Dialog

- Import project
- See code
  - See case 1 in onCreateDialog
    - Dialog is based now on class ProgressDialog
  - See how progress is updated


Dialog
Click to display a dialog
Click to display a progress dialog
Click to display a detailed progress dialog
Downloading files...
18% 18/100
OK
Cancel

Context

- We have avoided the issue of Context
- A context object provides a reference to some element (A), so that new actions/elements can be executed in reference to it (A)
  - An activity is the reference of a displayed dialog
- When in doubt, use `getBaseContext()` to get the context reference

Android Development

Session 4

Javier Poncela

Contents

Working in the Background

- Some tasks are better performed in the background, in parallel with other tasks

- Android provides several mechanisms
  - Services
  - Background threads
  - AsyncTasks
  - Alarms

Services

Services

- A service is an application in Android that runs in the background without needing to interact with the user
  - Respond to events even when application is not active
  - Executed in the main thread of the application
- Services have a higher priority than inactive Activities, so they're less likely to be killed by the system
  - If killed it will be restarted automatically when resources are available
- Examples of use
  - Update data from/to a server
  - Receive long responses
  - Long calculations
  - Music players
  - …

Running a Service

- Started
  - When it is started by calling `startService()`
  - Usually, a started service performs a single operation and does not return a result to the caller
  - When the operation is done, the service should stop itself or the caller should do it
  - Do not return a result
- Bound
  - When it is started by calling `bindService()`
  - Offers a client-server interface that allows components to interact with the service, get requests, deliver results
  - Runs as long as one application component is bound to it
  - Multiple components can bind to the service at once. Service is destroyed when all of them unbind
- Services may run using both mechanisms

Creating Services

- Extend class Service
- Define methods
  - onCreate
    - When first created
  - onStartCommand (Started)
    - When start request
  - onBind (Bound)
    - When bind request

```
public class MyService extends Service {

  @Override
  public void onCreate() {
    ...
  }

  @Override
  public IBinder onBind(Intent intent) {
    ...
    return null;
  }

  @Override
  public int onStartCommand(Intent intent,
                 int flags, int startId) {
    ...
    return Service.START_STICKY;
  }
}
```

Restarting Services

- Restarting behaviour, when service is terminated by OS, is controlled by `onStartCommand()` returned value
  - START_STICKY: Standard behavior, onStartCommand will be called any time your Service restarts after being terminated by the OS.
    - OS does not redeliver last intent
    - For example, services explicitly started and stopped
  - START_NOT_STICKY: Used for Services that are started to process specific actions or commands. Following termination by the OS, it will restart only if there are pending start calls.
    - For example, services that handle specific requests
  - START_REDELIVER_INTENT: Combination of previous two modes
    - Recreate service with last intent
    - Used to ensure that requested commands are completed

Manifest

- Include a <service> item within the application node

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
   xmlns:android=http://schemas.android.com/apk/res/android
   package="iobm.examples.services">
   <application android:icon="@drawable/icon"
                android:label="@string/app_name">
     <activity android:name=".HelloAndroid"
               android:label="@string/app_name">
       <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER"/>
       </intent-filter>
     </activity>

     <service android:name=".MyService" />

   </application>
</manifest>
```

Services – Starting, Stopping, Passing Data

- Prepare Intent (including data to pass)

```
Intent myIntent = new Intent(MyService.ORDER_PIZZA);
myIntent.putExtra("TOPPING", "Margherita");
```

- Start the Service

```
MyService service= startService(myIntent);
```

- Stop the Service
  - From caller

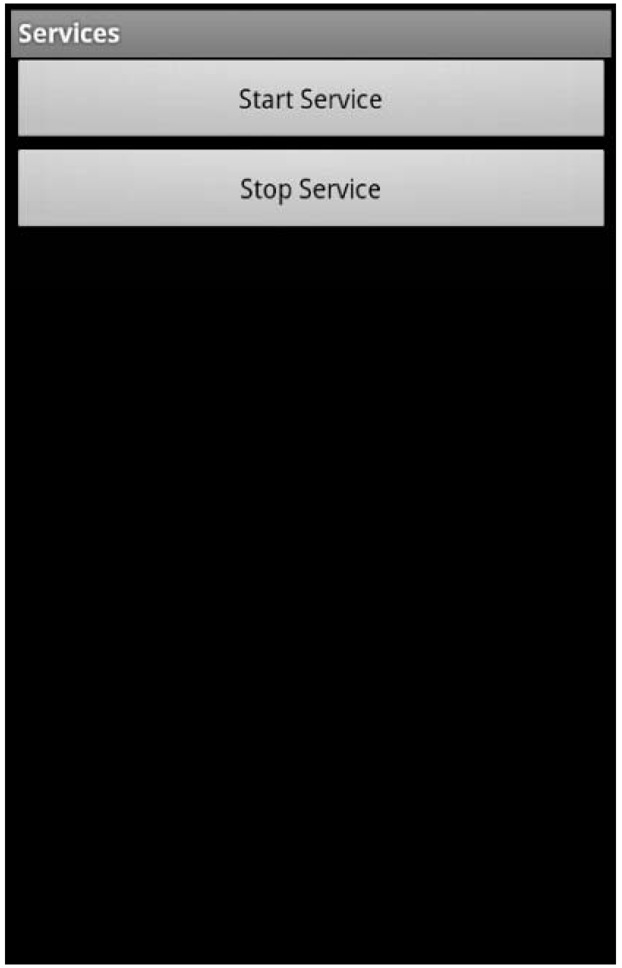
```
stopService(new Intent(this, service.getClassName());
stopService(new Intent(this, MyService);
```

  - From self

```
stopSelf(startId);
```

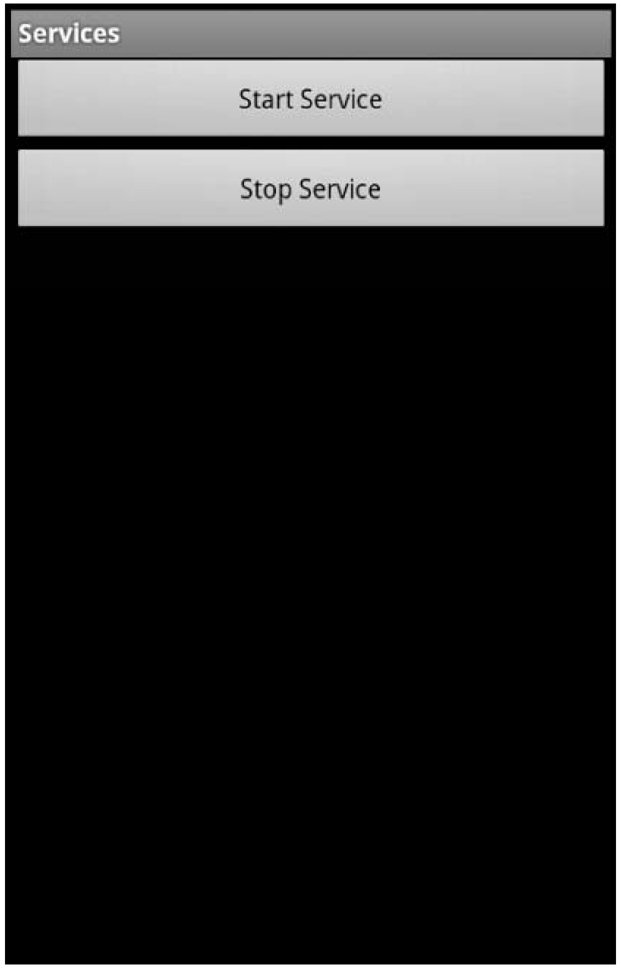
Let´s Try… Project Services

- Import project
- See Manifest file
  - Buttons are linked to functions
  - Service registration
- Uncomment sections /* Start Service */
  - ServicesActivity.java
    - Function startService
  - MyService.java
    - Function onStartCommand



Let´s Try… Project Services

- See code ‘ServicesActivity’
  - Call to Service
  - Passing data (urls)
- See code ‘MyService’
  - Retrieving data (urls)
  - Result on screen
- My click ‘Back’ and service is still running
  - Toasts appear on screen



Binding Services

- Service must implement onBind()

```
private final IBinder binder = new MyBinder();

@Override
public IBinder onBind(Intent intent) {
  return binder;
}

public class MyBinder extends Binder {
  MyService getService() {
    return MyService.this;
  }
}
```

- Return a reference to class Binder (or extended class)

Binding Services

- Activity must be connected to the Service
  - ServiceConnection

```
// Reference to the service
private MyService serviceBinder;

// Handles the connection between the service and activity
private ServiceConnection mConnection = new ServiceConnection() {
  public void onServiceConnected(ComponentName className, IBinder
                                                                  service) {
    // Called when the connection is made.
    serviceBinder = ((MyService.MyBinder)service).getService();
  }

  public void onServiceDisconnected(ComponentName className) {
    // Received when the service unexpectedly disconnects.
    serviceBinder = null;
  }
};
```

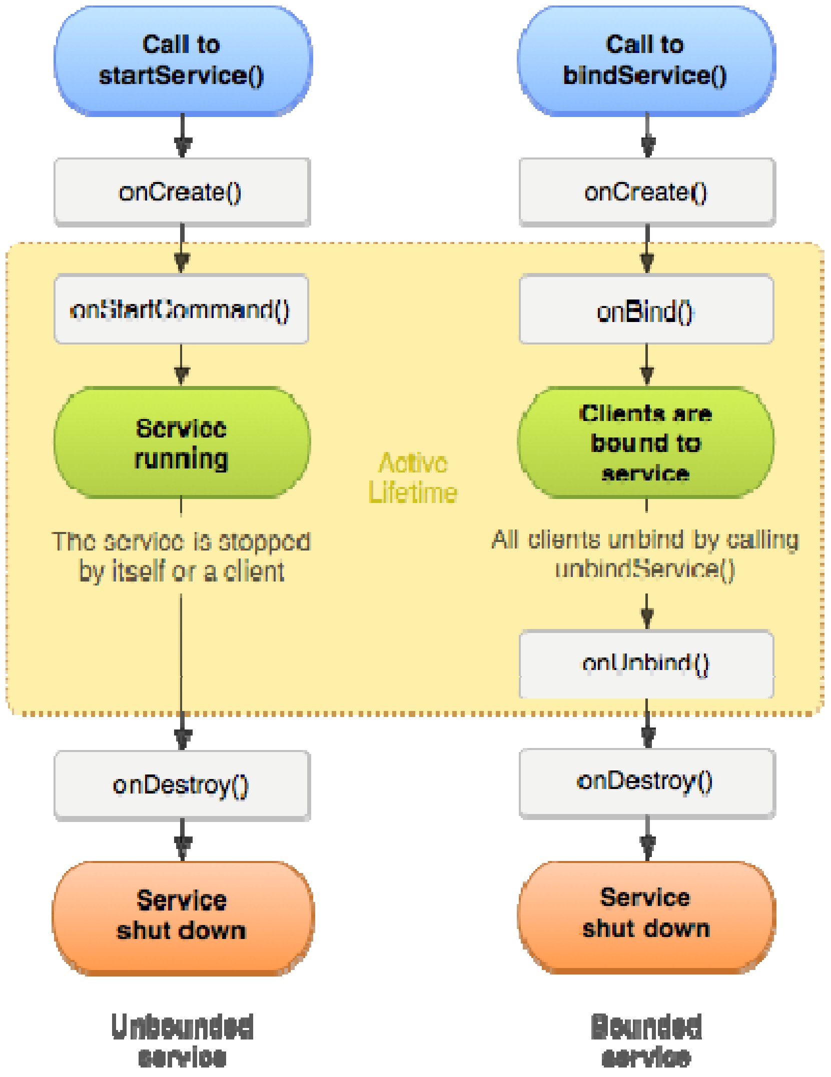
Binding Services

- Bind the Service

```
Intent bindIntent = new Intent(MyActivity.this, MyService.class);
bindService(bindIntent, mConnection, Context.BIND_AUTO_CREATE);
```

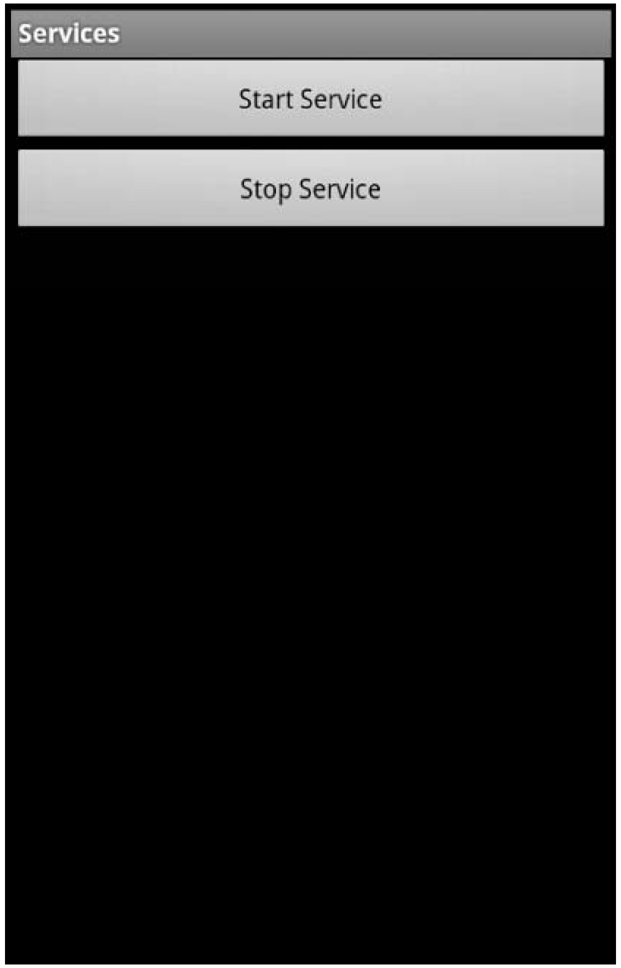
- Once the Service has been bound, all its public methods and properties are available through the serviceBinder object obtained from the onServiceConnected handler.

Bound Services – Lifecycle



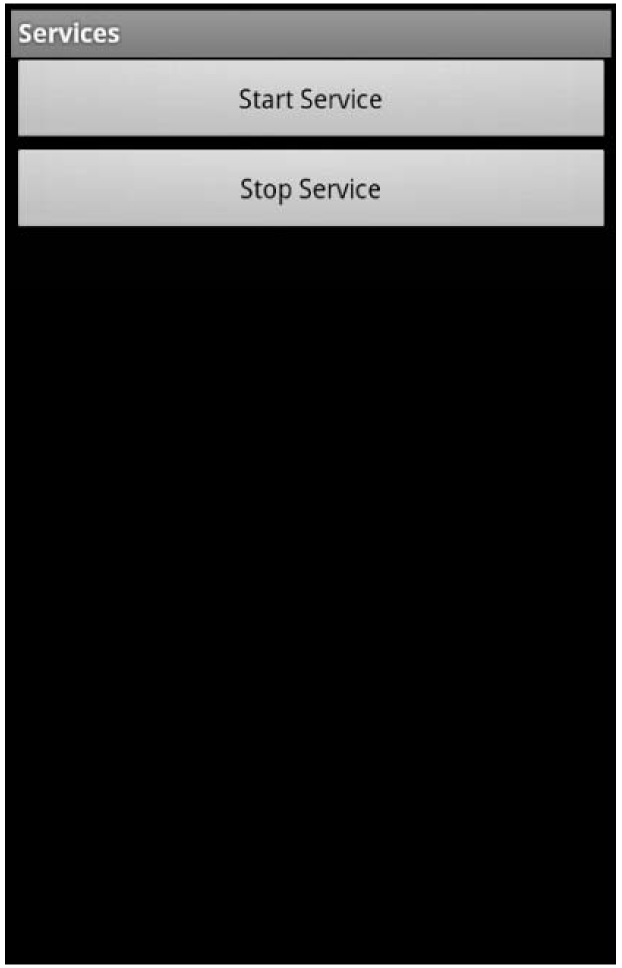
Let´s Try… Project Services

- Import project
- Uncomment sections /* Bind Service */
  - Re-comment section /* Start Service */
  - ServicesActivity.java
    - Function startService
  - MyService.java
    - Function onStartCommand



Let´s Try… Project Services

- See code ‘ServicesActivity’
  - Call to Service
  - ServiceConnection functions
- See code ‘MyService’
  - Retrieving data (urls)
  - Result on screen
- Click ‘Back’ and service is still running
  - Toasts appear on screen



AsyncTasks

Asynchronous Tasks

- To ensure that your applications remain responsive, it’s good practice to move all slow, time-consuming operations off the main application thread and onto a child thread

- Alternatives
  - Use your own Threads, Handlers and synchronization with GUI
  - AsyncTask: provides all above

Using AsyncTask

- Create a new class extending AsyncTask

