Android Application Development

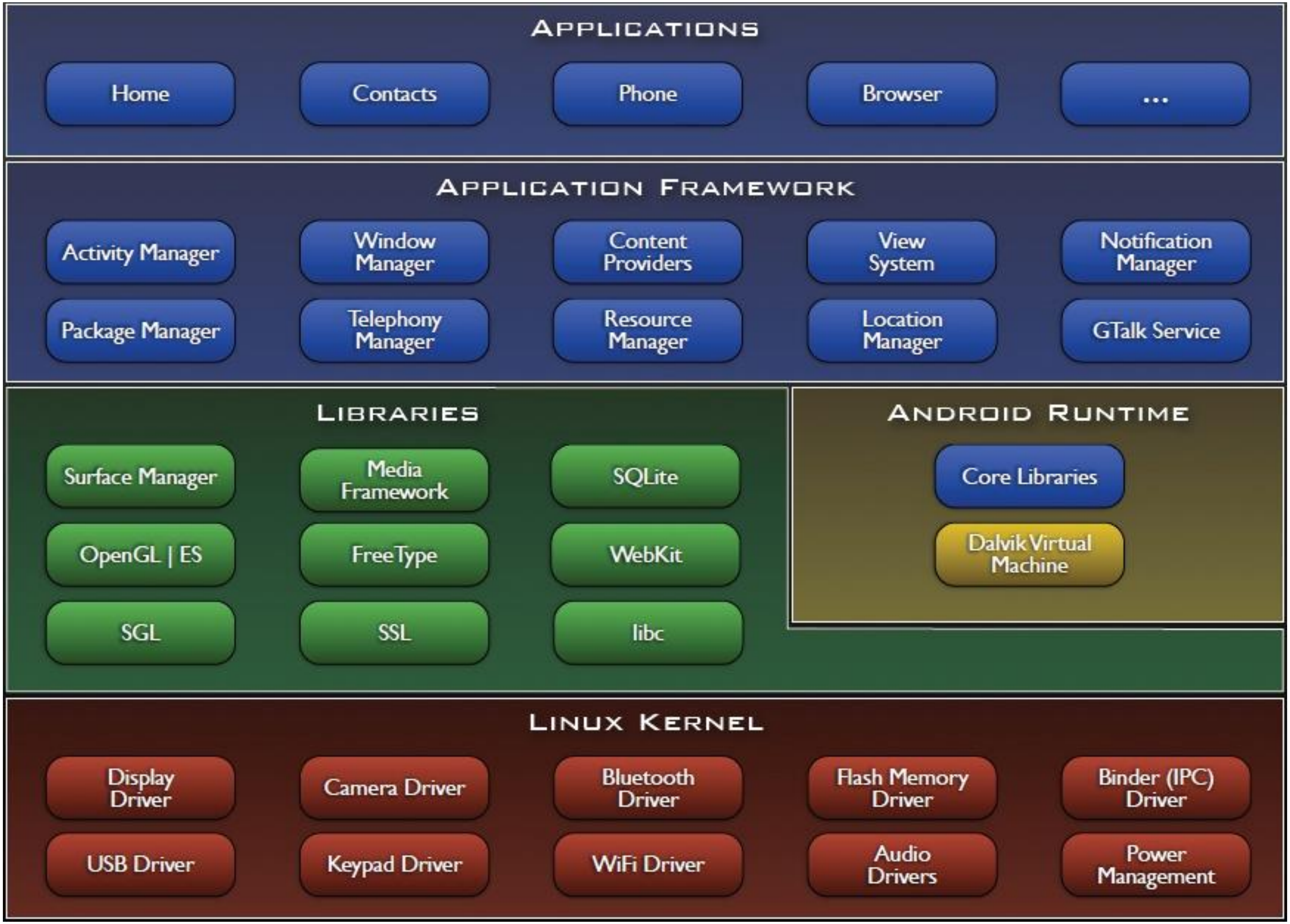
What is Android?

What is Android?

- An open source Linux-based operating system intended for mobile computing platforms

- Includes a Java API for developing applications

- It is **not** a device or product


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
GTalk Service
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Android Buzz words

- Android has a lot of “buzz” now
  - Newness
  - Coolness
  - Googleness
- UI and graphics made simple(r)
- Advanced Java skills

What Skills Will You Learn?

- Reinforce the basics: OOP, decomposition, etc.

- Separation of UI design and functionality
- XML and resource files
- Events and Listeners
- Callback methods
- Threads

What do you need in order to learn Android?

What Should You Already Know?

- Java!
  - inheritance, method overriding
  - interfaces, casting
  - exceptions
  - debugging
  - reading API documentation

- Eclipse
  - easy to pick up quickly, though

Need Phones?

- The emulator that is part of the Android toolset for Eclipse is quite good (though a bit slow)

- You may be able to get free “developer phones” from Google

VERSIONS OF ANDROID

➢ Android Beta

- ✓ First Version of Android.
- ✓ The focus of Android beta is testing incorporating usability.
- ✓ Android beta will generally have many more problems on speed and performance.

➢ Android Astro 1.0

- ✓ First full version of android.
- ✓ Released on September 23, 2008.
- ✓ Wi-Fi and Bluetooth support.
- ✓ Quite slow in operating.
- ✓ copy and paste feature in the web browser is not present.

Android Cupcake 1.5

- ✓ Released on April 30, 2009.
- ✓ Added auto-rotation option.
- ✓ Copy and Paste feature added in the web browser.
- ✓ Increased speed and performance but not upto required level.

Android Donut 1.6

- ✓ Released on September 15, 2009.
- ✓ Voice search and Search box were added.
- ✓ Faster OS boot times and fast web browsing experience.
- ✓ Typing is quite slower.

Android Éclair 2.0/2.1

- ✓ Released on ctober 26, 2009.
- ✓ Bluetooth 2.1 support.
- ✓ Improved typing speed on virtual keyboard, with smarter dictionary.
- ✓ no Adobe flash media support.

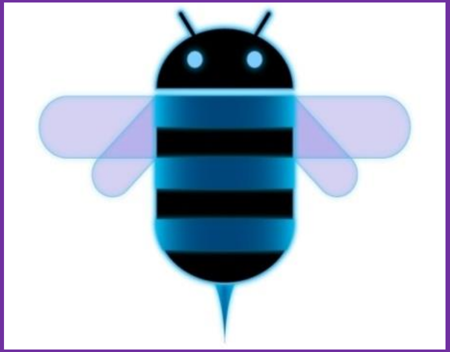
➢ Android Froyo 2.2

- ✓ Released on May 20, 2010.
- ✓ Support for Adobe Flash 10.1
- ✓ Improved Application launcher with better browser
- ✓ No internet calling.

➢ Android Gingerbread 2.3

- ✓ Released on December 6, 2010.
- ✓ Updated User Interface with high efficiency and speed
- ✓ Internet calling
- ✓ One touch word selection and copy/paste.
- ✓ New keyboard for faster word input.
- ✓ More successful version of Android than previous versions.
- ✓ not supports multi-core processors.

➢ Android Honeycomb 3.0



- ✓ Released on February 22, 2011.
- ✓ Support for multi-core processors
- ✓ Ability to encrypt all user data.
- ✓ This version of android is only available for tablets.

Android IceCreamSandwich(ICS) 4.0

- ✓ Released on November 14, 2011.
- ✓ Virtual button in the UI.
- ✓ A new typeface family for the UI, Roboto.
- ✓ Ability to shut down apps that are using data in the background.

Android JellyBean 4.1

- ✓ Released on June 27, 2012.
- ✓ Multichannel audio.
- ✓ Smoother user interface.

Android KitKat 4.4

- ✓ Released on June 25, 2014.
- ✓ UI updates for Google Maps navigation and alarms.
- ✓ Offline music playback.

LIMITATIONS:-

- Making source code available to everyone inevitably invites the attention of hackers.
- Android operating system uses more amount of battery as compared to normal mobile phones.
- As there are so many user sometimes it becomes difficult to connect all the users.
- As we call Android is world of applications we continuously need to connected with the internet which is not possible for all the users.

FUTURE SCOPE:-

- Android is now stepping up in next level of mobile internet.
- There are chances of Android Mobile sales becomes more then iPhone in next two years.
- Google may launch another version of android that starts L because Google is launching all the android versions in the alphabetical order.
- There are chances of Android may become the widely used operating system in world.

Android tools.

What's given by Android for development?

- ADB.
- AVD manager and emulator.
- Android devices.
- SDK.

Android tools – **ADB.**

The “Android Debug Bridge” is:

•A command line tool.

•handles communication with a development unit

•a client-server program that includes three components:

1. A client, which runs on your computer.
2. A server, which runs as a background process.
3. A daemon, which runs as a background process on each development unit.

