Programmation objets, web et mobiles (JAVA)

Programmation Androïd

Licence 3 Professionnelle - Multimédia

Philippe Esling (esling@ircam.fr)

Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)

UPMC SORBONNE UNIVERSITÉS

ircam Centre Pompidou

UPMC SORBONNE UNIVERSITÉS

Sommaire

- Un peu d'historique
- Andoid OS comme *middleware*
  - Applications et évènements gérés par le middleware
  - Une approche déclarative des IHM en XML
  - Une configuration en XML
  - Linux et Java
    - sans la JVM mais avec une DVM
- Principes de base
- *Ce n'est qu'une introduction …*

Android : les objectifs

- http://www.android.com
- http://www.OpenHandsetAlliance.com

  “… Open Handset Alliance™, a group of 47 technology and mobile companies have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.

  • Together we have developed Android™, the first complete, open, and free mobile platform.
  • We are committed to commercially deploy handsets and services using the Android Platform. “

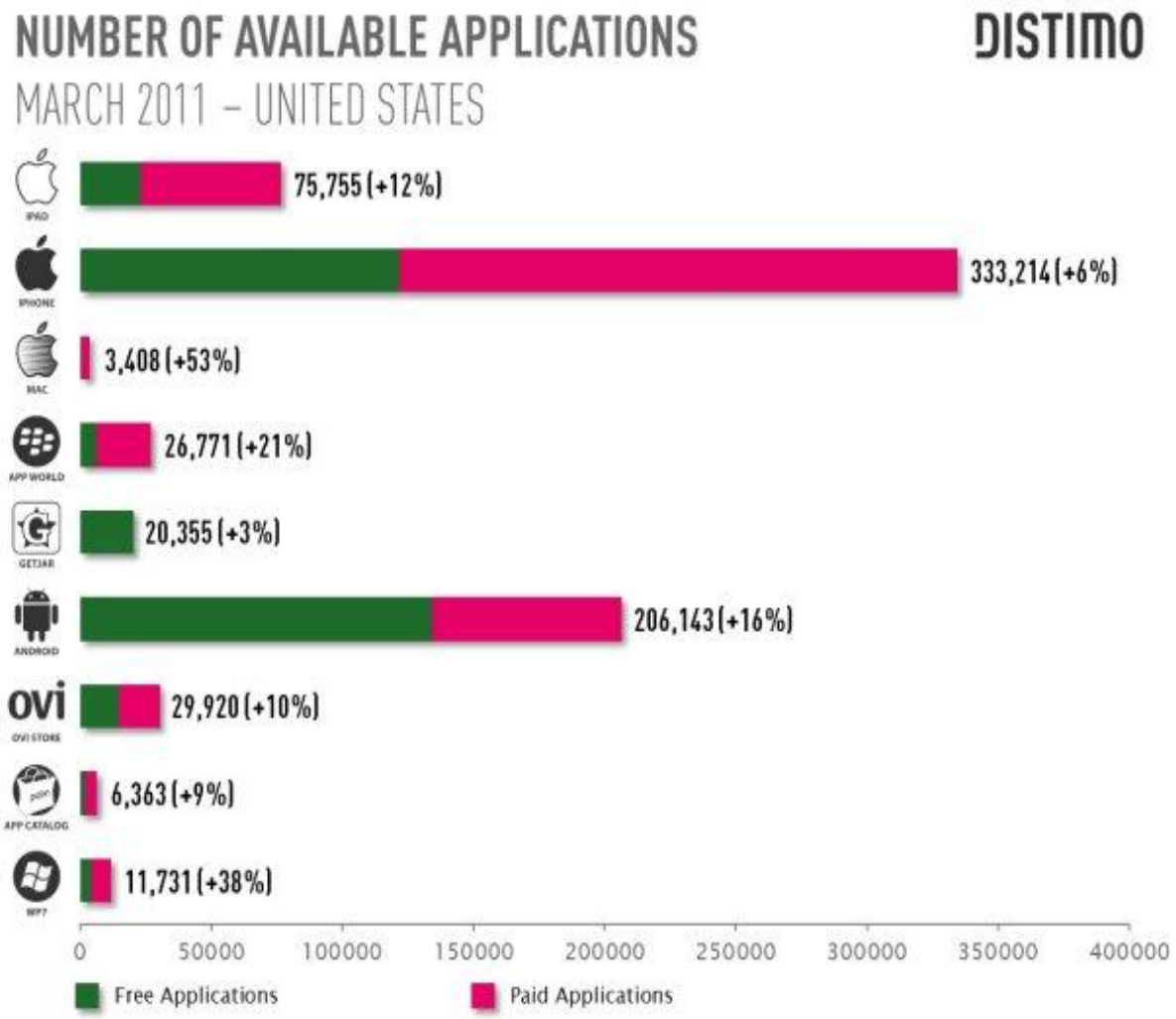
Android ?

- Une plate forme ouverte
- + un ensemble de librairies, de composants logiciels pour les systèmes embarqués et mobiles
  - Un système d'exploitation
    - **Linux**
  - Un intergiciel (middleware)
  - Nombreuses librairies
    - IHM,
    - Téléphonie,
    - Multimédia,
    - Capteurs,
    - Internet, cartographie

Pourquoi Android ?

- Indépendant d'une architecture matérielle
  - Avec des contraintes matérielles à respecter par les intégrateurs
- Dédié aux systèmes embarqués mais pas seulement
- Ambitions de Google/Apple
  - Marketing
  - Les applications
    - Nombreuses et gratuites sur AndroidMarket
    - Nombreuses et payantes sur AppStore

Applications gratuites (2011)



Les autres

- Apple
- Microsoft
- Nokia
- Palm
- Research in Motion (BlackBerry)
- Symbian
- Quid de JavaFX et javaME ? ….
  - javaFX/Flash prometteur mais :
    - Lancé en 2009, qui l'utilise ?
      - http://java.sun.com/javafx/1/tutorials/core/
  - javaME obsolète ?
    - Smartphone ? Pour les pays riches et émergents ?

Projections

Table 1
Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units)

OS	2010	2011	2012	2015
Symbian	111,577	89,930	32,666	661
Market Share (%)	37.6	19.2	5.2	0.1
Android	67,225	179,873	310,088	539,318
Market Share (%)	22.7	38.5	49.2	48.8
Research In Motion	47,452	62,600	79,335	122,864
Market Share (%)	16.0	13.4	12.6	11.1
iOS	46,598	90,560	118,848	189,924
Market Share (%)	15.7	19.4	18.9	17.2
Microsoft	12,378	26,346	68,156	215,998
Market Share (%)	4.2	5.6	10.8	19.5
Other Operating Systems	11,417.4	18,392.3	21,383.7	36,133.9
Market Share (%)	3.8	3.9	3.4	3.3
Total Market	**296,647**	**467,701**	**630,476**	**1,104,898**

Source: Gartner (April 2011)

Android les grandes lignes

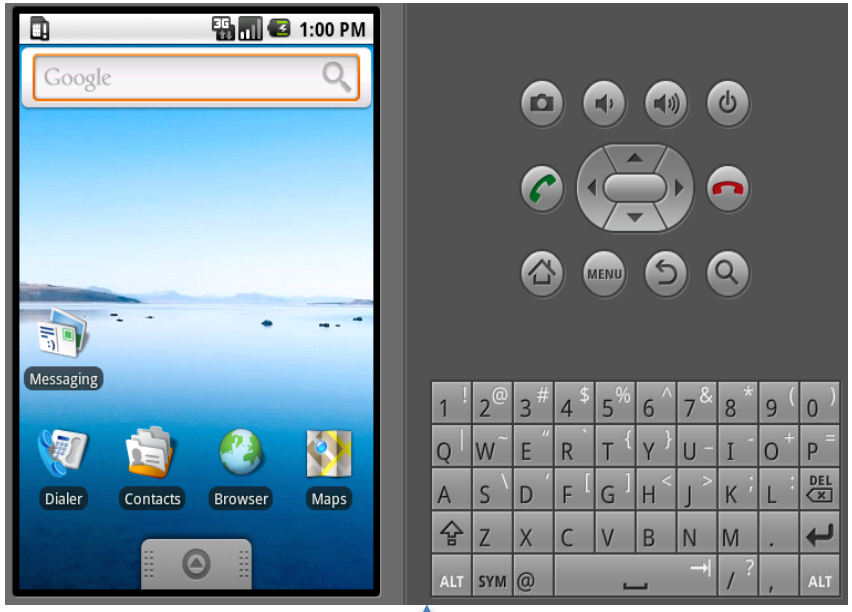
- Composants Android
- Outils de Développement
- Architecture Logicielle
- Développement
  - en java (☺)
  - quelques directives
  - configurations en syntaxe XML
- Deux exemples
  - Démonstration
    - Deux exemples en quelques lignes de java

Composants Android

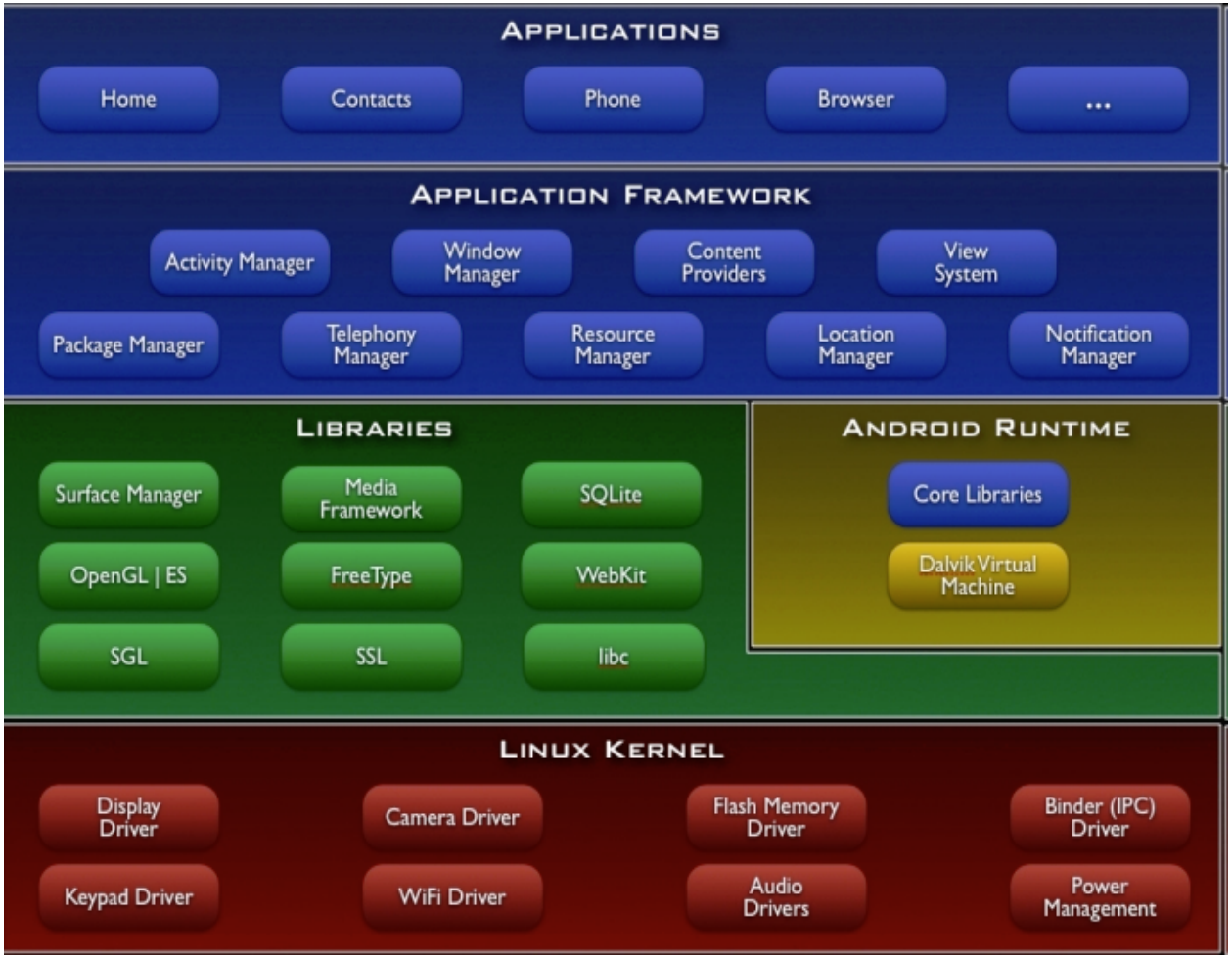
- Framework de déploiement d'applications
- Dalvik comme machine virtuelle (à registres != JVM à pile)
- Navigateur intégré, WebKit (webkit safari, Google Chrome…)
- SQLite (Base de données)
- Multimédia support PNG, JPG, GIF, MPEG4, MP3
- Dépendant du matériel
  - GSM
  - Bluetooth, EDGE, 3G, WiFi
  - Caméra, GPS, boussole et accéléromètre
  - Température,

Outils de développement

- SDK Android
  - –En ligne de commandes
  - –Plug-in sous eclipse
  - –Émulateur
  - –Débogueur
  - –Traces fines d'exécution
  - –Tests unitaires
  - –Outils de mise au point
    - •Mesure de mémoire et performance



Composants Android



http://developer.android.com/guide/basics/what-is-android.html

Android OS

- **Un ensemble d'API**
  - http://developer.android.com/guide/basics/what-is-android.html

Middleware Android OS (extrait)

- View System listes, boutons,… navigateur (WebView)
- Resource Manager, accès aux String, aux descriptifs de lihm
  - R.java …
- Activity Manager gestion du cycle de vie d'une application
  - Une application android peut être composée de plusieurs activités
- Content Providers Accès aux données d'autres applications
  - partage, persistance de données
- Notification Manager autorise des alertes dans la barre de statut
- TelephonyManager

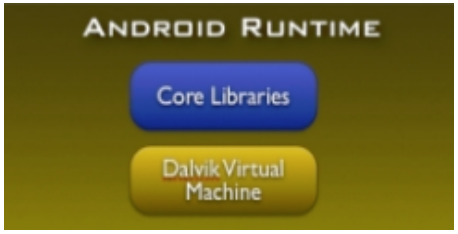
Librairies

C/C++ …

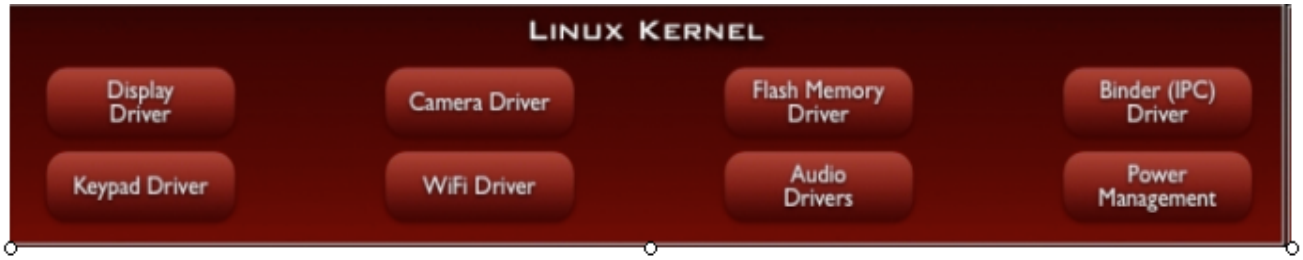
- SGL comme moteur pour le 2D
- FreeType comme fontes de caractères

Dalvik VM, au lieu de JVM

- Machines à registres



- Chaque application à sa propre DVM
  - –Communication inter-applications assurée par le middleware
  - –Multi thread assuré par Linux
  - –Accès aux capteurs par le noyau Linux



Développement d'une application

Pour développer une application, nous allons utiliser Eclipse et des mécanismes simples

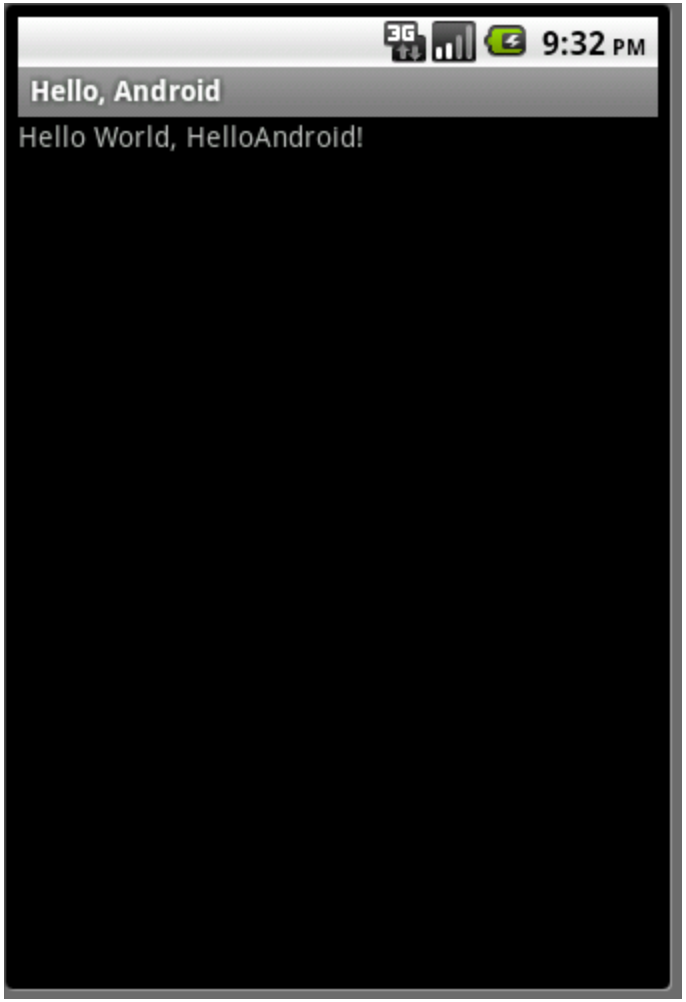
1. Obtention des .class
2. Génération de l'APK, (Android Package file)

1. Obtention des .class

- Fichier de configuration XML
  - Un source Java est généré, le fichier de ressources R.java
    - Configuration de l'application, IHM, String, …
      - Approche déclarative de l'IHM
- java
  - Paquetage java.lang,
    - Attention ce ne sont pas les librairies du JavaSE *(android.jar n'est pas rt.jar)*
- Compilateur javac de Sun/Oracle
  - javac –bootclasspath android.jar android/introduction/*.java

1er exercice

- Créer une application très simple
- Lancer sur émulateur
- Lancer sur téléphone
- Examiner sa structure



Google Tutorial

- Commençons par créer un projet
- (Et vérifier que le SDK Eclipse marche)
- Start Eclipse (Start -> All Programs -> Eclipse)
- Create an Android Virtual Device (AVD)
- Create a New Android Project

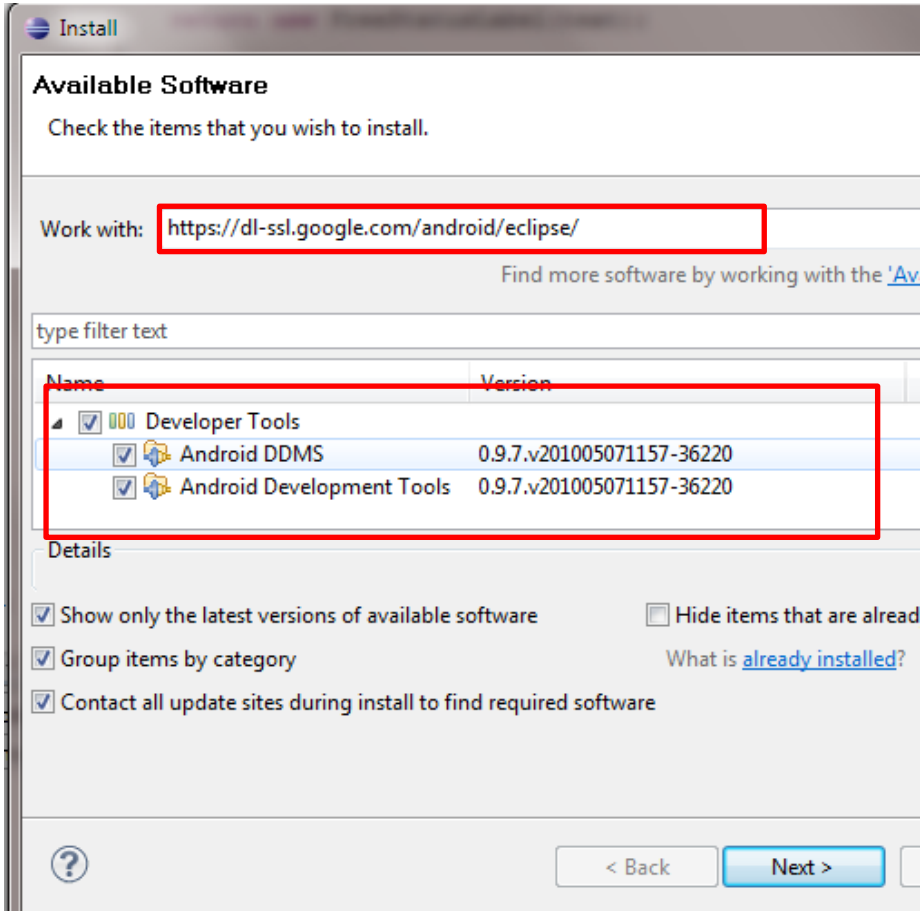
Setup Android SDK

- Download Android SDK and extract the zip file to an arbitrary folder
  - http://androidappdocs.appspot.com/sdk/index.html
  - E.g.: extract to C:\
  - The SDK will be used by ADT in eclipse

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r06-windows.zip	23293160 bytes	7c7fcec3c6b5c7c3df6ae654b27effb5
Mac OS X (intel)	android-sdk_r06-mac_86.zip	19108077 bytes	c92abf66a82c7a3f2b8493ebe025dd22
Linux (i386)	android-sdk_r06-linux_86.tgz	16971139 bytes	848371e4bf068dbb582b709f4e56d903

