



Applications

Marco Ronchetti
Università degli Studi di Trento

Android Applications

An Android *application* typically **consists of one or more related, loosely bound activities** for the user to interact with.

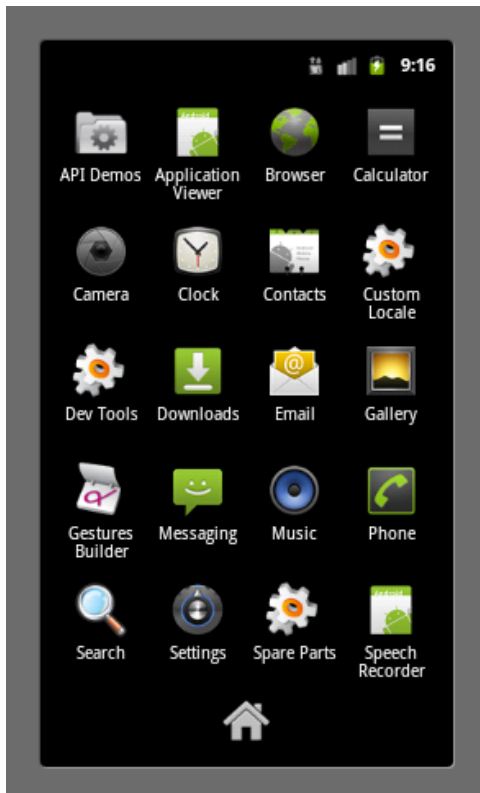
Android has an **application launcher** available at the Home screen, typically in a sliding drawer which displays applications as icons, which the user can pick to start an application.

Android ships with a rich set of applications that may include email, calendar, browser, maps, text messaging, contacts, camera, dialer, music player, settings and others.

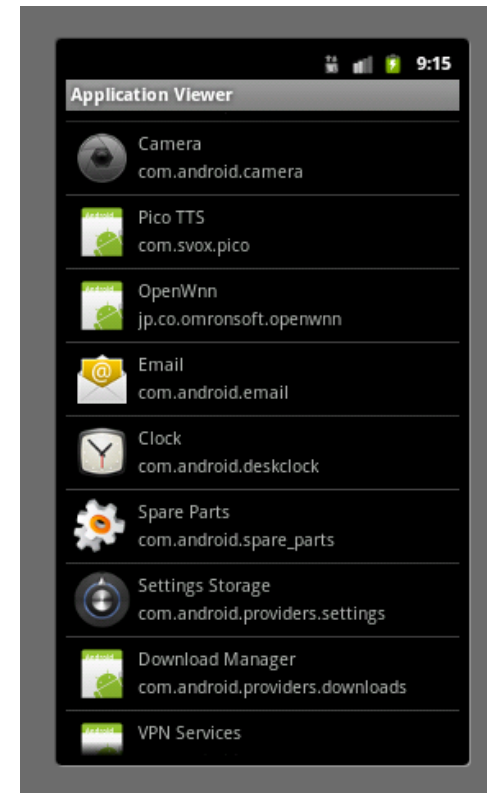
2



Application Launcher



You can replace it



3

See e.g.

<http://xiaphx.wordpress.com/2011/06/12/create-application-launcher-as-a-list/>



Application package

An application is a single APK (application package) file. An APK file roughly has three main components.

- **Dalvik executable**: all your Java source code compiled down to Dalvik executable. This is the code that runs your application.
- **Resources**: everything that is not code (images, audio/video clips, XML files describing layouts, language packs, and so on).
- **Native libraries**: e.g. C/C++ libraries.



Signing applications

Android applications must be **signed** before they can be installed on a device

To distribute your application commercially, you'll want to sign it with your own key.



Distributing applications

Unlike the iPhone, on Android, there can be **many different Android stores or markets**. Each one can have its own set of policies with respect to what is allowed, how the revenue is split, and so on.

