Android: Introduction, présentation

jean-michel Douin, douin au cnam point fr
version : 10 février 2016

Notes de cours

Sommaire

- **Intergiciel (middleware)**
  - **Les éléments de base**

- **Android OS comme *intergiciel***
  - **Applications et évènements gérés par le middleware**
  - **Approche déclarative**
  - **Composants, classes de base**
    - **Activity,**
    - **BroadcastReceiver,**
    - **Service,**
    - **ContentProvider,**
    - **Intent.**
  - **Patrons de conception utilisés**
    - *Publish/subscribe*
    - *Chaîne de responsabilités*
    - *Fabrique méthode*
    - *MVC, Modèle Vue Contrôleur*
    - *Façade*
    - *Procuration*
    - *Médiateur*
    - *Stratégie*
    - *…*

Bibliographie utilisée… *à compléter*

Mobile Middleware, Architecture, Patterns and Practice, Sasu Tarkoma,ed. Wiley 2009

[Sheng-De Wang]
Le point de départ de ce support
Auteur: Prof. Sheng-De Wang
http://*intw2-2010.cs.pu.edu.tw/a1.**ppt***
http://www.dre.vanderbilt.edu/~schmidt/cs282/PDFs/android-binder-ipc.pdf

[Sch06]Le site de douglas Schmidt
http://www.cs.wustl.edu/~schmidt/patterns.html
ce site http://www.cs.wustl.edu/~schmidt/POSA/

[Kra06]
La présentation de S. Krakowiak faite à l 'école d 'été ICAR 2006
http://sardes.inrialpes.fr/ecole/2006/ http://www2.lifl.fr/icar/Chapters/Intro/intro.html

[VKZ05]
le site de Uwe Zdun : http://www.infosys.tuwien.ac.at/Staff/zdun/
ce livre : http://www.infosys.tuwien.ac.at/Staff/zdun/remotingPatterns/index.html
ce support http://www.infosys.tuwien.ac.at/Staff/zdun/teaching/evs/evs.pdf

informations générales http://www.edlin.org/cs/patterns.html

http://www.powershow.com/view/133aa5-MjUxY/Mobile_Middleware_Course_Principles_and_Patterns_Sasu_Tarkoma_powerpoint_ppt_presentation

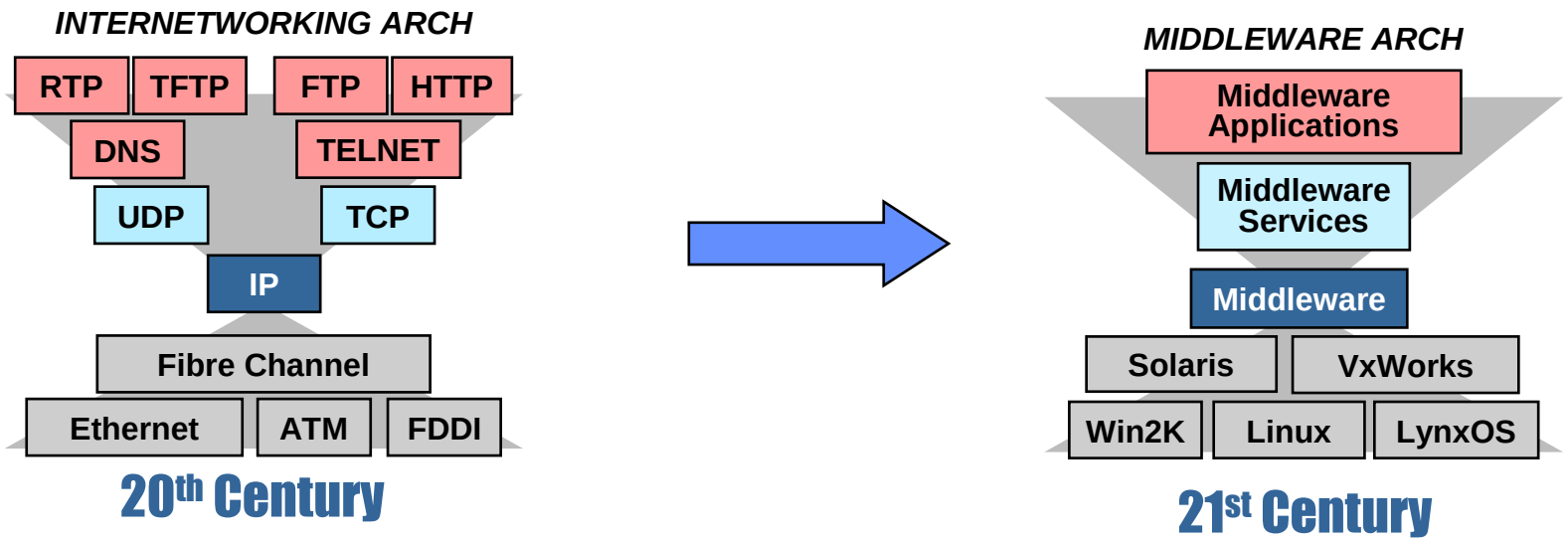
http://developer.android.com/resources/index.html