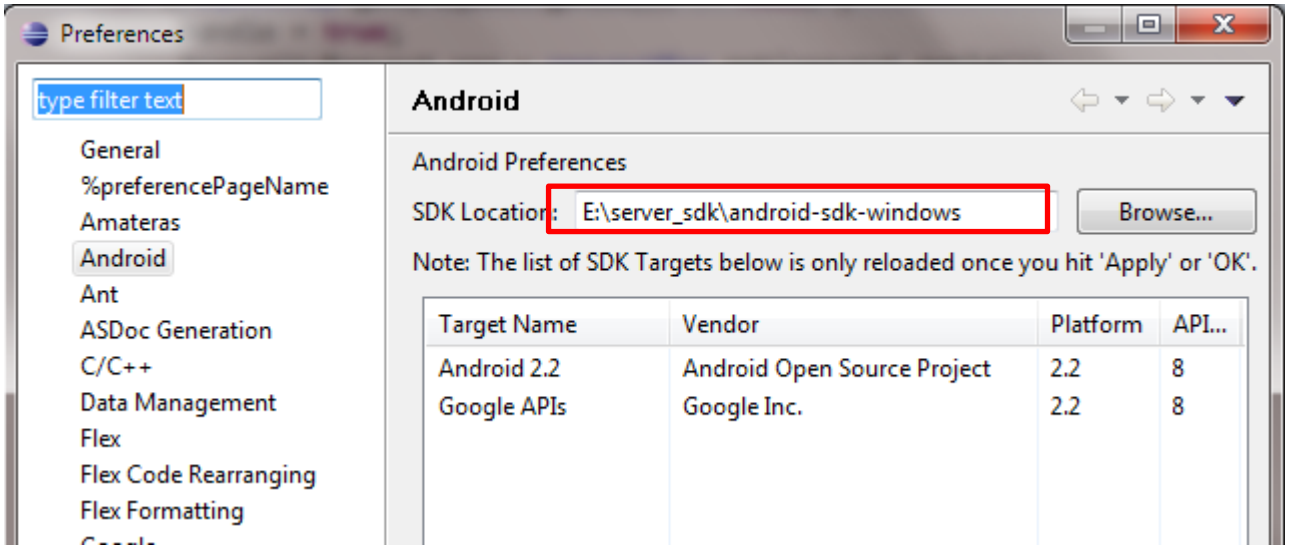
Setup ADT plugin

- Install Eclipse ADT plugin
  - Eclipse must be J2EE edition, 3.5 recommended
  - Update site: https://dl-ssl.google.com/android/eclipse/
  - Install all the plugins in the repository
  - Restart needed after installation



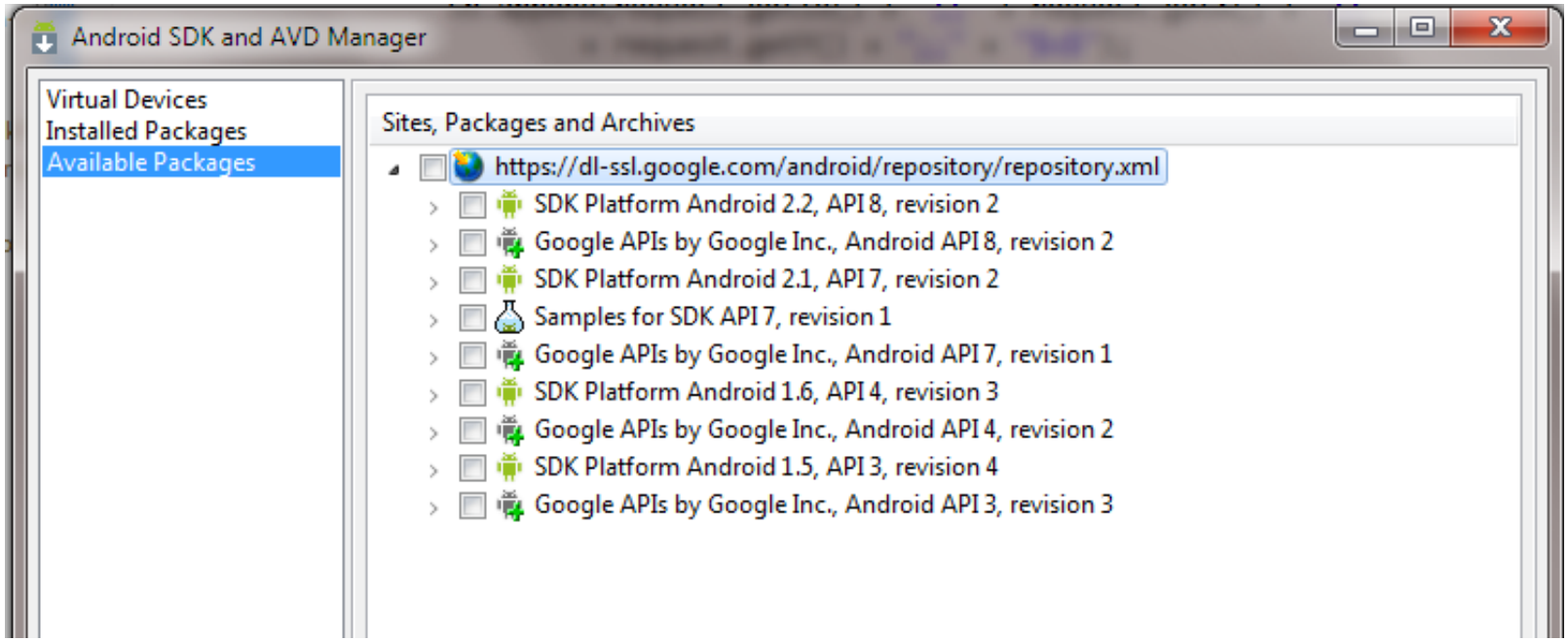
Configure ADT Plugin

- Open eclipse Window->Preferences, select Android
- Setup the SDK location as the folder where you extracted the downloaded SDK zip file



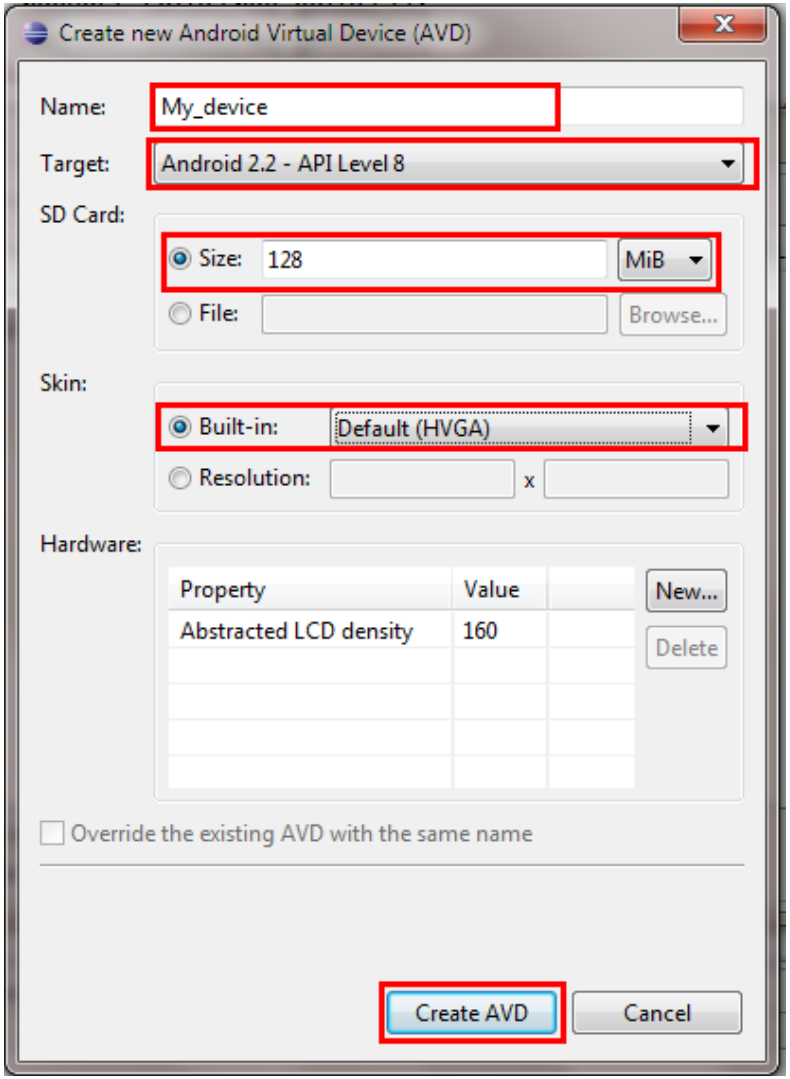
Setup SDK APIs

- Open Window->Android SDK and AVD Manager
- Click *Available Packages* and then choose proper APIs to install, the latest may be the best



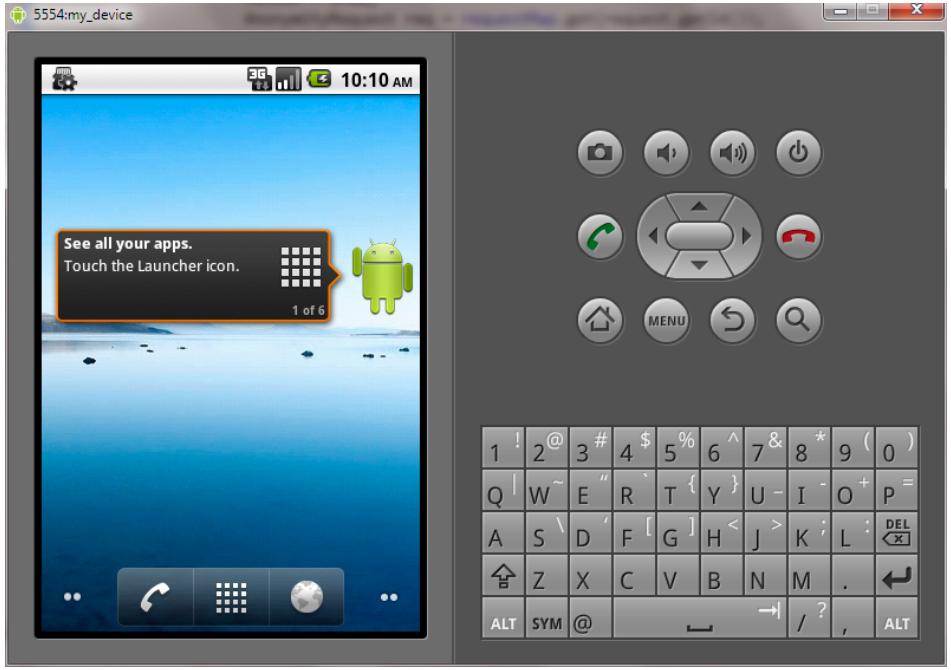
Setup Emulators

- After SDK APIs installation, click *Virtual Devices*
- Click *new, there will be a dialog*
  - input a name
  - choose a running target and a skin
  - specify the SD card size



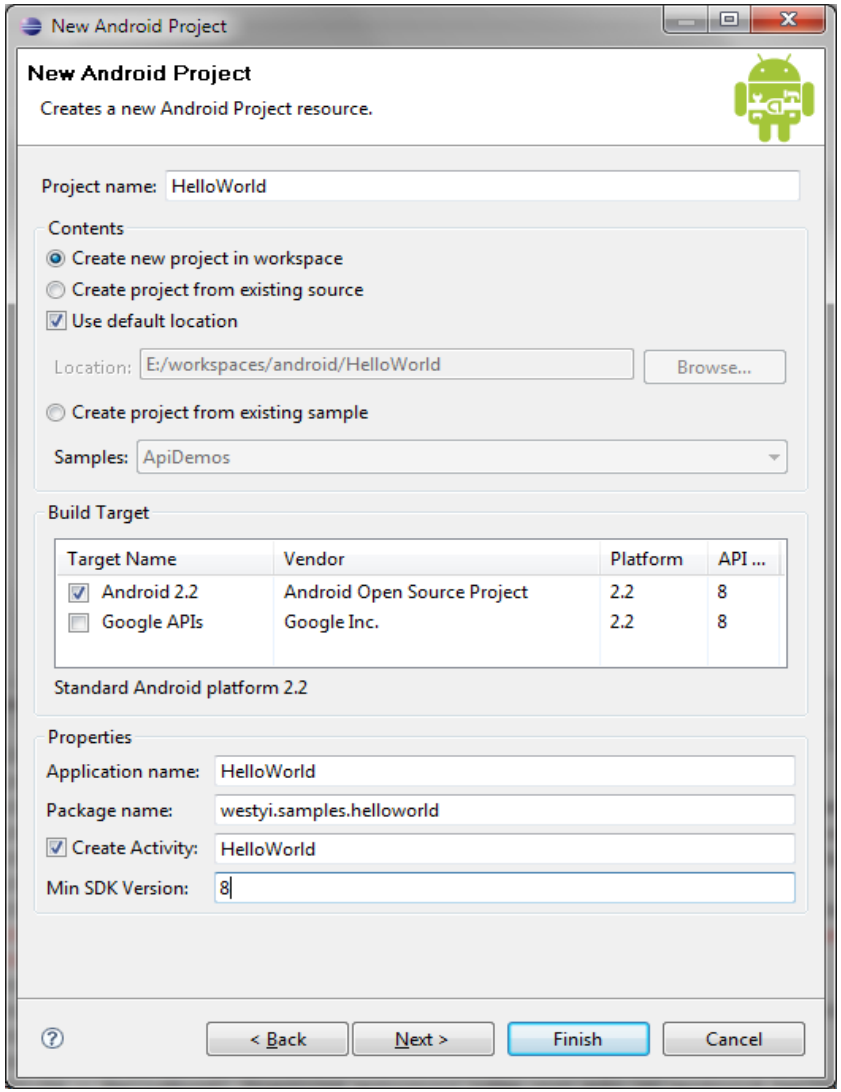
Ready…

- Now you may start the AVD
  - Click start to start the new AVD
  - First start-up may take a very long time



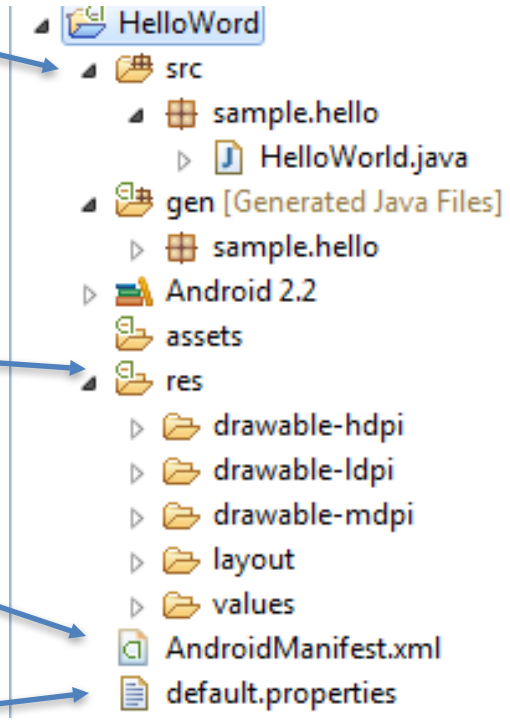
Create a new Android Project

- Open File->New->Android project
  - Project name
  - Build Target
  - Application name
  - Package name
  - Create Activity



Hello World Project

- **src**: source folder
- **gen**: SDK generated file
- **android 2.2**: reference lib
- **assets**: binary resources
- **res**: resource files and resource description files
- **AndroidManifest.xml**: application description file
- **default.properties**: project properties file



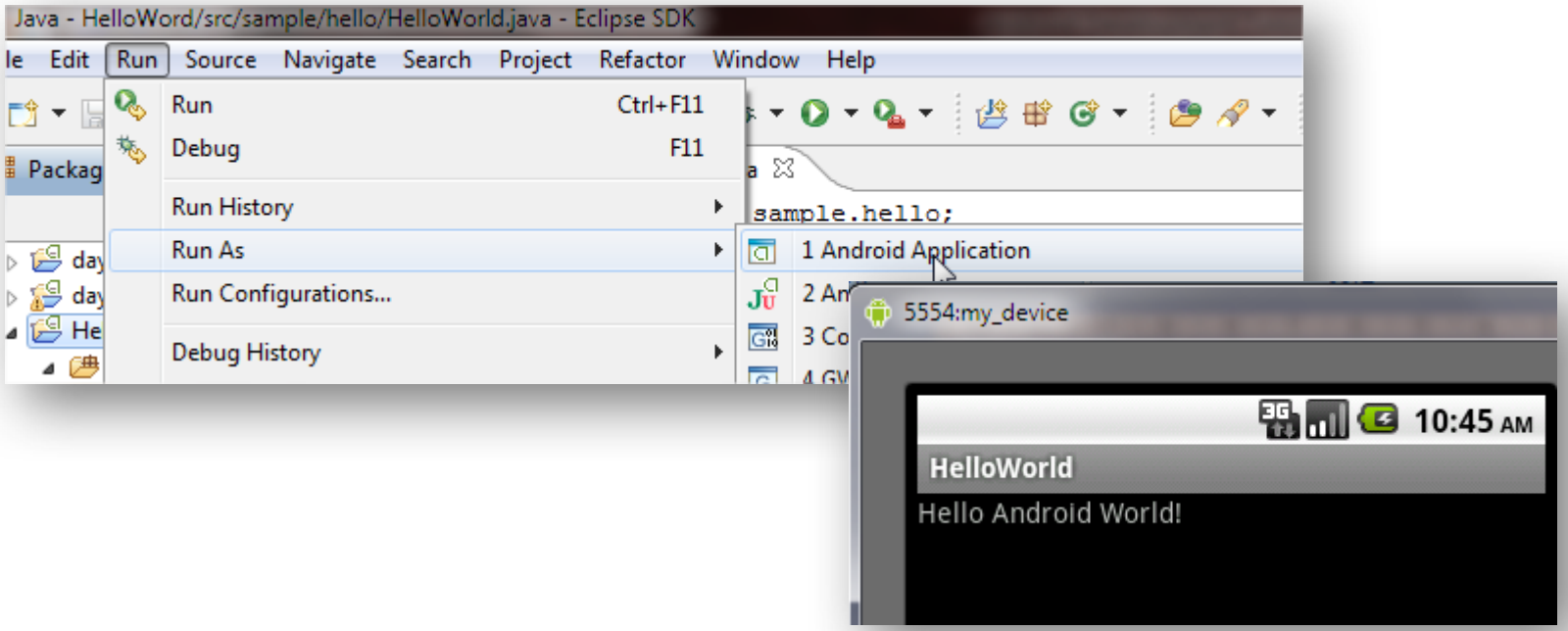
Say Hello World

- modify HelloWorld.java

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

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

Run Hello World

- Select *HelloWorld* Project, Run->Run as->Android Application
- ADT will start a proper AVD and run HelloWorld app on it



R Class

- Auto-generated: you shouldn’t edit it
- Contains IDs of the project resources
- Enforces good software engineering
- Use findViewById and Resources object to get access to the resources
  - Ex. Button b = (Button)findViewById(R.id.button1)
  - Ex. getResources().getString(R.string.*hello*));

Behind HelloWorld #1

- 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 {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

XML

- Used to define some of the resources
  - Layouts (UI)
  - Strings
- Manifest file
- Shouldn’t usually have to edit it directly, Eclipse can do that for you
- Preferred way of creating UIs
  - Separates the description of the layout from any actual code that controls it
  - Can easily take a UI from one platform to another

Behind HelloWorld #2

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

main.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>
```

Linear Layout

TextView, display static text

A reference to String resource 'hello'

Strings

- In res/values
  - strings.xml
- Application wide available strings
- Promotes good software engineering
- UI components made in the UI editor should have text defined in strings.xml

- Strings are just one kind of ‘Value’ there are many others

Behind HelloWorld #3

- 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 #4

- 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 #5

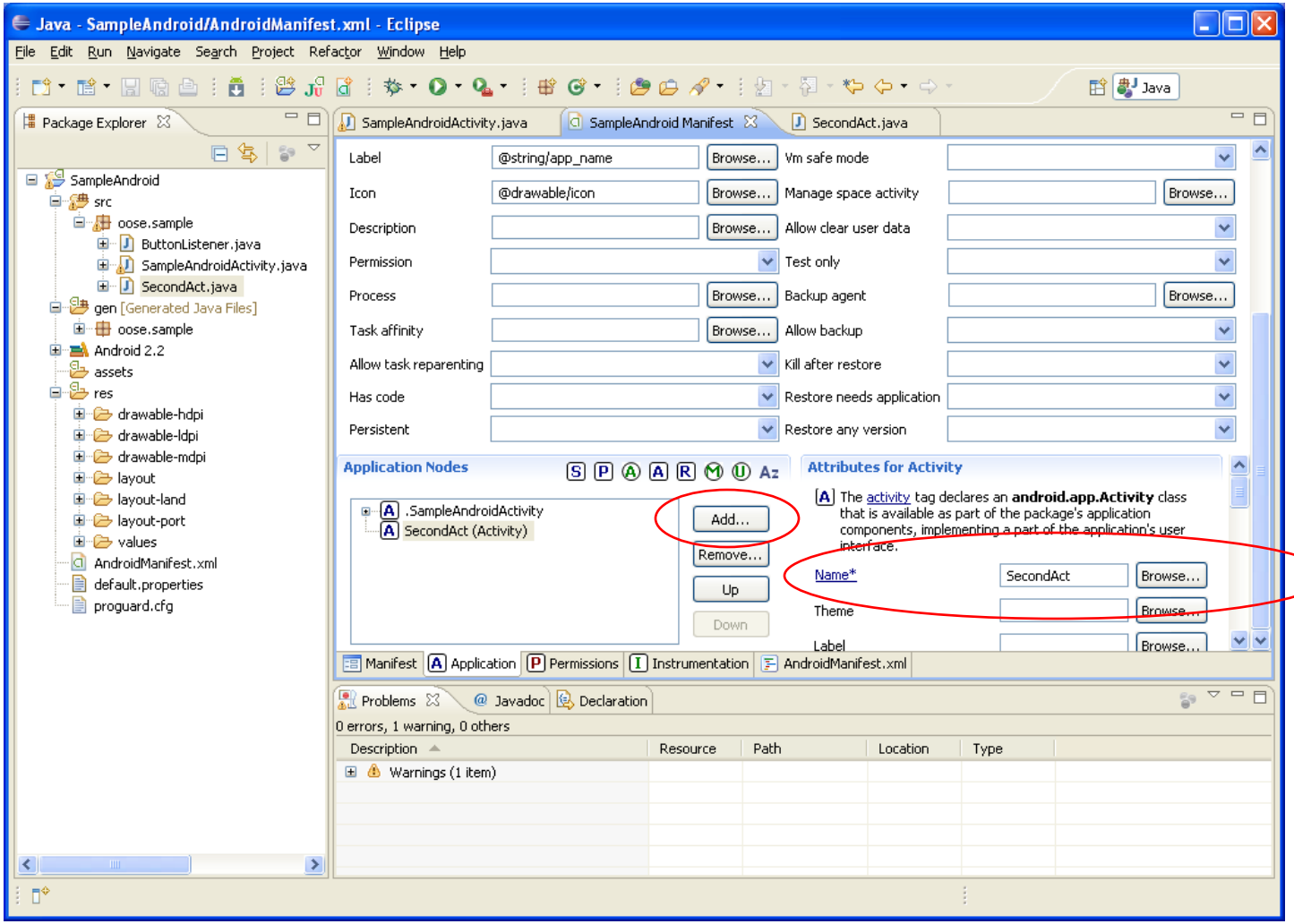
- AndroidManifest.xml describe the application
  - declare app's name, version, icon, permission, etc…
  - declare the application's components: activity, service ,receiver or provider