```
AsyncTask<[Input Parameter Type...],
          [Progress Report Type...],
          [Result Type]>
```

- doInBackground: Task to be done. Will be executed on the background thread `[Input Parameter Type...]`
  - Use publishProgress method to post progress updates to the UI
  - Return the final result for onPostExecute
- onProgressUpdate: Post interim updates to the UI
  - Can safely modify UI elements `[Progress Report Type...]`
- onPostExecute: When doInBackground has completed, the return value is passed in `[Result Type]`
  - Update the UI once the asynchronous task has completed

Running an AsyncTask

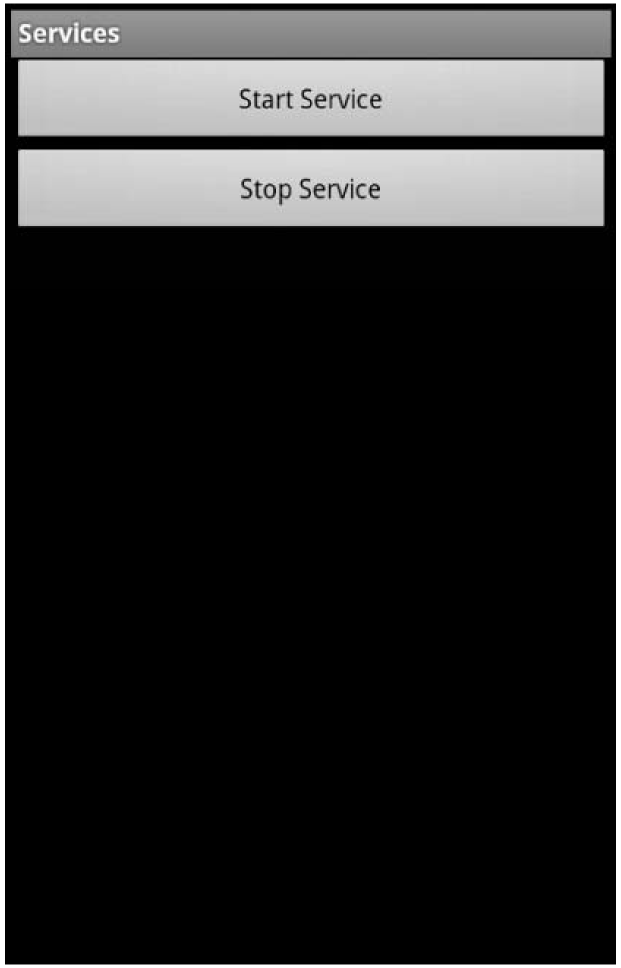
- When *execute* is called, Android does the following:
    1. runs *onPreExecute* in the main (UI) thread
    2. runs *doInBackground* in a background thread
    3. runs *onPostExecute* in the main (UI) thread

```
public void onButtonClick(View view) {
  new BackgroundTask().execute(editText.getText().toString());
}

class BackgroundTask extends AsyncTask<String, Void, String> {
  protected String doInBackground(String... inputs) {
    String reply = // do some background stuff...
    return reply; // this gets sent to onPostExecute
  }

  protected void onPostExecute(String result) {
    tv = (TextView)findViewById(R.id.display_view);
    tv.setText(result);
  }
}
```

Let´s Try… Project Services

- Import project

- See code ‘MyService’
  - Class DoBackgroundTask
    - Internal methods
  - How progress is reported
  - How progress is visualized
  - How finalization is indicated



Broadcast Receivers

Broadcast Receivers

- Android allows broadcasting events
  - Battery low, networks available, new data received, incoming phone call, received text message, ...
- Broadcast receivers are components that respond to such events
  - Background processes
- Events are broadcast as Intents: Broadcast Intents
- Broadcast Receivers must be registered (Manifest) indicating (`<intent-filter>`) which Intents it will listen for

Broadcast Receivers

- Creating a broadcast receiver
  - Executed when a broadcast Intent matches the Intent Filter
  - Applications don’t have to be running. They will be started automatically

```
public class MyBroadcastReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    //TODO: React to the Intent received.
  }
}
```

- Register a broadcast receiver (Manifest)
  - Includes action (event) listening for

```
<receiver android:name=".LifeformDetectedBroadcastReceiver">
  <intent-filter>
    <action android:name="com.paad.action.NEW_LIFEFORM"/>
  </intent-filter>
</receiver>
```

Broadcast Intents

- Construct the Intent and broadcast it (sendBroadcast)
  - Set the action, data and category

```
Intent intent = new Intent("com.paad.action.NEW_LIFEFORM");
intent.putExtra("lifeformName", lifeformType);
intent.putExtra("longitude", currentLongitude);
intent.putExtra("latitude", currentLatitude);
sendBroadcast(intent);
```

Native Android Broadcast Actions

- Android broadcasts Intents for many system Services

  http://developer.android.com/reference/android/content/Intent.html

Action	Description
ACTION_BOOT_COMPLETED	Sent when the platform completes booting
ACTION_TIME_TICK	Sent every minute to indicate that time is ticking
ACTION_TIME_CHANGED	Sent when the user changes the time
ACTION_TIMEZONE_CHANGED	Sent when the user changes the time zone
ACTION_PACKAGE_ADDED	Sent when a package is added to the platform
ACTION_PACKAGE_REMOVED	Sent when a package is removed from the platform
ACTION_BATTERY_CHANGED	Sent when the battery charge level or charging state changes
ACTION_CAMERA_BUTTON	Fired when the camera button is clicked
ACTION_MEDIA_EJECT	If the user chooses to eject the external storage media, this event is fired first
ACTION_NEW_OUTGOING_CALL	Broadcast when a new outgoing call is about to be placed. Intercept outgoing calls.

Ordered and Sticky Broadcast Intents

- Ordered broadcasts
  - Using sendBroadcast, the Intent will be received by all registered Broadcast Receivers
    - Cannot control the order and cannot propagate results
  - When order in which the Broadcast Receivers receive the Intent is important, use sendOrderedBroadcast
    - sendOrderedBroadcast(intent, null);
    - Intent will be delivered to all registered Receivers in order of priority.

- Sticky broadcasts
  - Remain in the system until explicitly removed
  - New broadcast receivers registrations will receive the event
    - sendStickyBroadcast(intent);
    - removeStickyBroadcast(intent);

Handling Ordered Broadcasts

- Passing results

```
public void onReceive(Context context, Intent intent) {
  ...
  tmp = getResultData() != null ? getResultData() : "";
  setResultData(tmp + ":Receiver 1:");
}
```

- Aborting propagation

```
public void onReceive(Context context, Intent intent) {
  ...
  if (condition && isOrderedBroadcast()) {
    abortBroadcast();
  }
  ...
}
```

Handling Sticky Broadcast

- `isInitialStickyBroadcast` returns true if the receiver is currently processing the held (cached) broadcast event

```
public void onReceive(Context context, Intent intent) {
  if (intent.getAction().equals(Intent.ACTION_BATTERY_CHANGED)) {
    String age = "Reading taken recently";
    if (isInitialStickyBroadcast()) {
      age = "Reading may be stale”;
    }
    state.setText("Current Battery Level" + String.valueOf(
               intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1))
               + "\n" + age);
}
```

Registering Broadcast Receivers

- Static registration
  - In Manifest
  - Register at boot time or when package is added

- Dynamic registration
  - At runtime
    - `registerReceiver, unregisterReceiver`
    - `IntentFilter`

```
public void onCreate(BundlesavedInstanceState) {
  ...
  registerReceiver(newReceiver1(),
              new IntentFilter(“MyPackage.CustomBroadIntent”));
  ...
  unregisterReceiver(newReceiver1());
}
```

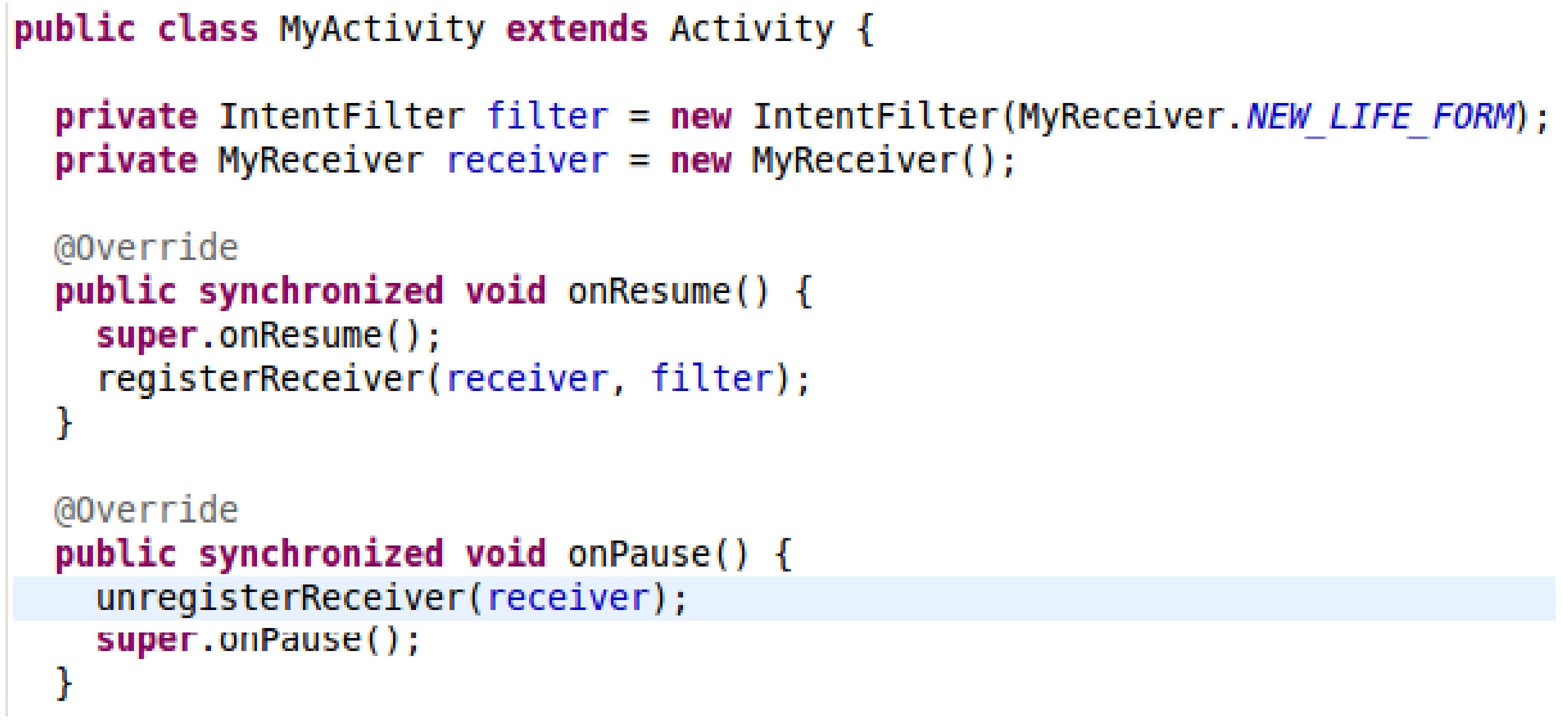
Some More Comments on Event Handling

- onReceive() should be short-lived (< 10s)
  - Application must remain responsive
  - If event handling take a long time, consider starting a Service, rather than performing complete operation in onReceive()
- BroadcastReceivers can't start asynchronous operations
  - e.g., showing a dialog, binding to a Service, starting an Activity via startActivityForResult
  - May happen receiver is no longer in memory when response is received
- Debugging tip: Log BroadcastReceivers that match an Intent