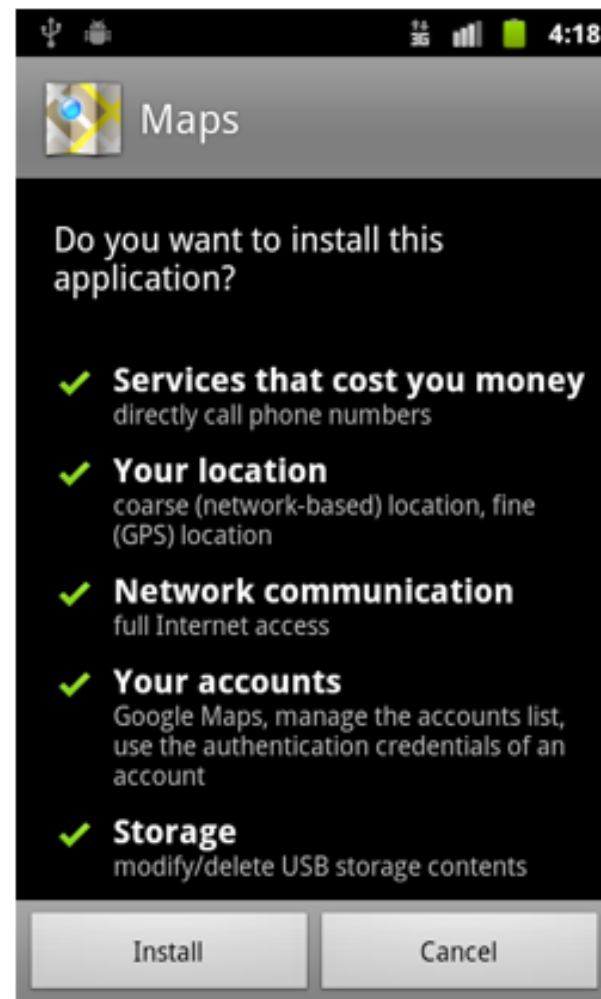
The biggest market currently is Android Market run by Google

Applications can also be distributed **via the web**. When you download an APK file from a website by using the Browser, the application represented by the APK file automatically gets installed on your phone.

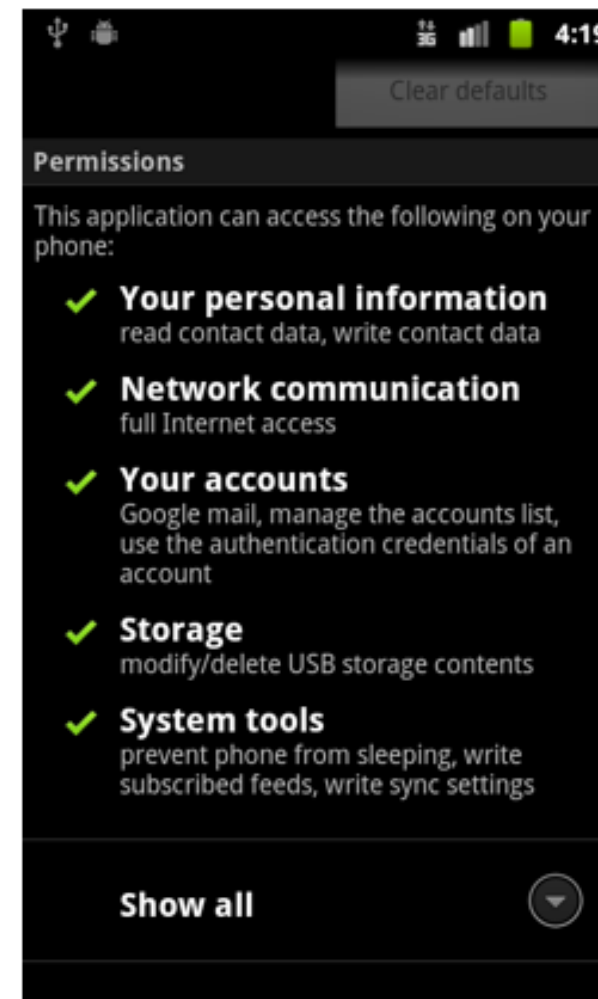


Granting and checking permissions

Permissions at Application Install -- Google Maps



Permissions of an Installed Application -- gMail



7



Impostazioni->Altro->Gestione Applicazioni -> ...