```
<?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.LAUNCHER"/>
                </intent-filter>
        </activity>
</application>
<uses-sdk android:minSdkVersion="8" />
</manifest>
```

Manifest File (1)

- Contains characteristics about your application
- When have more than one Activity in app, NEED to specify it in manifest file
  - Go to graphical view of the manifest file
  - Add an Activity in the bottom right
  - Browse for the name of the activity
- Need to specify Services and other components too
- Also important to define permissions and external libraries, like Google Maps API

Manifest File – Adding an Activity



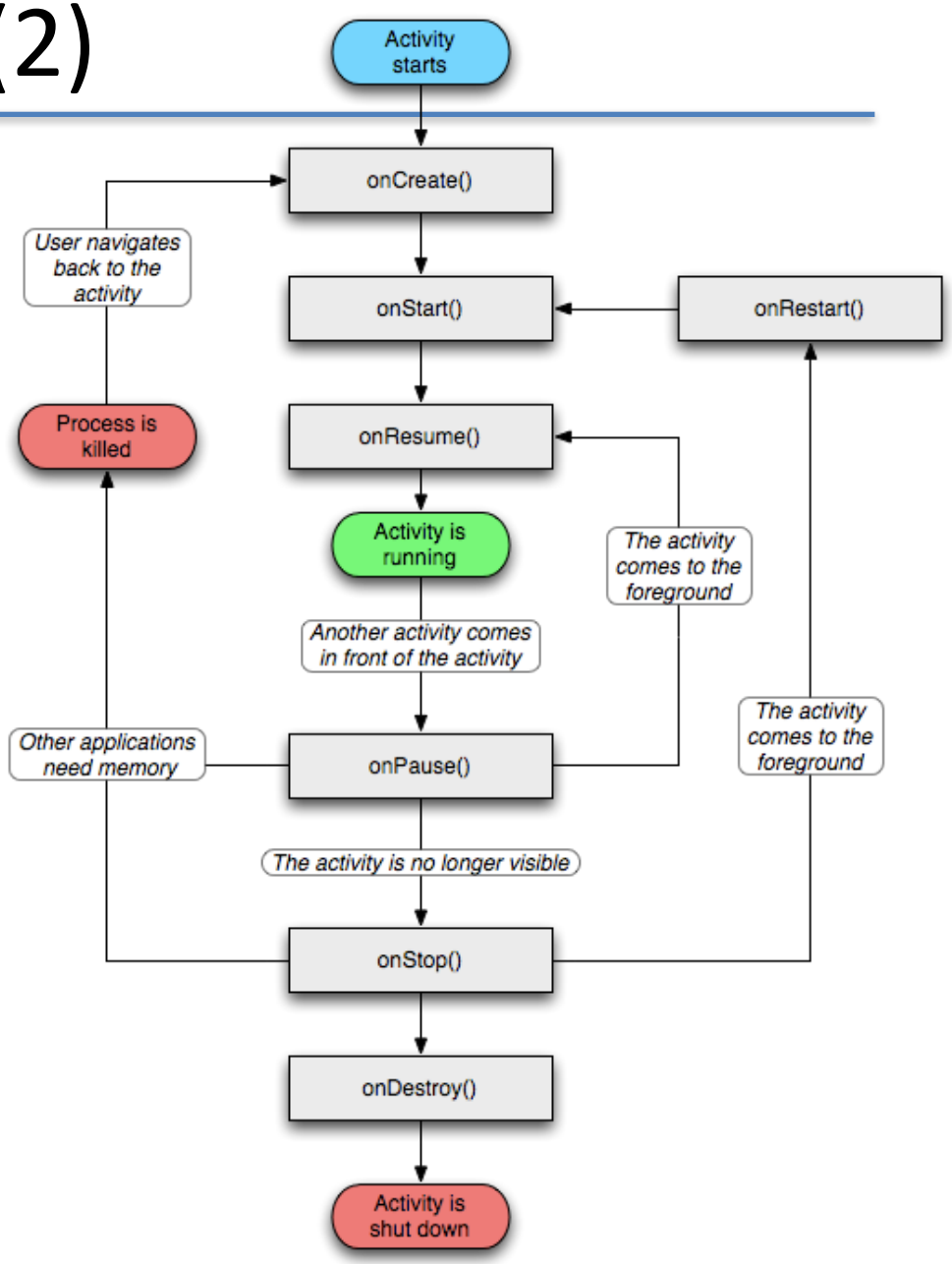
Android Programming Components

- Activity
  - http://developer.android.com/guide/topics/fundamentals/activities.html
- Service
  - http://developer.android.com/guide/topics/fundamentals/services.html
- Content Providers
- Broadcast Receivers
- Android in a nutshell:
  - http://developer.android.com/guide/topics/fundamentals.html

Activities (1)

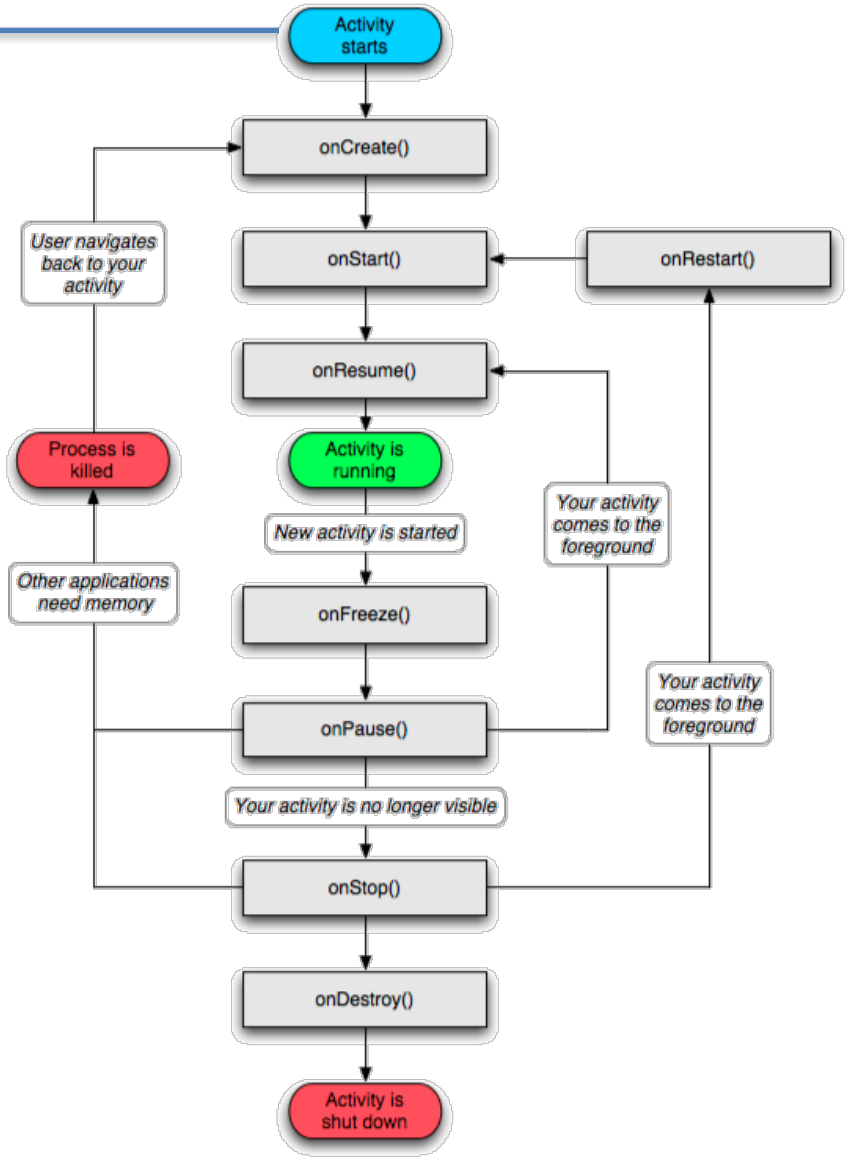
- The basis of android applications
- A single Activity defines a single viewable screen
  - the actions, not the layout
- Can have multiple per application
- Each is a separate entity
- They have a structured life cycle
  - Different events in their life happen either via the user touching buttons or programmatically

Activities (2)

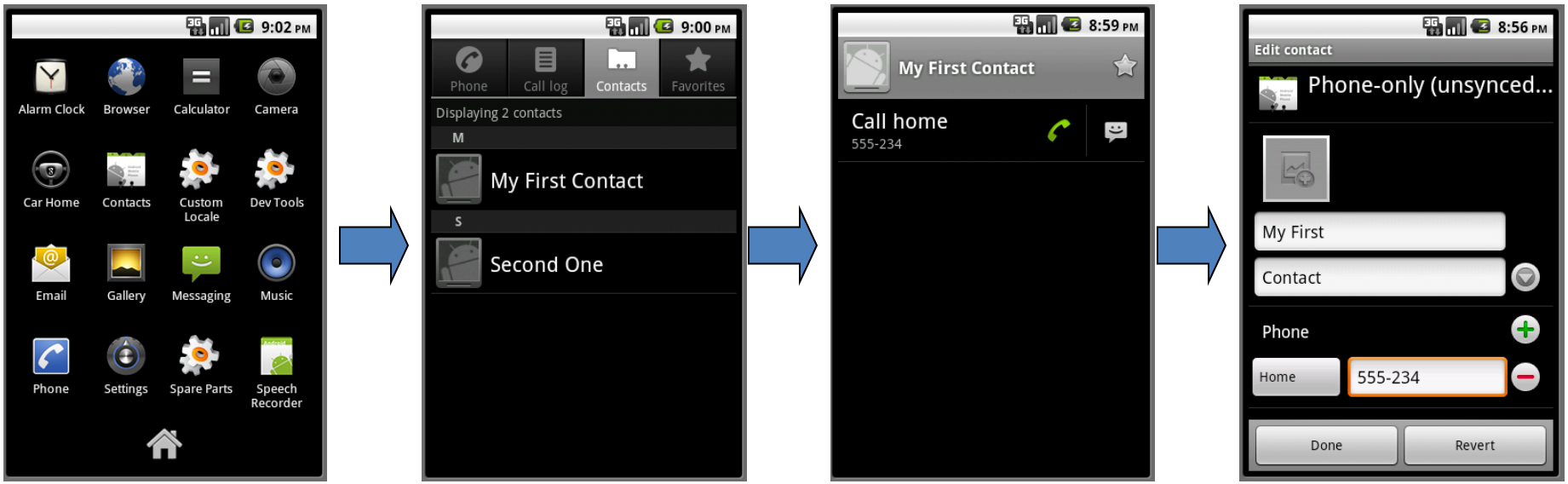


Activity

- An Android activity is focused on a single thing a user can do.
- Most applications have multiple activities



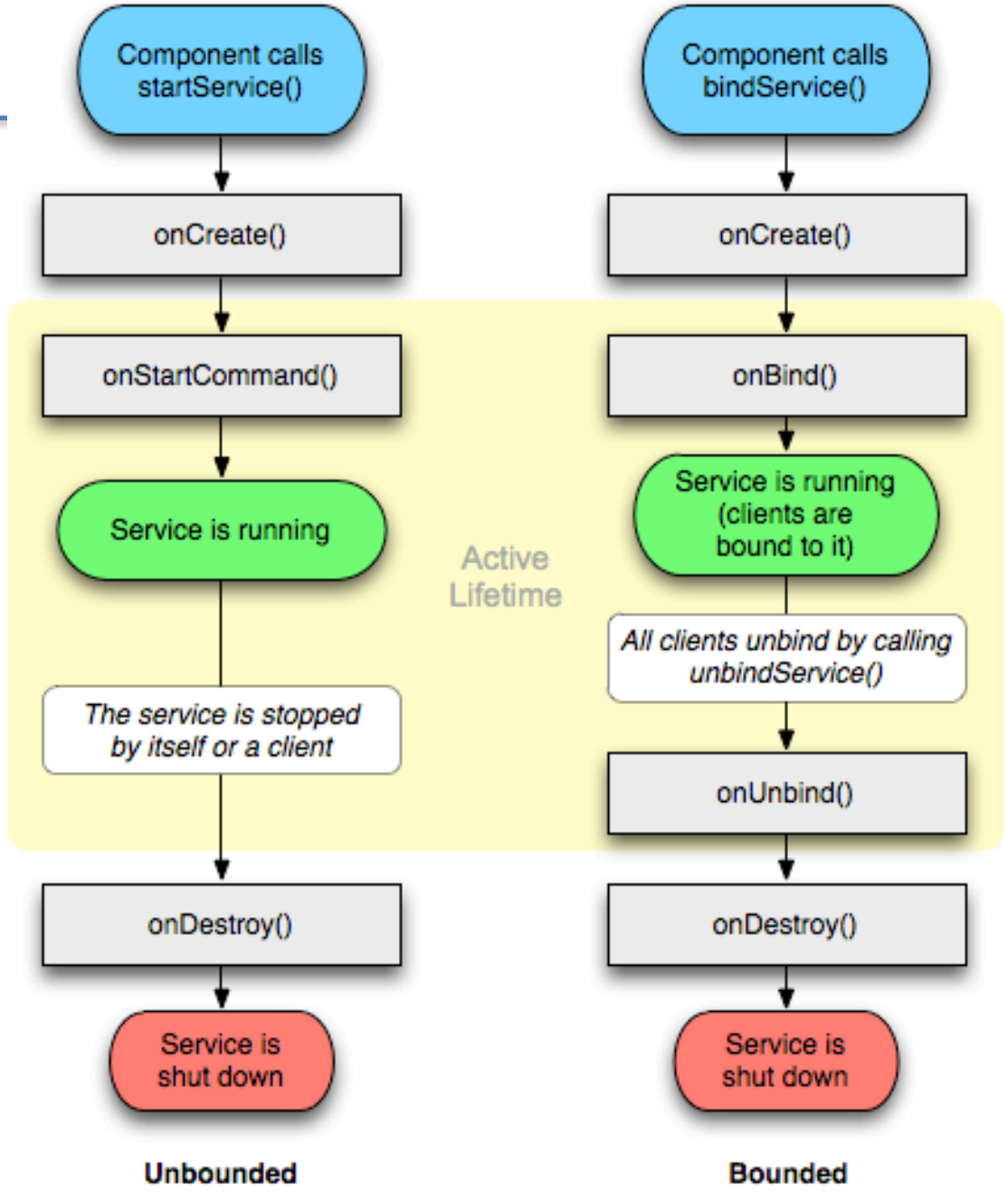
Activities start each other



Services (1)

- Run in the background
  - Can continue even if Activity that started it dies
  - Should be used if something needs to be done while the user is not interacting with application
    - Otherwise, a thread is probably more applicable
  - Should create a new thread in the service to do work in, since the service runs in the main thread
- Can be bound to an application
  - In which case will terminate when all applications bound to it unbind
  - Allows multiple applications to communicate with it via a common interface
- Needs to be declared in manifest file
- Like Activities, has a structured life cycle

Services (2)


Component calls startService()
onCreate()
onStartCommand()
Service is running
The service is stopped by itself or a client
onDestroy()
Service is shut down
Unbounded
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
Bounded

Exercice 2

- Build up an app that you can input your greetings and display your greetings
  - Input: EditText
  - Display: TextView
  - Of course, we have to add an 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:text="" android:id="@+id/editText"
    android:layout_width="fill_parent" android:layout_height="wrap_content"></EditText>
<Button android:text="Show Greetings" android:id="@+id/showBtn"
    android:layout_width="wrap_content" android:layout_height="wrap_content"></Button>
<TextView android:layout_width="fill_parent" android:id="@+id/textView"
    android:layout_height="wrap_content" android:text="@string/hello" />
```

Cheat – Look at the « Layout tab » at the bottom of Eclipse ☺

Beyond HelloWorld #2

- modify HelloWorld.java
  - firstly get the references declared in main.xml

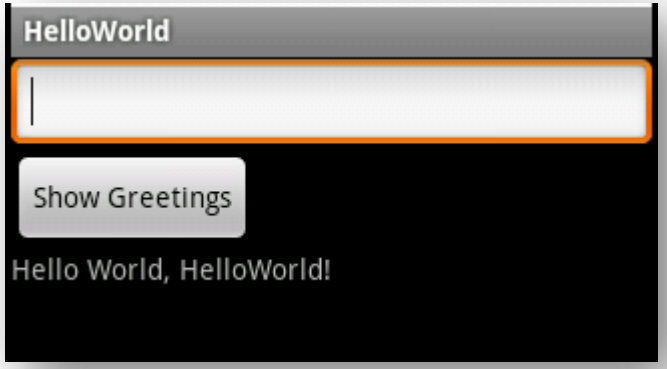
```
setContentView(R.layout.main);

final EditText edit = (EditText) findViewById(R.id.editText);
final TextView view = (TextView) findViewById(R.id.textView);
final Button btn = (Button)findViewById(R.id.showBtn);
```

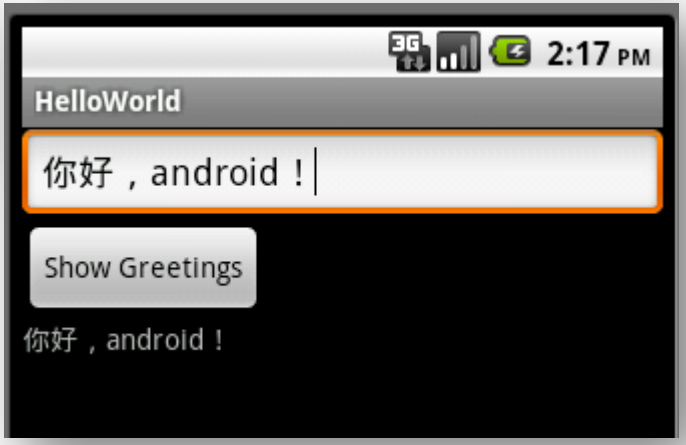
  - then add event response for Button

```
btn.setOnClickListener(new OnClickListener(){

    @Override
    public void onClick(View arg0) {
        view.setText(edit.getText());
    }
});
```

Beyond HelloWorld #3

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



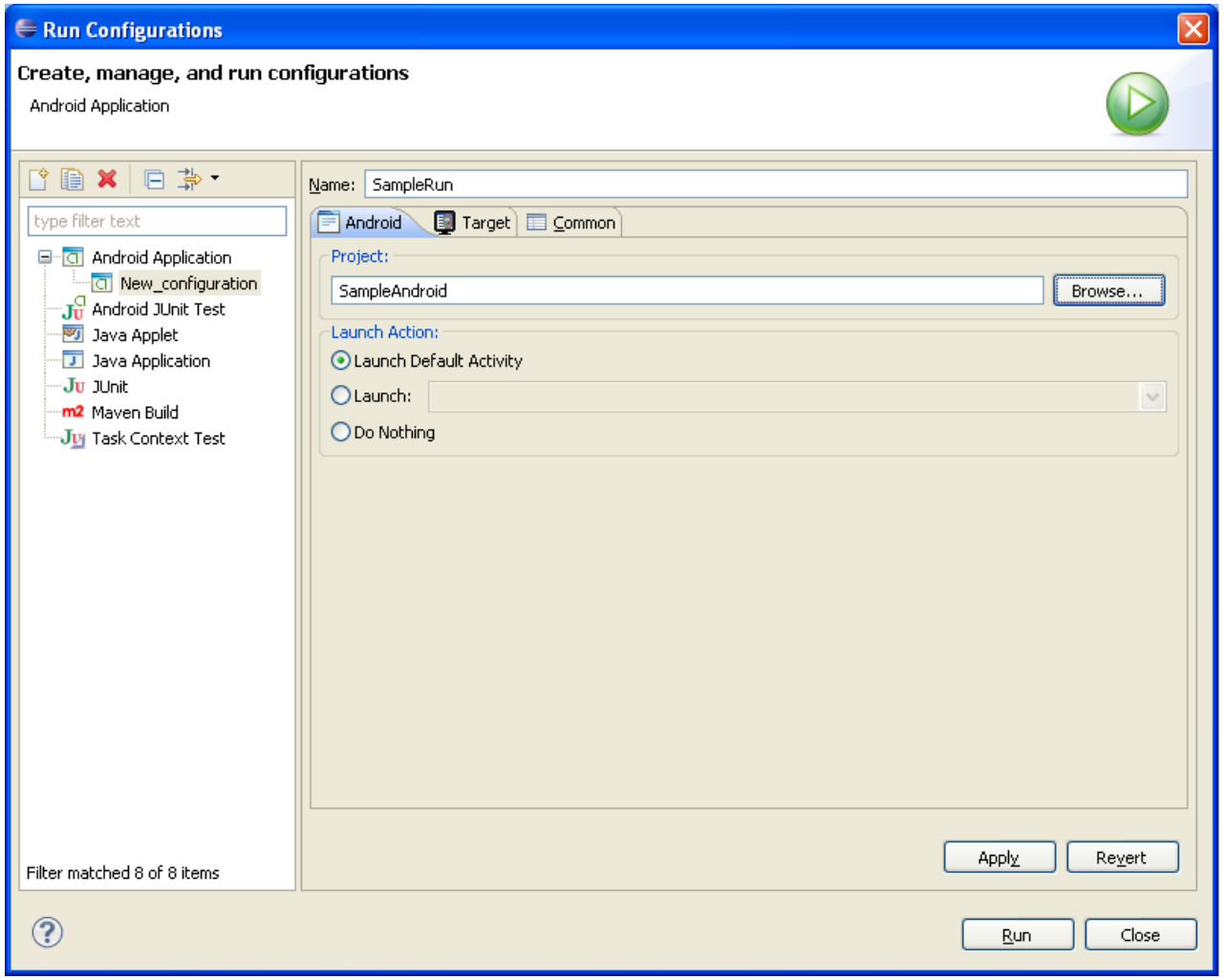


- Quite easy, isn’t it?