```
Intent.setFlag(FLAG_DEBUG_LOG_RESOLUTION)
```

Saving Resources with Broadcast Receivers

- Register the receiver when the activity is visible and unregister when not



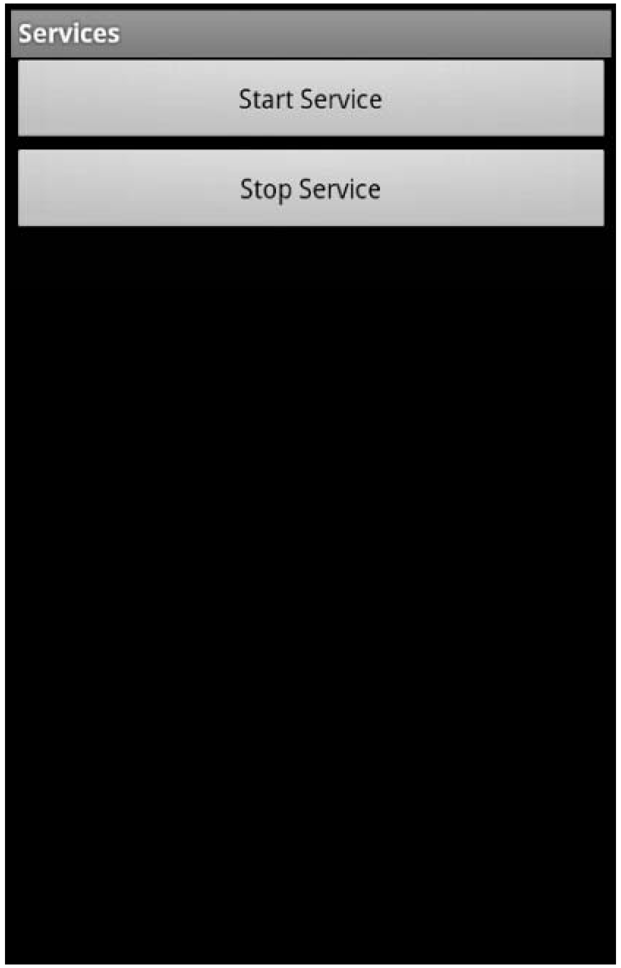
```java
public class MyActivity extends Activity {

  private IntentFilter filter = new IntentFilter(MyReceiver.NEW_LIFE_FORM);
  private MyReceiver receiver = new MyReceiver();

  @Override
  public synchronized void onResume() {
    super.onResume();
    registerReceiver(receiver, filter);
  }

  @Override
  public synchronized void onPause() {
    unregisterReceiver(receiver);
    super.onPause();
  }
```

Let´s Try… Project Services

- Import project

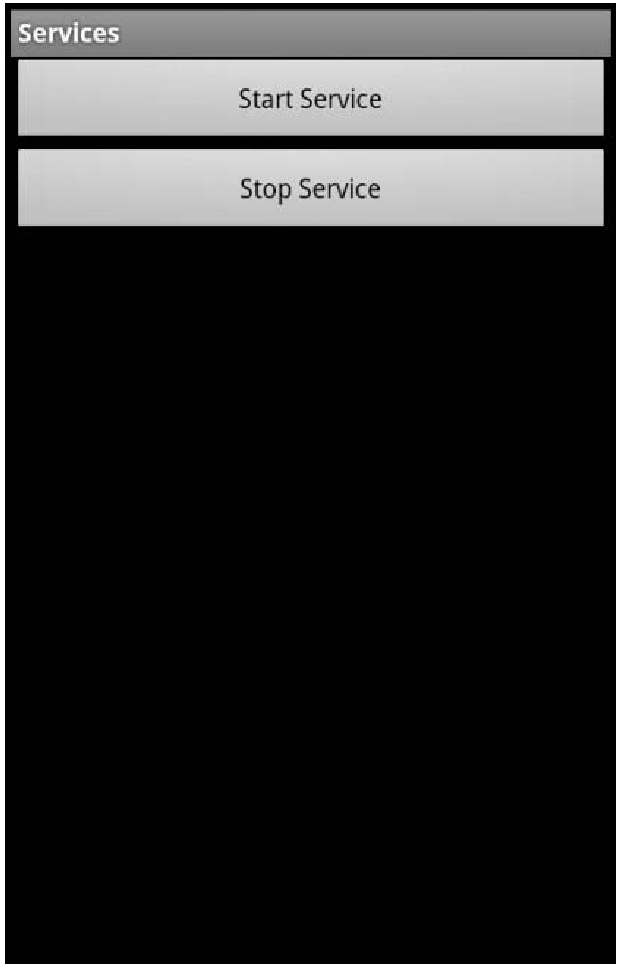
- Uncomment sections
  /* Broadcast Receiver */
  - Re-comment sections /* Start Service */ and /* Bind Service */
  - ServicesActivity.java
    - Function startService

- See Manifest
  - Service registration



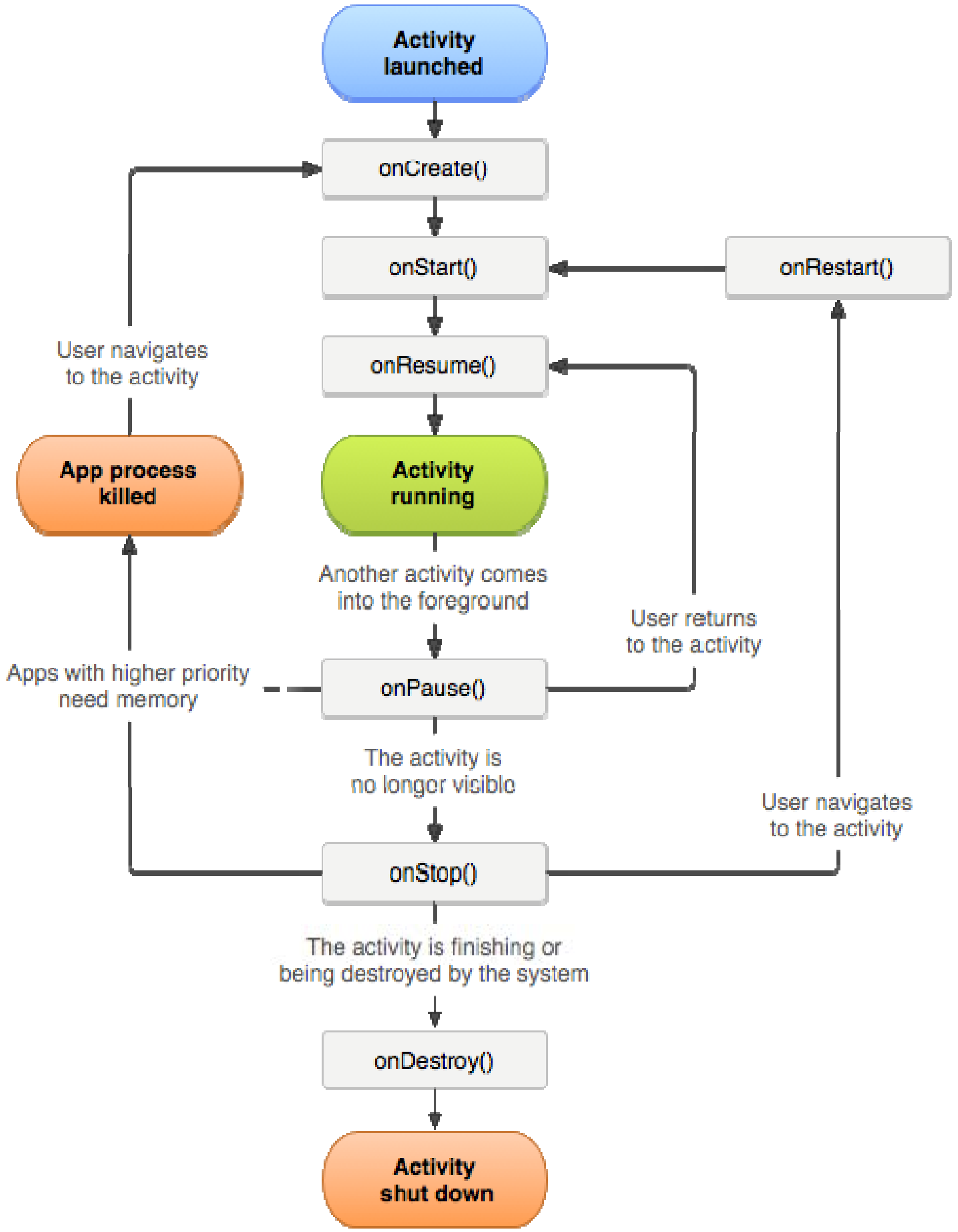
Let´s Try… Project Services

- See code ‘ServicesActivity’
  - Call to Service
  - IntentService is a service that creates a worker thread
  - BroadcastReceiver
    - onReceive
  - onResume
    - Registers the receiver
  - onPause
- See code ‘MyIntentService’
  - onHandleIntent
  - Log start and end of download
    - See log output
  - setAction, sendBroadcast



Activity Lifecycle

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()
- onRestart()



Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Services Revisited – IntentService

- Handles asynchronous requests on demand on a single background thread
  - It isn't affected by most user interface lifecycle events
- Limitations
  - Can't interact directly with your user interface
    - To update UI, send results to Activity via Broadcast Receiver
  - Work requests run sequentially
  - An operation can't be interrupted
- Provides default implementation of
  - onBind(): returns null
  - onStartCommand(): sends intent to work queue
- Requires to implement
  - Constructor
  - onHandleIntent(): It is run automatically on a thread

IntentService

- Create service

```
public class RSSPullService extends IntentService {
    protected void onHandleIntent(Intent workIntent) {
        String dataString = workIntent.getDataString();
        // Do work here, based on the contents of dataString
        ...
    }
}
```

- Call service

```
mServiceIntent = new Intent(getActivity(), RSSPullService.class);
mServiceIntent.setData(Uri.parse(dataUrl));
getActivity().startService(mServiceIntent);
```

- Receive result/update

```
Intent localIntent =
  new Intent(Constants.BROADCAST_ACTION)
                 .putExtra(Constants.EXTENDED_DATA_STATUS, status);
LocalBroadcastManager.getInstance(this).sendBroadcast(localIntent);
```

Android Development

Session 5

Javier Poncela

Contents

User Preferences

User Preferences

- Android provides the Shared Preferences mechanism to help save simple application data
  - Application settings: color, font size, URL, contact person, user name, password (hidden), ...
- Saving to a file may be cumbersome... or to a database... use the in-built preferences mechanism
  - Save pairs of key-value
- Android provides a system of permissions to access the stored preferences
  - Private
  - World Readable
  - World Writable

<u>Limited data types</u>

String, boolean, float, int, and long

Working with SharedPreferences

- Follow these steps
  - Get a handle to the SharedPreference object (file ‘name’)

```
prefsPrivate =
       getSharedPreferences(String name, int accessMode)
```

  - Get the editor

```
prefsPrivateEditor = prefsPrivate.edit()
```

  - Store values

```
prefsPrivateEditor.putString(
                      SharedPrefTestInput.KEY_PRIVATE,
                      inputPrivate.getText.toString());
```

  - Commit changes

```
prefsPrivateEditor.commit();
```

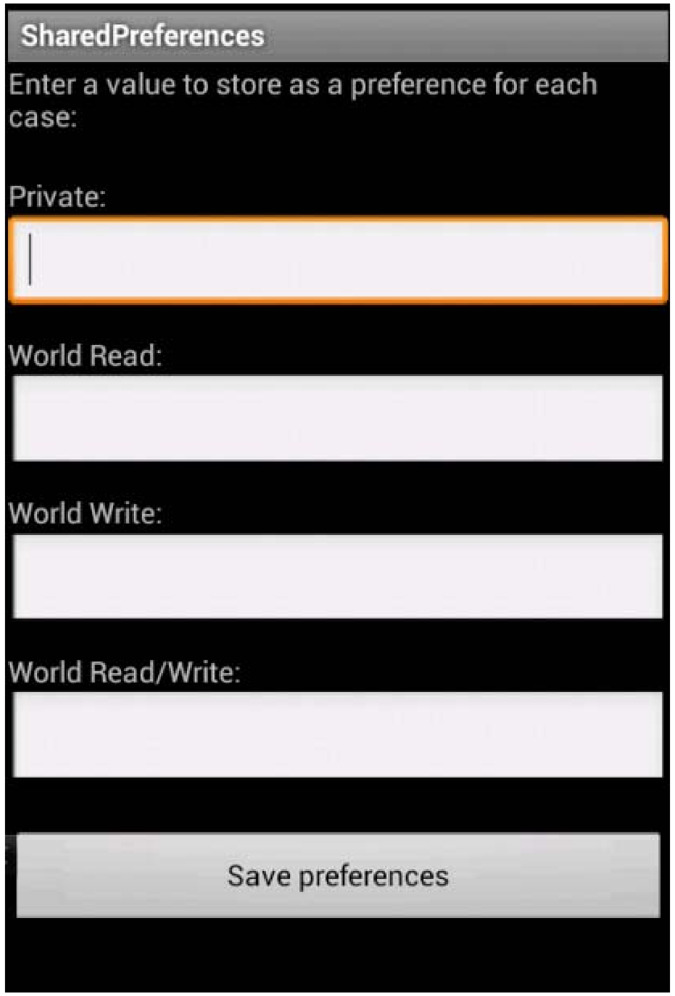
Let’s Try… Project SharedPreferences

- Import the project

- 4 permission modes
  - 3 basic + 1 combined (R/W)

- See code SharedPrefTestInput
  - onCreate
    - Strings for preferences
    - Variables
    - Preference setting process

- See code SharedPrefTestOutput
  - How to retrieve stored preferences


SharedPreferences

Enter a value to store as a preference for each case:

Private:

World Read:

World Write:

World Read/Write:

Retrieving Shared Preferences

- From the same application
  - All preferences are accessible, independent of its permission

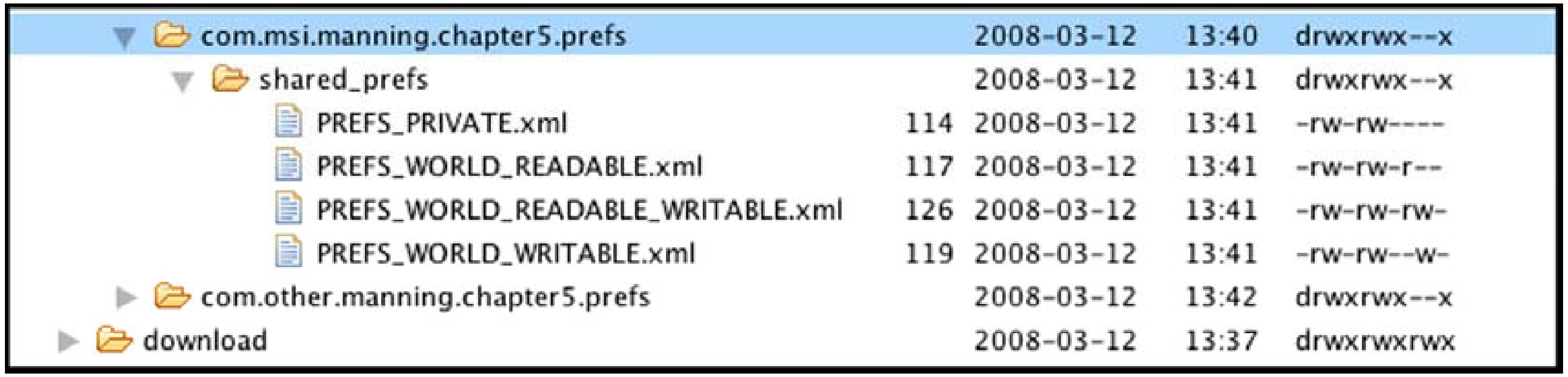
- From other application
  - Access depends on permission
    - Private: No
    - World_Readable: Only reading
    - World_Writable: Only writing

- How to retrieve a preference
  - ‘default’: returned if preference not found/accessible