Android tools – **Emulator.**

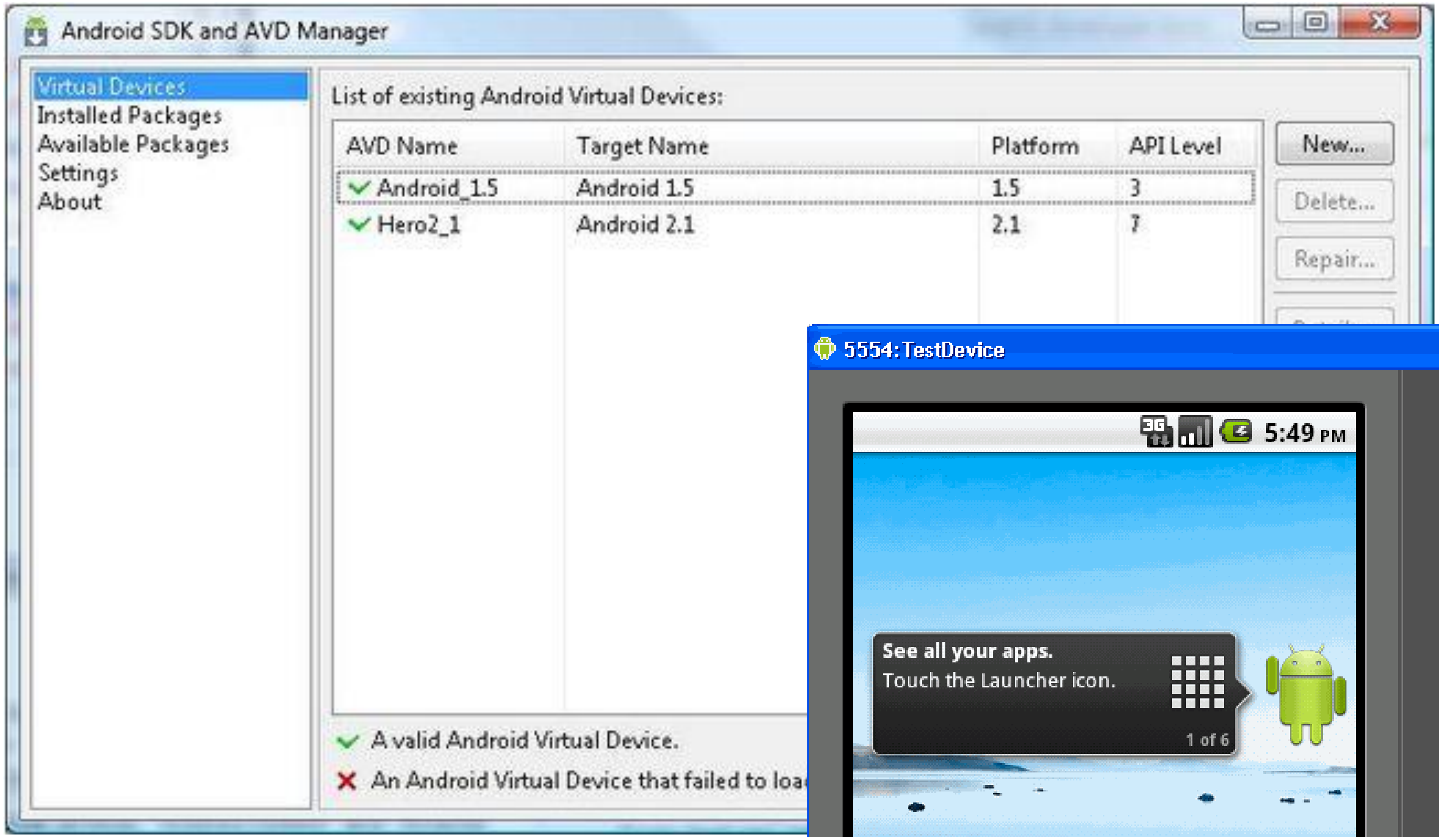
Runs on your computer.

Mouse/keyboard to mimic touch and key events.

Mimics all hardware and software features.

- Different screen sizes and densities.
- Emulating phone events (SMS, phone calls).
- Spoofing location events.
- Different API levels.

Android tools – **Emulator.**


Android SDK and AVD Manager

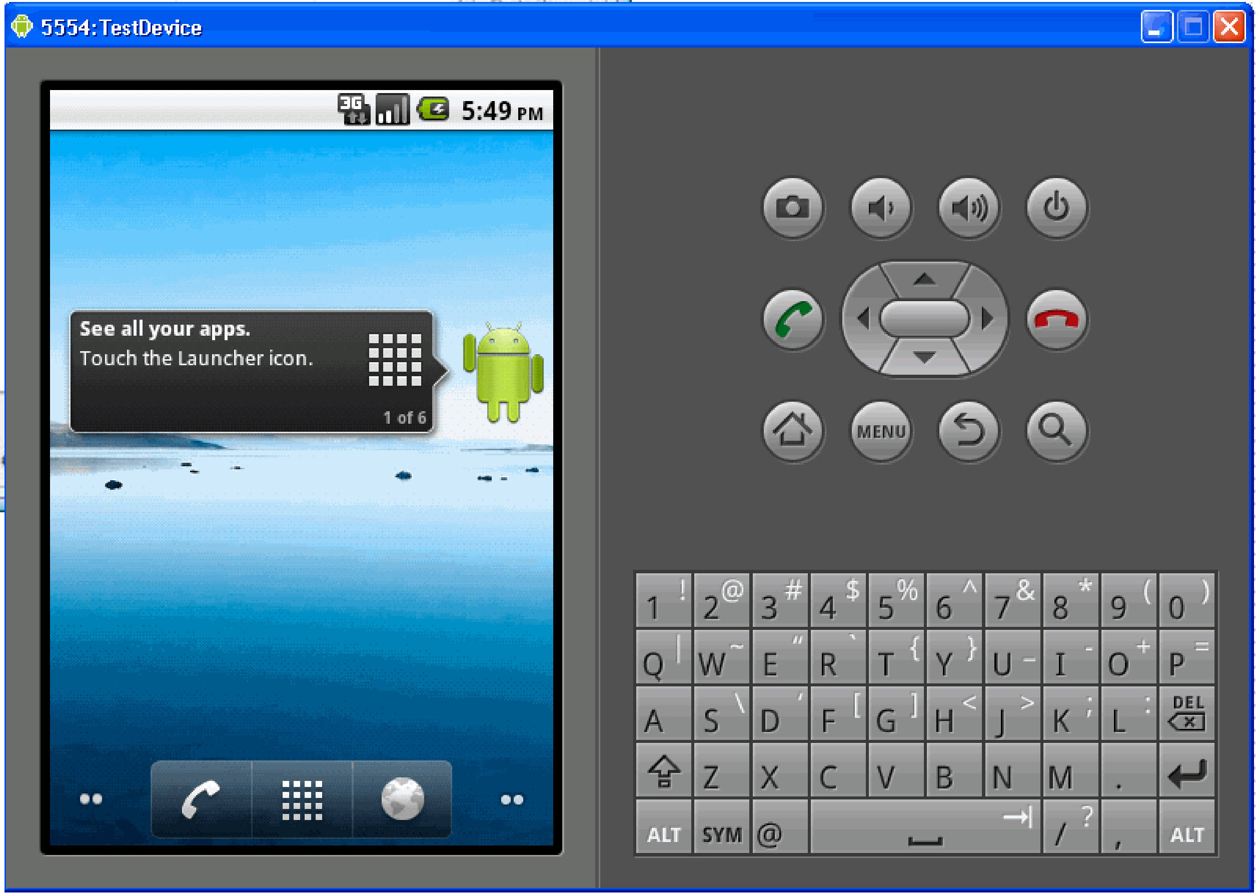
Virtual Devices
Installed Packages
Available Packages
Settings
About

List of existing Android Virtual Devices:

AVD Name	Target Name	Platform	API Level
✓ Android_1.5	Android 1.5	1.5	3
✓ Hero2_1	Android 2.1	2.1	7

New...
Delete...
Repair...

✓ A valid Android Virtual Device.
✗ An Android Virtual Device that failed to loa


5554:TestDevice

5:49 PM

See all your apps.
Touch the Launcher icon.
1 of 6

Android tools – **Device.**

- All android devices are debug-able via USB “out of the box” (no root is needed).
- To enable:
  - Settings>applications:
    - Allow unknown resources.
  - Settings>applications>development:
    - Allow USB debugging.
    - Stay awake.

Android tools – **SDK.**

Programming language: JAVA + XML.

Create standard android application by using and utilizing the applications framework.

Most popular IDE:

- Eclipse + ADT.

Other possible IDEs:

- intellij.
- Net beans.

Assignment

What is smart device?

What is smart device computing?

Features of Smart device OS.

Brief note on app development languages.

What is Android?

Explain Android Architecture.

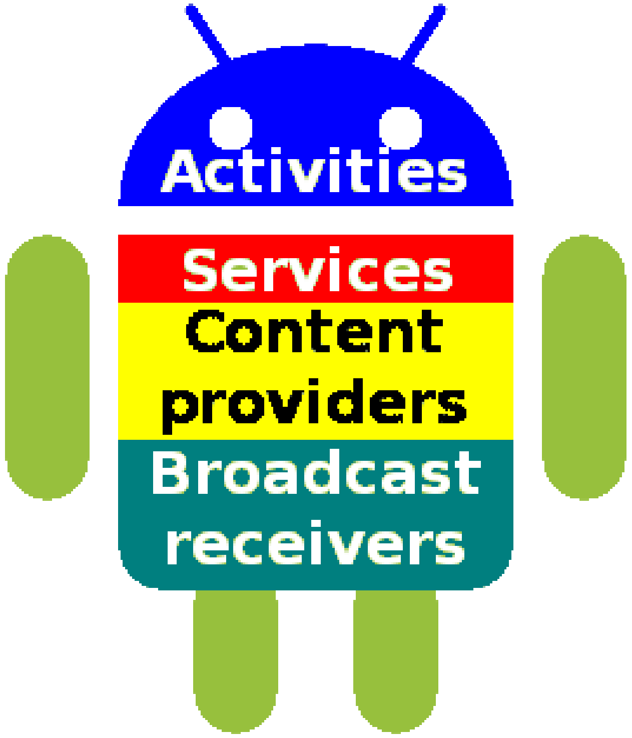
Android versions with features.

Android app development tools

Android building blocks

Android building blocks

The components of an android application and their connecting methods.



The (Android) application.