Security

Android has a security framework.

<http://source.android.com/devices/tech/security/index.html>

The Android File System can be encrypted.

Encryption on Android uses the dm-crypt layer in the Linux kernel.



Security model

Android OS is a multi-user Linux in which **each application is a different user**.

By default, the system assigns each application a unique Linux user ID (the ID is unknown to the application). The system sets permissions for all the files in an application so that **only the user ID assigned to that application can access them**.

Each process has its own virtual machine (VM), so an application's code runs **in isolation from other applications**.

By default, every application runs in its own Linux process.



Principle of least privilege

Principle of least privilege (or “need to know”)

Each application, by default, has access only to the components that it requires to do its work and no more.

A variation of “information hiding”, or “Parnas’ principle”.



Data sharing

It's possible to arrange for two applications to **share the same Linux user ID**, in which case they are able to access each other's files.

Applications with the same user ID can also arrange to **run in the same Linux process and share the same VM** (the applications must also be signed with the same certificate).

An application can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more. **All application permissions must be granted by the user at install time.**



Process lifetime

Android

- starts the process when any of the application's components need to be executed,
- shuts down the process when
 - it's no longer needed
 - the system must recover memory for other applications.



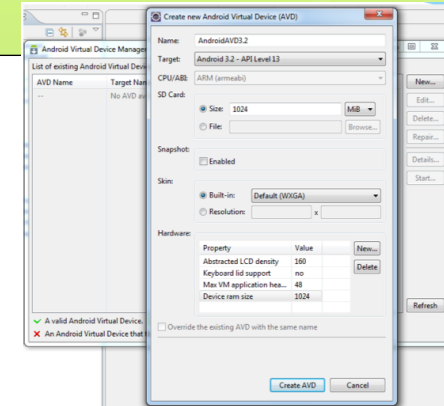


Getting started: Installing IDE and SDK

Marco Ronchetti
Università degli Studi di Trento

Get ready in four steps

1. Get Eclipse
2. Get the Android SDK
3. Run Eclipse, and install the ADT plugin into it
4. In Eclipse, setup the emulator (choose the ADV)



You're ready to go!

For details, come to the Laboratory lecture, or else see:

<http://developer.android.com/sdk/installing.html>

14

or take one of the many other tutorials on the net.



Alternative: Android Studio

<https://developer.android.com/sdk/installing/studio.html>

It is only version 0.42...



15



Warning on Linux 64 bits!

The Android SDK is 32 bit, therefore on an 64 bit Linux system you need to have the package ia32-libs installed. For Ubuntu you can do this via the following command:

```
apt-get install ia32-libs
```



Tools behind the scenes

dx

- allows to convert Java .class files into .dex (Dalvik Executable) files.

aapt (Android Asset Packaging Tool)

- packs Android applications into an .apk (Android Package) file.

adb (Android debug bridge)

ADT (Android Development Tools for Eclipse)

- A development tool provided by Google to perform automatic conversion from .class to .dex files and to create the apk during deployment. It also provides debugging tools, and an Android device emulator.



ADV - Android Virtual Device

An **emulator configuration** that lets you model an actual device by defining hardware and software options

An AVD consists of:

- **A hardware profile**
 - Defines the hardware features of the virtual device (whether it has a camera, a physical QWERTY keyboard or a dialing pad, how much memory it has etc.
- **A mapping to a system image:**
 - You can define what version of the Android platform will run on the virtual device
- Other options: the **emulator skin** (screen dimensions, appearance, etc.), **emulated SD card**
- A **dedicated storage area** on your development machine:
 - the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area.



ADV - Android Virtual Device

You create an AVD:

- with the graphical **AVD Manager** in Eclipse
 - See <http://developer.android.com/guide/developing/devices/managing-avds.html>
- from the command line (**\$ android create avd**),
 - see <http://developer.android.com/guide/developing/devices/managing-avds-cmdline.html>





Getting started: Hello Android

Marco Ronchetti
Università degli Studi di Trento

android.app.application

How shall we start?