```
getString(String key, String default)
```

Where Are Preferences Stored?

- Preferences are stored as XML files in the /data/data/PACKAGE_NAME/ shared_prefs path
  - Every application or package has its own user ID
  - Linux like access permissions



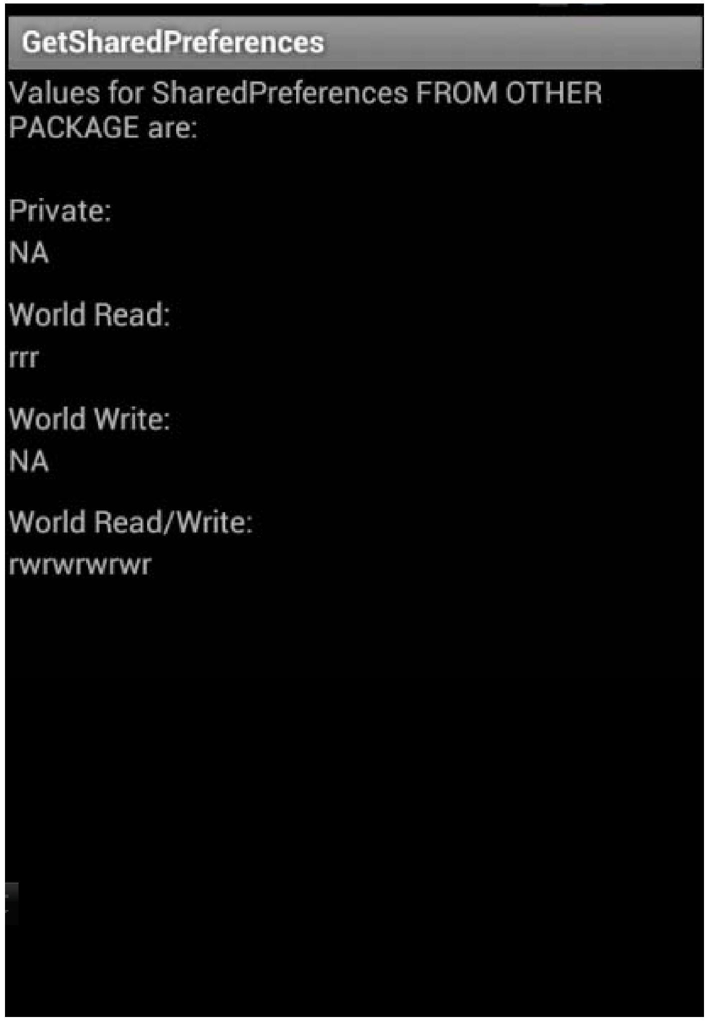
```
▼ com.msi.manning.chapter5.prefs                         2008-03-12  13:40  drwxrwx--x
  ▼ shared_prefs                                         2008-03-12  13:41  drwxrwx--x
      PREFS_PRIVATE.xml                             114  2008-03-12  13:41  -rw-rw----
      PREFS_WORLD_READABLE.xml                      117  2008-03-12  13:41  -rw-rw-r--
      PREFS_WORLD_READABLE_WRITABLE.xml             126  2008-03-12  13:41  -rw-rw-rw-
      PREFS_WORLD_WRITABLE.xml                      119  2008-03-12  13:41  -rw-rw--w-
▶ com.other.manning.chapter5.prefs                       2008-03-12  13:42  drwxrwx--x
▶ download                                               2008-03-12  13:37  drwxrwxrwx
```

- Tricky part: The path is built from the Context. So, to get files from another application you have to know and use that application's Context

```
createPackageContext(“package.example.prefs",
                             Context.MODE_WORLD_WRITEABLE);
```

Let’s Try… Project GetSharedPreferences

- Import the project
- See code SharedPrefTestOtherOutput
  - onStart
    - Gets context
    - Retrieves preferences
  - Compare output with previous example



```
GetSharedPreferences
Values for SharedPreferences FROM OTHER
PACKAGE are:

Private:
NA
World Read:
rrr
World Write:
NA
World Read/Write:
rwrwrwrwr
```

File Storage

Using Files

- DISCLAIMER: File Storage is very much Java-based

- Creating files
  - File will be stored at "data/data/[PACKAGE_NAME]/files/file.name"

```
fos = openFileOutput("filename.txt", Context.MODE_PRIVATE);
fos.write(createInput.getText().toString().getBytes());
```

- Reading files

```
fis = this.openFileInput("filename.txt");
byte[] reader = new byte[fis.available()];
while (fis.read(reader) != -1) {}
this.readOutput.setText(new String(reader));
```

Accessing Resource Files

- Files as raw (static) resources (res/raw)
  - For example, configuration file at compile time

```
Resources resources = this.getResources();
InputStream is = null;
is = resources.openRawResource(R.raw.people);
byte[] reader = new byte[is.available()];
while (is.read(reader) != -1) {}
this.readOutput.setText(new String(reader));
```

- XML file resources (res/xml)

```
XmlPullParser parser =
this.getResources().getXml(R.xml.people);
StringBuffer sb = new StringBuffer();
try {
  while (parser.next() != XmlPullParser.END_DOCUMENT) {
    ...
```

External Storage

- SD card
  - Standard java.io.File and related objects can be used to create, read and remove files on the /sdcard path

```
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() + “/MyFiles”);
File file = new File(directory, “textfile.txt”);
FileInputStream fIn = new FileInputStream(file);
InputStreamReader isr = new InputStreamReader(fIn);
```

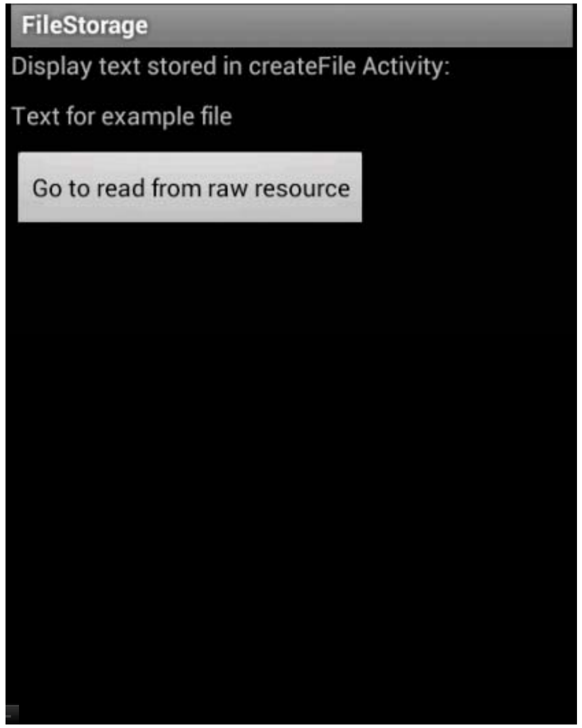
  - Manifest

```
<uses-permission
   android:name=”android.permission.WRITE_EXTERNAL_STORAGE”>
</uses-permission>
```

Let’s Try… Project FileStorage

- Import the project

- Several activities to demonstrate filesystem access
  - Begin with first activity
  - Go up to the last one



Display text stored in createFile Activity:

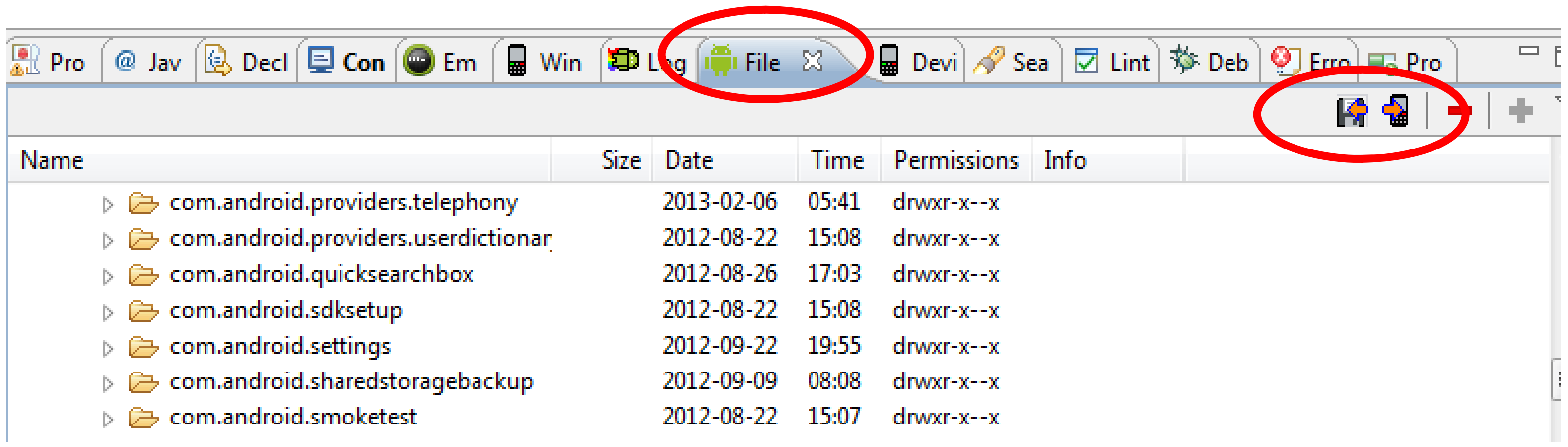
Text for example file

- XML file read

```
<people>
<person firstname="John" lastname="Ford" />
<person firstname="Alfred" lastname="Hitchcock" />
<person firstname="Stanley" lastname="Kubrick" />
<person firstname="Wes" lastname="Anderson" />
</people>
```

Eclipse File Explorer

- Show file system in device
- Transfer files to/from device



Name	Size	Date	Time	Permissions	Info
com.android.providers.telephony		2013-02-06	05:41	drwxr-x--x	
com.android.providers.userdictionar		2012-08-22	15:08	drwxr-x--x	
com.android.quicksearchbox		2012-08-26	17:03	drwxr-x--x	
com.android.sdksetup		2012-08-22	15:08	drwxr-x--x	
com.android.settings		2012-09-22	19:55	drwxr-x--x	
com.android.sharedstoragebackup		2012-09-09	08:08	drwxr-x--x	
com.android.smoketest		2012-08-22	15:07	drwxr-x--x	

Databases

Databases

- For saving relational data, a database is much more efficient
  - For example, contact information
- Android uses the SQLite database system
- The database created for an application is only accessible to itself
  - Other applications will not be able to access it
- The SQLite database created programmatically is stored in the /data/data/<package_name>/databases folder

Creating the DBAdapter Helper Class

- Encapsulate the complexities of accessing the data
  - Makes access transparent to the calling code
  - If the database changes, no need to update code

_id	name	email
15	Ritoo	rkhan@edu.pk
27	Umair	uahmed@edu.pk
12	Kumar	kumar@edu.pk

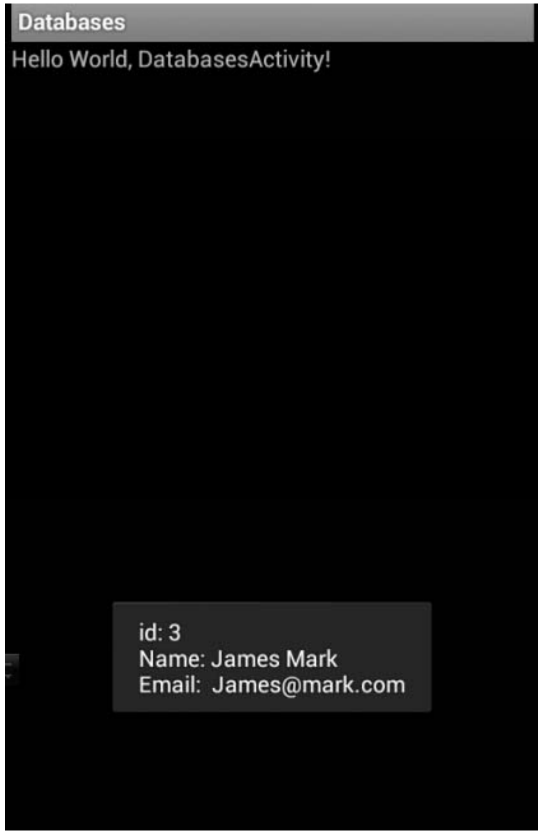
- Steps
  - Define constants for each field of the database
  - Create class DatabaseHelper extending SQLiteOpenHelper
    - Override onCreate and onUpdate
  - Define to
    - Open, close, insert contact, delete contact, get contact, get all contacts, update contact

Using the Database

```
db.open();

// Insert
id = db.insertContact(“Aamir Ahmed”, “ahmed@edu.pk”);

// Retrieve
Cursor c = db.getContact(2);
if (c.moveToFirst())
  DisplayContact(c);

// Update
db.updateContact(1, “Aamir Ahmed”, “ahmed@edu.pk”);