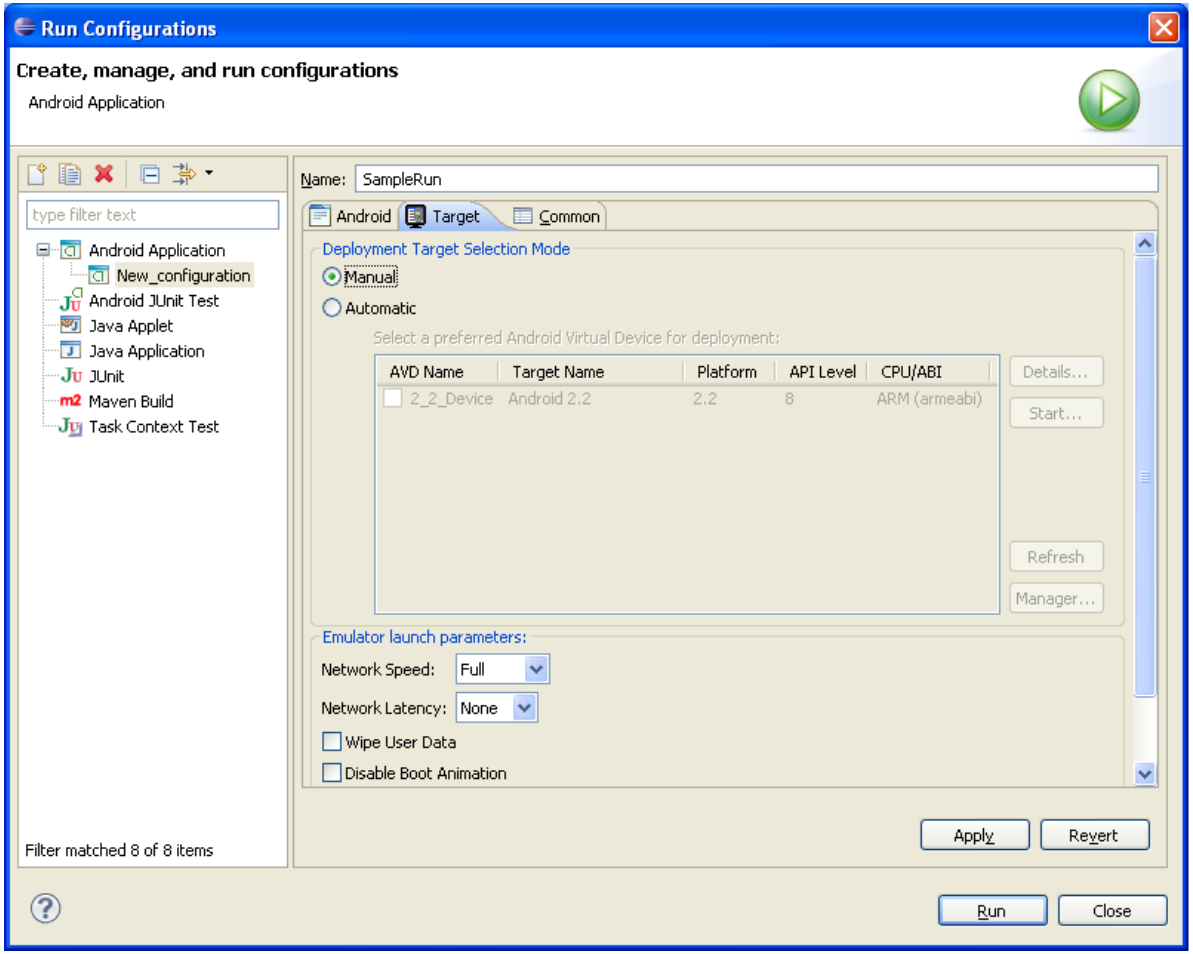
Running in Eclipse (1)

- Similar to launching a regular Java app, use the launch configurations
- Specify an Android Application and create a new one
- Specify activity to be run
- Can select a manual option, so each time program is run, you are asked whether you want to use the actual phone or the emulator
  - Otherwise, it should be smart and use whichever one is available

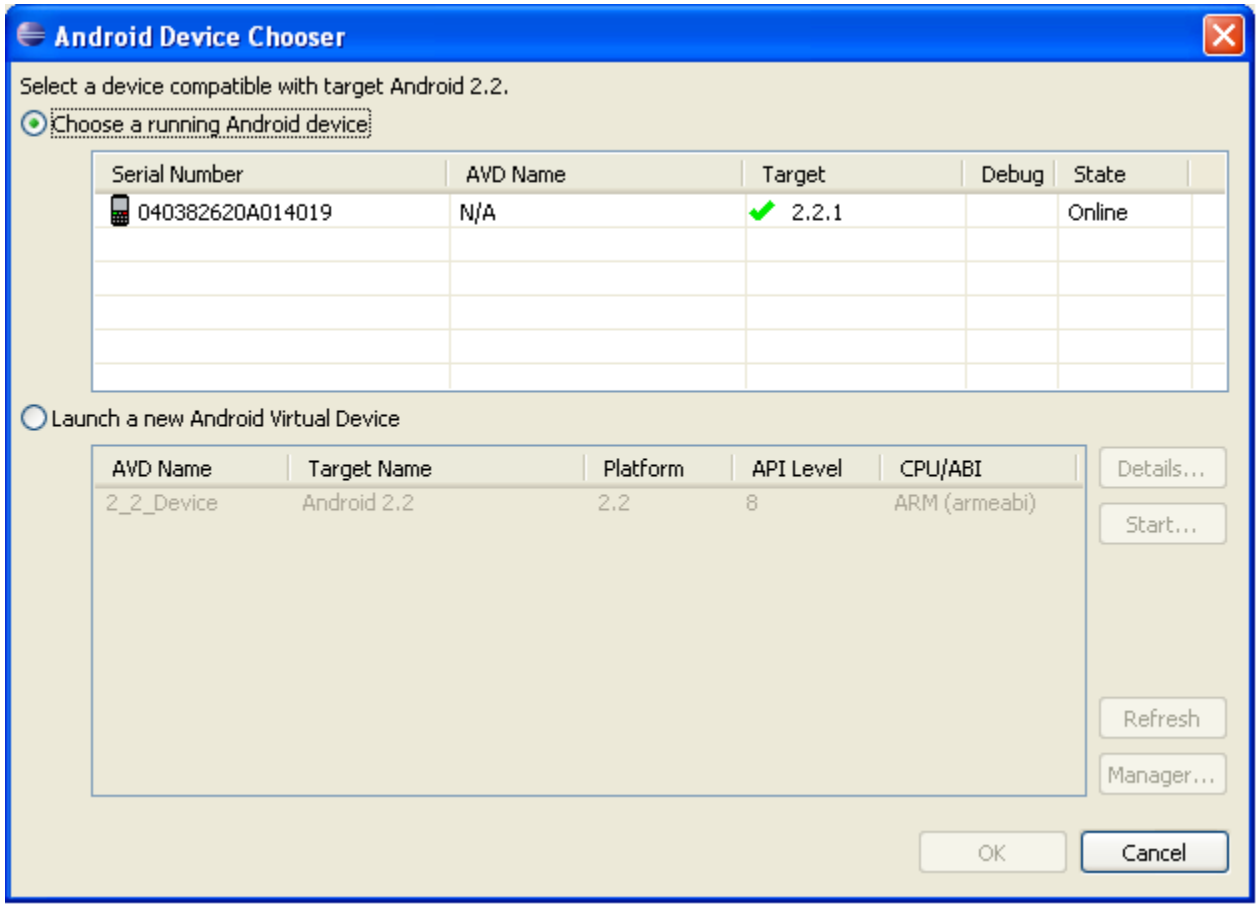
Running in Eclipse (2)



Running in Eclipse (3)



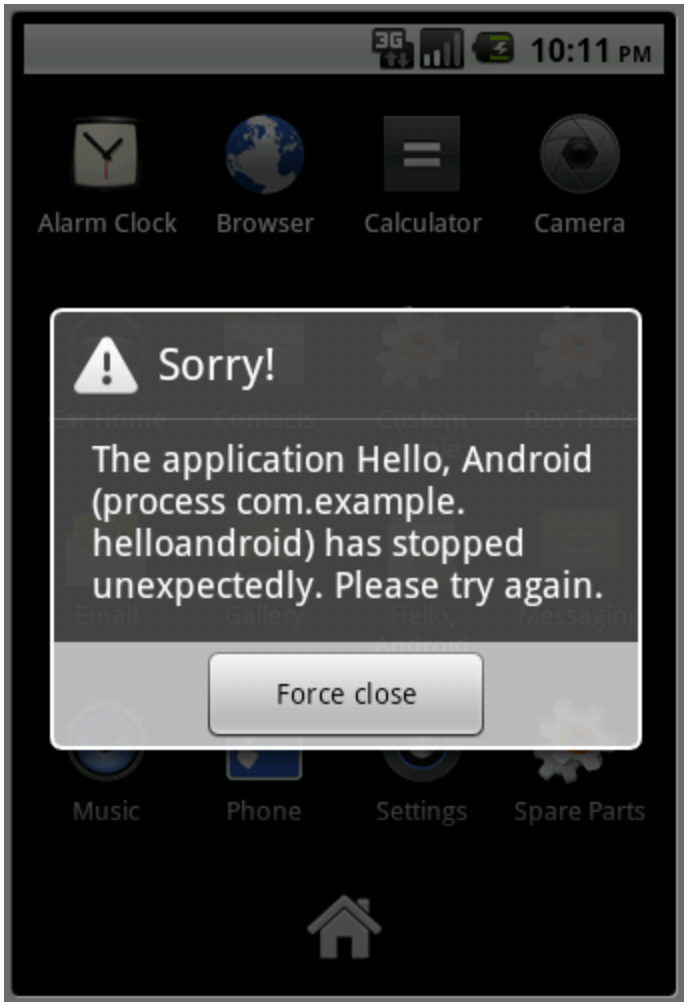
Running in Eclipse (4)



Introduce a bug

```
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);
         Object o = null;
         o.toString();
         setContentView(R.layout.main);
      }
   }
```

Run it!


10:11 PM
Alarm Clock
Browser
Calculator
Camera
Sorry!
The application Hello, Android (process com.example. helloandroid) has stopped unexpectedly. Please try again.
Force close
Music
Phone
Settings
Spare Parts

USB Debugging

- Should be enabled on phone to use developer features
- In the main apps screen select Settings -> Applications -> Development -> USB debugging (it needs to be checked)

Android Debug Bridge

- Used for a wide variety of developer tasks
  - Read from the log file
  - Show what android devices are available
  - Install android applications (.apk files)
- In the ‘platform-tools’ directory of the main android sdk directory
  - Recommend putting this directory and the ‘tools’ directory on the system path
- adb.exe

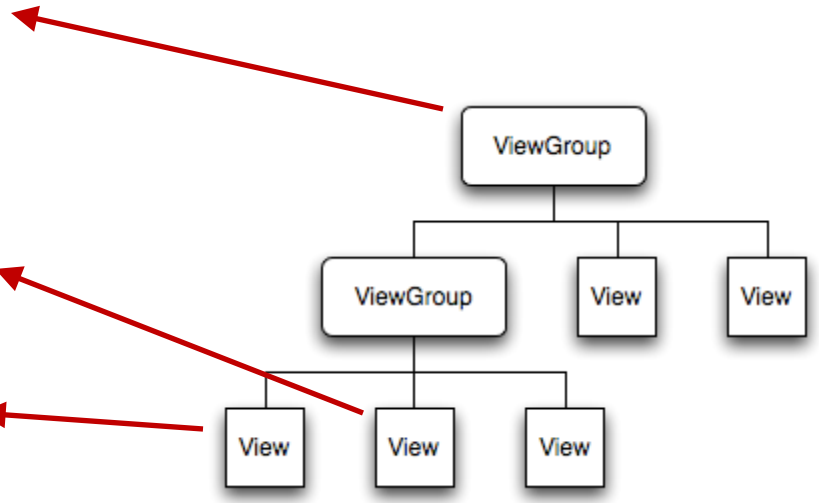
Debugging

- Instead of using traditional System.out.println, use the Log class
  - Imported with android.util.Log
  - Multiple types of output (debug, warning, error, …)
  - Log.d(<tag>,<string>)
- Can be read using logcat.
  - Print out the whole log, which auto-updates
    - adb logcat
  - Erase log
    - adb logcat –c
  - Filter output via tags
    - adb logcat <tag>:<msg type> *:S
    - can have multiple <tag>:<msg type> filters
    - <msg type> corresponds to debug, warning, error, etc.
    - If use Log.d(), then <msg type> = D
- Reference
  - http://developer.android.com/guide/developing/debugging/debugging-log.html

res/layout/**main.xml**

- Declares layouts & widgets for the activity

```
<?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" >

    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

    <Button
        android:id="@+id/hello_button"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="Press Me" />

</LinearLayout>
```



Tree from: http://developer.android.com/guide/topics/ui/index.html

Various Layouts

LinearLayout

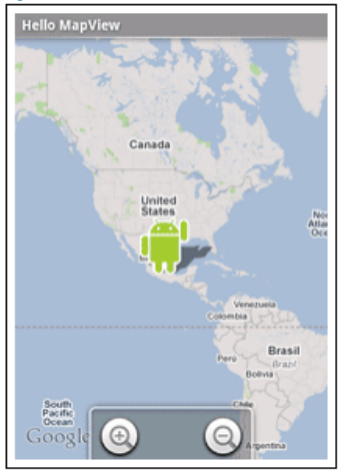
RelativeLayout

TableLayout

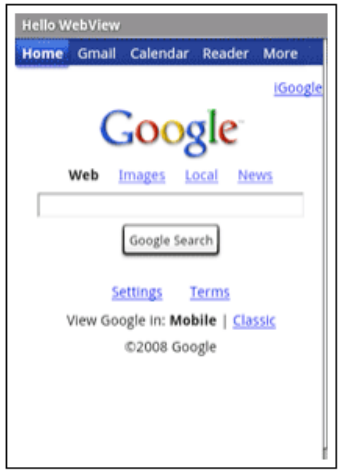
http://developer.android.com/resources/tutorials/views/index.html

Various Widgets

MapView



WebView

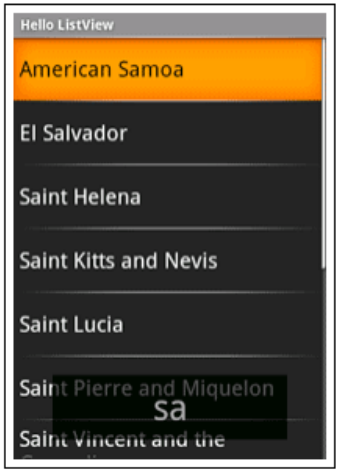


DatePicker

Spinner

AutoComplete

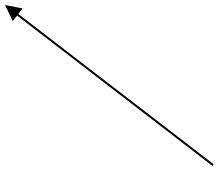
ListView



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

HelloAndroid.java

```
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);
        }
}
```



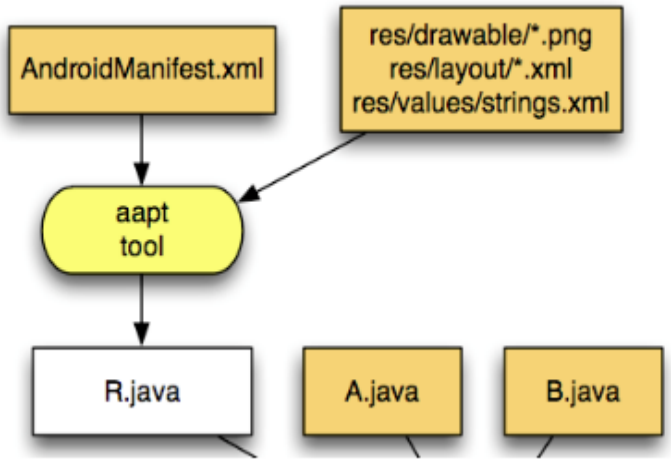
Set the layout of the view as described in the main.xml layout

Développement 1/2

Fichier de configuration, AndroitManifest.xml

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

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

  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".Demo"
            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>
```

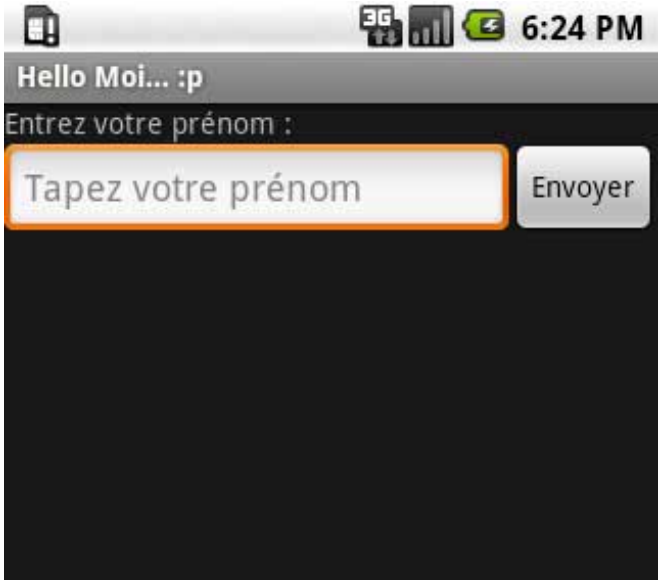
Fichier de Ressources XML associé




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  …
</LinearLayout>
```

- Répertoire res/
  - –Projet eclipse
- res/layout/main.xml

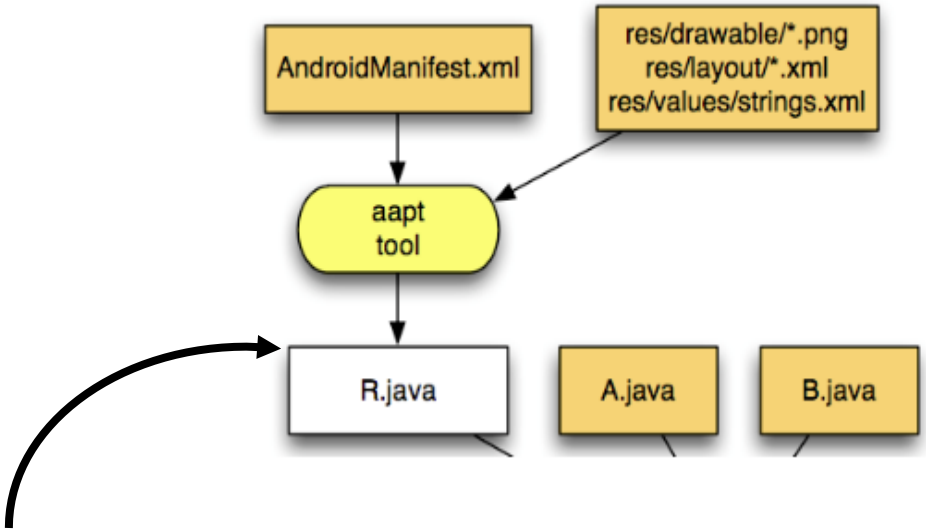
Fichier de Ressources XML associé




```
<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:id="@+id/TextViewPrenom' android:layout_width="fill_parent »
android:layout_height="wrap_content »
android:text="@string/prenom »/>

                <LinearLayout  android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content » >

                  <EditText android:id="@+id/EditTextPrenom"
                              android:layout_width="wrap_content"
                              android:layout_height="wrap_content"
                               android:layout_weight="1"
                               android:layout_gravity="bottom"
                              android:hint="@string/prenomHint »/>

                  <Button android:id="@+id/ButtonEnvoyer"
                              android:layout_width="wrap_content"
                              android:layout_height="wrap_content"
                              android:text="@string/bouton »/>
               </LinearLayout>

               <TextView android:id="@+id/TextViewHello"
                  android:layout_width="wrap_content"
                  android:layout_height="fill_parent"
                  android:layout_gravity="center_horizontal"
                  android:textSize="@dimen/dimMessage"
                  android:textColor="@color/couleurMessage »/>

</LinearLayout>
```

UPMC SORBONNE UNIVERSITÉS

Le fichier R.java

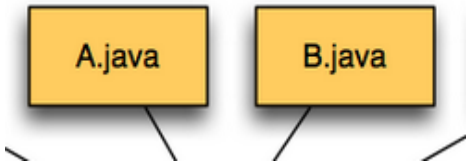



```
package test.biblio;
public final class R {
   …
   public static final class layout {
      public static final int main=0x7f030000;
   }}
```

- /test/biblio/R.java AUTO-GENERATED FILE. DO NOT MODIFY.