As we know already, there is no main...

But there is an "application" class in the API.
(actually, `android.app.application`)

Probably we should `subclass` that, like we do with
`java.applet.Applet` or with
`javax.servlet.http.HttpServlet`?

21



NO!

Application is a base class ONLY for keeping a global application state.

We need to subclass another thing: **Activity**



HelloAndroid

```
package com.example.helloandroid;

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

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



HelloAndroid

```
package com.example.helloandroid;

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

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

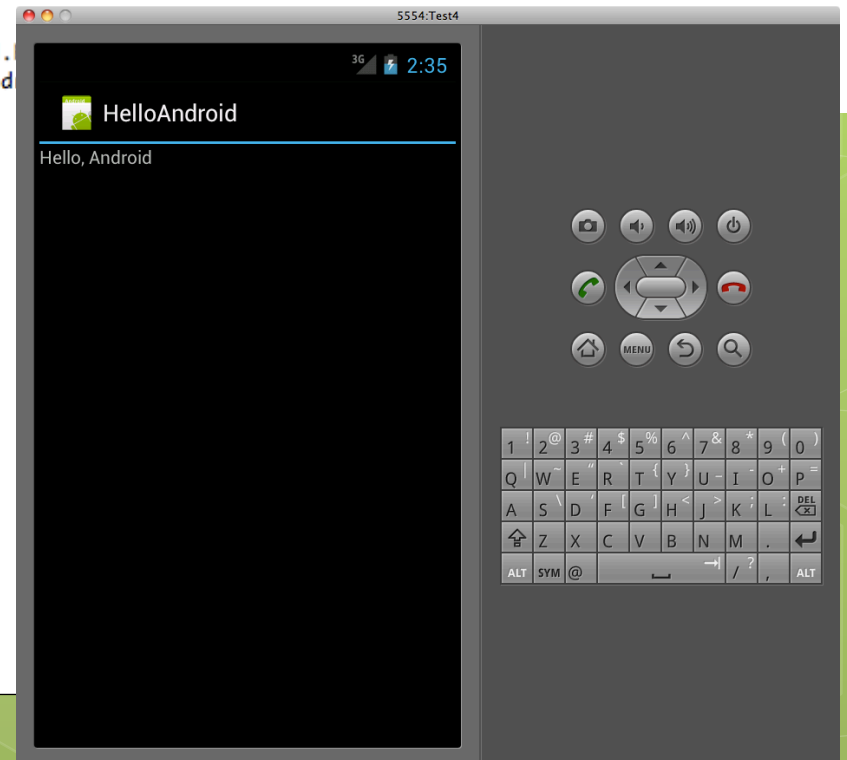


Launching the emulator...

Problems Javadoc Declaration Console

Android

```
[2012-02-28 14:33:05 - HelloAndroid] -----
[2012-02-28 14:33:05 - HelloAndroid] Android Launch!
[2012-02-28 14:33:05 - HelloAndroid] adb is running normally.
[2012-02-28 14:33:05 - HelloAndroid] Performing com.example.helloandroid.HelloAndroidActivity activity launch
[2012-02-28 14:33:05 - HelloAndroid] Automatic Target Mode: launching new emulator with compatible AVD 'Test4'
[2012-02-28 14:33:05 - HelloAndroid] Launching a new emulator with Virtual Device 'Test4'
[2012-02-28 14:33:07 - Emulator] 2012-02-28 14:33:07.475 emulator-arm[3911:80b] Warning once: This application, or a library it uses, is using
[2012-02-28 14:33:07 - Emulator] emulator: WARNING: Unable to create sensors port: Connection refused
[2012-02-28 14:33:07 - HelloAndroid] New emulator found: emulator-5554
[2012-02-28 14:33:07 - HelloAndroid] Waiting for HOME ('android.process.acore') to be launched...
[2012-02-28 14:33:39 - HelloAndroid] HOME is up on device 'emulator-5554'
[2012-02-28 14:33:39 - HelloAndroid] Uploading HelloAndroid.apk onto device 'emulator-5554'
[2012-02-28 14:33:39 - HelloAndroid] Installing HelloAndroid.apk...
[2012-02-28 14:34:04 - HelloAndroid] Success!
[2012-02-28 14:34:04 - HelloAndroid] Starting activity com.example.helloandroid.
[2012-02-28 14:34:05 - HelloAndroid] ActivityManager: Starting: Intent { act=and
```