An application consists of

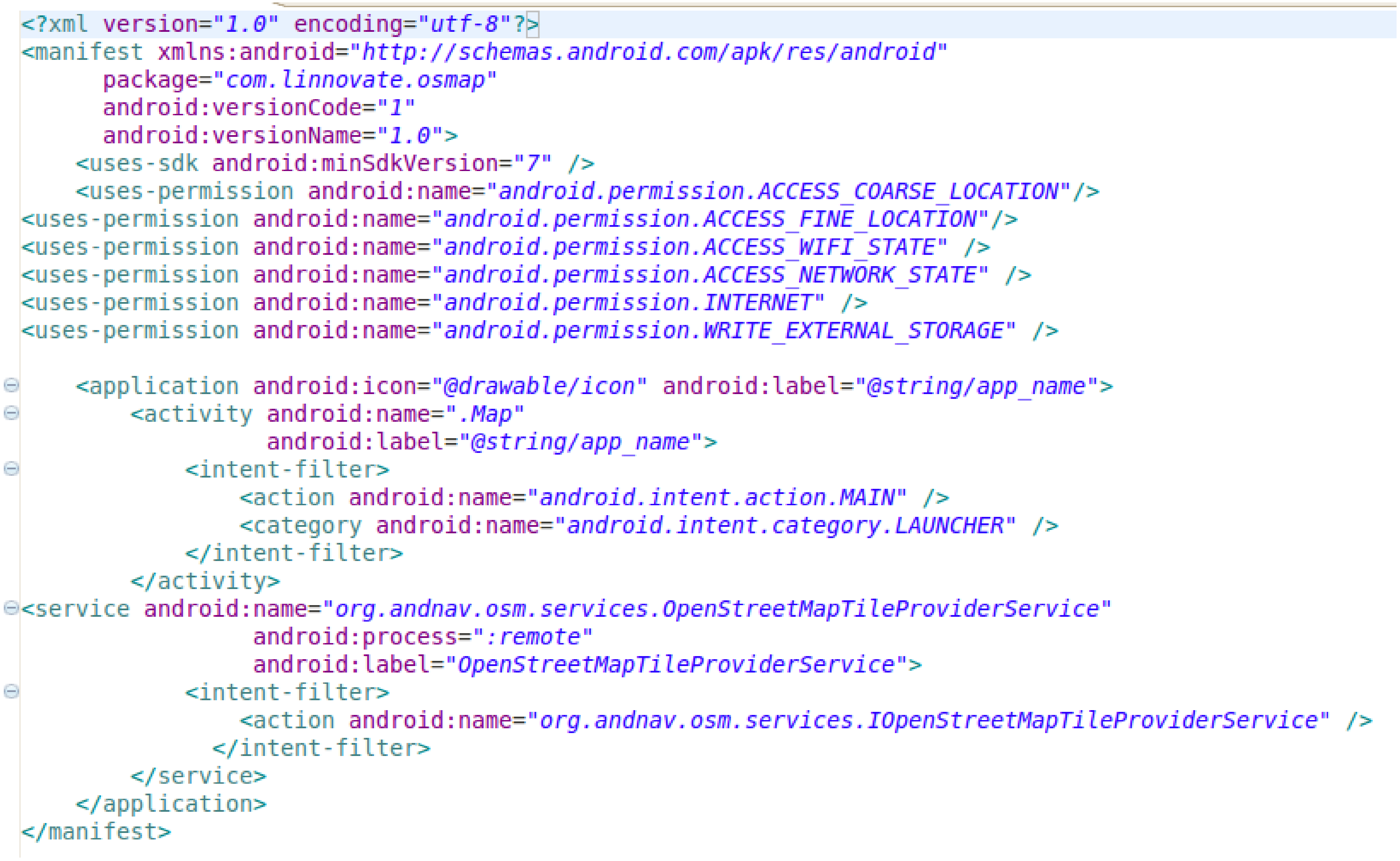
- A manifest xml.
- A set of:
  - Activities.
  - Services.
  - Broadcast receivers.
  - Content providers.

The Manifest

The Manifest is essentially a list of components that the application uses or requires, such as:

- The components of the applications.
- User permissions .
- The minimum API level.
- Hardware and software features.
- API dependencies.