Un premier source java, juste pour la syntaxe…

```
package test.biblio;
import android.app.Activity;
import android.os.Bundle;
```




```
public class Demo extends Activity {
    ....
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main); // association IHM <-> Activity

        ....
}
}
```

```
package test.biblio;
public final class R {
   ...
   public static final class layout {
      public static final int main=0x7f030000;
   }}
```

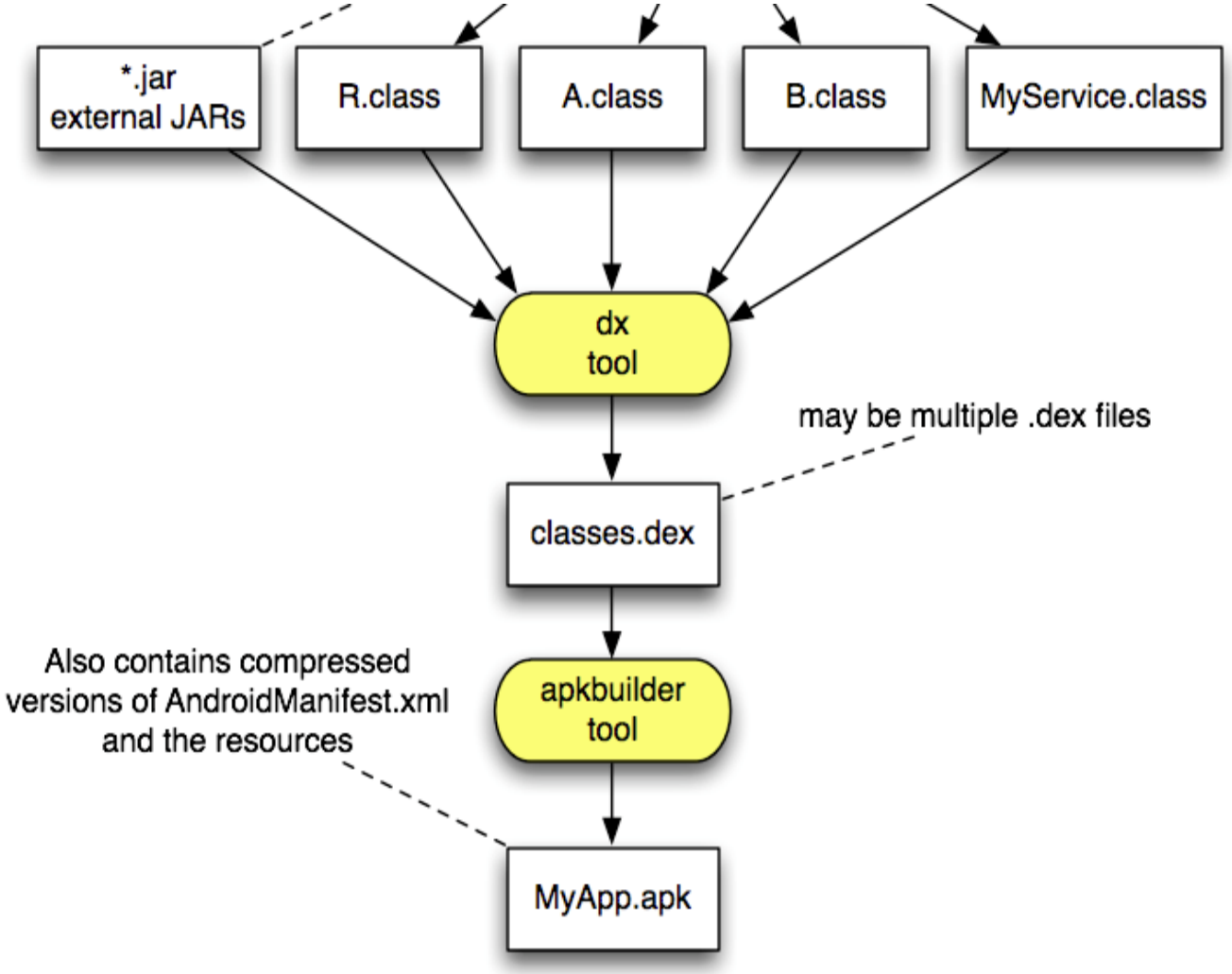
Un premier source java, juste pour la syntaxe…

```
public class Demo extends Activity {
      private EditText editText;
      private Button button;

public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

      setContentView(R.layout.main); // association //récupération de l'EditText grâce à son ID
      editText = (EditText) findViewById(R.id.EditTextPrenom);
      //récupération du bouton grâce à son ID
      button = (Button) findViewById(R.id.ButtonEnvoyer);
      //on applique un écouteur d'évenement au clique sur le bouton
      button.setOnClickListener(
        new OnClickListener() {
              @Override
              public void onClick(View v) {
                    ?? // On récupère le texte écrit dans l'EditText
                    ?? // On crée une pop-up Toast
                    ?? // On affecte le texte dans l'objet TextView
              }
        }
      );}
}
```

Obtention de l'application

- De tous les .class en .dex
  - De la JVM à la machine Dalvik
  - D'une machine à pile en machine à registres
- Génération de l'application .apk
  - Une archive signée
  - Téléchargement : émulateur ou mobile

Génération de l'application



Source : http://stuffthathappens.com/blog/wp-content/uploads/2008/11/android_flow.png

Développement 2/2, suite

- Du .class en .dex
  - Assemblage de tous les .class vers un .dex
  - Une machine par application, un processus Linux
  - Les applications communiquent via l'intergiciel
    - Une application peut être composée de plusieurs activités
    - Les activités communiquent via des variables globales, de la mémoire persistante,…
- Génération de l'application .apk
  - Assemblage, édition des liens
  - Une archive signée
  - Téléchargement : émulateur ou mobile

Introduction aux Applications

- Une présentation, un vocabulaire

- Mots-clés
  - Applications,
  - Communication, évènements, intentions,
  - Services en tâche de fond,
  - Persistance.

- Une Application est composée d'une ou de plusieurs *Activity*
  - *Une activity*
  - Surcharge de certaines méthodes,
    - Du déjà vu : Applet, MIDlet, Servlet,…
  - Le cycle de vie est imposé par le framework
    - Déjà vu : pour une Applette init() puis start() …

Vocabulaire

- Les Essentiels
  - Activity
  - BroadcastReceiver
  - Service
  - ContentProvider

Introduction … Classes

- Activity
  - Une interface utilisateur
    - Démarre d'autres activités, émet des évènements(intentions, intent)
  - Une configuration de type XML, permissions, librairies,
- BroadcastReceiver
  - Bus de messages
  - Émission et réception d'intentions
- Service
  - Pas d'interface, un service à rendre, en tache de fond
  - Intention de servir
- ContentProvider
  - Données rendues persistantes (pour d'autres applications)
  - Un fichier, base SQLite

Deux exemples, deux Activity

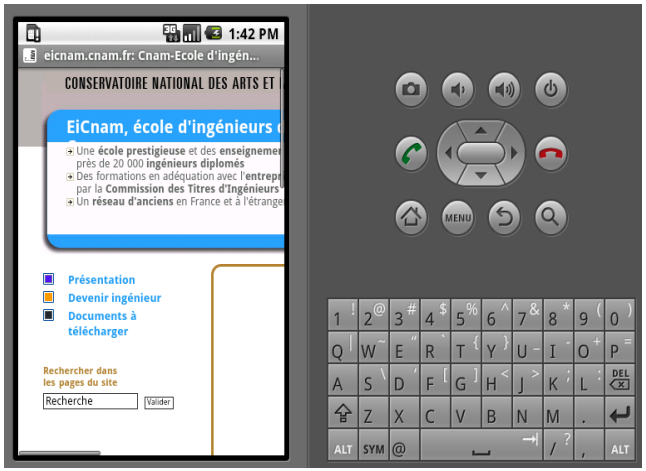
1. Installation d'un navigateur en 2 lignes (WebView)

2. Une toute petite IHM
   - Un écran constitué d'un bouton, d'un écouteur,
   - A chaque clic, l'heure est affichée !

Activity Usage du WebKit

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

public class BrowserDemo extends Activity {
   private WebView browser;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      browser=(WebView)findViewById(R.id.webWiew);
      browser.loadUrl("http://upmc.fr/");
   }
}
```

2 lignes



OnCreate est déclenché par le framework Android

```
public class BrowserDemo extends Activity {
    private WebView browser;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

// installation de l'IHM
        setContentView(R.layout.main);
// accès au composant graphique
        browser=(WebView)findViewById(R.id.webView);

        browser.loadUrl("http://www.upmc.fr/");
    }
}
```

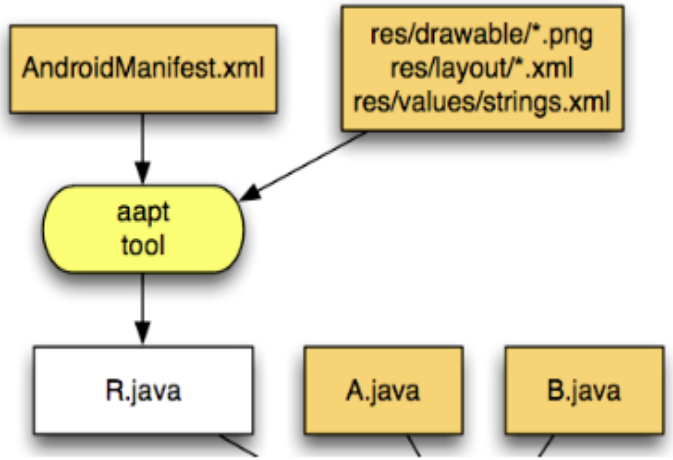
R.layout.*main*, R.id.*webView ?*

Retour sur la configuration XML

- R.id.*webView ? R.layout.main ?*

  – *En Entrée*
    - *Fichiers de configuration XML*

  – *En Sortie*
    - *Source Java, R.java*



Une IHM, deuxième exemple

Une IHM

- Un bouton, un écouteur, un clic et l'heure est affichée !
- En approche *traditionnelle*
  - Tout est codé en Java IHM comprise
- En approche *déclarative*
  - Usage d'XML pour la configuration, de java pour l'utilisation

Click pour actualiser l'heure

```
import android.app.Activity;
import android.os.Bundle;
import static android.view.View.OnClickListener ;
import android.widget.Button;
import java.util.Date;


public class Now extends Activity implements OnClickListener {
  private Button btn;

  @Override
  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    btn = new Button(this);       // <- un bouton
    btn.setOnClickListener(this);// <- un écouteur auprès de cette vue

    setContentView(btn);          // <- le bouton occupe l'écran
  }

  public void onClick(View view) {  // <- à chaque click
    btn.setText(new Date().toString());
  }
}
```

Discussion : Vue apparentée swing
Ici un MVC à lui tout seul …

Activity, méthodes à redéfinir

- MonActivity extends `android.app.Activity;`

```
@Override
  protected void onCreate(Bundle
   savedInstanceState){
```

Core Components-Activity #1

- 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
- Activity.setContentView() method is used to set a certain hierarchy of view objects

Core Components-Activity #2

- Activities are activated by asynchronous messages called *intents*
  - An intent is an Intent object that holds the content of the message
  - The action being requested or the URI of the data to act on
- The <intent-filter> label in AndroidManifest.xml file specifies the Intent that can start the Activity

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

  - declares the main activity, it will be started automatically when the app starts
- An activity is launched (or given something new to do) by passing an Intent object to Context.startActivity() or Activity.startActivityForResult()

Other Core Components

- Service
  - A *service* doesn't have a visual user interface, runs in the background for a period of time
- Broadcast receivers
  - a component that does nothing but receive and react to broadcast announcements
- Content providers
  - A *content provider* makes a specific set of the application's data available to other applications.
  - The data can be stored in the file system, in an SQLite database, or in any other manner that makes sense

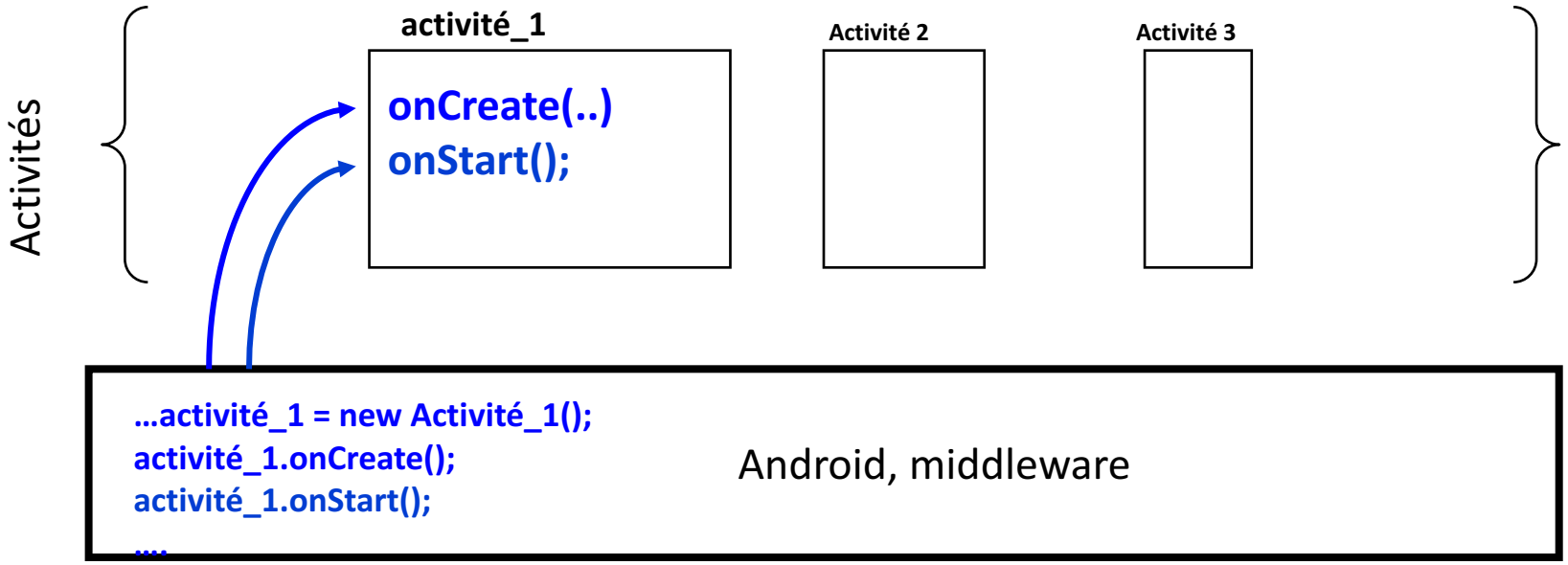
android.app.Activity

package android.app;

```
public class Activity extends ApplicationContext {
        protected void onCreate(Bundle savedInstanceState){
          protected void onStart();
          protected void onRestart();
          protected void onResume();
          protected void onPause();
          protected void onStop();
          protected void onDestroy();
        … etc …
}
```

induit un cycle de vie imposé par le « framework »

Inversion de Contrôle… Rappel




```
public class Activity extends ApplicationContext {   1
      protected void onCreate(Bundle savedInstanceState);
      protected void onStart();
       …..}
```

http://developer.android.com/reference/android/app/Activity.html

Petites précisions, rappels …

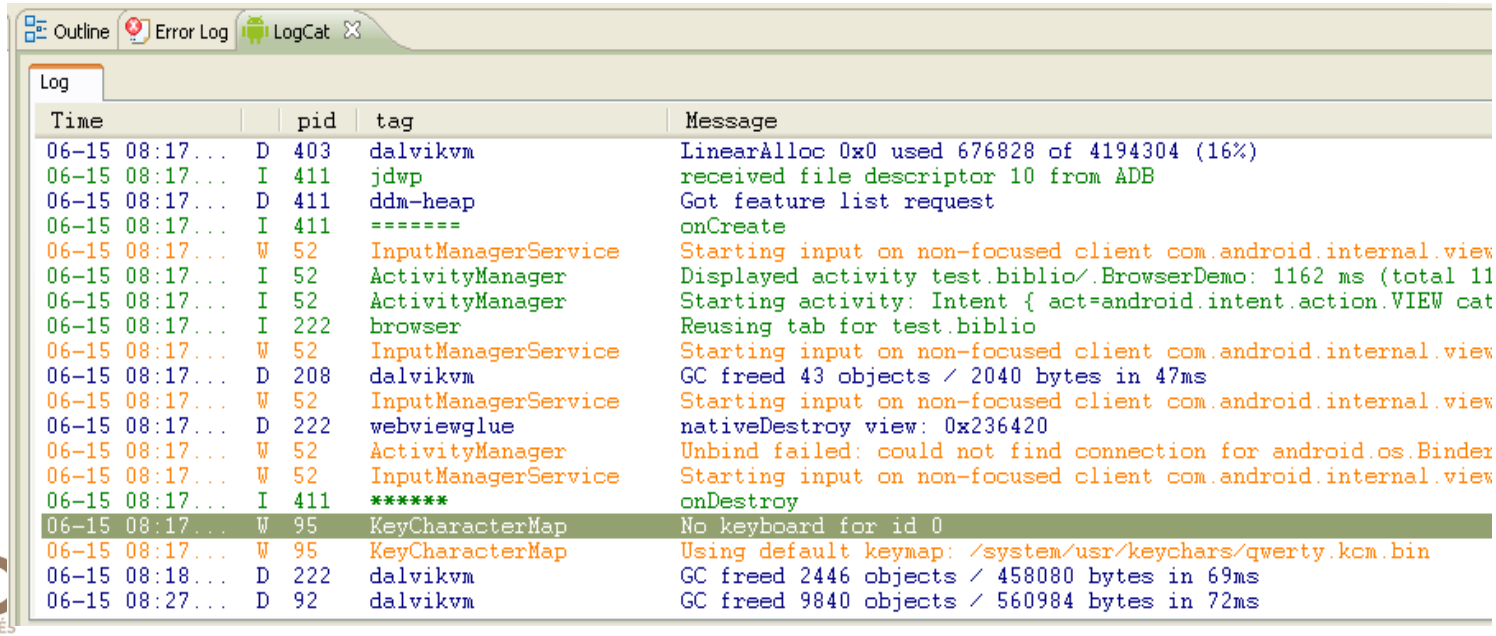
- Une application s'exécute dans un processus Linux

- Depuis ce processus
  - les méthodes d'une activité sont appelées selon un certain ordre
    - onCreate …onPause…. onResume… onPause …. onResume… onDestroy

  - L'appel de onDestroy n'engendre pas l'arrêt du processus initiateur

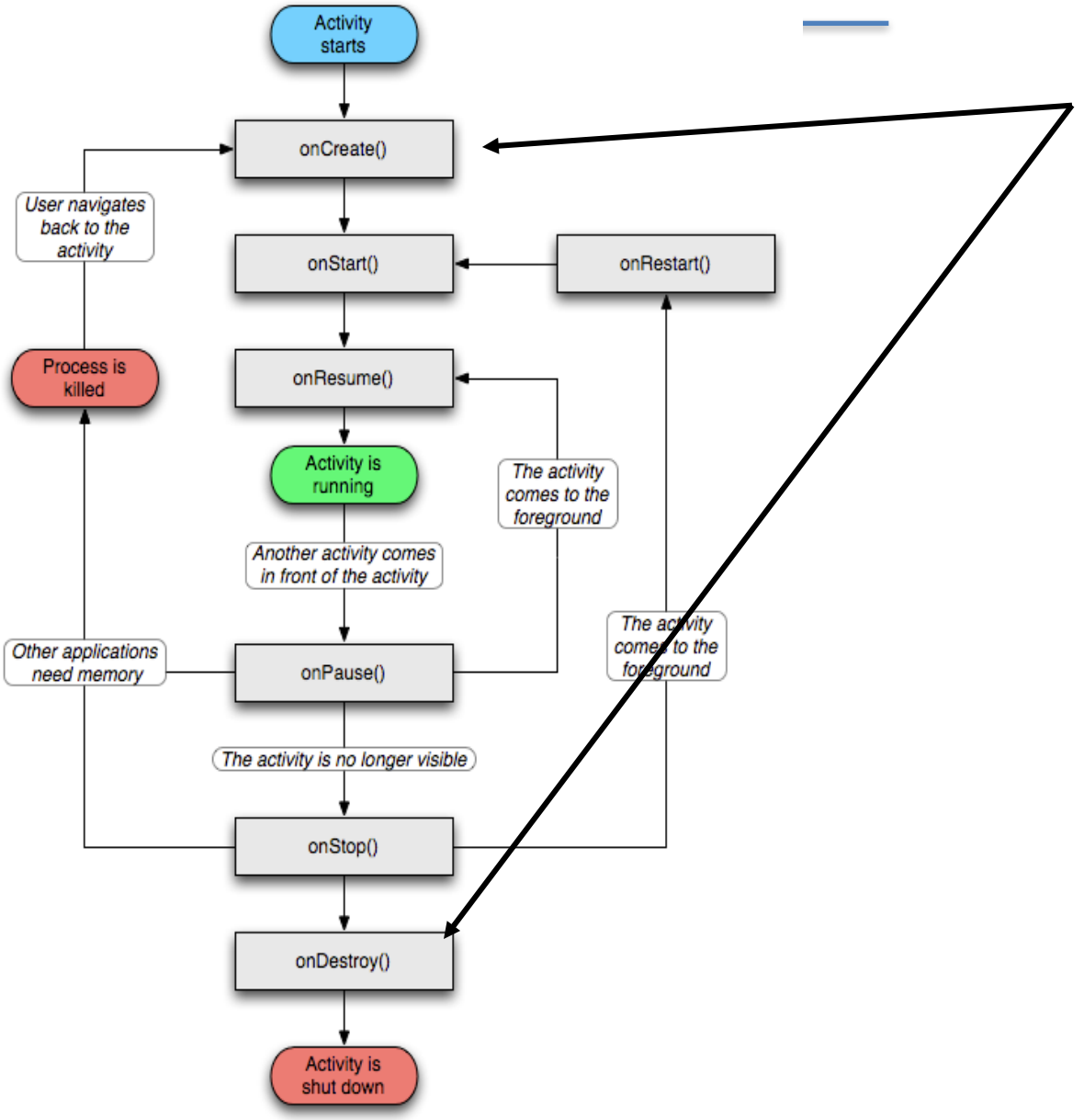
Démonstration, Activity dans tous ses états

```
public class BrowserDemo extends Activity {
  private WebView browser;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i("=======","onCreate");
        // cf. page précédente
    }
    public void onDestroy(){
      super.onDestroy();
      Log.i("******", "onDestroy");
    }
}
```



Time		pid	tag	Message
06-15 08:17...	D	403	dalvikvm	LinearAlloc 0x0 used 676828 of 4194304 (16%)
06-15 08:17...	I	411	jdwp	received file descriptor 10 from ADB
06-15 08:17...	D	411	ddm-heap	Got feature list request
06-15 08:17...	I	411	=======	onCreate
06-15 08:17...	W	52	InputManagerService	Starting input on non-focused client com.android.internal.view
06-15 08:17...	I	52	ActivityManager	Displayed activity test.biblio/.BrowserDemo: 1162 ms (total 11
06-15 08:17...	I	52	ActivityManager	Starting activity: Intent { act=android.intent.action.VIEW cat
06-15 08:17...	I	222	browser	Reusing tab for test.biblio
06-15 08:17...	W	52	InputManagerService	Starting input on non-focused client com.android.internal.view
06-15 08:17...	D	208	dalvikvm	GC freed 43 objects / 2040 bytes in 47ms
06-15 08:17...	W	52	InputManagerService	Starting input on non-focused client com.android.internal.view
06-15 08:17...	D	222	webviewglue	nativeDestroy view: 0x236420
06-15 08:17...	W	52	ActivityManager	Unbind failed: could not find connection for android.os.Binder
06-15 08:17...	W	52	InputManagerService	Starting input on non-focused client com.android.internal.view
06-15 08:17...	I	411	******	onDestroy
06-15 08:17...	W	95	KeyCharacterMap	No keyboard for id 0
06-15 08:17...	W	95	KeyCharacterMap	Using default keymap: /system/usr/keychars/qwerty.kcm.bin
06-15 08:18...	D	222	dalvikvm	GC freed 2446 objects / 458080 bytes in 69ms
06-15 08:27...	D	92	dalvikvm	GC freed 9840 objects / 560984 bytes in 72ms

En résumé : émulateur + LogCat , traces bien utiles

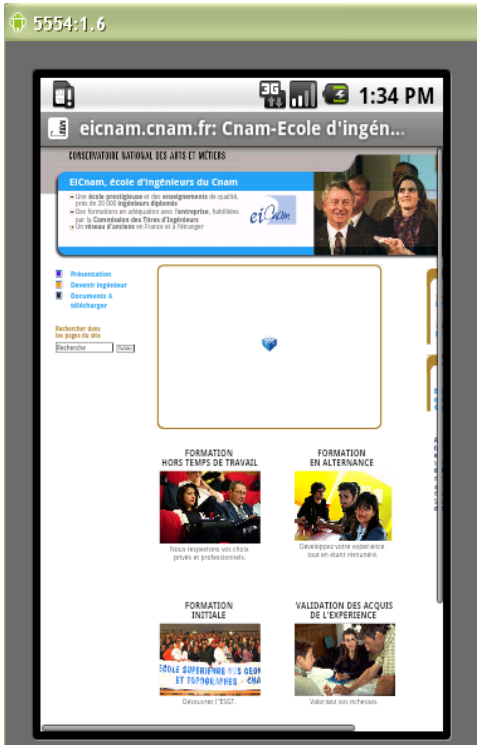


Activity dans tous ses états