HelloAndroid: questions.

```
package com.example.helloandroid;
```

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

- What is an Activity?
- What is onCreate?
- What is a Bundle?
- What is R?

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

26



- What is a TextView??

```
TextView tv = new TextView(this);  
tv.setText("Hello Android");
```



Dissecting the HelloWorld

Marco Ronchetti
Università degli Studi di Trento

android.app

Class Activity

Class Activity

[java.lang.Object](#)

└ [android.content.Context](#)

└ [android.content.ContextWrapper](#)

└ [android.view.ContextThemeWrapper](#)

└ **android.app.Activity**

All Implemented Interfaces:

[ComponentCallbacks](#), [KeyEvent.Callback](#), [LayoutInflater.Factory](#), [View.OnCreateContextMenuListener](#), [Window.Callback](#)

Direct Known Subclasses:

[ActivityGroup](#), [AliasActivity](#), [ExpandableListActivity](#), [ListActivity](#)

An activity is a single, focused thing that the user can do.

Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(int)`.

Doesn't it reminds you of "JFrame" and "setContentPane()?"



android.app Class Activity

Class Activity

```
java.lang.Object
├── android.content.Context
│   ├── android.content.ContextWrapper
│   │   └── android.view.ContextThemeWrapper
│   └── android.app.Activity
```

Interface to global information about an application environment.

All Implemented Interfaces:

[ComponentCallbacks](#), [KeyEvent.Callback](#), [LayoutInflater.Factory](#), [View.OnCreateContextMenuListener](#), [Window.Callback](#)

Direct Known Subclasses:

[ActivityGroup](#), [AliasActivity](#), [ExpandableListActivity](#), [ListActivity](#)

An activity is a single, focused thing that the user can do.

Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(int)`.

Doesn't it reminds you of "JFrame" and "setContentPane()?"



Class Activity

While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `R.attr.windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`).



Resources

You should always **externalize resources** (e.g. images and strings) from your application code, so that you can:

- **maintain them independently.**
- **provide alternative resources, e.g.:**
 - different languages
 - different screen sizes

Resources must be organized in your project's **res/** directory, with various sub-directories that group resources by type and configuration.



The R class

When your application is compiled, aapt generates the **R class**, which contains resource IDs for all the resources in your `res/` directory.

For each type of resource, there is an R subclass (for example, **R.layout** for all layout resources) and for each resource of that type, there is a static integer (for example, **R.layout.main**). This integer is the **resource ID** that you can use to retrieve your resource.



R.Java in gen/

```
/* 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 com.example.helloandroid;  
public final class R {  
    public static final class attr {  
    }  
    public static final class drawable {  
        public static final int ic_launcher=0x7f020000;  
    }  
    public static final class layout {  
        public static final int main=0x7f030000;  
    }  
    public static final class string {  
        public static final int app_name=0x7f040001;  
        public static final int hello=0x7f040000;  
    }  
}
```