A manifest example



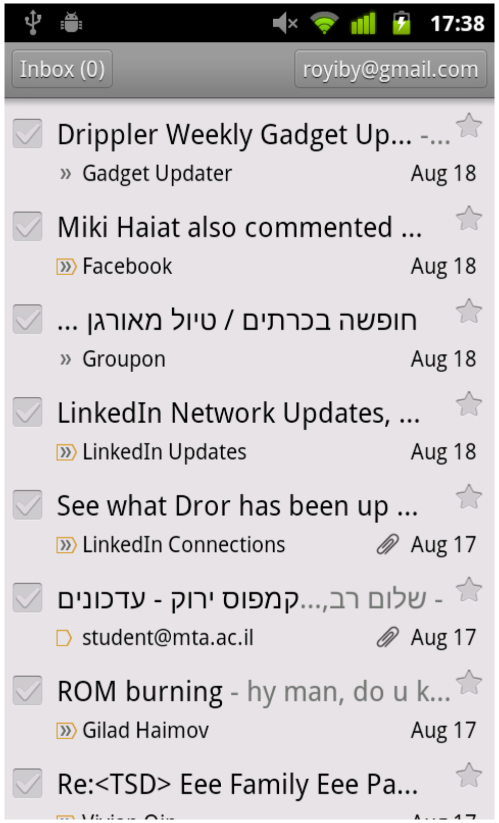
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.linnovate.osmap"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Map"
                  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="org.andnav.osm.services.OpenStreetMapTileProviderService"
                    android:process=":remote"
                    android:label="OpenStreetMapTileProviderService">
            <intent-filter>
                <action android:name="org.andnav.osm.services.IOpenStreetMapTileProviderService" />
              </intent-filter>
        </service>
    </application>
</manifest>
```

. The Activity

- An *activity* represents a single screen with a user interface (Java file + xml layout file).
- An application usually consists of multiple activities that are loosely bound to each other.
- One Activity is flagged as “main” and it is started at the application launch time.
- An Activity can launch other activities to create an app UI workflow.
- The Activity has an Intent attribute that determines how android treats it.

Activity explained with an example.

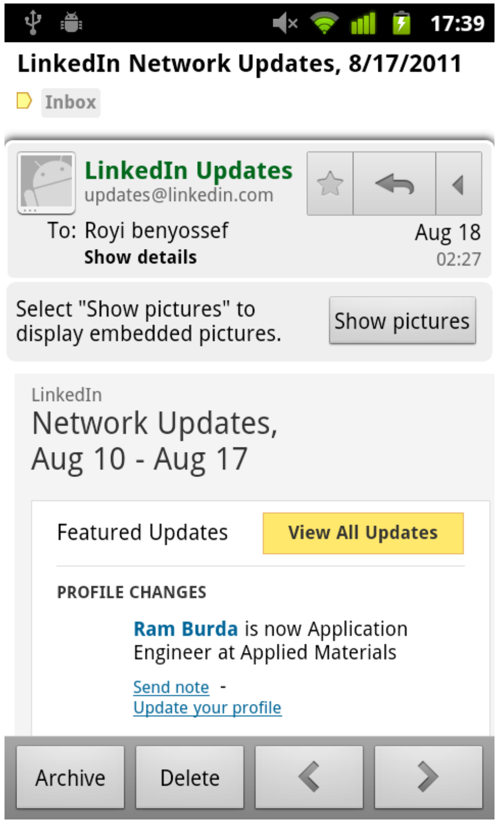


17:38

Inbox (0) royiby@gmail.com

Drippler Weekly Gadget Up... -...	Gadget Updater	Aug 18
Miki Haiat also commented ...	Facebook	Aug 18
חופשה בכרתים / טיול מאורגן ...	Groupon	Aug 18
LinkedIn Network Updates, ...	LinkedIn Updates	Aug 18
See what Dror has been up ...	LinkedIn Connections	Aug 17
- שלום רב,...קמפוס ירוק - עדכונים	student@mta.ac.il	Aug 17
ROM burning - hy man, do u k...	Gilad Haimov	Aug 17
Re:<TSD> Eee Family Eee Pa...		

A screen that displays a list of emails.



17:39

LinkedIn Network Updates, 8/17/2011

Inbox

LinkedIn Updates
updates@linkedin.com
To: Royi benyossef
Show details
Aug 18
02:27

Select "Show pictures" to display embedded pictures. Show pictures

LinkedIn
Network Updates,
Aug 10 - Aug 17

Featured Updates View All Updates

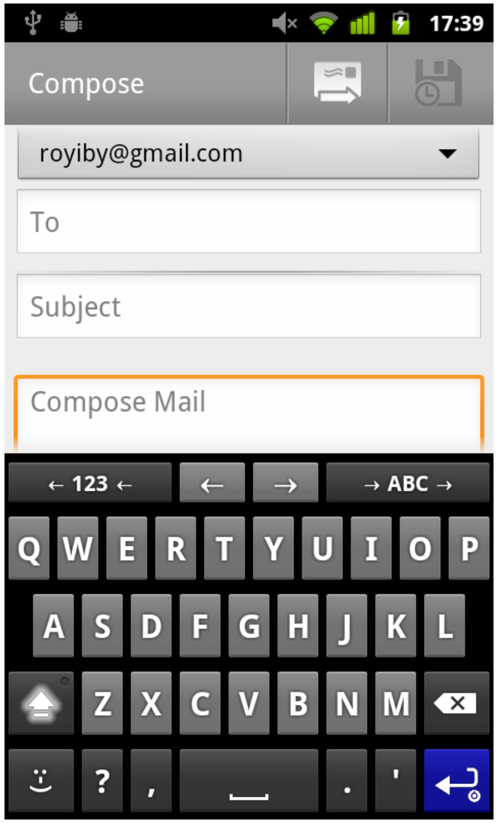
PROFILE CHANGES

Ram Burda is now Application Engineer at Applied Materials

Send note -
Update your profile

Archive Delete

A screen that displays a single email.



17:39

Compose

royiby@gmail.com

To

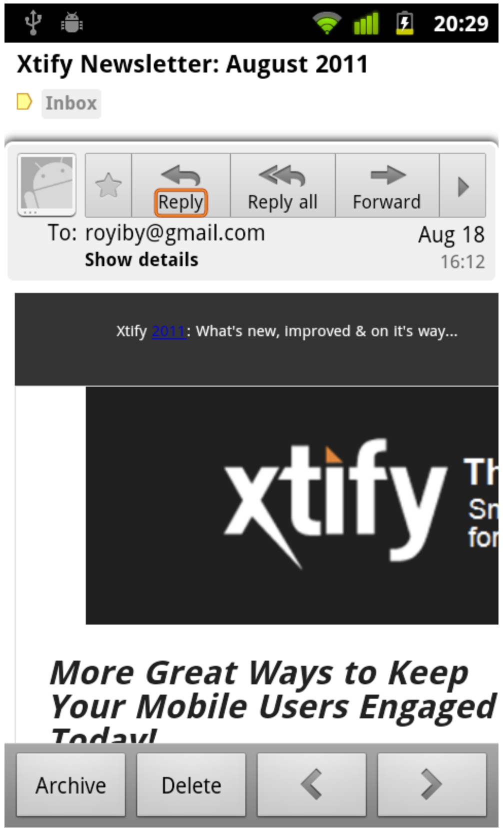
Subject

Compose Mail

A screen that enables you to compose an email.

Each and every one of these is an Activity!

Starting an Activity.



20:29

Xtify Newsletter: August 2011

Inbox

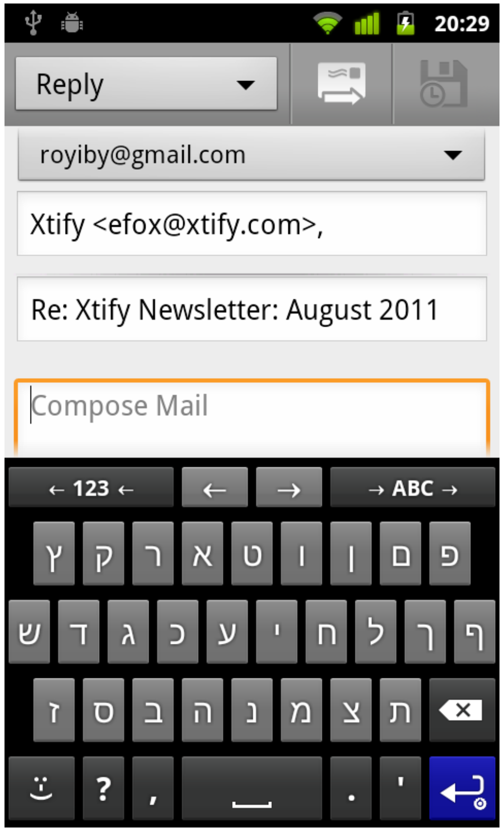
To: royiby@gmail.com Aug 18

Show details 16:12

Xtify 2011: What's new, improved & on it's way...

More Great Ways to Keep Your Mobile Users Engaged Today!

When you press reply:



20:29

royiby@gmail.com

Xtify <efox@xtify.com>,

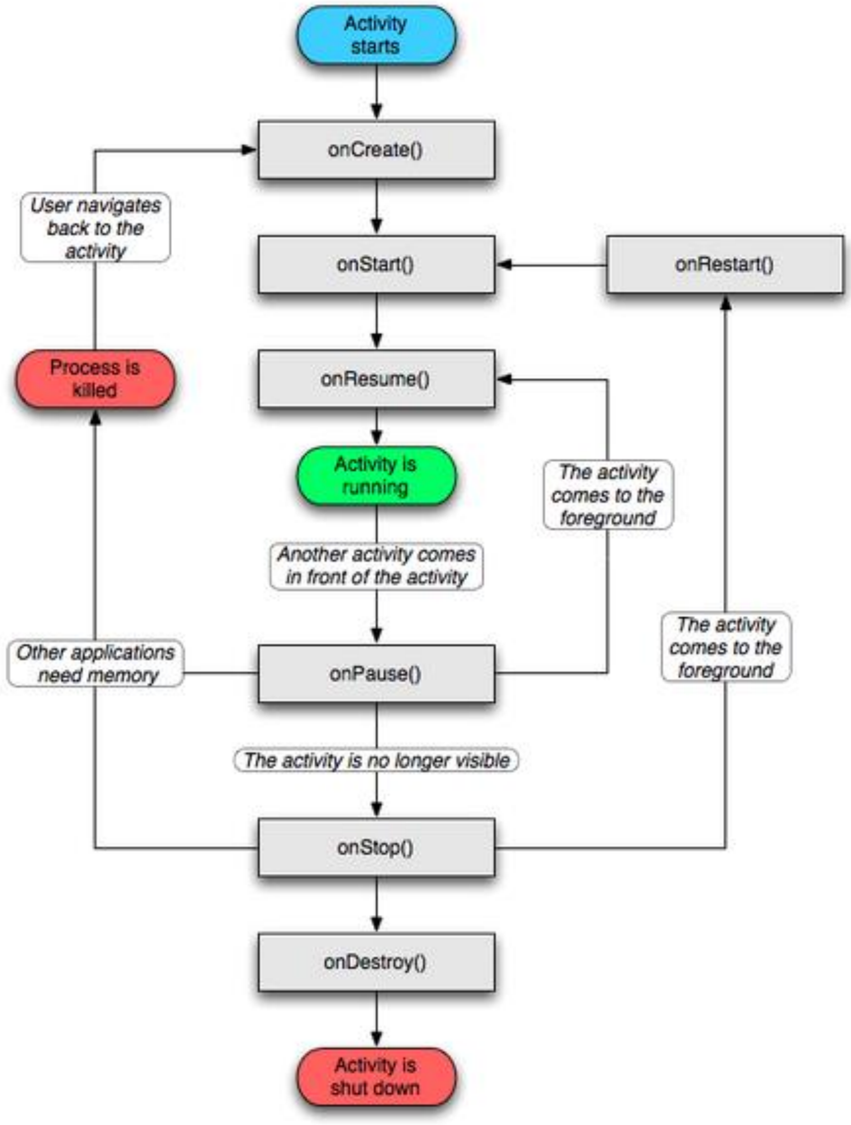
Re: Xtify Newsletter: August 2011

Compose Mail

- The compose Activity gets called.
- The recipients are sent via the "Extras" attribute of the intent.

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

The Activity life-cycle.


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

- There are 6 stages in the Android Activity life cycle.
- These stages allow you to perform tasks in their correct time of the workflow.

The Service

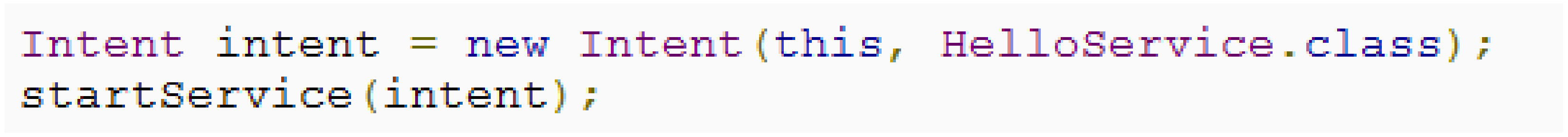
An application component that can perform long-running operations in the background and does not provide a user interface.

The service has two forms:

- A "started“ service is called by an app component and runs in the background even if the application that called it was closed.
- A “bound” service is bound by an app component and is used as a client-server bridge, this sort of service is terminated when the application that is bound to it is closed.

Starting a service.

From the activity perform:

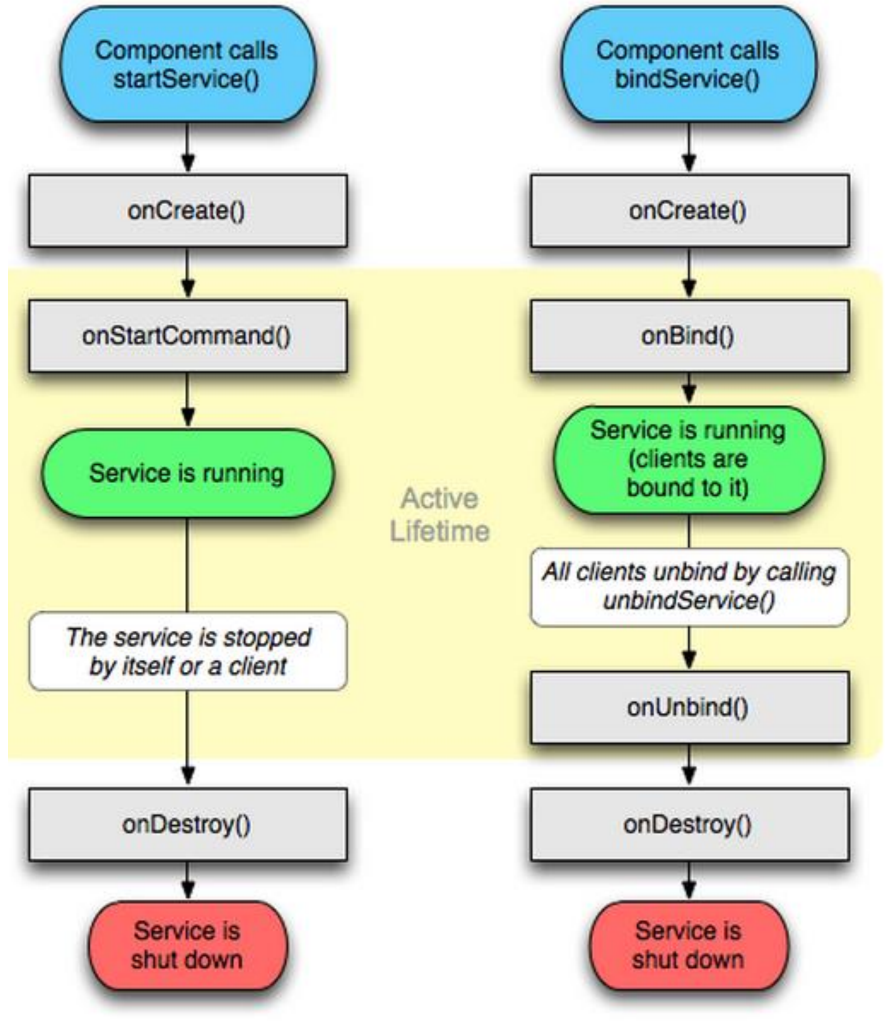


```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

The startService() method returns immediately.