```
public class BrowserDemo extends Activity {

  public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);Log.i("=======","onCreate");}

   public void
   onStart(){super.onStart();Log.i("=======","onStart");}

   public void onResume(){
     super.onResume();
     Log.i("=======","onResume");
   }

   public void onPause(){
     super.onPause();
     Log.i("=======","onPause");
   }

   public void onStop(){
     super.onStop();
     Log.i("******","onStop");
   }

    public void onDestroy(){
      super.onDestroy();
      Log.i("******", "onDestroy");
    }
}
```

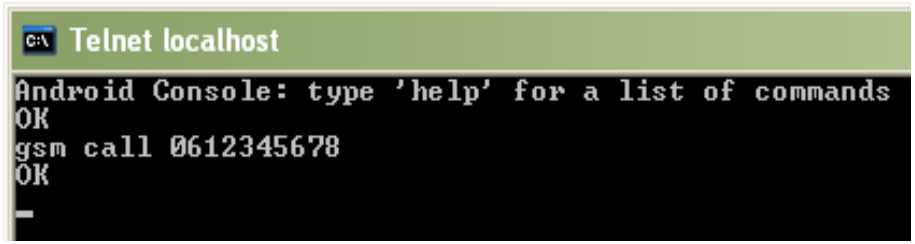
OnPause -> onResume

1)

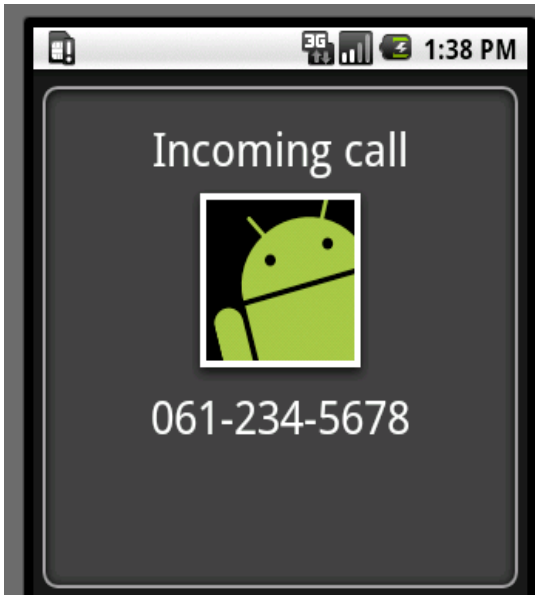


OnPause

2-1) telnet localhost 5554



2-2)



2)

OnResume

3)

Vous pouvez tout simuler avec telnet

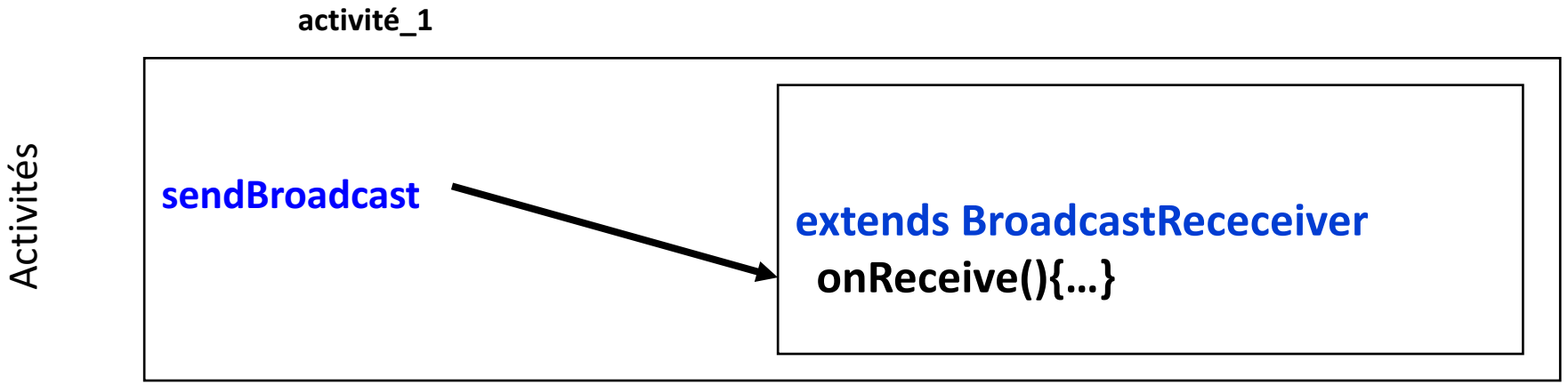
UPMC SORBONNE UNIVERSITÉS

Intention de … porter un toast

- Intent

```
Toast.makeText(context,intent.getStringExtra("message"),3000).show();
```

Android : Intents, comme Intention



Il faut :

- Un receveur

```
public class Receiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    ….
```

- Enregistrement auprès du middleware
- Une intention d'émettre
- Ajuster la configuration

```
<receiver android:name=".TestBroadcastReceiver"
android:label= "Test Broadcast receiver" />
```

Intention de

Un receveur, qui porte un toast : abonné

```
public static class Receiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context,intent.getStringExtra("message"),3000).show();
  }
}
```

Enregistrement auprès du middleware : abonnement

```
Receiver receiver = new Receiver();
registerReceiver(receiver, new IntentFilter("PORTER_UN_TOAST"));
registerReceiver(receiver,new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
```

Émission de l'intention : notification

```
intent = new Intent("PORTER_UN_TOAST");
intent.putExtra("message", getResources().getString(R.string.message));
sendBroadcast(intent);
```

- Intent.*ACTION_AIRPLANE_MODE_CHANGED, un abonné à cet Intent*

Intention… auprès du middleware

Enregistrement auprès du middleware :

```
Receiver receiver = new Receiver();
registerReceiver(receiver, new IntentFilter("PORTER_UN_TOAST"));

registerReceiver(receiver,new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
```

- Intent.*ACTION_AIRPLANE_MODE_CHANGED, je suis maintenant abonné à cet Intent*

Avec

```
private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
```

et

```
registerReceiver(receiver,new IntentFilter(SMS_RECEIVED));
```

Je suis maintenant abonné aux réception de SMS
(si la permission est installée dans le AndroidManifest.xml)

Sous Android

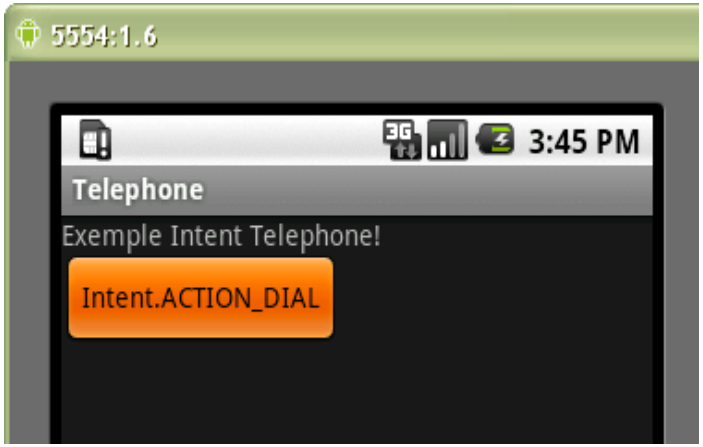
- Mediator, classe Context
  - http://developer.android.com/reference/android/content/Context.html
- Subscriber, classe BroadcastReceiver
  - http://developer.android.com/reference/android/content/BroadcastReceiver.html
- X,Y les thèmes, classe Intent
  - http://developer.android.com/reference/android/content/Intent.html
- IntentFilter
  - http://developer.android.com/reference/android/content/IntentFilter.html

Intention de téléphoner ☺

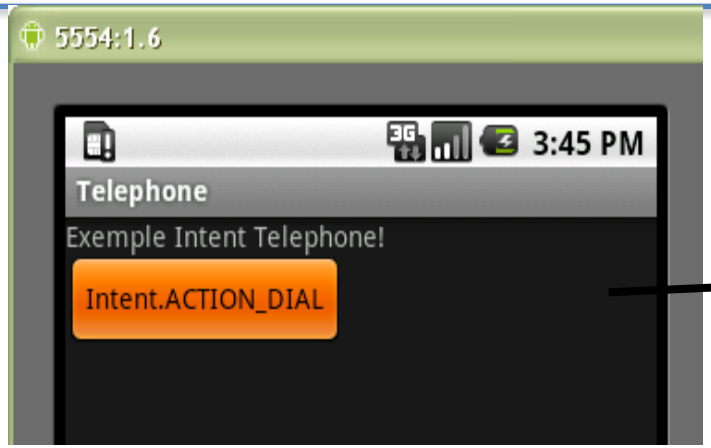
- Intent
- Comme Communication
- Démarrer une activité prédéfinie : téléphoner

Intention de téléphoner

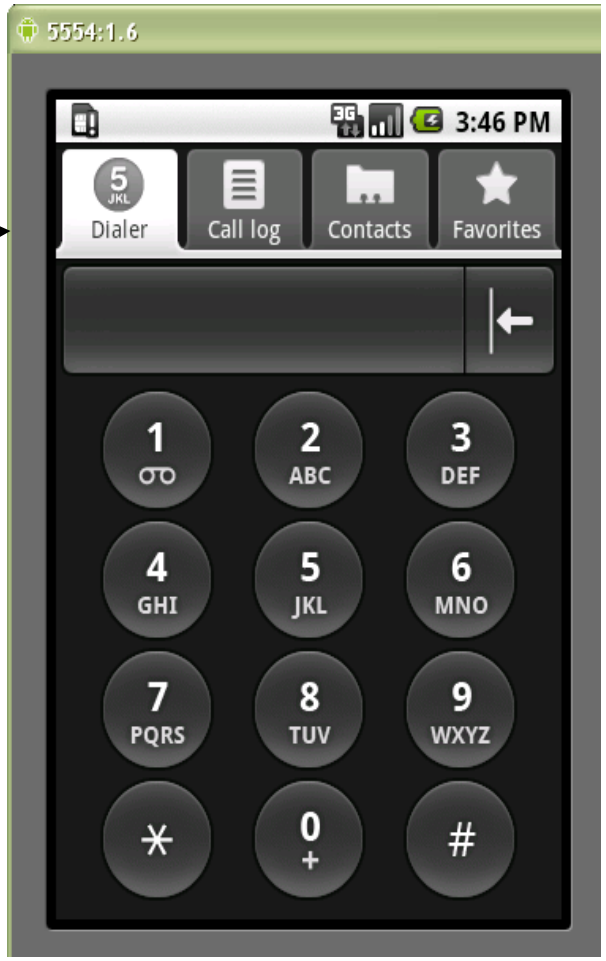
```
public class Telephone extends Activity implements  OnClickListener {

public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      ((Button)this.findViewById(R.id.Button01)).setOnClickListener(this);
   }

   public void onClick(View v){
      Intent intent = new Intent(Intent.ACTION_DIAL);
      startActivity(intent);
   }
}
```

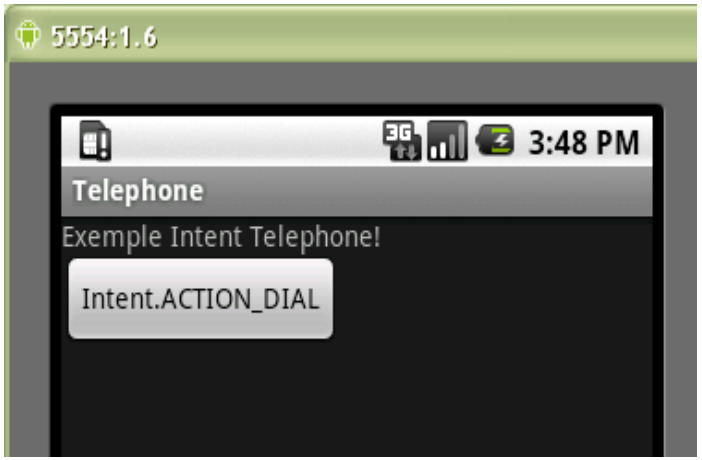


En Images, activité de téléphoner



Click
empiler





Click
dépiler

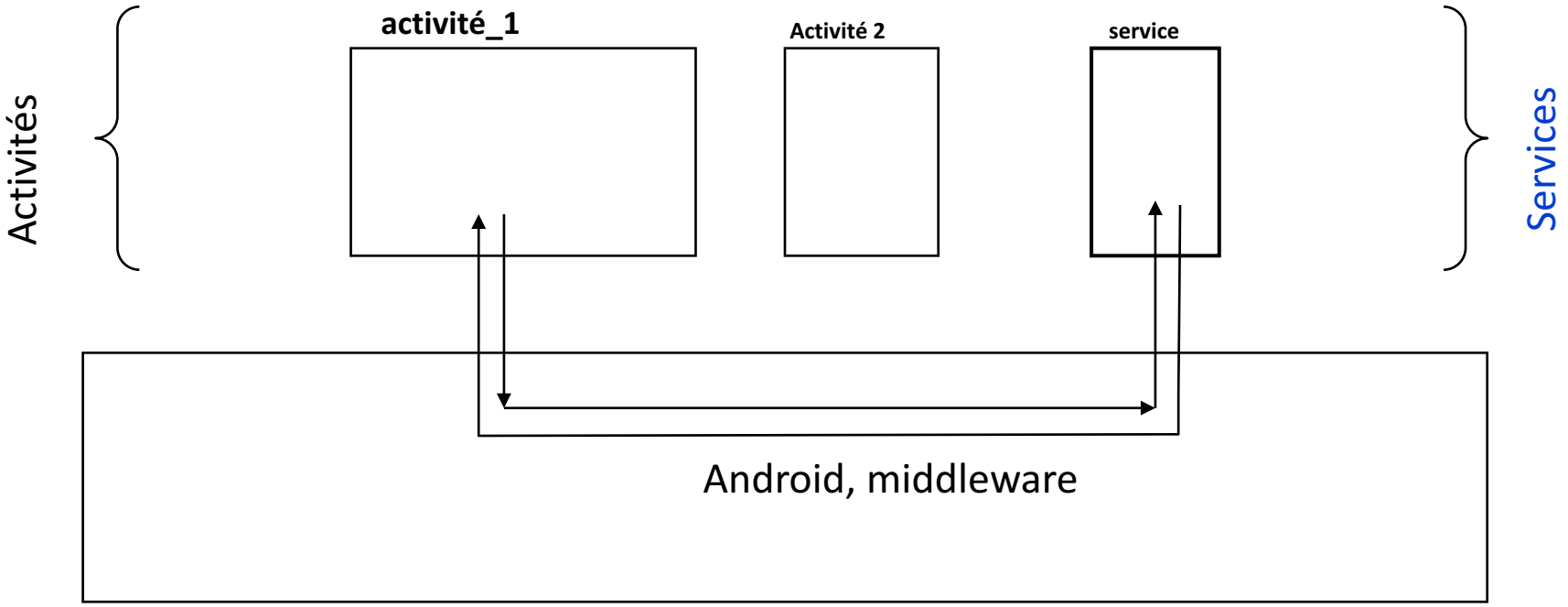
Activités un résumé

- Activity
  - Cycle de vie
  - Géré par le framework
- startActivity
  - Une activité peut en démarrer une autre
  - Transmission des paramètres par les intentions
- Communication entre deux activités
  - Discussion
- Une Application peut contenir plusieurs activités
  - Communication entre activités d'applications différentes
    - Exemple des contacts

Service

- En tache de fond
- Toujours disponible
  - Locaux
    - Service personnel, inaccessible pour les autres applications
  - Distants
    - Rmi en plus simple … même machine
    - J'écoute un fichier audio mp3, pendant que je navigue sur le web

Un Service



- Service distant,
  - Découverte du service par une intention …

SMSService : un exemple

- Un compteur de SMS reçus
  - A chaque sms reçu un compteur est incrémenté
  - Service accessible par toute application

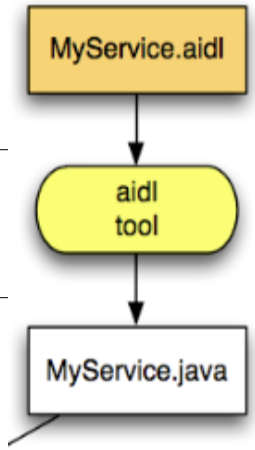
- Le service a l'intention de recevoir des SMS
  - IntentFilter filter = new IntentFilter(*SMS_RECEIVED*);
  - registerReceiver(new SMSReceiver(), filter);

- Un client recherchera ce service via le middleware

- Le service : SMSService

Service, aidl (android interface description language)

```
package cnam.android;

interface SMSService{
  void start();
  void stop();
  long received();
}
```




```
/*
This file is auto-generated.  DO NOT MODIFY.
*/
package cnam.android;
import …;
public interface SMSService extends android.os.IInterface{

/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder implements cnam.android.SMSService{
…
```

Le Client recherche le service

private SMSService statsSMS;

```
bindService(new Intent(SMSService.class.getName()),
            connexion,
            Context.BIND_AUTO_CREATE);
}


// Traitement asynchrone de la réponse venant de l'intergiciel

private ServiceConnection connexion = new ServiceConnection() {
  public void onServiceConnected(ComponentName name, IBinder service) {
    Log.v("service","onServiceConnected()");

    // réception de la souche
    Client.this.statsSMS = SMSService.Stub.asInterface(service);
  }

  public void onServiceDisconnected(ComponentName name) {
    Client.this.statsSMS = null;
  }
};
```

Le service 1/3, réveillé à chaque SMS

```
public class SMSServiceImpl extends Service {

private static final String TAG = "SMSServiceImpl";
private static final String SMS_RECEIVED =
   "android.provider.Telephony.SMS_RECEIVED";

    private boolean active;
    private int     countSMSReceived;

  public void onCreate() {
    super.onCreate();
    IntentFilter filter = new
   IntentFilter(SMS_RECEIVED);
    registerReceiver(new SMSReceiver(), filter);
}
```

Le service 2/3, réveillé à chaque SMS

```
private class SMSReceiver extends BroadcastReceiver{
  public SMSReceiver(){
    Log.v("SMSReceiver","SMSReceiver()");
  }

@Override
public void onReceive(Context context, Intent
  intent) {
  Log.v("SMSReceiver","onReceive()");
  if(active) SMSServiceImpl.this.countSMSReceived++;
}
}
```

Le service 3/3, La souche le stub fournie à la demande

```
public IBinder onBind(Intent arg0) {
  return new SMSServiceStubImpl();
}
```

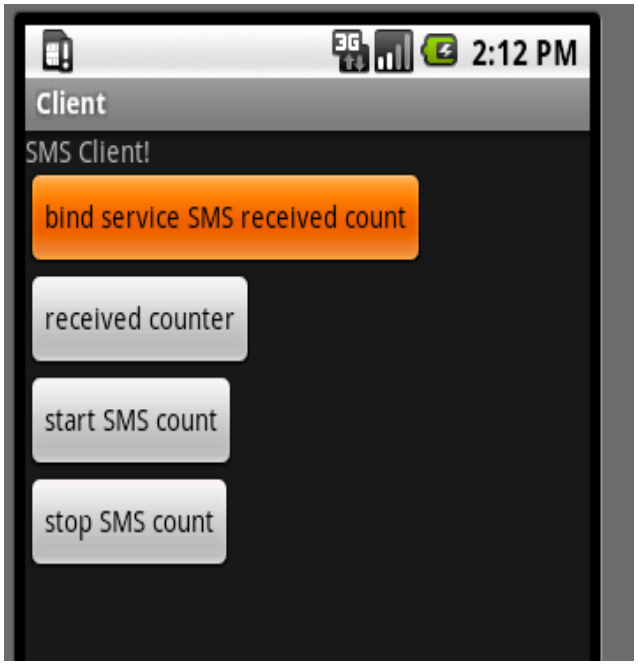
```
public class SMSServiceStubImpl extends SMSService.Stub{

  public long received() throws android.os.RemoteException{
    return SMSServiceImpl.this.countSMSReceived;
  }