Première partie : Introduction

- **Intergiciel, fonctionnalités attendues**

- **Android**
  - **Un premier aperçu en quelques diapositives**
    - **Le vocable**
    - **Architecture logicielle**
    - **Le simulateur, les API**
    - **Une démonstration**

Tendances, historique



- **[Sch06]**

Fonctions attendues de l 'intergiciel

- **de [Kra06]**
- **Proposer une API de haut Niveau**
  - **Masquer l 'hétérogénéité**
  - **Rendre la répartition invisible**
  - **Faciliter la programmation répartie**

→ Fonctions orientées: Réseau « fixe », internet, web, SOA
(Service Oriented Architecture)

Intergiciel et mobile

- **Sécurité**
  - Confidentialité, intégrité, contrôle des informations par l'utilisateur
- **Mobile comme « Baie d'accueil » des applications**
  - Installation, au sein d'un « sandbox »
    - Liste des applications, annuaire,
  - Éligibilité, sélection de la « bonne » application,
  - Découverte au « run time » d'un service
- **Déploiement, mise à jour, maintenance**
  - Substitution d'une application par une autre
- **Communication**
  - Réseaux
    - communications synchrones, asynchrones, interruptions de services
    - Notifications inhérentes: sms, push …
- **Persistance**
  - Embarquée
- **Consommation**
  - Batterie

Android

- **Sécurité**
  - Linux, processus …
- **Mobile comme « Baie d'accueil » des applications**
  - Un processus une application une DVM (Dalvik Virtual Machine)
    - (Vue logique), « sandbox » inhérent
- **Déploiement, mise à jour, maintenance**
  - Prise en charge par l'intergiciel
- **Communication**
  - Interne, externe -> intergiciel, sms, push …
- **Persistance**
  - -> intergiciel, SQLLite

Android les grandes lignes

- **Composants Android**
- **Outils de Développement**
- **Architecture Logicielle**

- **Développement**
  - **en java avec quelques directives et configurations en syntaxe XML**

- **Un exemple, une démonstration**
  - **Un exemple en quelques lignes de java et d'XML**

Composants Android

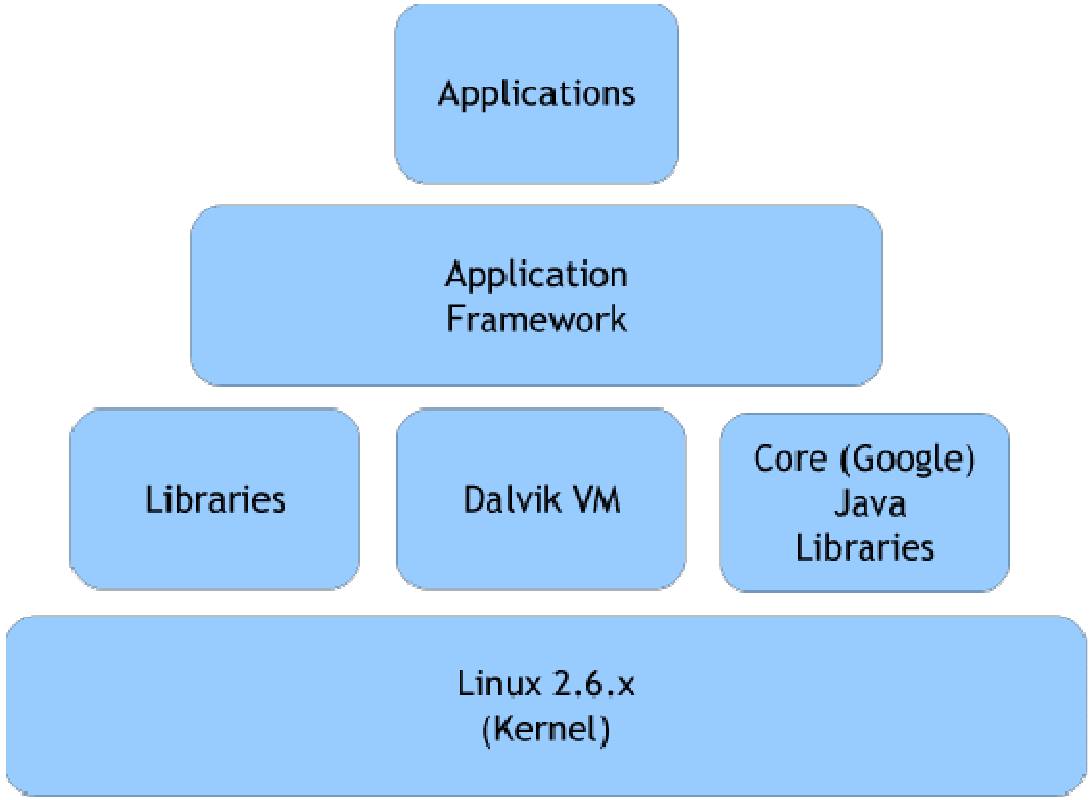
- **Linux**
  - **Processus / DVM / Application**
- **Framework de déploiement d'applications**
  - **De nombreuses librairies,**
    - **Navigateur intégré, WebKit** (webkit utilisé par safari, Google Chrome…)
    - **SQLite**
- **En natif, pilotes**
  - **Dépendant du matériel**
    - **GSM**
    - **Bluetooth, EDGE, 3G, WiFi**
    - **Caméra, GPS, boussole et accéléromètre**
    - **Température,**
    - **…**