If the service is not already running, the android system calls onCreate(), then calls onStartCommand() else it only calls the onStartCommand() method.

The service life cycle.


Component calls startService()
onCreate()
onStartCommand()
Service is running
The service is stopped by itself or a client
onDestroy()
Service is shut down
Active Lifetime
Component calls bindService()
onCreate()
onBind()
Service is running (clients are bound to it)
All clients unbind by calling unbindService()
onUnbind()
onDestroy()
Service is shut down

The same principle as in the activity life cycle applies here only the stages are different.

As you can see the bound and started services have a similar cycle with a twist:

The bound service is only “alive” while the binding application is “alive”.

The Broadcast receivers.

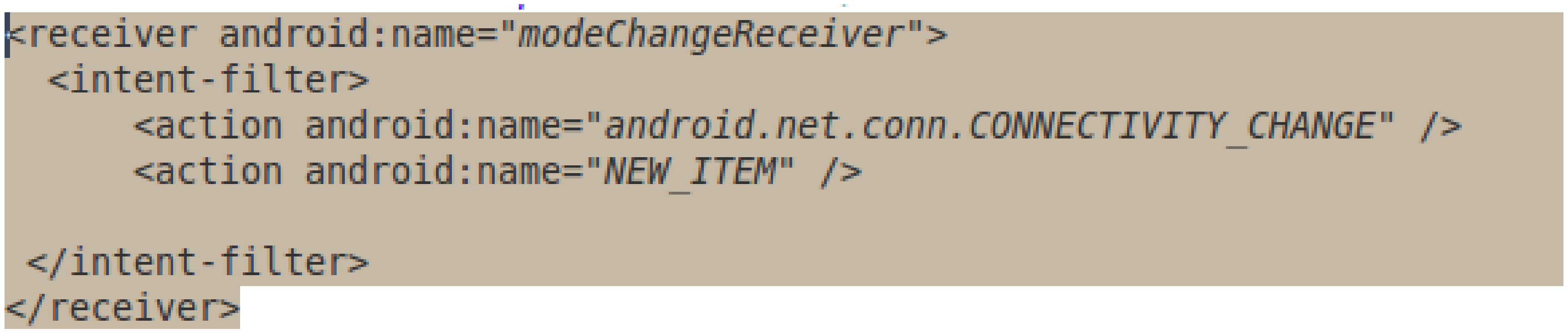
a component that responds to system-wide broadcast announcements.

The announcement can originate either from a system event (screen on/off) or a custom activity event.

When this protocol is customized it facilitates message sending between applications and activities that should work together.

The Broadcast receivers - example.

The manifest tag:

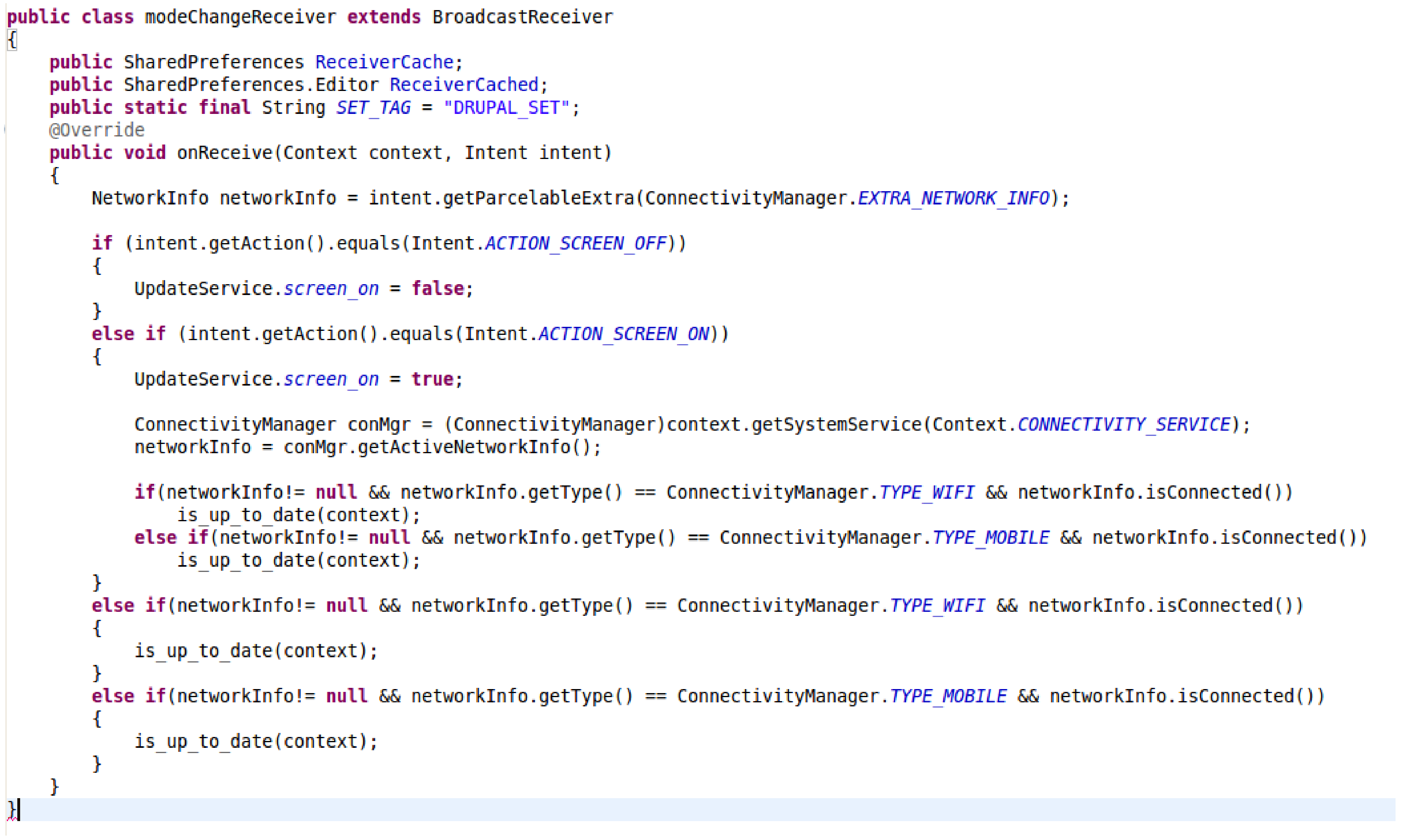


```
<receiver android:name="modeChangeReceiver">
  <intent-filter>
      <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
      <action android:name="NEW_ITEM" />

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

The Broadcast receivers - example.

The broadcast receiver file:



```
public class modeChangeReceiver extends BroadcastReceiver
{
    public SharedPreferences ReceiverCache;
    public SharedPreferences.Editor ReceiverCached;
    public static final String SET_TAG = "DRUPAL_SET";
    @Override
    public void onReceive(Context context, Intent intent)
    {
        NetworkInfo networkInfo = intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);

        if (intent.getAction().equals(Intent.ACTION_SCREEN_OFF))
        {
            UpdateService.screen_on = false;
        }
        else if (intent.getAction().equals(Intent.ACTION_SCREEN_ON))
        {
            UpdateService.screen_on = true;

            ConnectivityManager conMgr = (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
            networkInfo = conMgr.getActiveNetworkInfo();