// Delete
db.deleteContact(1)

db.close();
```

Let's Try… Project Databases

- Import the project
- Run it once
- See code Databases
  - onCreate
    - Some blocks are commented
    - Uncomment and set parameters at will
    - Run again
- See code Adapter
  - Includes the DatabaseHelper
  - Includes functions to query and modify the database


Databases

Hello World, DatabasesActivity!

id: 3
Name: James Mark
Email: James@mark.com

Android Development

Session 6

Javier Poncela

Contents

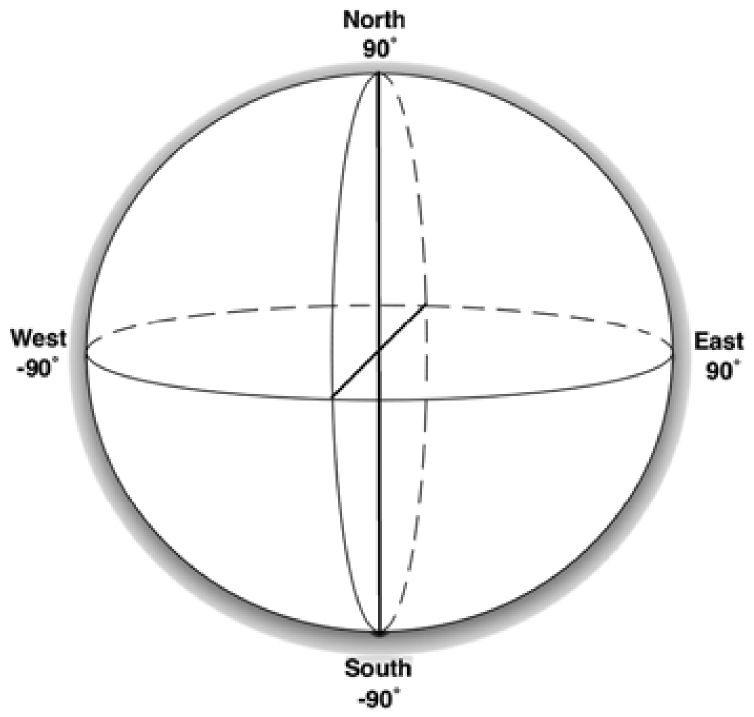
Mobility, Location, Mapping

- Mobile applications can benefit from being location-aware, e.g.,
  - Routes from current to desired location
  - Searching for stores near a current location
  - Displaying info on nearby resources

- Android allows applications to determine & manipulate location
  - Define map-based Activities using Google Maps
  - Use Overlays you can annotate maps and handle user input to contextualize map display
  - Use Location-Based Services (LBS) to
    - Pinpoint geographic locations
    - Convert between coordinates and real-world addresses

Location Representation

- A location represents a position on Earth
- A Location instance consists of:
  - Latitude, longitude, a UTC timestamp
  - Optionally, altitude, speed, and bearing



Location Based Services (LBS)

Using Location Based Services

- LBS is an umbrella term used to describe the different technologies used to find a device's current location

- The two main LBS elements are
  - Location Manager: Provides hooks to LBS services
  - Location Providers: Location-finding technology

- Using the Location Manager it is possible to
  - Obtain current location
  - Track movement
  - Set proximity alerts for detecting movement into and out of a specified area
  - Find available Location Providers

Location Providers

- Multitude of location sources
  - GPS
    - Most accurate: 1-5 meters
    - Only works outdoors
    - Quickly consumes battery power
  - Cell-ID
    - Determine user location according to connected cell phone tower
    - Very low accuracy due to large coverage area of cell tower
    - Requires no extra battery power
  - WiFi
    - Determine user location according to recorded location of WiFi APs
    - Only available if Google has recorded WiFi location
    - Accuracy: ~60 m

Selecting a Location Provider

- Find available providers

```
List<String> providers – locationManager.getProviders(true);
```

  - LocationManager.GPS_PROVIDER, LocationManager.NETWORK_PROVIDER

- Find providers using criteria

```
String bestProvider;

Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);

bestProvider = locationManager.getBestProvider(criteria, true);
List<String> matchingProviders =
                   locationManager.getProviders(criteria, true);
```

Finding the Location

- Access the Location Manager and get and instance of the Service

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(serviceString);
```

- Set appropriate permissions in the Manifest file

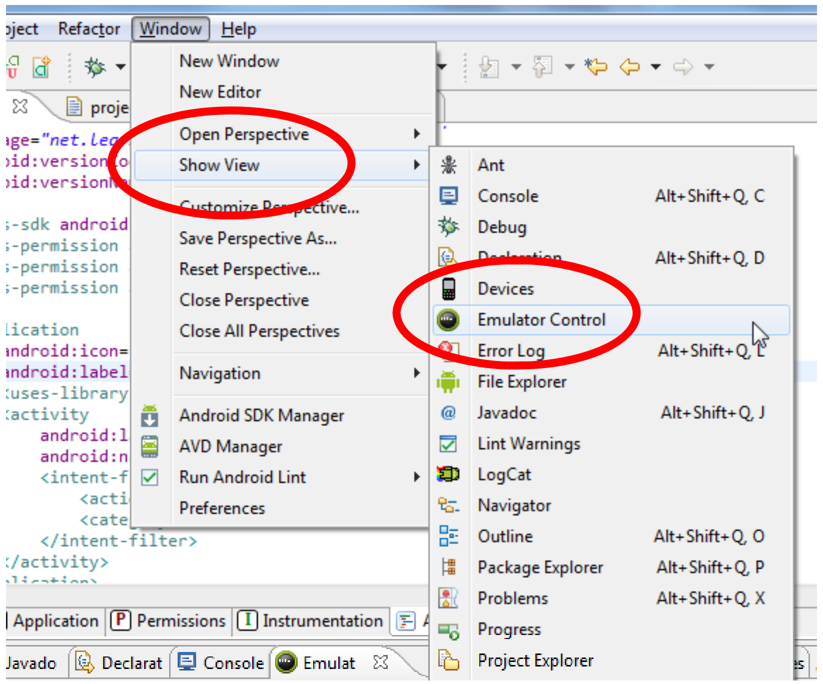
```
<uses-permission android:name=
     “android.permission.ACCESS_FINE_LOCATION"/>  /* GPS */
<uses-permission android:name=
    "android.permission.ACCESS_COARSE_LOCATION"/> /* Network */
```

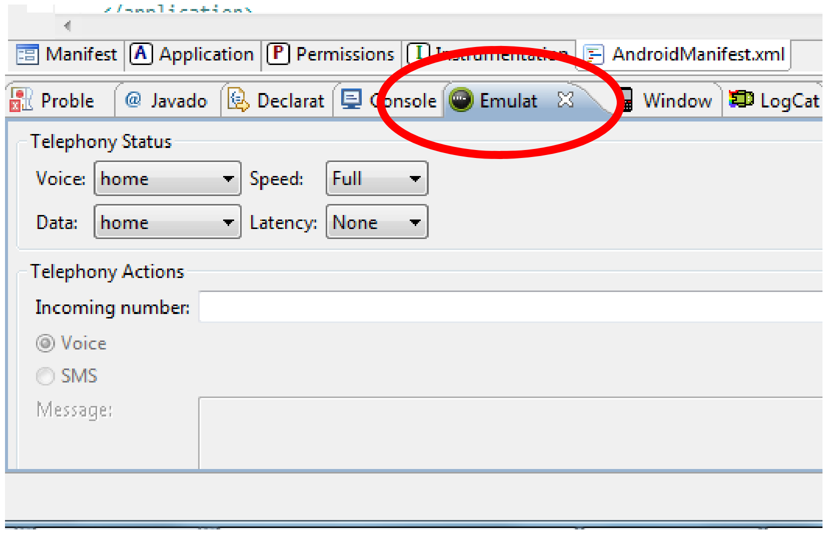
- Get last location fix

```
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```

Emulating the Location

- Location-based services are dependent on device hardware to find the current location
- When developing and testing it is possible to emulate the location information
- Open the Emulator Control window





Telephony Status

Voice: home Speed: Full

Data: home Latency: None

Telephony Actions

Incoming number:

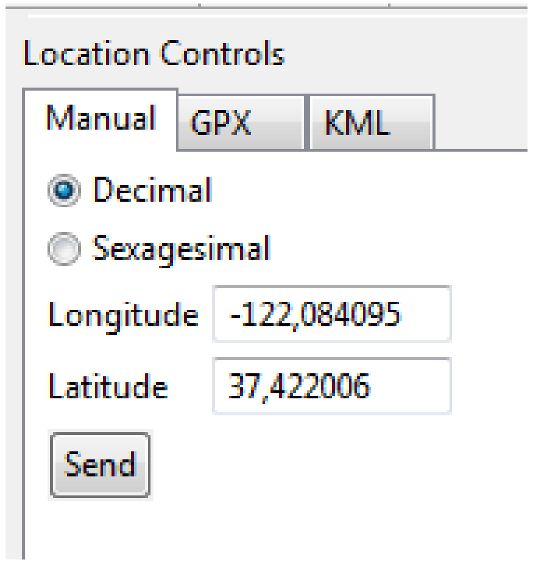
[✓ Voice]

SMS

Message:

Emulating the Location

- Manual introduction
  - Decimal or sexagesimal
- Lists of points
  - GPX (GPS eXchange format)
  - KML (Keyhole Markup Language)


Location Controls

(● Decimal) ○ Sexagesimal

Longitude: -122,084095

Latitude: 37,422006

Description	Latitude Degrees	Longitude degrees	Latitude decimal	Longitude decimal
Karachi, Pakistan	24º53’38”N	67º1’42”E	24,894	67,028
Mount Everest, Nepal	27°59’ N	86°56’ E	27,983	86,933
Ayer’s Rock, Australia	25°23’ S	131°05’ E	-25,383	131,083
Golden Gate Bridge, California	37°49’ N	122°29’ W	37,816	-122,483

GPX example

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1"
 version="1.1"
 creator="Charlie Collins - Hand Rolled"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.topografix.com/GPX/1/1
 http://www.topografix.com/GPX/1/1/gpx.xsd">

   <metadata>
      <name>Sample Coastal California Waypoints</name>
        <desc>Test waypoints for use with Android</desc>
      <time>2008-11-25T06:52:56Z</time>
      <bounds minlat="25.00" maxlat="75.00"
       minlon="100.00" maxlon="-150.00" />
   </metadata>

   <wpt lat="41.85" lon="-124.38">
      <ele>0</ele>
      <name>Station 46027</name>
      <desc>Off the coast of Lake Earl</desc>
   </wpt>
   <wpt lat="41.74" lon="-124.18">
      <ele>0</ele>
      <name>Station CECC1</name>
      <desc>Crescent City</desc>
```

1 Define root gpx element

2 Include metadata stanza

3 Supply waypoint elements

Individual points
+
Tracks

KHL example

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">

   <Placemark>
      <name>Station 46027</name>
      <description>Off the coast of Lake Earl</description>
      <Point>
         <coordinates>-124.38,41.85,0</coordinates>
      </Point>
   </Placemark>

   <Placemark>
      <name>Station 46020</name>
      <description>Outside the Golden Gate</description>
      <Point>
         <coordinates>-122.83,37.75,0</coordinates>
      </Point>
   </Placemark>

   <Placemark>
      <name>Station 46222</name>
      <description>San Pedro Channel</description>
      <Point>
         <coordinates>-118.31,33.61,0</coordinates>
        </Point>
   </Placemark>

</kml>
```

1 Define root kml element

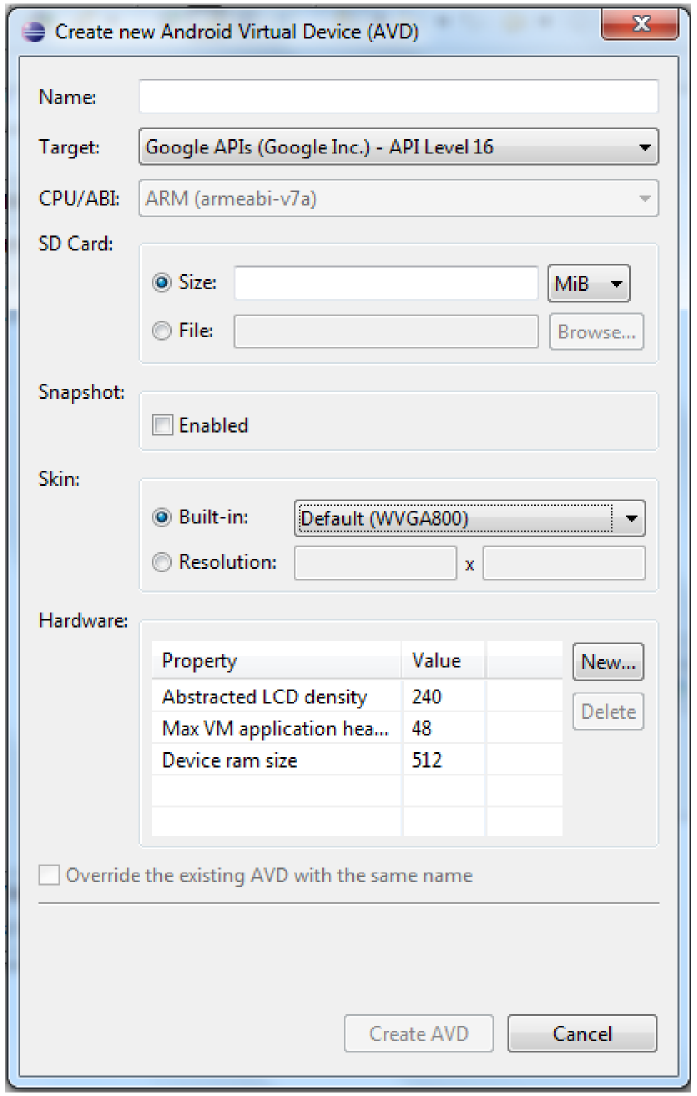
2 Capture information with Placemark

3 Use a Point

4 Supply coordinates for Point

Create new Android Virtual Device (AVD)

- Displaying maps in Android requires virtual devices with Google APIs enabled
- Steps to create a new AVD
  - In Eclipse: Window -> AVD Manager
  - Click New
  - Give a name
  - Select Target: Google APIs
  - Set SD card size
  - Skin: WVGA800
  - Hardware: If additional hardware must be emulated
  - Click Create AVD



Name:

Target: Google APIs (Google Inc.) - API Level 16

CPU/ABI: ARM (armeabi-v7a)

SD Card:
[✓ Size:] MiB
File:

Snapshot:
☐ Enabled

Skin:
[✓ Built-in:] Default (WVGA800)
Resolution: x

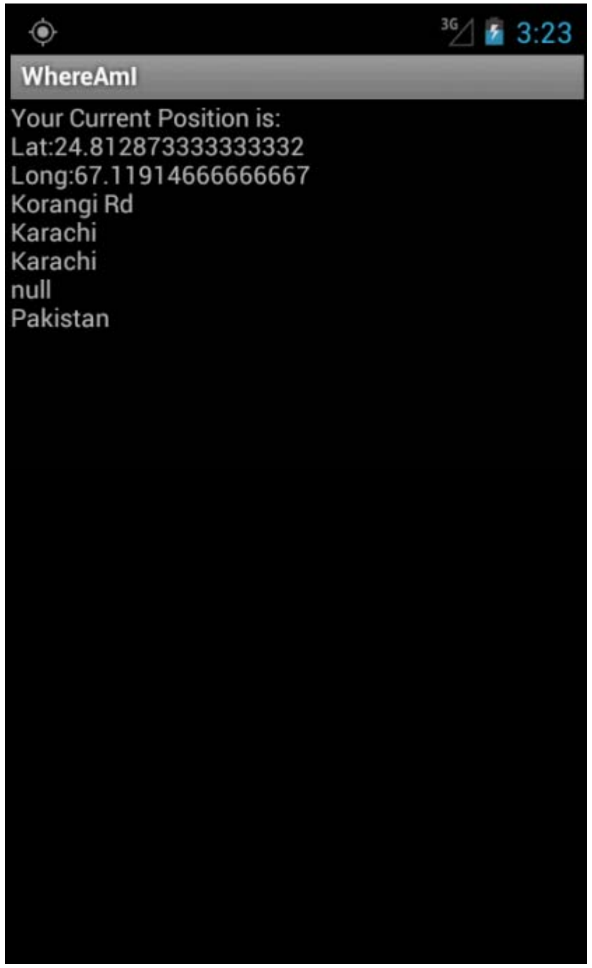
Hardware:

Property	Value
Abstracted LCD density	240
Max VM application hea...	48
Device ram size	512

☐ Override the existing AVD with the same name

Let's Try… Project WhereAmI

- Import project
- Set location using Manual coordinates
  - GPS icon activates
- Run project
- See Manifest
  - <uses-permission> clause
- See code LBSActivity
  - onCreate
    - Use of Location Manager
    - Obtain last known location