Outils de développement

- **SDK Android**
  - –En ligne de commandes
  - –Plug-in sous eclipse

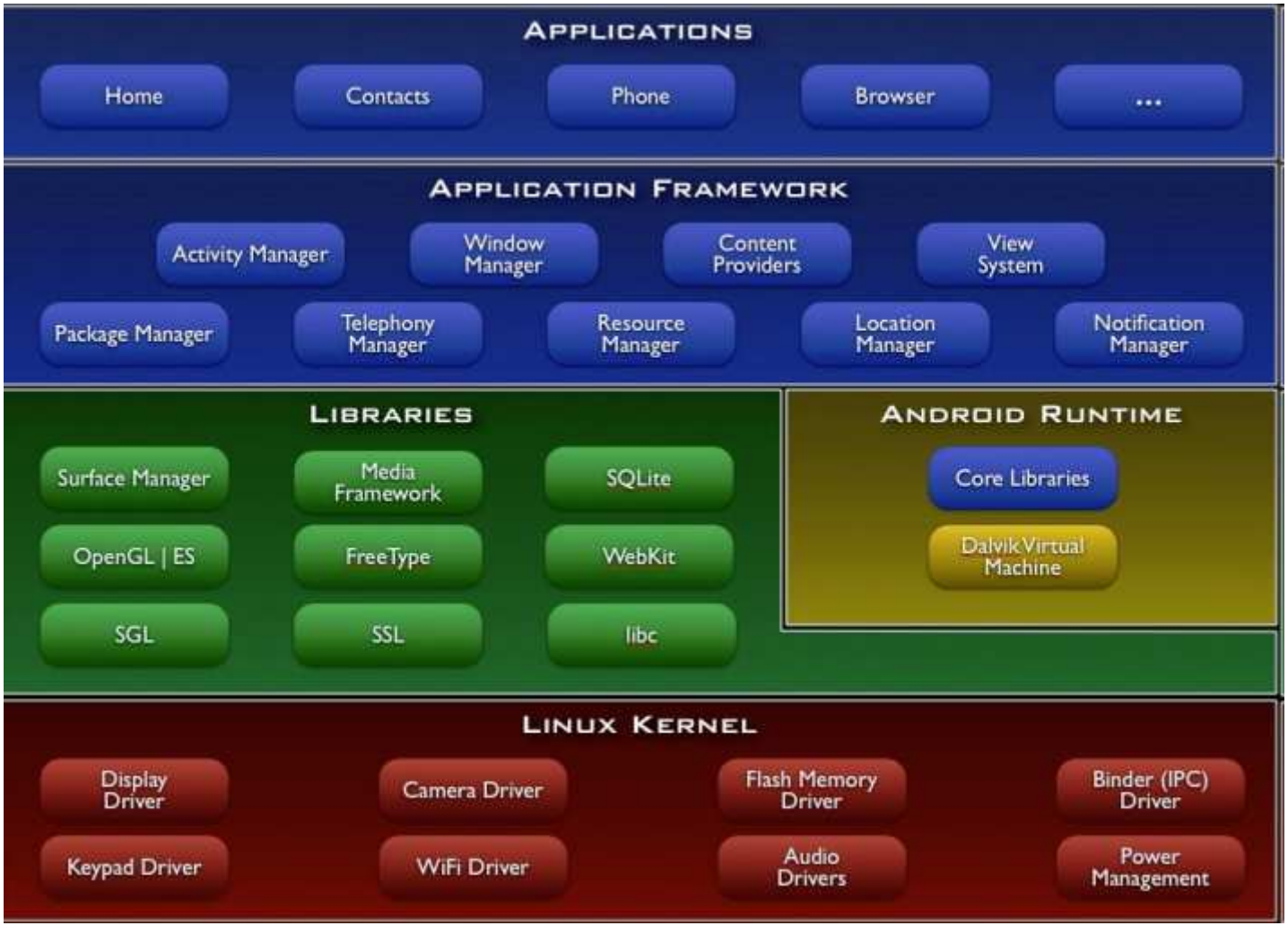
  - –Émulateur
  - –Débogueur
  - –Traces fines d'exécution
  - –Tests unitaires
  - –Outils de mise au point
    - •Mesure de mémoire et performance