            if(networkInfo!= null && networkInfo.getType() == ConnectivityManager.TYPE_WIFI && networkInfo.isConnected())
                is_up_to_date(context);
            else if(networkInfo!= null && networkInfo.getType() == ConnectivityManager.TYPE_MOBILE && networkInfo.isConnected())
                is_up_to_date(context);
        }
        else if(networkInfo!= null && networkInfo.getType() == ConnectivityManager.TYPE_WIFI && networkInfo.isConnected())
        {
            is_up_to_date(context);
        }
        else if(networkInfo!= null && networkInfo.getType() == ConnectivityManager.TYPE_MOBILE && networkInfo.isConnected())
        {
            is_up_to_date(context);
        }
    }
}
```

The Content providers.

Components that manage a shared set of application data.

The data is stored at one of the following formats:

1. In the file system.
2. In an SQLite database.
3. on the web.
4. More..

Essentially you can store it in any persistent storage location your application can access.

The Content providers.

Through the content provider, other applications can query or even modify the data (if the content provider allows it).

providers are also useful for reading and writing data that is private to your application and is not shared.

The Content providers – examples.

Here’s a few examples for system content providers:

Contacts.

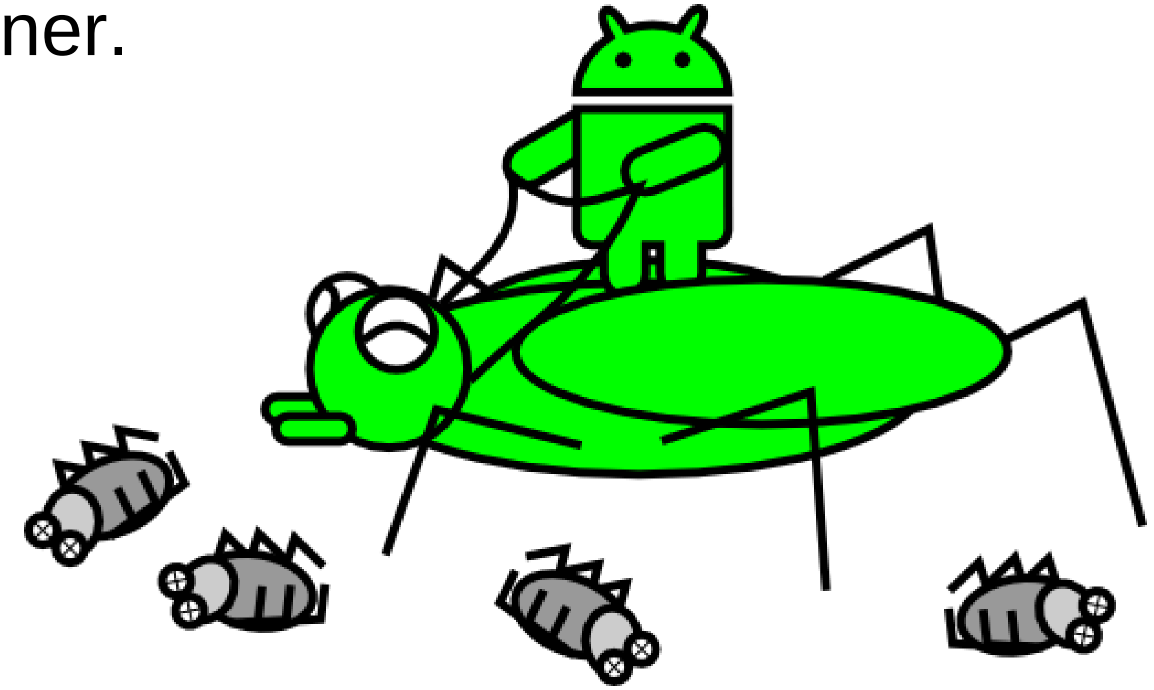
Text messages.

Phone calls.

Android debug tools.

Android provides a few precious tools for that:

- DDMS.
- Logcat.
- Hierarchy viewer.
- Monkey & monkey runner.

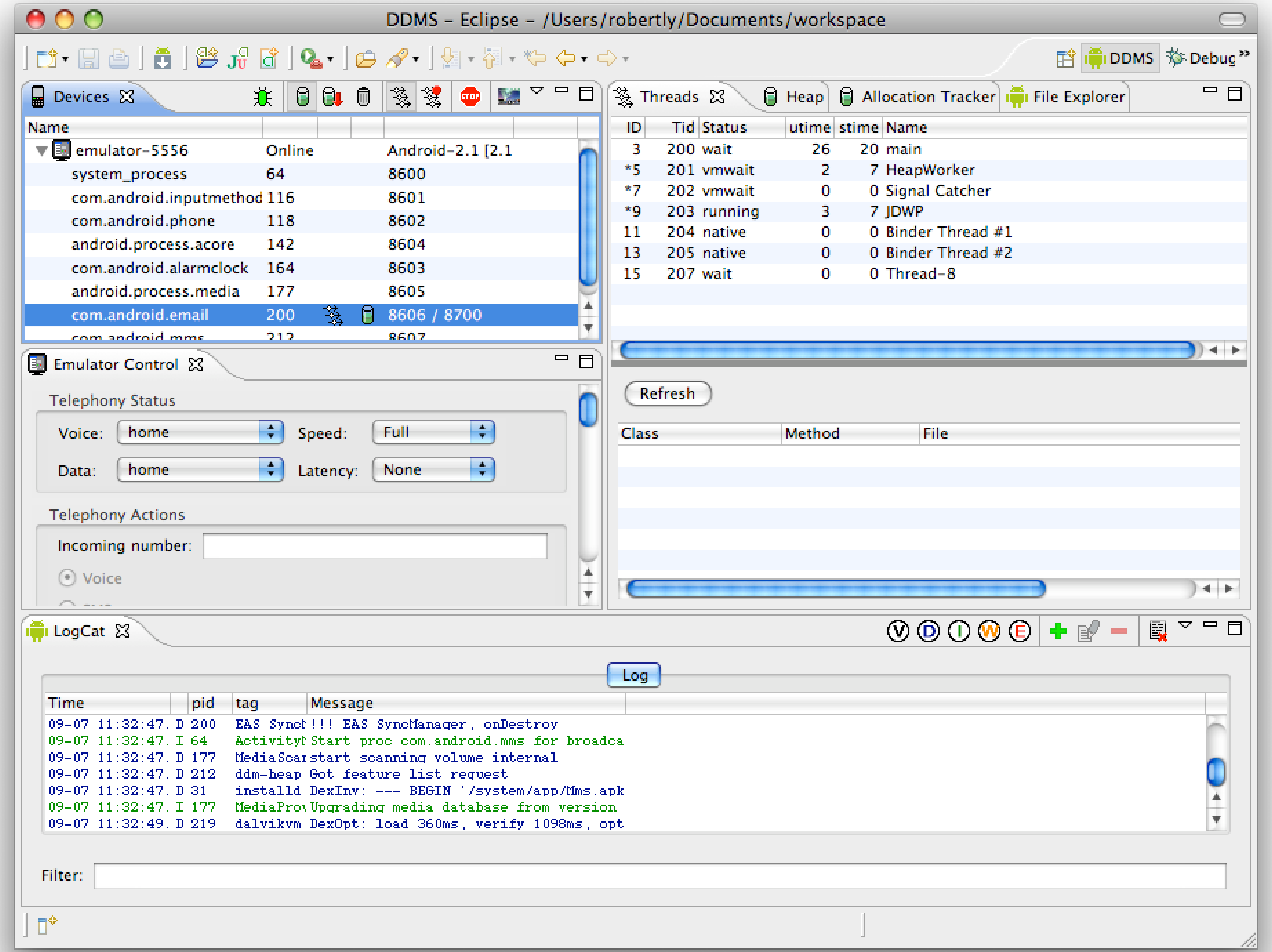


Android debug tools – **DDMS.**

The “Dalvik Debug Monitor Server (DDMS)” is:
A debugging tool that provides:

1. Port forwarding services.
2. Screen capture.
3. thread and heap information.
4. Logcat and process logging.
5. Mobile data spoofing (location, calls and SMS)
6. Memory dump.

Android debug tools – **DDMS.**



DDMS – Eclipse – /Users/robertly/Documents/workspace

Devices

Name			
emulator-5556	Online		Android-2.1 [2.1
system_process	64		8600
com.android.inputmethod	116		8601
com.android.phone	118		8602
android.process.acore	142		8604
com.android.alarmclock	164		8603
android.process.media	177		8605
com.android.email	200		8606 / 8700
com.android.mms	212		8607

Threads

ID	Tid	Status	utime	stime	Name
3	200	wait	26	20	main
*5	201	vmwait	2	7	HeapWorker
*7	202	vmwait	0	0	Signal Catcher
*9	203	running	3	7	JDWP
11	204	native	0	0	Binder Thread #1
13	205	native	0	0	Binder Thread #2
15	207	wait	0	0	Thread-8

Emulator Control

Telephony Status

Voice: home   Speed: Full

Data: home   Latency: None

Telephony Actions

Incoming number:

Voice

LogCat