  public void start(){
    active = true;
  }
  public void stop() throws RemoteException {
    active = false;
  }
}
```

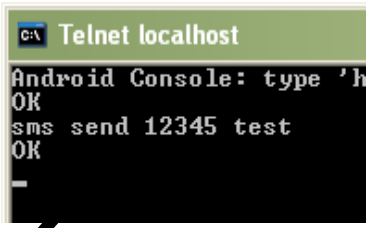
```
<service android:name=".SMSService"
        android:label= "mon service de SMS" />
```

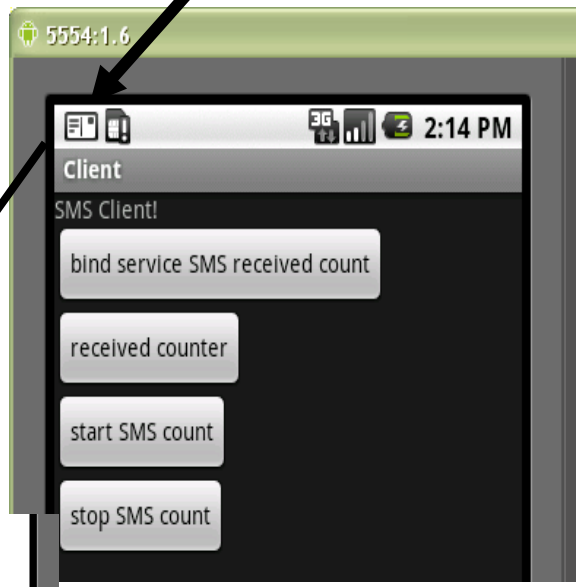
Le Client, une activity

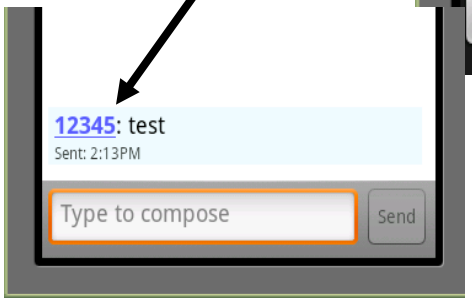
1. bind service
2. start SMS count

telnet localhost 5554
3. sms send 12345 test

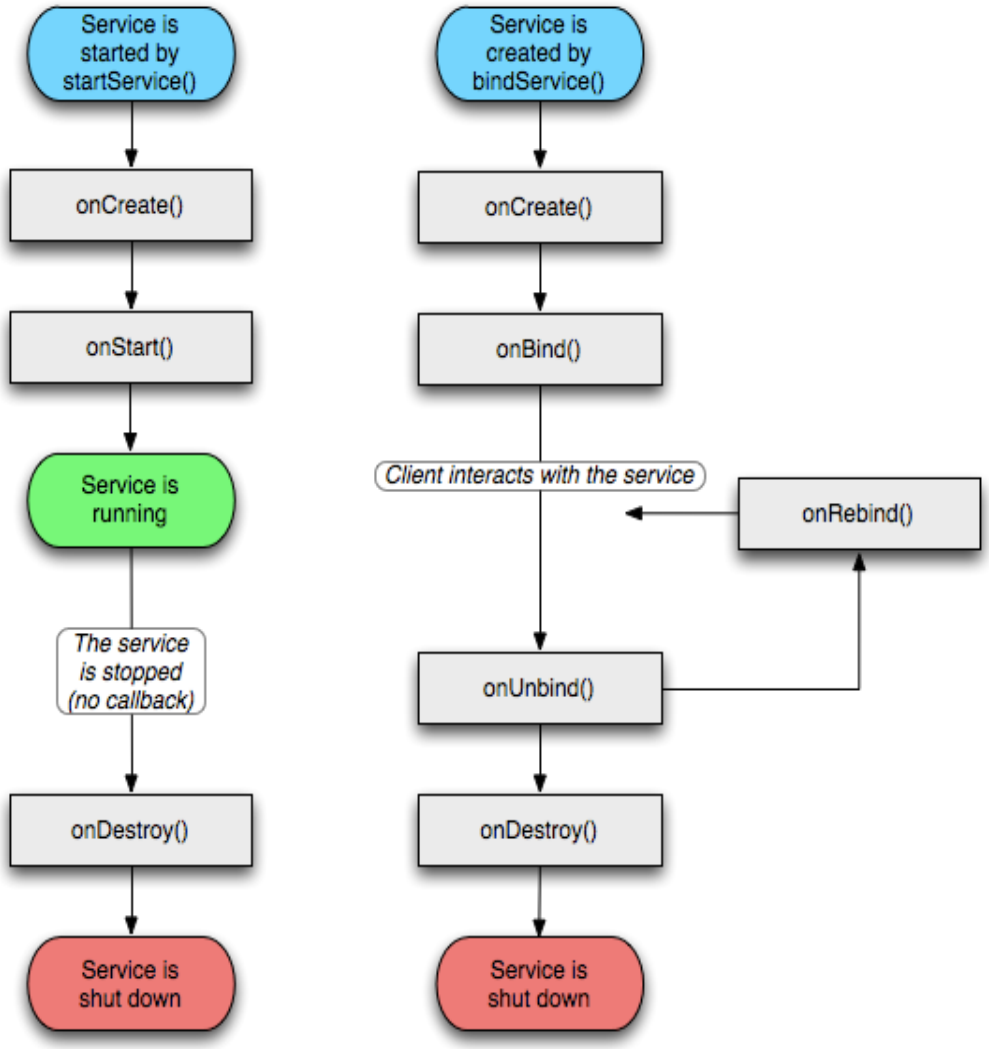








Services cycle de vie

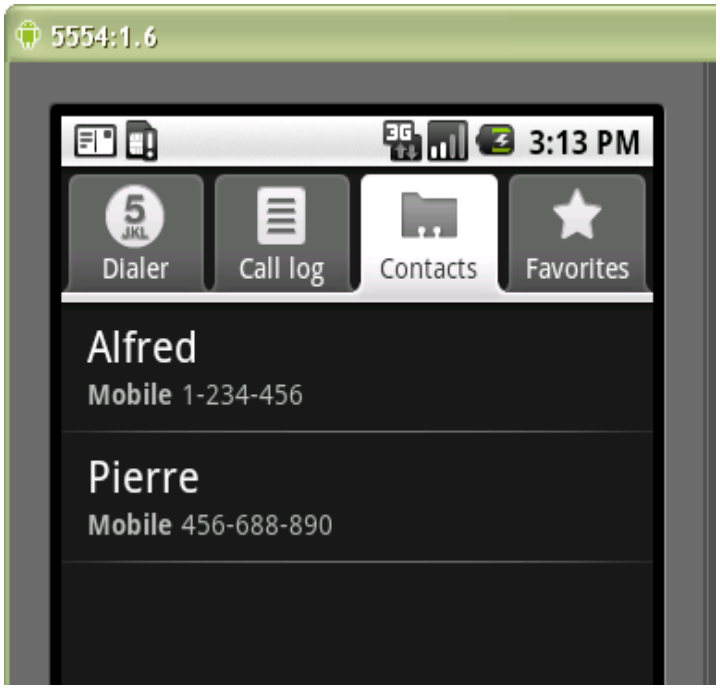


ContentProvider

- Sauvegarde et restitution de données
  - Données accessibles à toutes applications

- ContentProvider
  - Persistance
  - DAO/CRUD
    - Data Access Object
    - Create Retrieve Update et Delete

Mes Contacts, un client
le *ContentProvider* prédéfini : phone



ContentProvider

- ContentProvider, *comme les contacts, accès*
  - ContentResolver resolver = getContentResolver();

  Identification du bon « Contentprovider » grâce à son URI

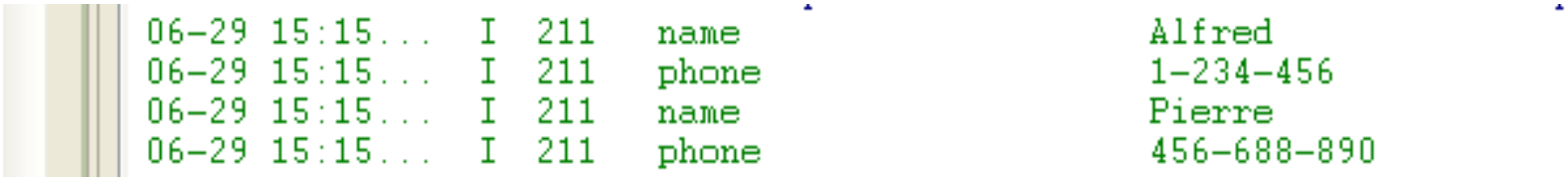
  - Uri uri = android.provider.Contacts.Phones.CONTENT_URI;
  - String s = uri.toString();
  - // assert s.equals("content://contacts/phone");

  - Appel de resolver.query
    - Pour une interrogation

  - Parcours à l'aide d'un « Cursor »

Cursor comme ResultSet

```
ContentResolver resolver = getContentResolver();
Uri uri = android.provider.Contacts.People.CONTENT_URI;
Cursor cr = resolver.query(uri,null,null,null,null);
cr.moveToFirst();
int columnName = cr.getColumnIndex(Contacts.People.NAME);
int columnPhone = cr.getColumnIndex(Contacts.People.NUMBER);
do{
  Log.i("name",cr.getString(columnName));
  Log.i("phone",cr.getString(columnPhone));
}while(cr.moveToNext());
```



Son propre ContentProvider

- public class MonPropreProvider extends ContentProvider

- Avec
  - Une Uri, l'accès à mon propre « *content provider* »
    - content://cnam.android.agenda/liste
    - content://….
  - Les méthodes
    - insert, update, delete et query
  - Et dans le fichier de configuration

```
<provider android:name="MonPropreProvider"
          android:authorities="cnam.android.agenda" />
```

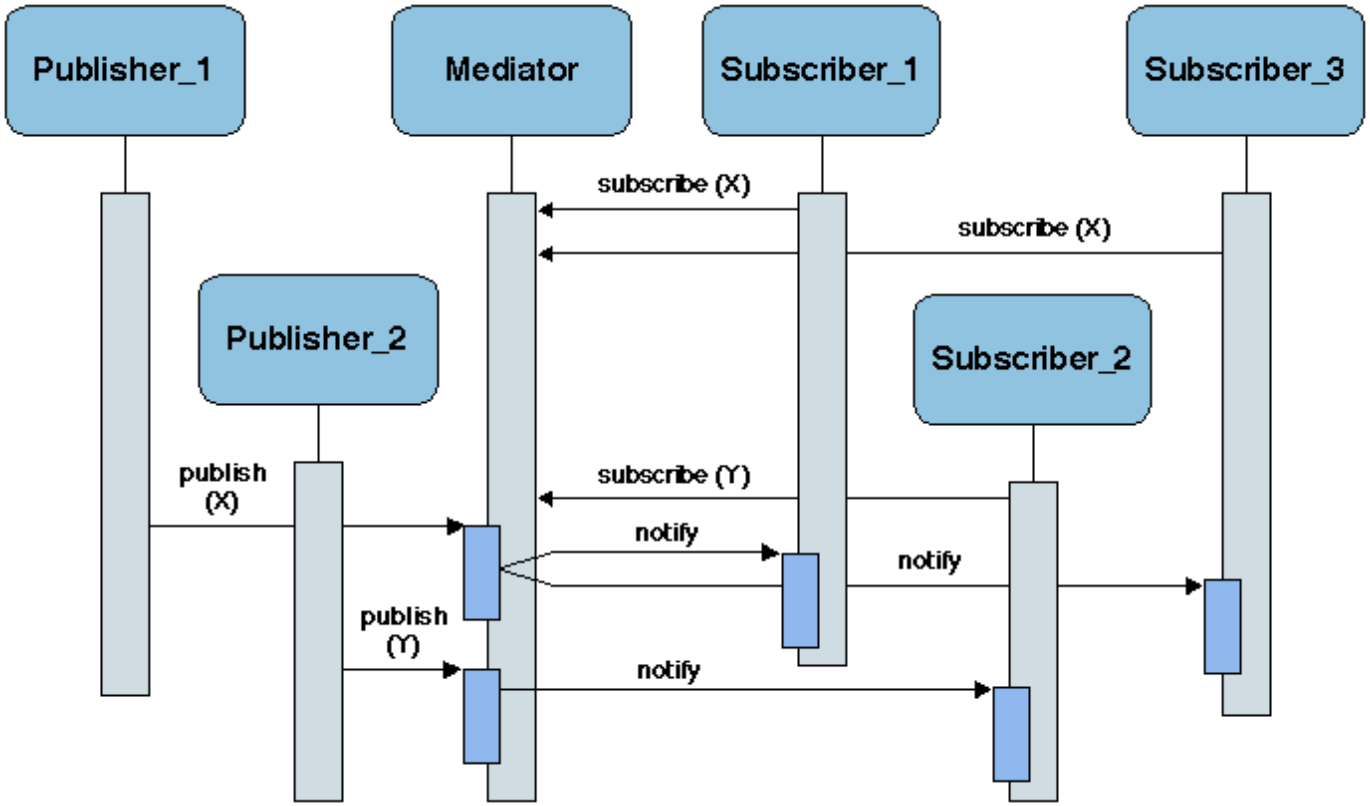
Exercice final

- Une application SMS
  - 3 Views
    - Contacts and stats
    - SMS Check
    - Parameters
  - N Activities (?)
  - 1 Service – SMS receive
  - L'application doit au minimum
    - Récupérer la liste des contacts
    - Permettre de définir un ensemble de messages types
    - Afficher toutes les informations
  - Cycle de vie
    - L'application récupère les SMS
    - Calcule des statistiques (nombre de message par contact)
    - Effectue un envoi automatique de SMS de réponse en fonction des stats et des messages pré-enregistrés de réponse
  - Bonus ?

Communication inter applications

- Intent, comme notification
  - Souscription/Publication
  - Appeler, déclencher une autre activité
    - Avec éventuellement des paramètres et des résultats attendus
  - L'intergiciel sélectionne la « bonne application »
    - Selon les critères exigés par le souscripteur
    - Critères précisés lors de la publication
      - Nécessaire adéquation à l'exécution services fournis / services requis

Patron Publish-Subscribe/Intent



- Source: http://roboticswikibook.org, rappel

Sous Android

- Mediator, classe Context
  - http://developer.android.com/reference/android/content/Context.html
- Subscriber, classe BroadcastReceiver
  - http://developer.android.com/reference/android/content/BroadcastReceiver.html
- X,Y les thèmes, classe Intent
  - http://developer.android.com/reference/android/content/Intent.html
- IntentFilter
  - http://developer.android.com/reference/android/content/IntentFilter.html

3 Exemples

1. Un *publisher* s'adresse à l'intergiciel
   - Une activité souhaite téléphoner …
     - À la recherche d'une application prédéfinie

2. Un *subscriber* de SMS entrants
   - Une activité souhaite filtrer le contenu des SMS entrants…
     - Installation d'un *Receiver*

3. Un *publisher* et un *subscriber*
   - Une activité souhaite publier un résultat à l'intention d'autres activités
     - Le publisher d'adresse à l'intergiciel qui sélectionne les abonnés

Publisher: Un mobile à l'Intention de téléphoner

- Intent

- Démarrer une activité prédéfinie : téléphoner, *légitime…*

```
public class TelephonerActivity extends Activity {

  @Override
  protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_now);
  }

  public void onClick(View view) {
     Intent intent = new Intent();
     intent.setAction("android.intent.action.DIAL");
     this.startActivity(intent);
  }

}
```

- `android.app.Activity extends ….. extends android.content.Context`

Publisher : Téléphoner

- Le souscripteur a dû déjà souscrire …

- Discussions

```
Intent intent = new Intent();
intent.setAction("android.intent.action.DIAL");
```

  - À la recherche de la bonne application effectuée par Android
    - Ici un nom `("android.intent.action.DIAL");`
    - Plusieurs élus possibles
    - Composant logiciels,
    - Maintenance
    - …

- Le patron chaîne de responsabilités est utilisé

Subcriber : Recevoir un SMS (souscrire)

- Le souscripteur hérite de *android.content.BroadcastReceiver*
  - Et implémente ce qu'il faut faire lors de la notification

```
public class ReceiverSMS extends BroadcastReceiver{
    // à chaque SMS reçu
    public void onReceive(Context ctxt, Intent intent) {
    //
    }
}
```

// à chaque SMS reçu ?

Adéquation intention (Intent) ReceiverSMS par un filtre (IntentFilter)

Réalisation : tout en java ou directives XML …

Subcriber : Recevoir un SMS (souscrire)

- Tout java: un receveur/souscripteur au sein d'une activité

```
public class SMSActivity extends Activity {

  private static class ReceiverSMS extends BroadcastReceiver{
    // à chaque SMS reçu
    public void onReceive(Context ctxt, Intent intent) {
    // …
    }}

protected void onCreate(Bundle bundle) {
  super.onCreate(bundle);

  final String SMS_RECEIVED =
       "android.provider.Telephony.SMS_RECEIVED";

  IntentFilter filter = new IntentFilter(SMS_RECEIVED);
  registerReceiver(new ReceiverSMS(), filter);

  setContentView(R.layout.activity_now);
}
```

Subcriber : Le receveur en déclaratif

```
 <receiver android:name=".ReceveurDeSMS"
   android:exported="true" >
   <intent-filter android:priority="9999">
     <action
    android:name="android.provider.Telephony.SMS_RECEIVED"
    />
   </intent-filter>
</receiver>
```

```
public class ReceveurDeSMS extends BroadcastReceiver{
    // à chaque SMS reçu
    public void onReceive(Context ctxt, Intent intent) {
    //
    }
}
```

Chaîne de responsabilités

Publish/Subscribe

Plusieurs Receveurs/souscripteurs,

l'un deux peut arrêter la propagation,

une priorité peut être affecté à chaque receveur

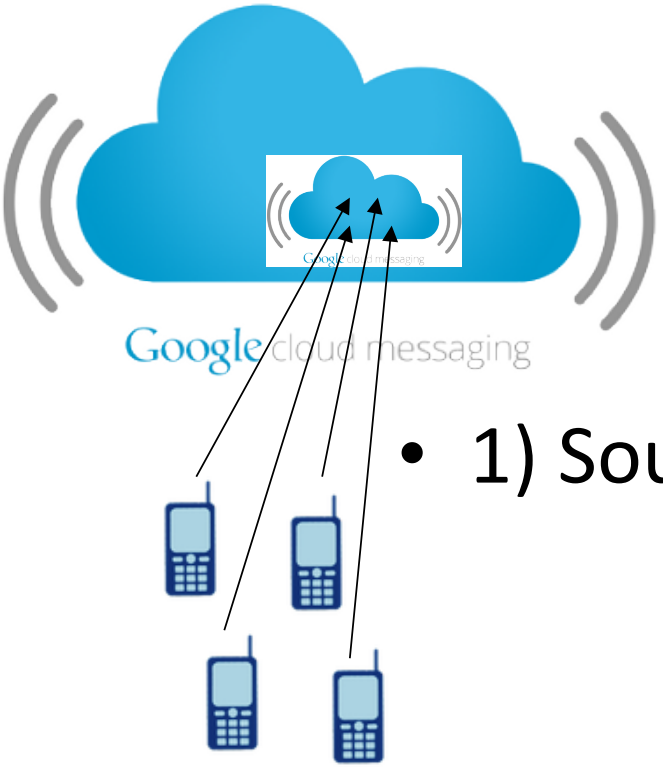
La liste des souscripteurs serait-elle :

une Chaîne de responsabilités avec priorités ?

Patron publish/subscribe « Over the air »

- Google Cloud Messaging
- Android 2.2
- Message <= 4ko
- Trafic illimité
  - Nécessite une inscription auprès de Google
    - Avec de préférence un compte gmail dédié

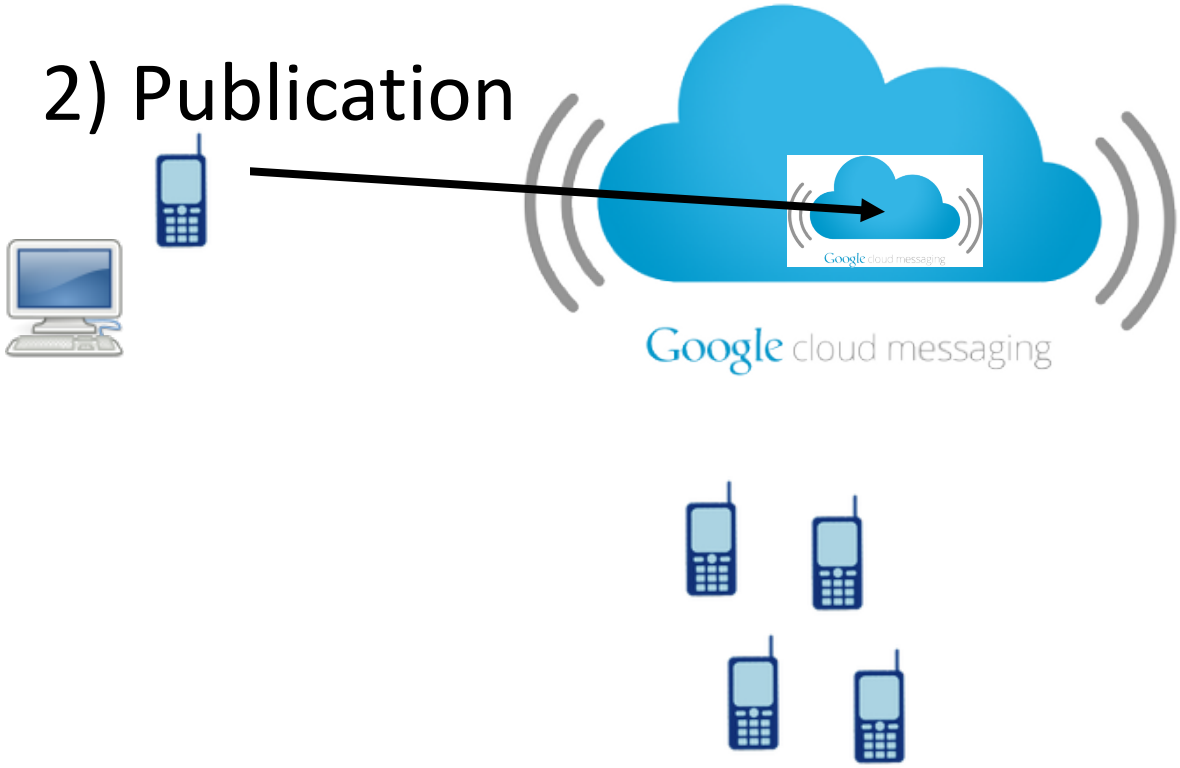
Publish/Subscribe « Over the air »



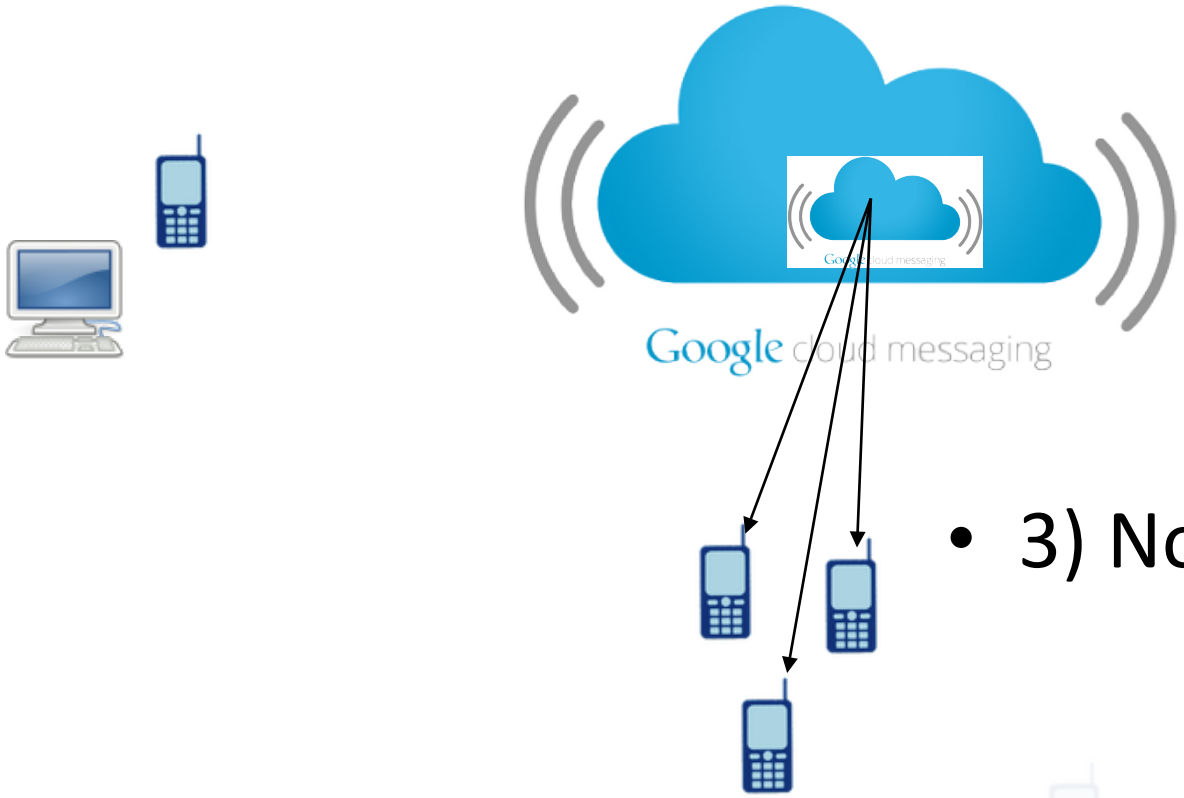
- 1) Souscription

Publish/Subscribe « Over the air »

- 2) Publication

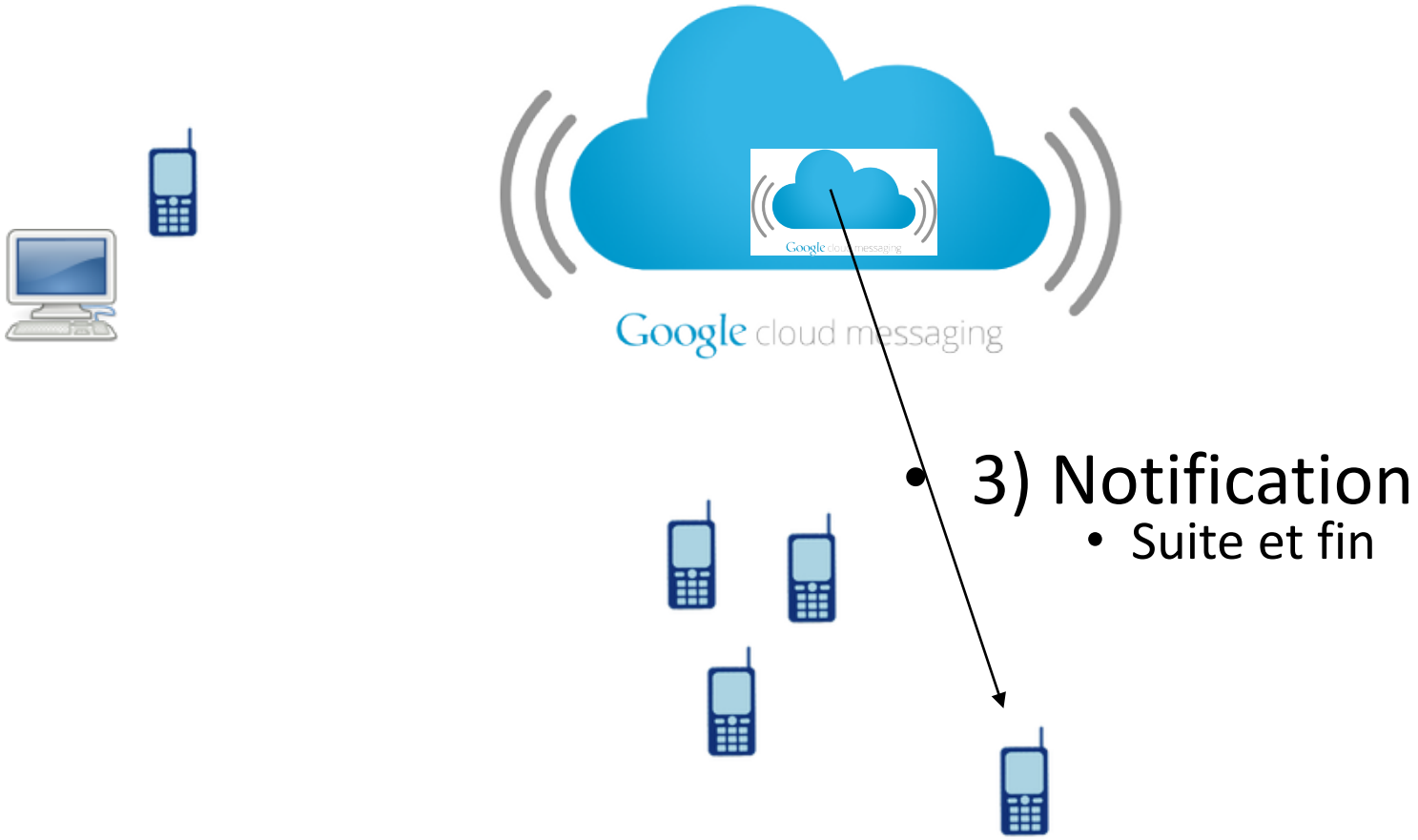


Publish/Subscribe « Over the air »



- 3) Notification

Publish/Subscribe « Over the air »



- 3) Notification
  - Suite et fin

SMS Sending

- STEP 1
  - In the AndroidManifest.xml file, add the two permissions - SEND_SMS and RECEIVE_SMS.
- STEP 2
  - In the main.xml, add Text view to display "Enter the phone number of recipient“ and "Message"
  - EditText with id txtPhoneNo and txtMessage
  - Add the button ID "Send SMS“

SMS Sending

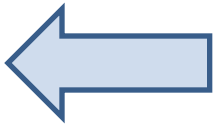
- Step 3 Import Classes and Interfaces