Framework ou baie d'accueil des applications



- **Source : http://www.linuxjournal.com/**

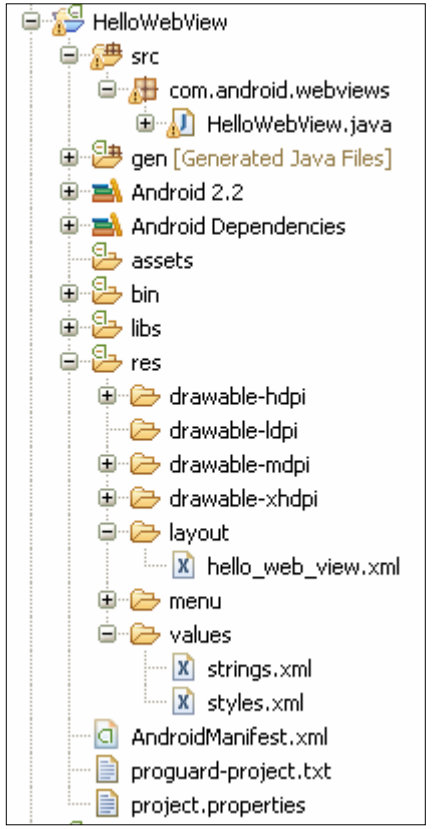
Architecture et API



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

Développement

- **Développement**
  - **En java, configurations en syntaxe XML**
    - **Une configuration de l'application**
      - **AndroidManifest.xml**
    - **Une déclaration de ressources en XML**
      - **comme l'IHM ou des String**
        - » res/layout/hello_web_view.xml
        - » res/values/strings.xml



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>
```

Premier exemple support Sheng Wang

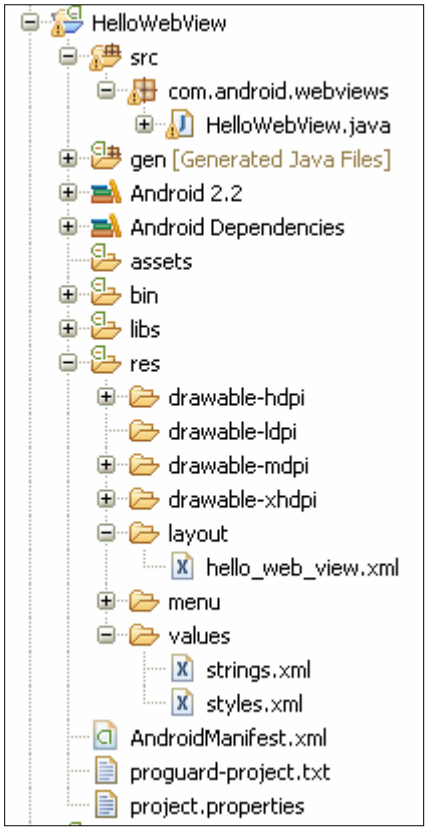
```
package com.android.webviews;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.webkit.WebView;
import android.webkit.WebViewClient;

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

      webview = (WebView) findViewById(R.id.webview);
      webview.getSettings().setJavaScriptEnabled(true);
      webview.loadUrl("http://www.cnam.fr");