```
Time              pid  tag       Message
09-07 11:32:47. D 200  EAS Sync! !!! EAS SyncManager, onDestroy
09-07 11:32:47. I 64   Activity! Start proc com.android.mms for broadca
09-07 11:32:47. D 177  MediaScar start scanning volume internal
09-07 11:32:47. D 212  ddm-heap Got feature list request
09-07 11:32:47. D 31   installd DexInv: --- BEGIN '/system/app/Mms.apk
09-07 11:32:47. I 177  MediaPro\ Upgrading media database from version
09-07 11:32:49. D 219  dalvikvm DexOpt: load 360ms, verify 1098ms, opt
```

Filter:

Android debug tools – **Logcat.**

A mechanism for collecting and viewing system debug output.

Filter levels:

- V — Verbose (lowest priority)
- D — Debug
- I — Info
- W — Warning
- E — Error

Invoke logs with a custom TAG in your app to monitor its operation:

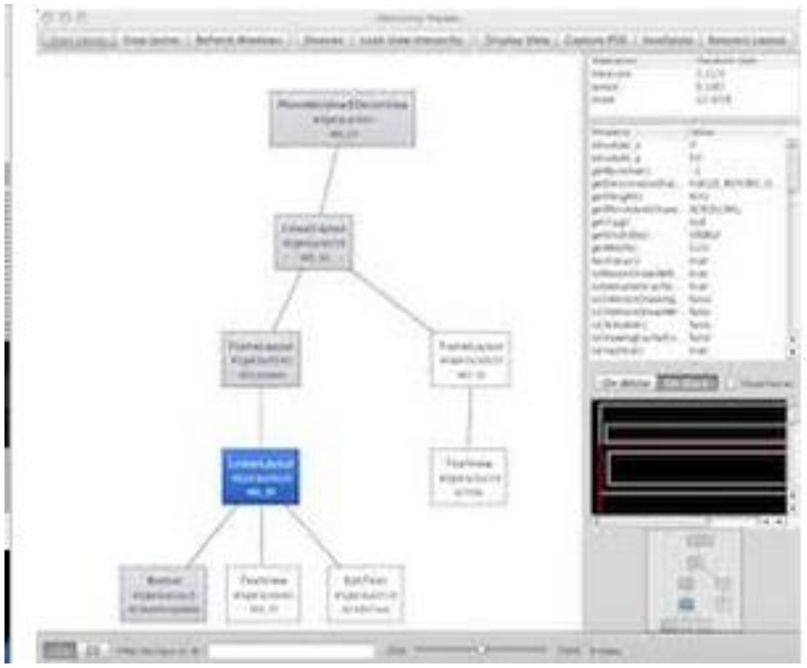
```
Log.v(TAG, "index=" + i);
```

Android debug tools – **Hierarchy viewer.**

Displays the View objects that form the UI of the Activity that is running on your device or emulator.

Start command: <sdk>/tools/hierarchyviewer

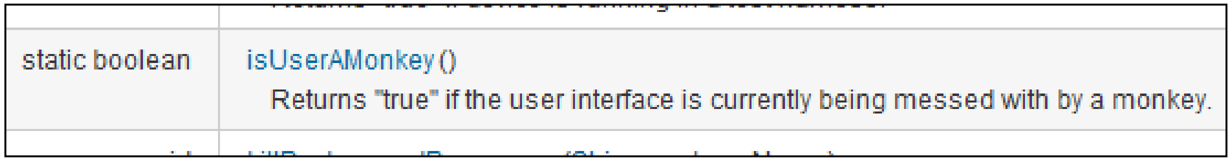
If #levels > 7 -> NOT GOOD!



Android debug tools – **Monkey.**

Creates random UI events.
Great UI durability test.
You can configure:

- The amount of event.
- Percentage of touch/motion/track
- The monkey’s “aggressiveness” (
  exception/error/security error etc.).



static boolean	isUserAMonkey () Returns "true" if the user interface is currently being messed with by a monkey.

Android debug tools – **Monkey runner.**

Enables using the monkey tool in a more precise way:

- Write a python code.
- The monkey performs it.

Great for recreating bugs and create tests to see when they are resolved.

“Hello, Android”

Creating Your First(?) Android App

1. Set up your development environment

2. Create a new Android project in Eclipse

3. Run it in the emulator

1. Set Up Your Android Environment

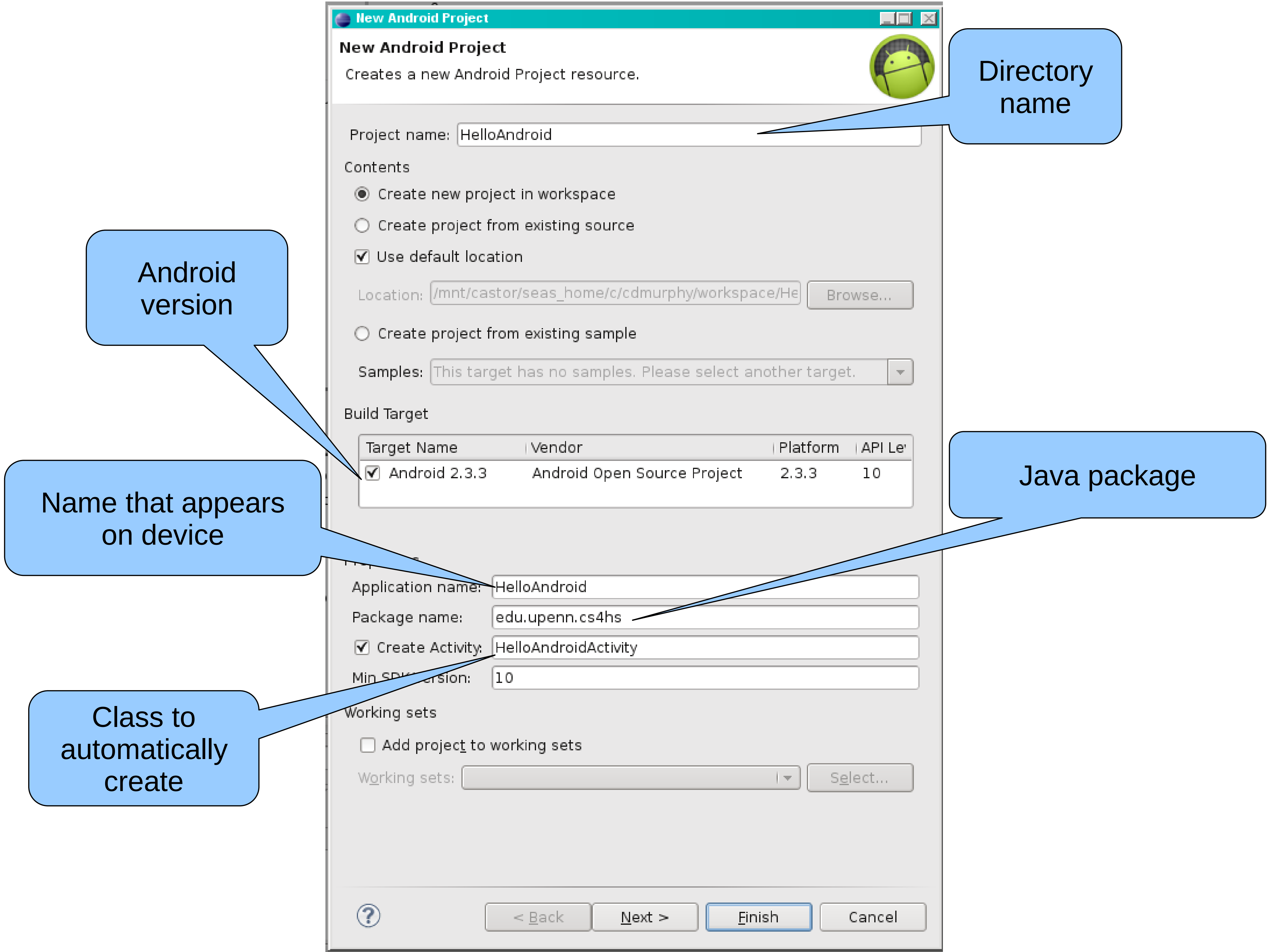
- http://developer.android.com/sdk

- Install Eclipse
- Install Android SDK (Android libraries)
- Install ADT plugin (Android development tools)

- Create AVD (Android virtual device)

2. Create an Android Project in Eclipse

- File → New → Project
- Select “Android Project”
- Fill in Project details...



New Android Project

New Android Project

Creates a new Android Project resource.

Project name: HelloAndroid

Contents

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

Location: /mnt/castor/seas_home/c/cdmurphy/workspace/He — Browse...

- ( ) Create project from existing sample

Samples: This target has no samples. Please select another target.

Build Target

	Target Name	Vendor	Platform	API Le
✓	Android 2.3.3	Android Open Source Project	2.3.3	10

Application name: HelloAndroid

Package name: edu.upenn.cs4hs

[✓] Create Activity: HelloAndroidActivity

Min SDK Version: 10

Working sets

[ ] Add project to working sets

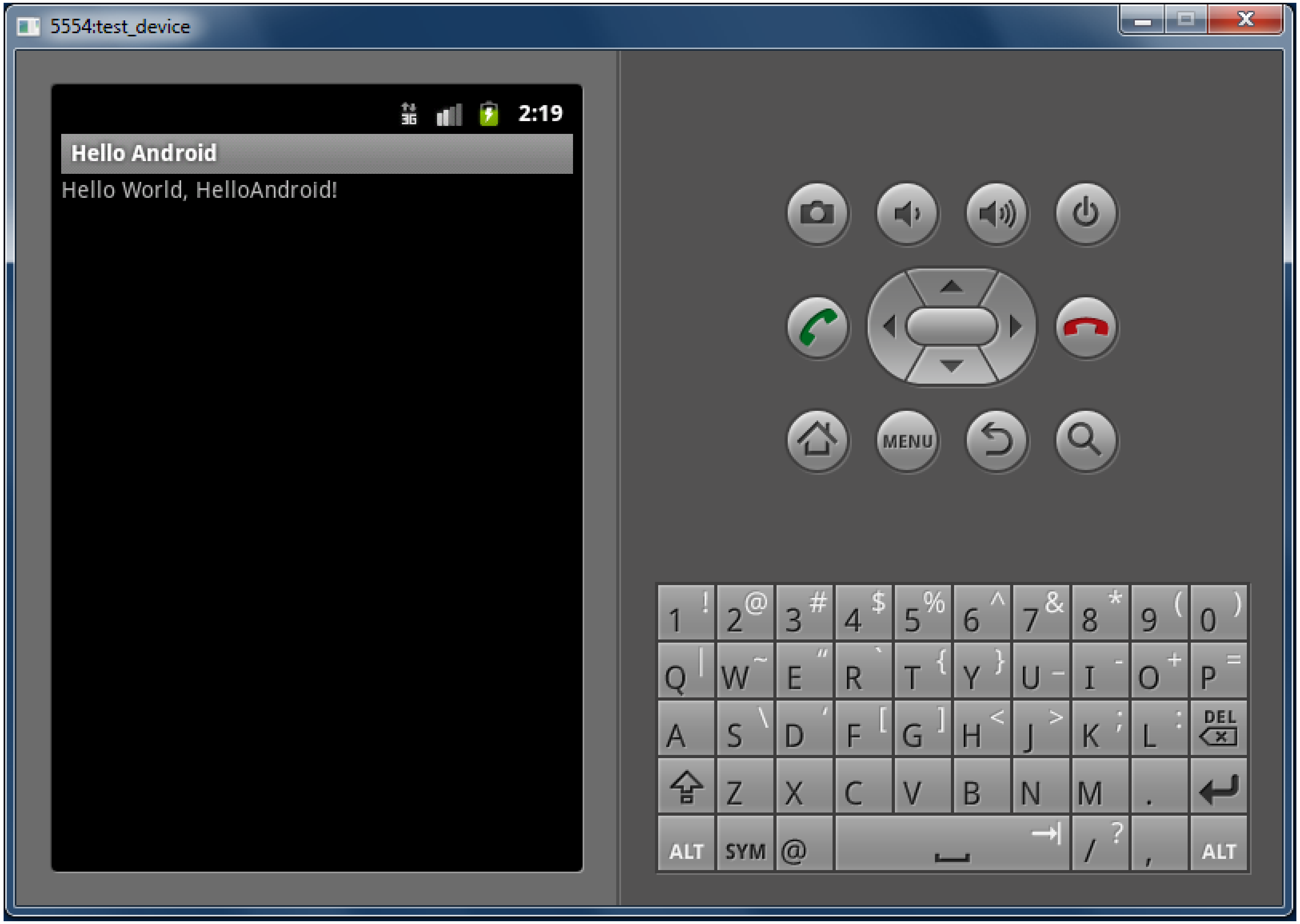
Working sets: — Select...

3. Run the Android Application

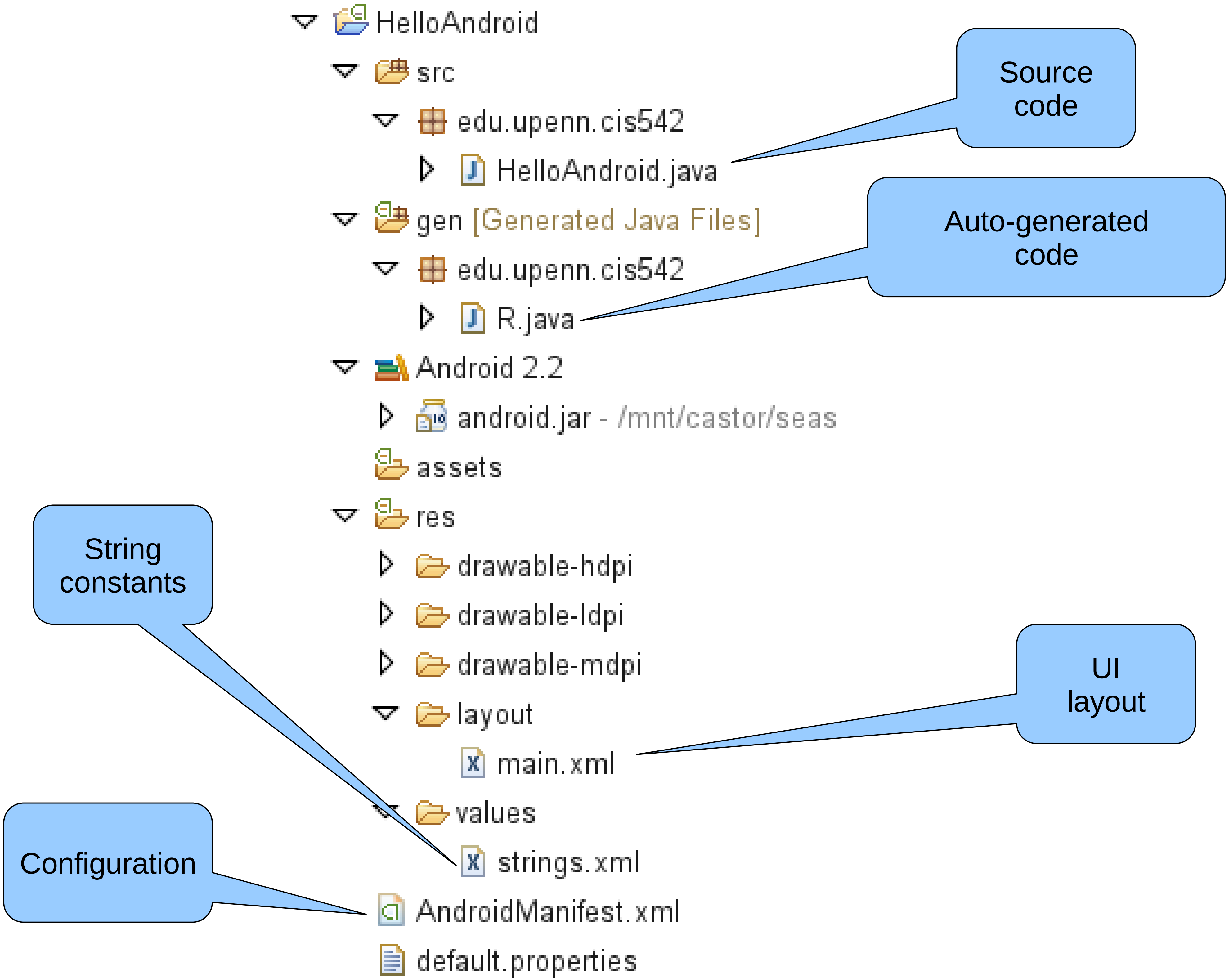
- Run → Run (or click the "Run" button)
- Select "Android Application"

- The emulator may take a few minutes to start, so be patient!

- You don't need to restart the emulator when you have a new version of your application



Hello World, HelloAndroid!