```
Your Current Position is:
Lat:24.812873333333332
Long:67.11914666666667
Korangi Rd
Karachi
Karachi
null
Pakistan
```

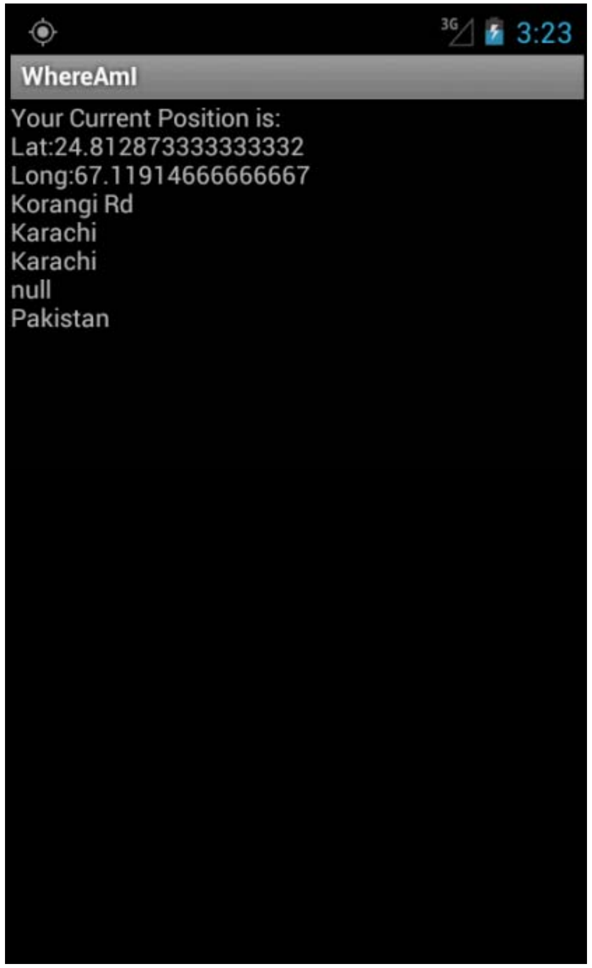
Tracking Movement

- Use a LocationListener to get updates whenever location changes, via requestLocationUpdates method
  - As battery might be a concern, operation can be optimized by specifying the minimum time and the minimum distance between location change updates

```
int t = 5000; // milliseconds
int distance = 5; // meters

LocationListener myLocationListener = new LocationListener() {
  public void onLocationChanged(Location location) { ... }
  public void onProviderDisabled(String provider){ ... }
  public void onProviderEnabled(String provider){ ... }
  public void onStatusChanged(String provider, int status,
                Bundle extras){ ... }  /* HW status changed */
};

locationManager.requestLocationUpdates(provider, t, distance,
                                           myLocationListener);
```

Let’s Try… Project WhereAmI

- Import project
- Set location using Manual coordinates
  - Run project
  - Update location
- See code LBSActivity
  - onCreate
    - Selection of Location Provider based on criteria
    - Obtain last known location
    - Listener registration
  - LocationListener
    - Implementation



```
Your Current Position is:
Lat:24.812873333333332
Long:67.11914666666667
Korangi Rd
Karachi
Karachi
null
Pakistan
```

Geocoding

- Geocoding lets you translate between street addresses and longitude/latitude map coordinates
  - Forward: Finds the latitude and longitude of an address
  - Reverse: Finds the street address for a given latitude and longitude
- Comments
  - Geocoding lookups are done on the server, requires Internet permission

```
<uses-permission android:name="android.permission.INTERNET"/>
```

  - Geocoder lookups are performed synchronously, so they will block the calling thread
  - Geocoding functions return a list of Address objects
  - Localize your results

```
Geocoder geocoder = new Geocoder(getApplicationContext(),
                                 Locale.getDefault());
```

Reverse Geocoding

- Returns street addresses for physical locations, specified by latitude/longitude pairs
  - Accuray will, of course, depend on the quality of database data

```
location = locationManager.getLastKnownLocation
                              (LocationManager.GPS_PROVIDER);

double latitude = location.getLatitude();
double longitude = location.getLongitude();

List<Address> addresses = null;

Geocoder gc = new Geocoder(this, Locale.getDefault());
try {
  addresses = gc.getFromLocation(latitude, longitude, 10);
}
catch (IOException e) {}
```

Forward Geocoding

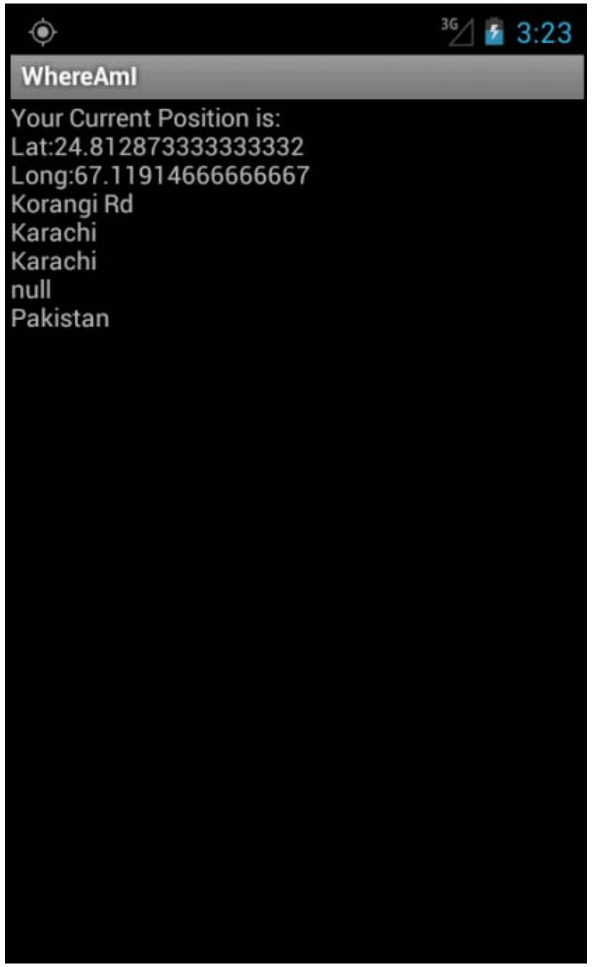
- Determines map coordinates for a given location
  - Valid location
    - Regular street addresses, postcodes, train stations, landmarks, named buildings, hospitals, ...

```
List<Address> result = geocoder.getFromLocationName
                         (aStreetAddress, maxResults);
```

```
Geocoder fwdGeocoder = new Geocoder(this, Locale.US);
String streetAddress = "160 Riverside Drive, New York, New York";

List<Address> locations = null;

try {
  locations = fwdGeocoder.getFromLocationName(streetAddress, 10);
}
catch (IOException e) {}
```

Let’s Try… Project WhereAmI

- Import project
- Set location using Manual coordinates
  - Lat: 24,894, Long: 67,028
  - Run project
- See Manifest
  - <uses-permission> clause
- See code LBSActivity
  - updateWithNewLocation
    - Geocoding
    - Response handling



```
Your Current Position is:
Lat:24.812873333333332
Long:67.11914666666667
Korangi Rd
Karachi
Karachi
null
Pakistan
```

Maps

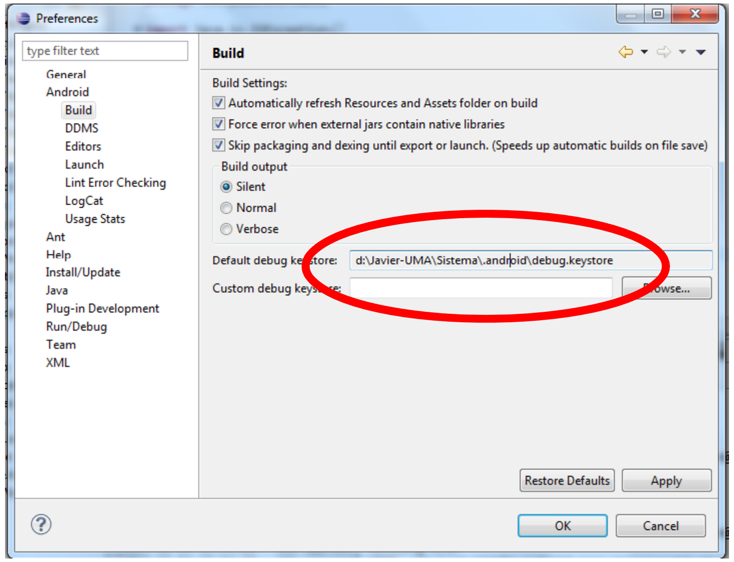
Map-based Activities

- A map-based activity will, at least, use the following elements
  - MapActivity: The base class you extend to create a new Activity.
  - MapView: The Map View control. Only can be used within a MapActivity
    - Present geographical data
    - Interact with the map
    - Modes: Street, satellite, traffic
    - Annotation using Overlays
- Using maps requires
  - Virtual Device with Google Maps enabled
  - A Google Maps API Key
  - Additional permissions

Obtaining Google Maps API Key

- Without an API key the Map View will not download the tiles used to display the map
  - Need a Gmail account

- Use the SDK Debug certificate
  - For development
    - Emulator or local connected device
  - Key valid for all projects in same machine/installation (apply only once)
  - Default debug keystore
    - `debug.keystore`
    - Windows -> Preference -> Android -> Build



Build Settings:

- [✓] Automatically refresh Resources and Assets folder on build
- [✓] Force error when external jars contain native libraries
- [✓] Skip packaging and dexing until export or launch. (Speeds up automatic builds on file save)

Build output: [● Silent] ○ Normal ○ Verbose

Default debug keystore: d:\Javier-UMA\Sistema\.android\debug.keystore

Custom debug keystore:

Obtaining Google Maps API Key

- Steps
  - Instructions

    `https://developers.google.com/maps/documentation/android/v1/mapkey`
  - Get the MD5 certificate of your signing certificate

```
> "c:\Program Files\Java\jre6\bin\keytool.exe" -list -alias
  androiddebugkey -keystore
  \users\<username>\.android\debug.keystore -storepass <password> -
  keypass <password>
>> Certificate fingerprint (MD5):
    94:1E:43:49:87:73:BB:E6:A6:88:D7:20:F1:8E:B5:98
```

  - Get the key

    `https://developers.google.com/maps/documentation/android/v1/maps-api-signup`
    - Accept terms
    - Introduce MD5 → `0pg7mjTX7wgDpmV-iG8aaq2_paA3an0DYp2VYLg`
  - Insert the key in the layout file

Using Maps

- Layout with MapView

```
<com.google.android.maps.MapView
   android:id="@+id/myMapView"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:enabled="true"
   android:clickable="true"
   android:apiKey="myMapKey"
 />
```

- Changes in Manifest
  - Add the library in the application node

```
<uses-library android:name="com.google.android.maps"/>
```

Using MapViews

- Default: Street map
  - Other views are possible

```
mapView.setSatellite(true);
mapView.setStreetView(true);
mapView.setTraffic(true);
```

- Display zoom controls

```
mapView.setBuiltInZoomControls(true);
```

Map Controller

- Use Map Controller to pan, move and zoom a MapView

```
MapController mapController = myMapView.getController();
```

- Map locations are represented by GeoPoint objects
  - Latitude and longitude measured in microdegrees

```
Double lat = 37.422006*1E6;
Double lng = -122.084095*1E6;
GeoPoint point = new GeoPoint(lat.intValue(), lng.intValue());
```

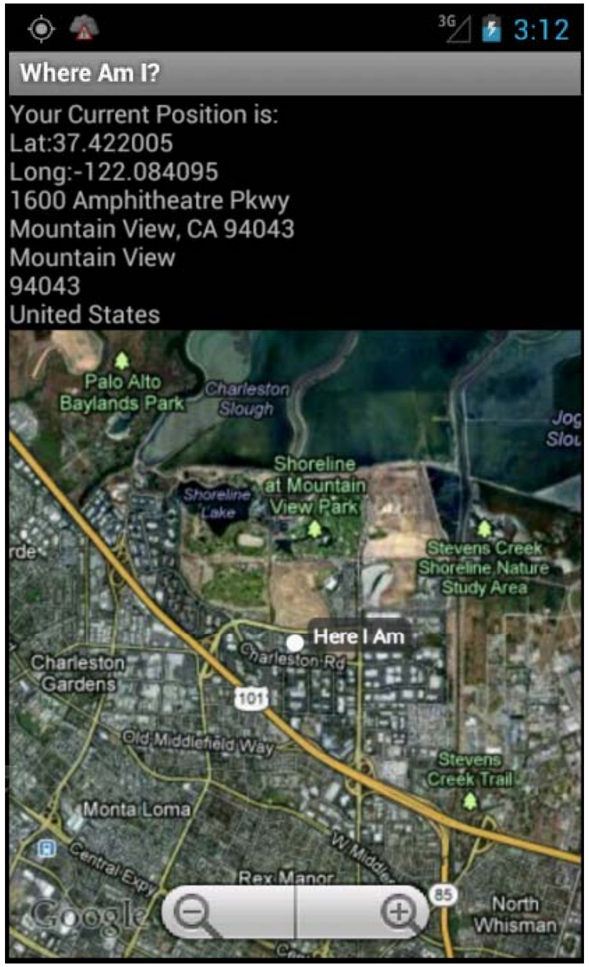
- Re-center, move and zoom the Map View