33



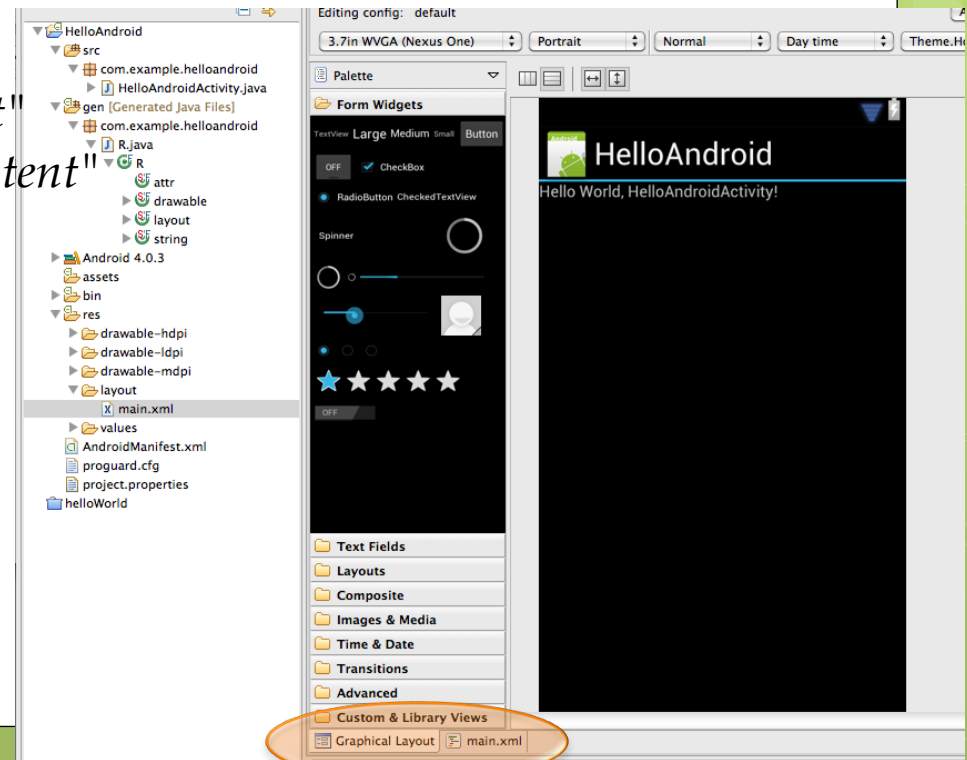
Res/layout/main.xml

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

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

```
</LinearLayout>
```

34



onCreate(Bundle b)

Callback invoked when the activity is starting.

This is where most initialization should go.

If the activity is being re-initialized after previously being shut down then this **Bundle** contains the data it most recently supplied in `onSaveInstanceState(Bundle)`, otherwise it is null.

Note: a Bundle is a sort of container for serialized data.



TextView

Displays text to the user and optionally allows them to edit it. A TextView is a complete text editor, however the basic class is configured to not allow editing; see EditText for a subclass that configures the text view for editing.

android.widget

Class TextView

java.lang.Object

└ [android.view.View](#)

└ android.widget.TextView

This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).

Doesn't it remind you the java.awt.Component?

All Implemented Interfaces:

[Drawable.Callback](#), [AccessibilityEventSource](#), [KeyEvent.Callback](#), [ViewTreeObserver.OnPreDrawListener](#)

Direct Known Subclasses:

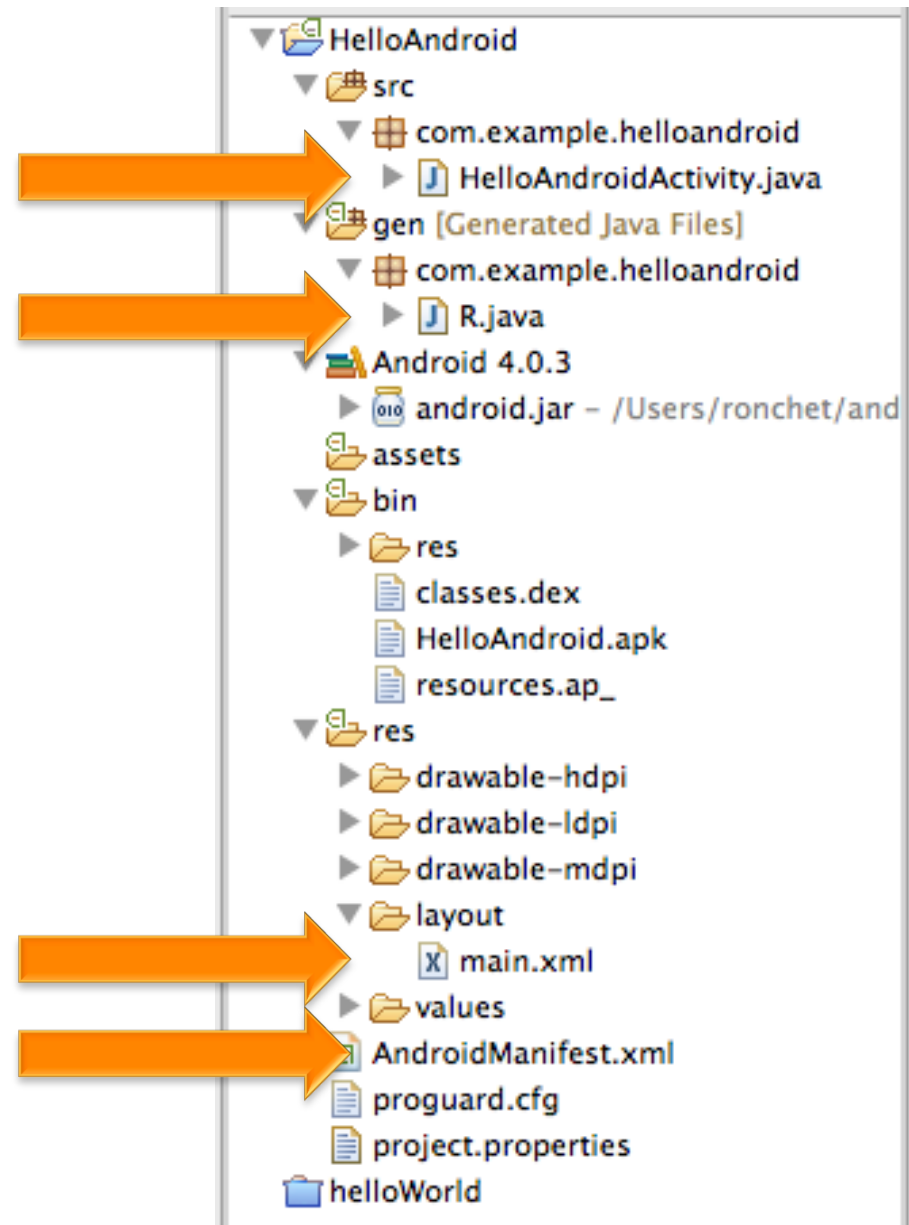
[Button](#), [CheckedTextView](#), [Chronometer](#), [DigitalClock](#), [EditText](#)

36

```
public class TextView
extends View
implements ViewTreeObserver.OnPreDrawListener
```



The project



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

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

```
  package="com.example.helloandroid"
```

```
  android:versionCode="1"
```

```
  android:versionName="1.0" >
```

```
  <uses-sdk android:minSdkVersion="15" />
```

```
  <application
```

```
    android:icon="@drawable/ic_launcher"
```

```
    android:label="@string/app_name" >
```

```
      <activity
```

```
        android:name=".HelloAndroidActivity"
```

```
        android:label="@string/app_name" >
```

```
        <intent-filter>
```

```
          <action android:name="android.intent.action.MAIN" />
```

```
          <category android:name="android.intent.category.LAUNCHER" />
```

```
        </intent-filter>
```

```
      </activity>
```

```
    </application>
```

```
</manifest>
```

Platform versions

Platform Version	API Level	VERSION_CODE
Android 4.0.3	15	ICE CREAM SANDWI
Android 4.0, 4.0.1, 4.0.2	14	ICE CREAM SANDWI
Android 3.2	13	HONEYCOMB MR2
Android 3.1.x	12	HONEYCOMB MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR MR1

Nov. 2011

Feb 2011

Dic 2010

Mag 2010

38



project.properties

```
# This file is automatically generated by Android Tools.  
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!  
#  
# This file must be checked in Version Control Systems.  
#  
# To customize properties used by the Ant build system use,  
# "ant.properties", and override values to adapt the script to your  
# project structure.  
  
# Project target.  
target=android-15
```