```
HelloAndroid
  src
    edu.upenn.cis542
      HelloAndroid.java
  gen [Generated Java Files]
    edu.upenn.cis542
      R.java
  Android 2.2
    android.jar - /mnt/castor/seas
  assets
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    layout
      main.xml
    values
      strings.xml
  AndroidManifest.xml
  default.properties
```

HelloAndroid.java

```java
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);
   }
}
```

main.xml

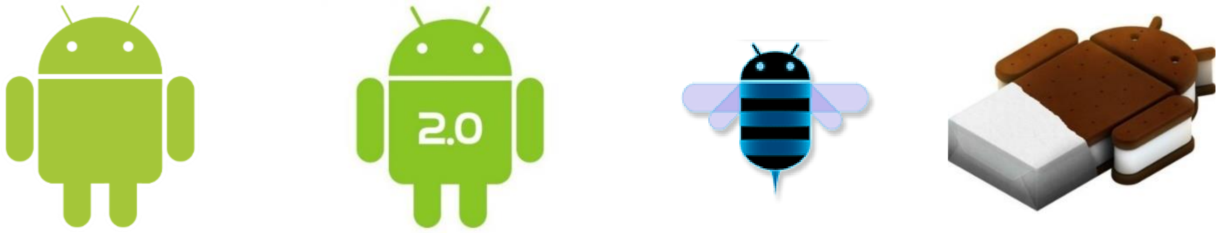
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
>
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"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

strings.xml

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

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
   package="edu.upenn.cis542"
   android:versionCode="1"
   android:versionName="1.0">
   <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>
   </application>
</manifest>
```



Programming with Android: User Interface & Layouts

Introduction to User Interface (UI)

- The user interface of an application is everything that the user can see and interact with.
- Android provides a variety of pre-built UI components such as structured layout objects and UI controls.
- E.g. : dialogs, notifications, and menus.

Two UI Approaches

Procedural	Declarative
Write Java code.	Write XML code Similar to HTML of a webpage.

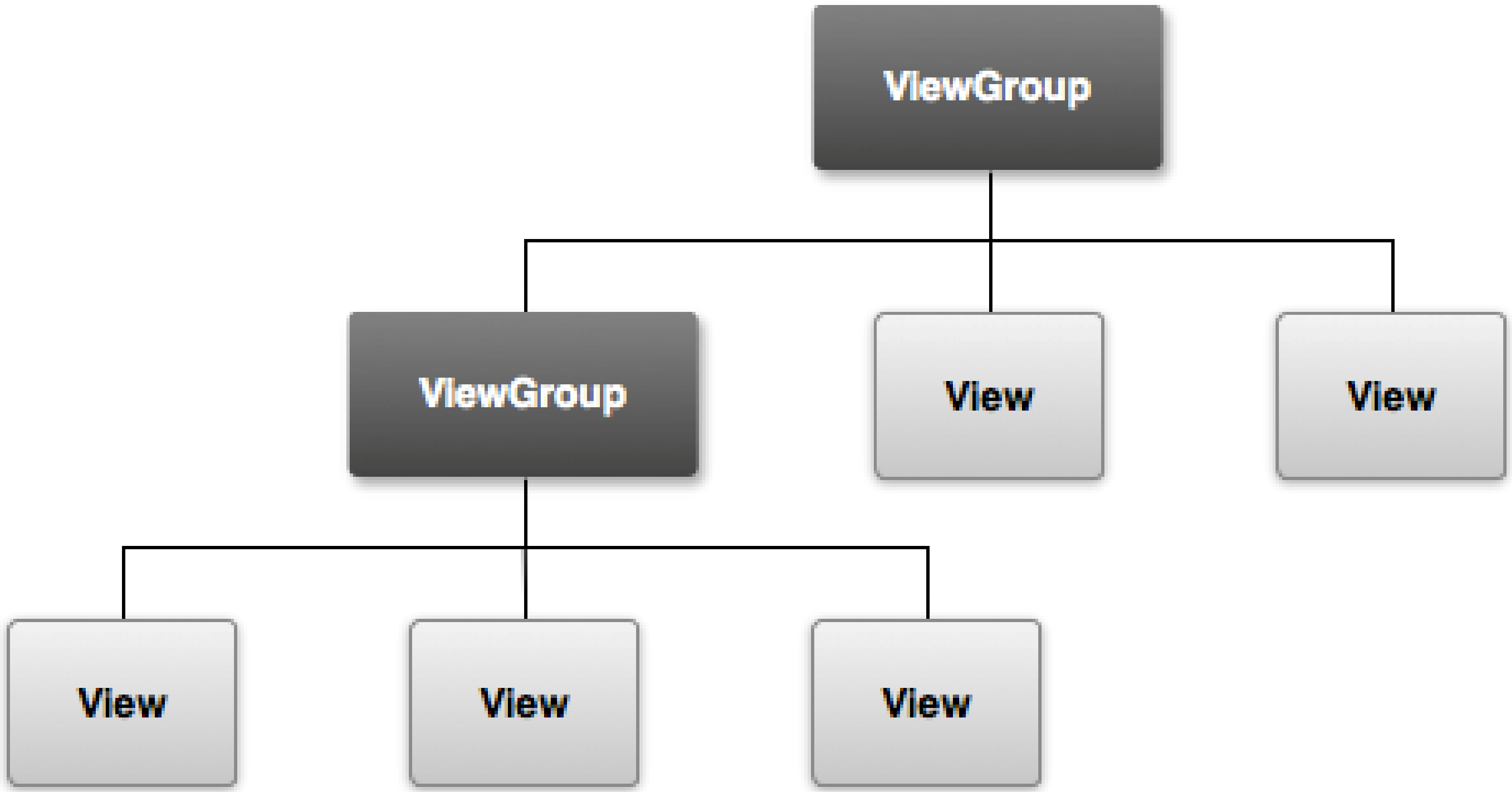
Two styles can be mixed :

- Start with XML and declare most of UI.
- Switch to Java and implement UI logic.

UI Controls

Some of the basic UI controls are :

- TextView
- EditText
- Button
- Radio Button
- Checkbox
- Spinner


ViewGroup
ViewGroup
View
View
View
View
View

outline

- Main difference between a Drawable and a View is reaction to events
- Could be declared in an XML file
- Could also be declared inside an Activity
- Every view has a unique ID
- Use findViewById(int id) to get it
- Views can be customized

ViewGroup

- ViewGroup is a view container
- It is responsible for placing other views on the display
- Every layout must extend a ViewGroup
- Every view needs to specify:
  - android:layout_height
  - android:layout_width
  - A dimension or one of match_parent or wrap_content

- There are following three concepts related to Android Event Handling:

Event Listeners − An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

Event Listeners Registration − Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

Event Handlers − When an event happens and we have registered and event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.