```
mapController.setCenter(point);
mapController.animateTo(point);
mapController.setZoom(1);
```

Note: Zoom = 1 → most distant view; Zoom = 21 nearest view

Let's Try… Project WhereAmIMaps

- Import project
- Get Google Maps API key
- Update GPS position before running
- See res\layout\main.xml
  - Insert key
  - MapView element
- See Manifest
  - <uses-library>
  - <uses-permission>



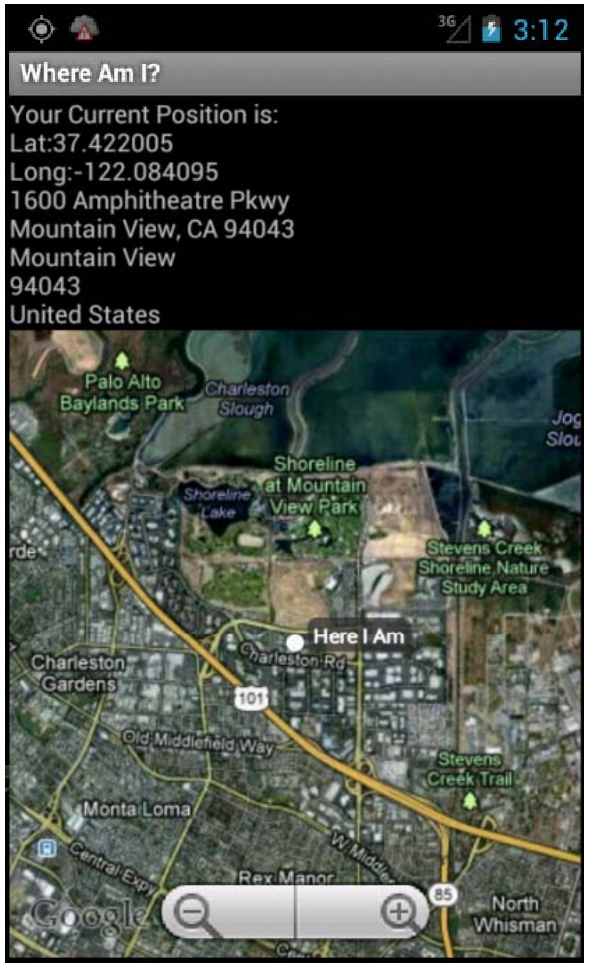
Your Current Position is:
Lat:37.422005
Long:-122.084095
1600 Amphitheatre Pkwy
Mountain View, CA 94043
Mountain View
94043
United States

Let's Try… Project WhereAmIMaps

- See code WhereAmI.java
  - Based on MapActivity
  - onCreate
    - MapView controls & configuration
  - mapController

- Others
  - Click on map to get zoom controls
  - Send new coordinates to move the map
    - Try small differences
  - Change map mode

```
myMapView.setSatellite(false/true)
```


Where Am I?
Your Current Position is:
Lat:37.422005
Long:-122.084095
1600 Amphitheatre Pkwy
Mountain View, CA 94043
Mountain View
94043
United States

Overlays

Overlays

- An overlay is a canvas with a transparent background that is layered onto a Map View
  - Enable to add annotations and click handling to MapViews
  - Can add as many Overlays as wished onto a single map

- How?
  - Extend class Overlay
  - Override draw(…) method
    - Draw the shape wanted
  - Override onTap(…) method
    - React to user clicks

Projections

- The canvas represents the visible display surface
- Projections translate between latitude/longitude coordinates (stored as GeoPoints) and x/y screen pixel coordinates (stored as Points)

```
Point myPoint = new Point();

Projection projection = mapView.getProjection();

// To screen coordinates
projection.toPixels(geoPoint, myPoint);
// To GeoPoint location coordinates
projection.fromPixels(myPoint.x, myPoint.y);
```

Drawing on the Canvas

- The canvas represents the visible display surface

```
public void draw(Canvas canvas, MapView mapView, boolean shadow) {
  Point myPoint = new Point();
  projection.toPixels(geoPoint, myPoint);

  // Create and setup your paint brush
  Paint paint = new Paint();
  paint.setARGB(250, 255, 0, 0);
  paint.setAntiAlias(true);
  paint.setFakeBoldText(true);

  // Create the circle
  int rad = 5;
  RectF oval = new RectF(myPoint.x-rad, myPoint.y-rad,
  myPoint.x+rad, myPoint.y+rad);

  // Draw on the canvas
  canvas.drawOval(oval, paint);
  canvas.drawText("Red Circle", myPoint.x+rad, myPoint.y, paint);
}
```

Handling Tap Events

- Override onTap(…)
  - Parameters
    - Geopoint
    - The MapView

```
public boolean onTap(GeoPoint point, MapView mapView) {
  // Perform hit test to see if this overlay is handling the click
  if ([ . . . perform hit test . . . ]) {
    [ . . . execute on tap functionality . . . ]
    return true;
  }

  // If not handled return false
  return false;
}
```

Adding and Removing Overlays

- Each MapView contains a list of Overlays currently displayed
  - Adding and removing items from the list is thread-safe and synchronized
- To add an Overlay onto a Map View, create a new instance of the Overlay and add it to the list

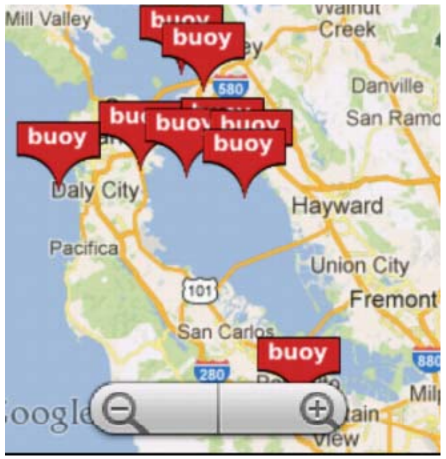
```
List<Overlay> overlays = mapView.getOverlays();
MyOverlay myOverlay = new MyOverlay();
overlays.add(myOverlay);
mapView.postInvalidate();
```

- Call `postInvalidate()` to redraw the MapView

Itemized Overlays

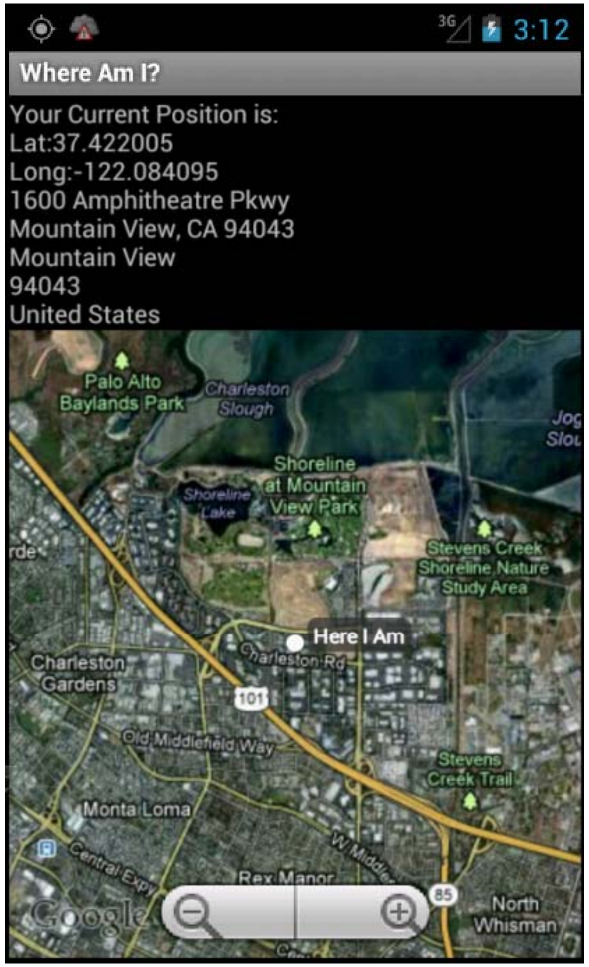
- Convenient shortcut for adding markers to a map, letting you assign a marker image and associated text to a particular geographical position
  - Handles the drawing, placement, click handling, focus control, and layout of each marker

```
List<Overlay> overlays =
              mapView.getOverlays();
MyItemizedOverlay markers;
Markers = new MyItemizedOverlay
             (r.getDrawable(R.drawable.marker));
overlays.add(markers);
```



Let's Try… Project WhereAmIMaps

- Import project
- See code WhereAmIMaps
  - onCreate
    - Overlay creation
- See code MyPositionOverlay
  - draw method
  - onTap should check user clicked on the overlay


Where Am I?
Your Current Position is:
Lat:37.422005
Long:-122.084095
1600 Amphitheatre Pkwy
Mountain View, CA 94043
Mountain View
94043
United States

Android Development

Session 7

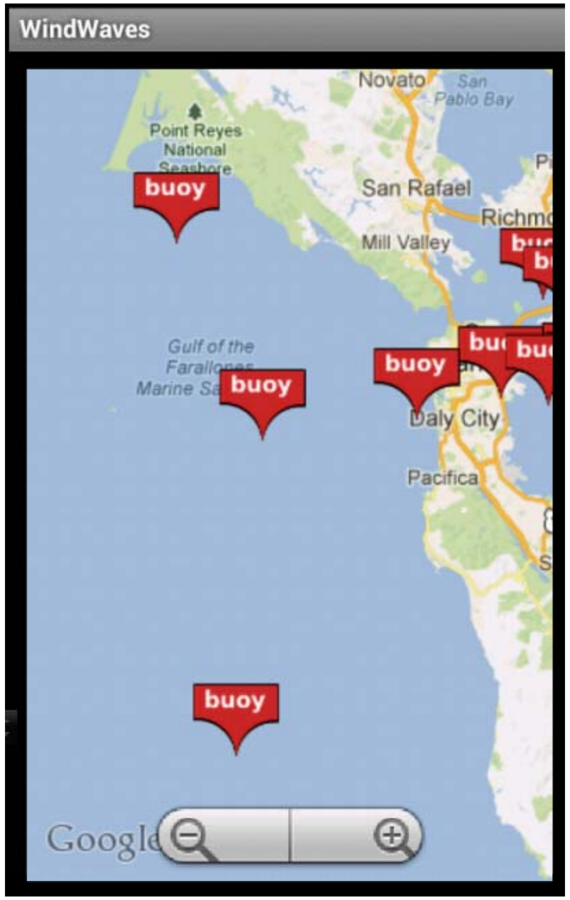
Javier Poncela

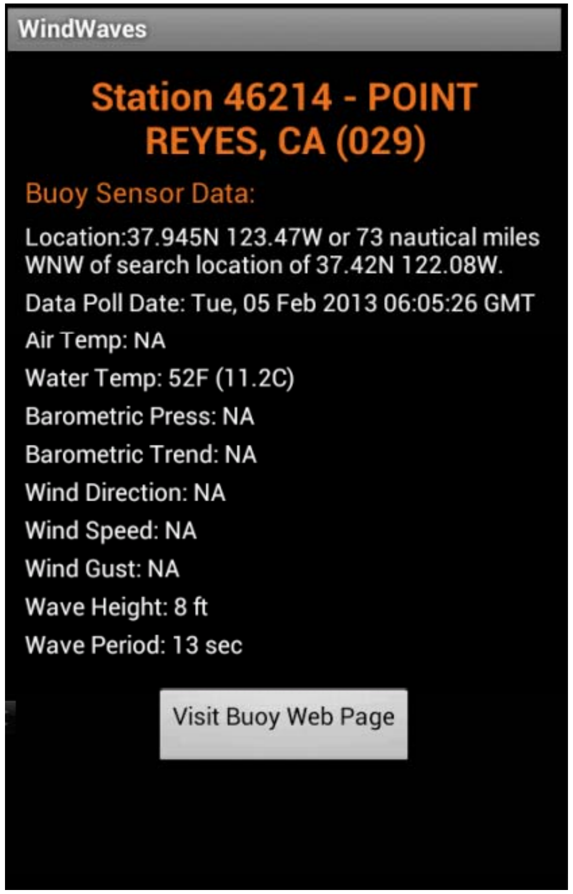
Contents

WindWaves

WindWaves

- Display buoy information




WindWaves

Station 46214 - POINT REYES, CA (029)

Buoy Sensor Data:

Location:37.945N 123.47W or 73 nautical miles WNW of search location of 37.42N 122.08W.

Data Poll Date: Tue, 05 Feb 2013 06:05:26 GMT

Air Temp: NA

Water Temp: 52F (11.2C)

Barometric Press: NA

Barometric Trend: NA

Wind Direction: NA

Wind Speed: NA

Wind Gust: NA

Wave Height: 8 ft

Wave Period: 13 sec

Visit Buoy Web Page

Conditions at 46013 as of
(9:50 pm PST on 02/04/2013)
0550 GMT on 02/05/2013:

Wind Direction (WDIR):	WNW (300 deg true)
Wind Speed (WSPD):	13.6 kts
Wind Gust (GST):	15.5 kts
Wave Height (WVHT):	6.2 ft
Dominant Wave Period (DPD):	13 sec
Average Period (APD):	7.6 sec
Atmospheric Pressure (PRES):	30.11 in
Pressure Tendency (PTDY):	+0.02 in (Steady)
Air Temperature (ATMP):	49.1 °F
Water Temperature (WTMP):	49.8 °F
Wind Chill (CHILL):	43.3 °F
Wind Speed at 10 meters (WSPD10M):	13.6 kts
Wind Speed at 20 meters (WSPD20M):	15.5 kts

Continuous Winds

Let’s Try… Project WindWaves

- Import project
- Run and test it briefly

Manifest

- Indication of main entry point
- List of all activities in the application
- Permission for Internet access

Buoy Data Structure

- BuoyData.java
  - Stores data of a buoy
  - toString

- BuoyOverlayItem.java
  - Stores the points to draw in the map

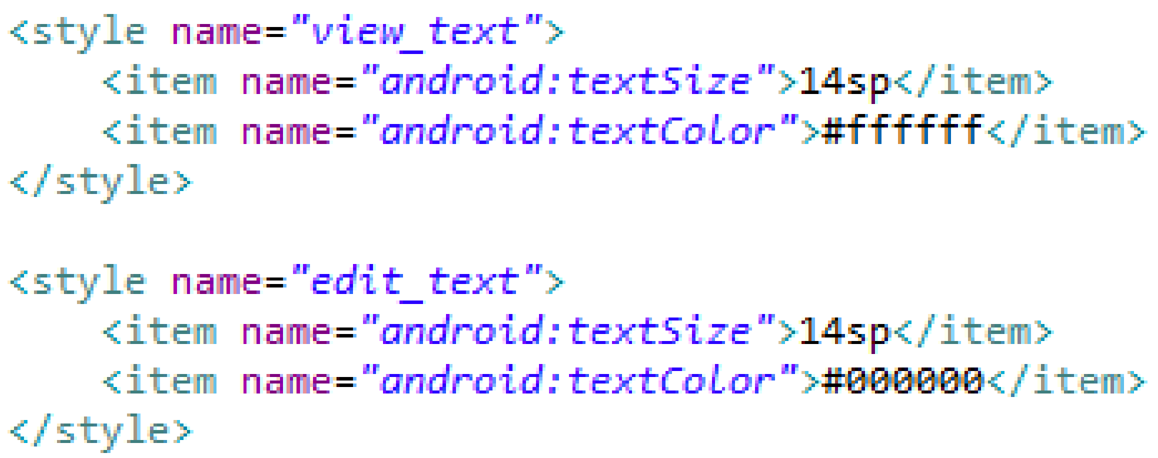
LocationHelper.java

- Auxiliary functions to
  - Create geoPoint
  - Parse geoRssPoint
  - Parse coordinates

styles.xml

- Located in res/values

- It is possible to define styles to draw elements
  - Texts, backgrounds, borders, …
  - Just reference the style to use in each item