```
import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
```

SMS Sending

- Step 4 Write the SMS class

```
public class SMS extends Activity {
  Button btnSendSMS;
  EditText txtPhoneNo;
  EditText txtMessage;

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

    btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
    txtPhoneNo = (EditText) findViewById(R.id.txtPhoneNo);
    txtMessage = (EditText) findViewById(R.id.txtMessage);

    btnSendSMS.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        String phoneNo = txtPhoneNo.getText().toString();
        String message = txtMessage.getText().toString();

        if (phoneNo.length()>0 && message.length()>0)
          sendSMS(phoneNo, message);
        else
          Toast.makeText(getBaseContext(),
            "Please enter both phone number and message.",
            Toast.LENGTH_SHORT).show();
      }
    });
  }
}
```



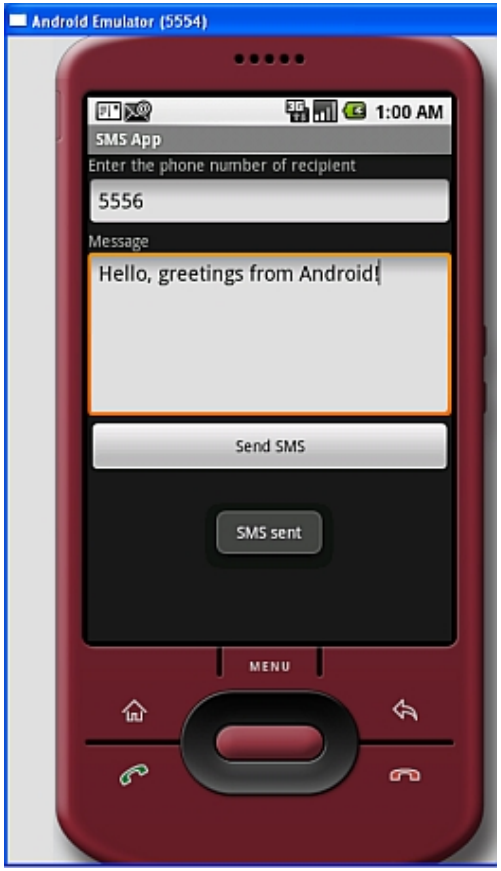
Input from the user (i.e., the phone no, text message and sendSMS is implemented).

SMS Sending

- Step 5
  - To send an SMS message, you use the SmsManager class. And to instantiate this class call getDefault() static method.
  - The sendTextMessage() method sends the SMS message with a PendingIntent.
  - The PendingIntent object is used to identify a target to invoke at a later time.

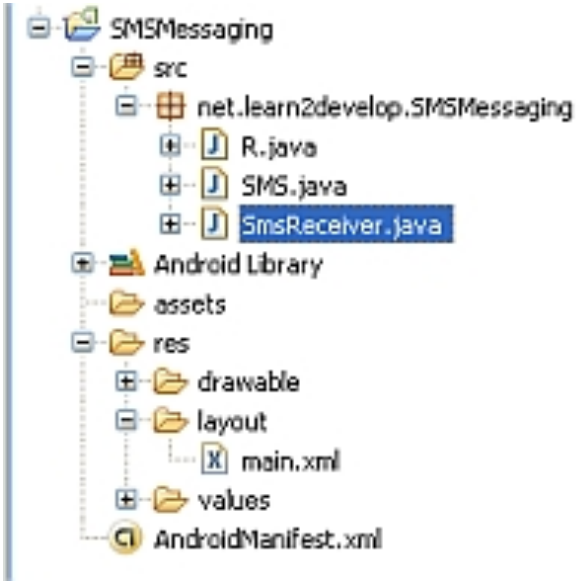
```
private void sendSMS(String phoneNumber, String message) {
  PendingIntent pi = PendingIntent.getActivity(this, 0,
    new Intent(this, SMS.class), 0);
  SmsManager sms = SmsManager.getDefault();
  sms.sendTextMessage(phoneNumber, null, message, pi, null);
}
```

SMS Sending


Android Emulator (5554)
1:00 AM
SMS App
Enter the phone number of recipient
5556
Message
Hello, greetings from Android!
Send SMS
SMS sent
MENU

Receiving SMS

- Step 1



Receiving SMS

- Step 2
  - In the AndroidManifest.xml file add the <receiver> element so that incoming SMS messages can be intercepted by the SmsReceiver class.

```
<receiver android:name=".SmsReceiver">
        <intent-filter>
          <action android:name=
            "android.provider.Telephony.SMS_RECEIVED" />
        </intent-filter>
  </receiver>
```

Receiving SMS

- Step 3

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsMessage;
import android.widget.Toast;
```

Receiving SMS

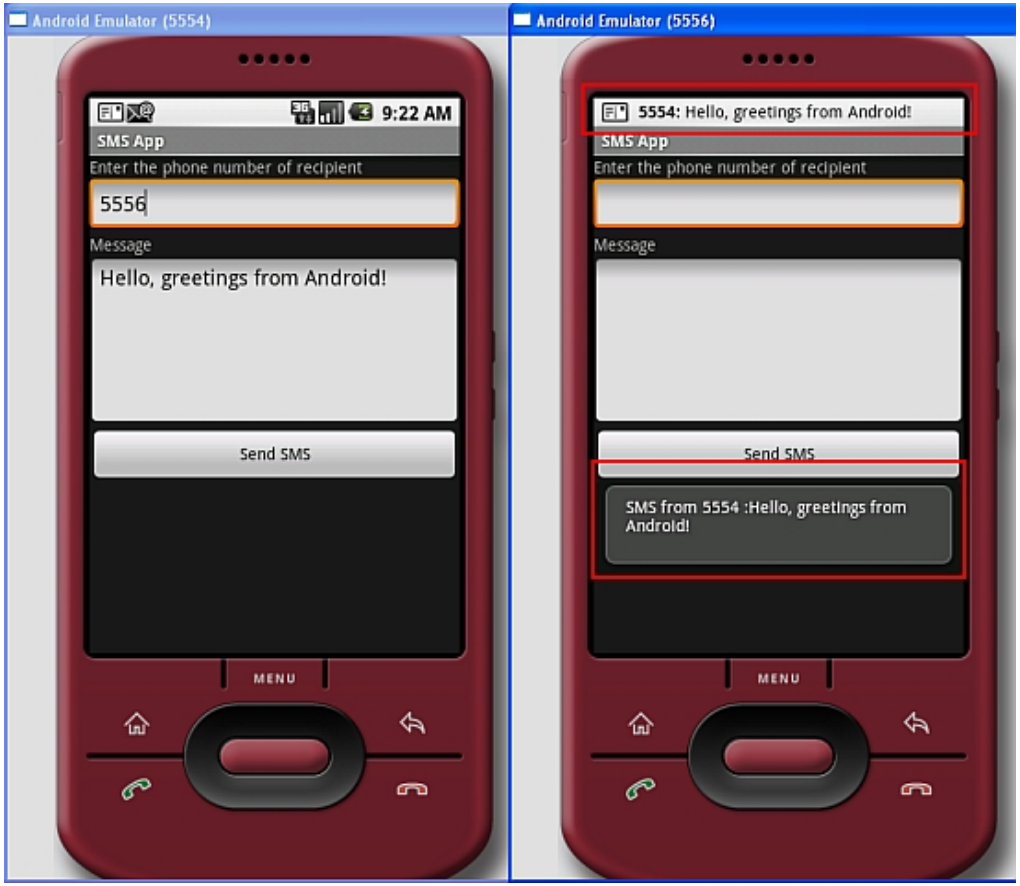
- Step 4

```
public class SmsReceiver extends BroadcastReceiver  {
  @Override
  public void onReceive(Context context, Intent intent) {
    //---get the SMS message passed in---
    Bundle bundle = intent.getExtras();
    SmsMessage[] msgs = null;
    String str = "";

    if (bundle != null){
      //---retrieve the SMS message received---
      Object[] pdus = (Object[]) bundle.get("pdus");
      msgs = new SmsMessage[pdus.length];

      for (int i=0; i<msgs.length; i++) {
        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
        str += "SMS from " + msgs[i].getOriginatingAddress();
        str += " :";
        str += msgs[i].getMessageBody().toString();
        str += "\n";
      }
      //---display the new SMS message---
      Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
    }
  }
}
```

In the SmsReceiver class, extend the BroadcastReceiver class and override the onReceive() method. The message is attached to the Intent

The messages are stored in a object array PDU format. To extract each message, you use the static createFromPdu() method from the SmsMessage class. The SMS message is then displayed using the Toast class

Receiving SMS



GPS : Add LocationManager

```
public class GPSSimulator extends Activity
{
    private LocationManager lm;
    private LocationListener locationListener;

    // Called when the activity is first created.
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // use the LocationManager class to obtain GPS locations
        lm = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);

        locationListener = new MyLocationListener();

        lm.requestLocationUpdates(
            LocationManager.GPS_PROVIDER, 0, 0, locationListener);
    }
}
```

GPS : Add LocationManager

```
private class MyLocationListener implements LocationListener {

  @Override
  public void onLocationChanged(Location loc) {
    if (loc != null) {
      Toast.makeText(getBaseContext(),
        "Location changed : Lat: " + loc.getLatitude() +
        " Lng: " + loc.getLongitude(),
        Toast.LENGTH_SHORT).show();
     }
  }

  @Override
  public void onProviderDisabled(String provider) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onProviderEnabled(String provider) {
    // TODO Auto-generated method stub
  }

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {
    // TODO Auto-generated method stub
  }
}
```

Test the GPSSimulator

- To test in Eclipse:
  1. Switch to DDMS view.
  2. Find the Location Controls in the Emulator Control tab.
  3. Click the GPX tab and click Load GPX.
  4. Locate and select the GPX file.
  5. Click Play to begin sending coordinates to the Emulator.

Add ability to use Google Maps

- Update the Manifest with two lines.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.android.GPSSimulator">

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

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

  <application android:icon="@drawable/icon"
    android:label="@string/app_name">

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

    <activity android:name=".GPS" 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>
```

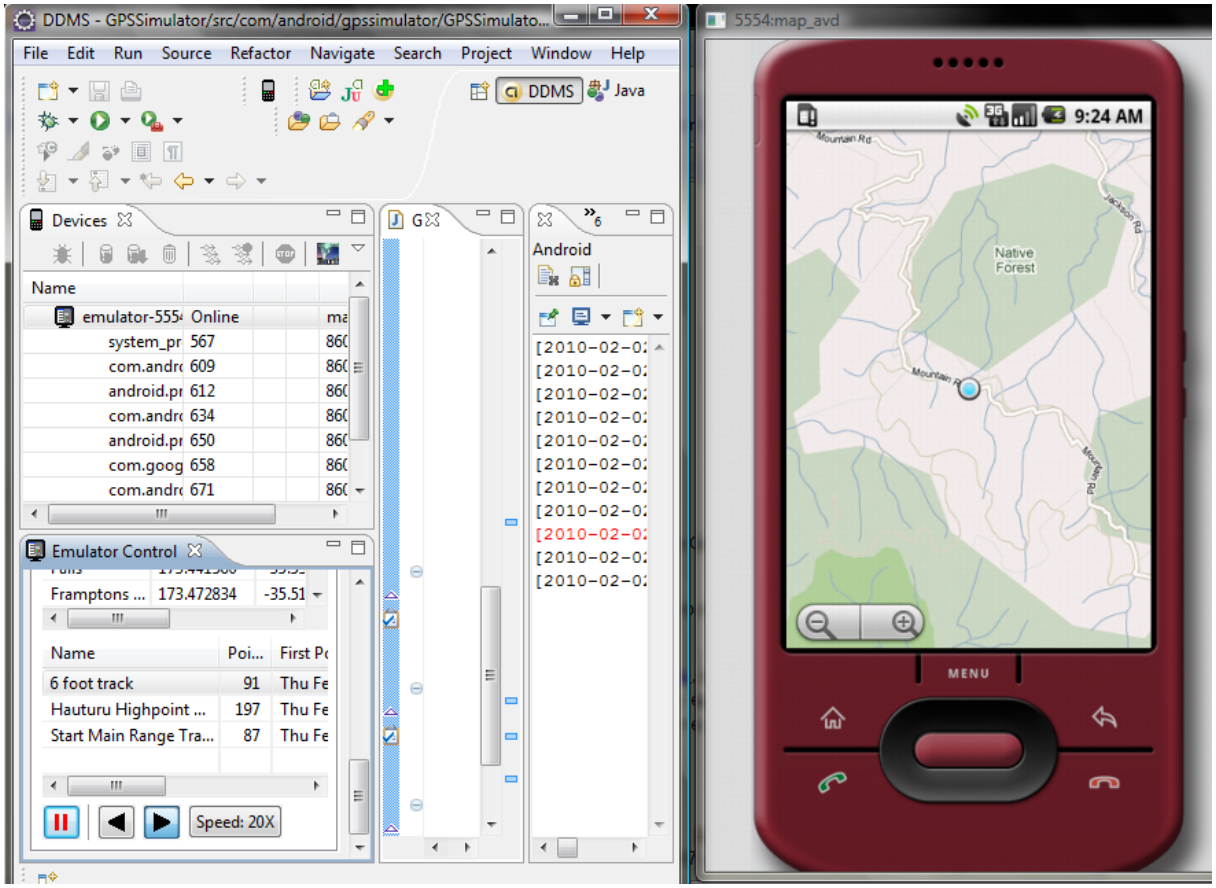
Add MapView to main.xml

```
<?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"
  >

  <com.google.android.maps.MapView
    android:id="@+id/mapview1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:enabled="true"
    android:clickable="true"
    android:apiKey="Your API Key Here" />

</LinearLayout>
```

GPSSimulator to use Google Maps

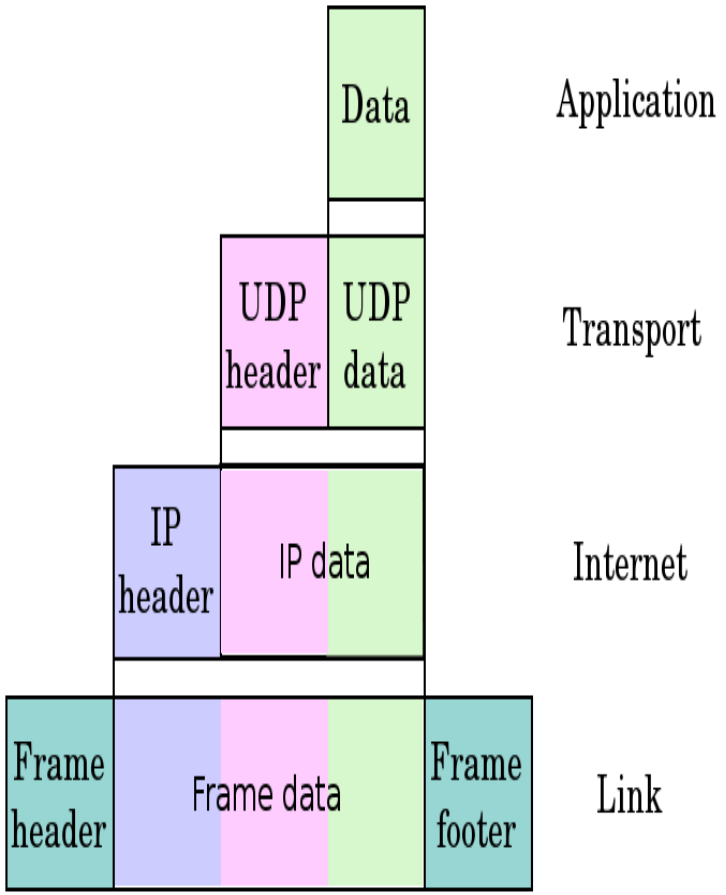
```
public class GPSSimulator extends MapActivity {
  private LocationManager lm;
  private LocationListener locationListener;
  private MapView mapView;
  private MapController mc;

  // Called when the activity is first created.
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // use the LocationManager class to obtain GPS locations
    lm = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);

    locationListener = new MyLocationListener();

    lm.requestLocationUpdates(
    LocationManager.GPS_PROVIDER, 0, 0, locationListener);

    mapView = (MapView) findViewById(R.id.mapview1);
    mc = mapView.getController();
  }

  @Override
  protected boolean isRouteDisplayed() {
    return false;
  }

  private class MyLocationListener implements LocationListener {

    @Override
    public void onLocationChanged(Location loc) {
      if (loc != null) {
        Toast.makeText(getBaseContext(),
          "Location changed : Lat: " + loc.getLatitude() +
          " Lng: " + loc.getLongitude(),
          Toast.LENGTH_SHORT).show();

        GeoPoint p = new GeoPoint(
        (int) (loc.getLatitude() * 1E6),
        (int) (loc.getLongitude() * 1E6));
        mc.animateTo(p);
        mc.setZoom(16);
        mapView.invalidate();
      }
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }
  }
}
```

View the Location on the Map



Internet Layers

- The Internet, is based on a layered architecture called the TCP/IP stack.
  - Link Layer
    - Protocols: ARP and RARP
  - Internet Layer
    - Protocols: IP, ping, etc.
  - Transport
    - Protocols: TCP and UDP
  - Application Layer
    - Protocols: HTTP, FTP, DNS, etc.



Client-Server Communication

- A server machine is identified on the Internet by some IP address
- Daemons are the processes running in the background which are listening all the time for connection requests from clients on a particular port number.
- Once a connection request comes into the server on a given port, the corresponding daemon can choose to accept it, and if so, a connection is established.
- Then the application layer protocol is typically used for the client to get or send data to the server.