```xml
<style name="view_text">
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#ffffff</item>
</style>

<style name="edit_text">
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#000000</item>
</style>
```

StartActivity.java

- Entry point
- Layout: startactivity.xml
  - ScrollView with a RelativeLayout with 3 items
- Starts a thread to delay entering the next activity
  - When time expires, call startActivity

MapViewActivity.java

- Main activity (it’s a MapActivity)
  - Layout: mapview_activity.xml

- onCreate
  - Load marker icon
  - Create list of OverlayItems

- onStart
  - Get location, animate to it
  - Get bouy data

- Options Menu
  - onCreateOptionsMenu
  - onMenuItemSelected

MapViewActivity.java

- getBuoyData
  - Show progress dialog
  - Create new thread and call class to retrieve data
  - Parse the list of received buoys
  - Update UI (handler)
- getBuoyOverlayItems
  - Create list of buoy Overlays
- handler
  - Update list of buoy Overlays
  - Redraw map
- LocationListener
  - Listen to location updates

BuoyItemizedOverlay.java

- To represent overlays on the map
- draw
  - Relies on super, as it provides an image for the marker
- onTap
  - Presents dialog with extra info on clicked buoy
  - Loads layout buoy_selected.xml
    - Styles used

BuoyDetailActivity.java

- Shows more detail on buoy
  - Layout: buoydetail_activity.xml

- Calls activity to display buoy URL

```
startActivity(new Intent(Intent.ACTION_VIEW,
          Uri.parse(BuoyDetailActivity.buoyData.link)));
```

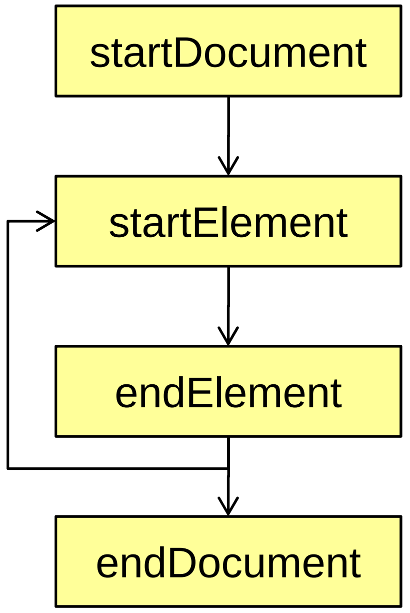
NDBCFetcher.java

- Retreives buoy data from server
  - Creates query
  - Sends request to the server
  - Creates reader and parser
  - Assigns handler to parser
  - Initiates parsing

NDBCHandler.java

- Parses retrieved information from server

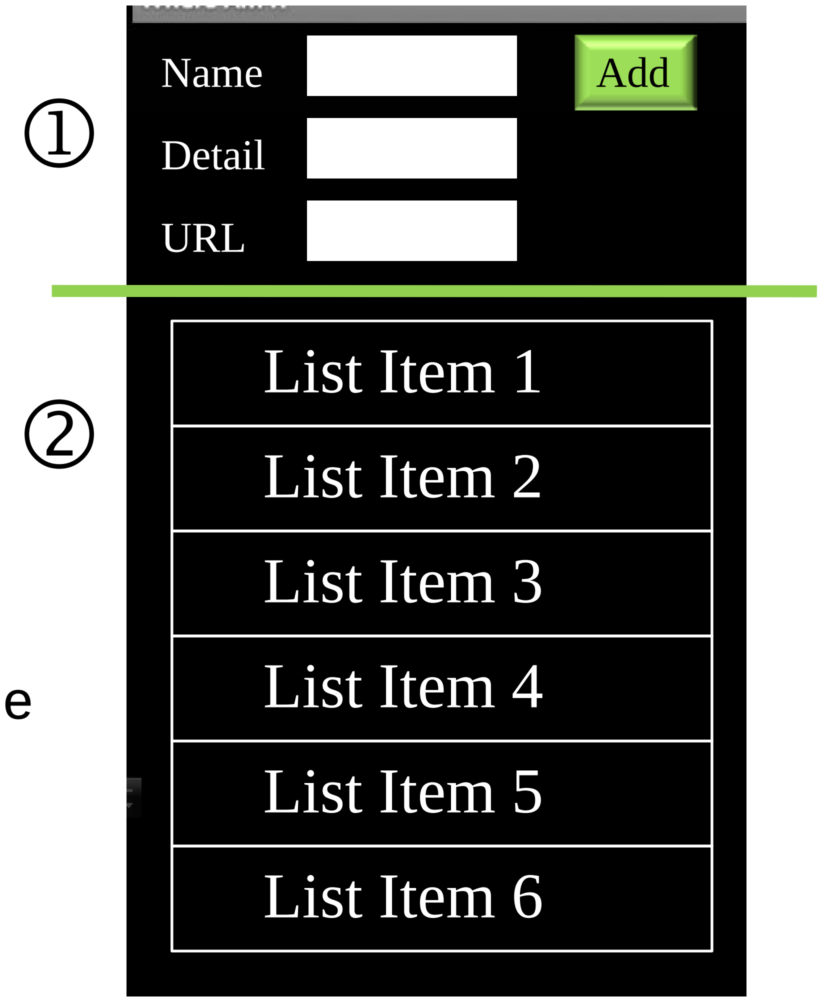
- Main functions
  - startDocument: start processing
  - startElement: creates Buoy entry
  - endElement: sets job attributes
  - endDocument: finish processing
  - characters: parses tokens



Hands On

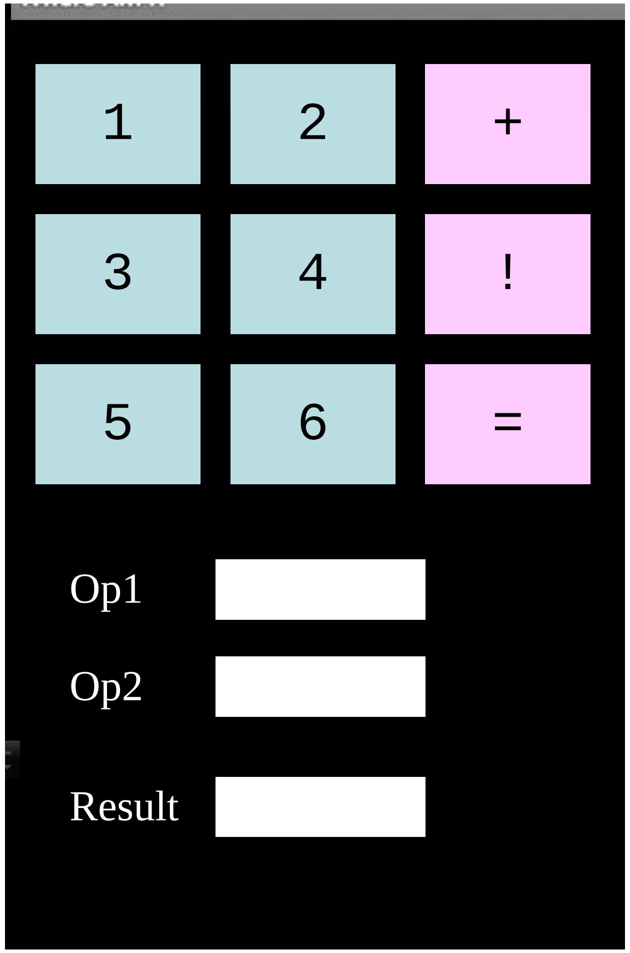
List of Items

- Screen 1
  - Introduce item + Detail + URL
  - Button Add → Show list
- Screen 2
  - Show list of items
  - Clicking item → get Details
- Screen 3
  - Show details + Button
  - Click button → show webpage



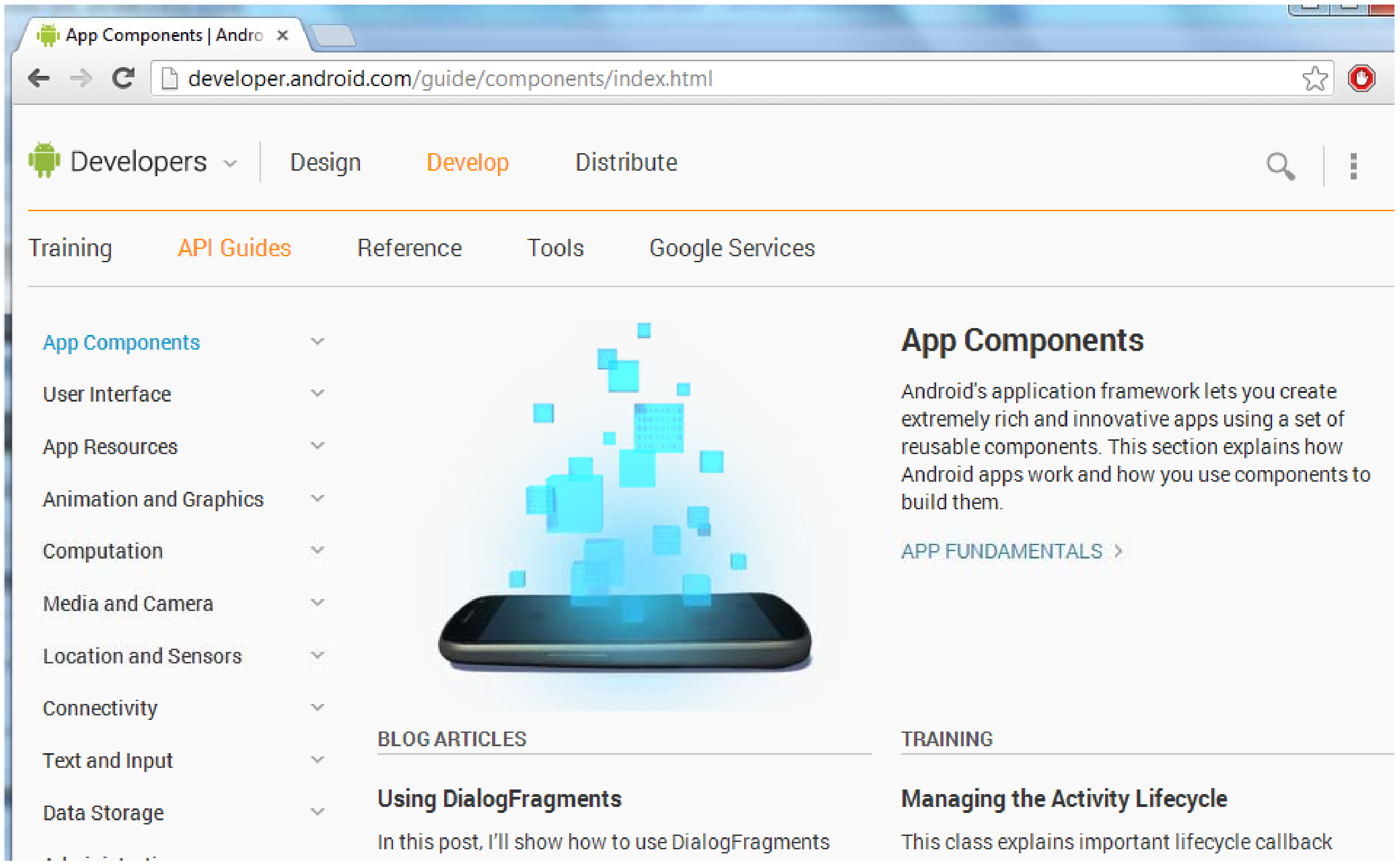
Calculator

- Basic implementation
  - Some digits
  - One simple operation
  - One complex operation
- Operands are shown in text boxes when number is input
- For the complex operation use
  - ProgressDialog
  - AsyncTask
  - … update progress (sleep between calculation cycles)



Further…

http://developer.android.com/guide/components/index.html


App Components | Andro ×

developer.android.com/guide/components/index.html

Developers Design Develop Distribute

Training API Guides Reference Tools Google Services

- App Components
- User Interface
- App Resources
- Animation and Graphics
- Computation
- Media and Camera
- Location and Sensors
- Connectivity
- Text and Input
- Data Storage

App Components

Android's application framework lets you create extremely rich and innovative apps using a set of reusable components. This section explains how Android apps work and how you use components to build them.

APP FUNDAMENTALS ›

BLOG ARTICLES

Using DialogFragments

In this post, I'll show how to use DialogFragments

TRAINING

Managing the Activity Lifecycle

This class explains important lifecycle callback

Further…

Learning Android
Android Cookbook
Programming Android
Android Application Testing Guide
Android User Interface Development
Beginner's Guide
Android 3.0 Application Development Cookbook
Pro Android
Beginning Android 4
Android Apps for Absolute Beginners
Professional Android 2 Application Development
Beginning Android Application Development
The Android Developer's Cookbook
Android in Practice
Android in Action
Sams Teach Yourself Android Application Development in 24 Hours
Hello, Android
Android Application Development All-in-One for Dummies

and many more …