    }
```

Démonstration

- **Démonstration**

Puis

- **Les composants essentiels**



Vocabulaire

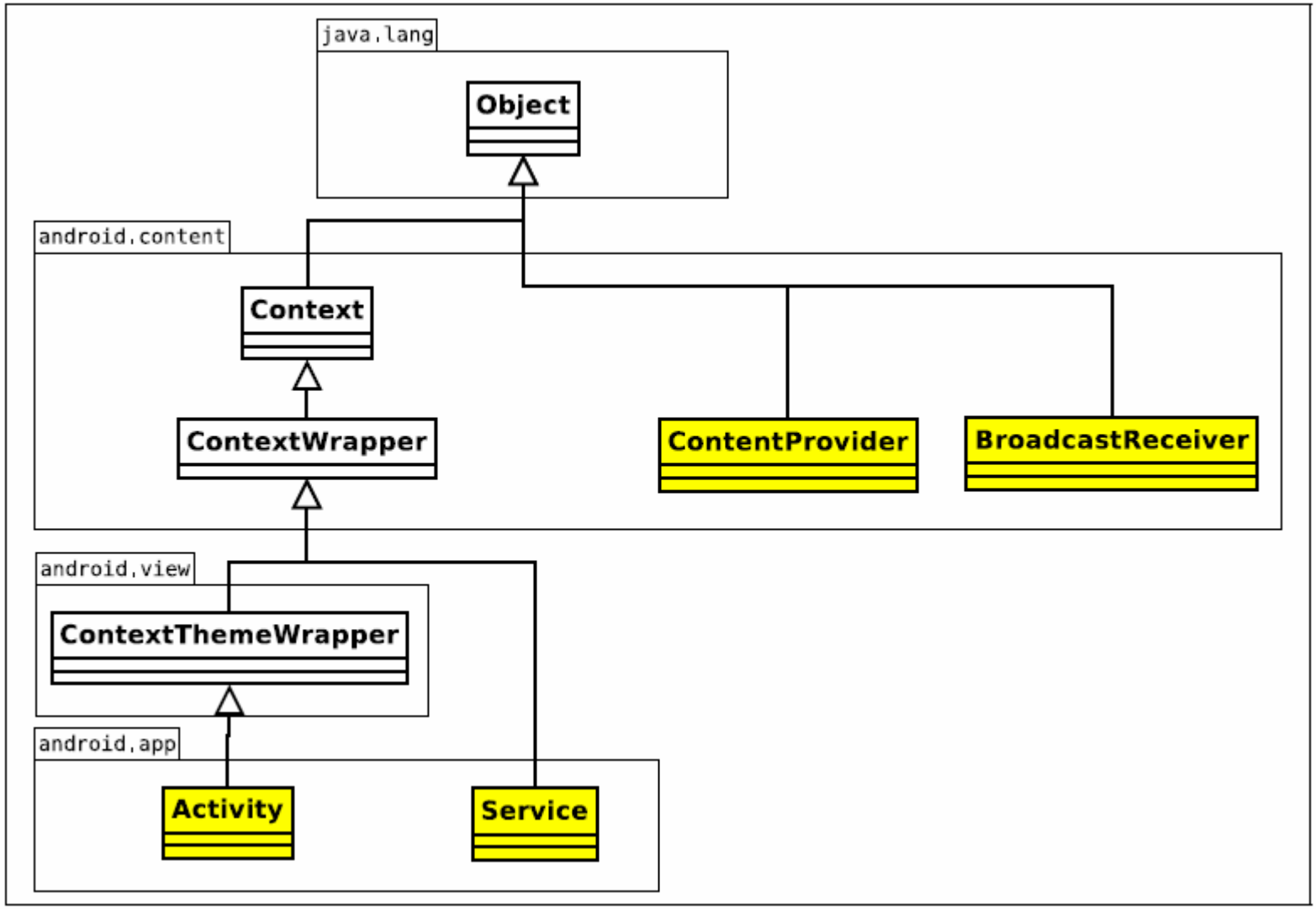
- **Les Essentiels**
  - **Activity**
    - **Une activité est associée à une IHM**
  - **BroadcastReceiver**
    - **Réception d'un message transmis par un émetteur via Android**
  - **Service**
    - **Une tache de fond sans IHM**
  - **ContentProvider**
    - **La persistance des données**
  - **Intent**
    - **Les notifications**
  - **IntentFilter**
    - **Les filtres des notifications candidates**

Introduction … Classes

- **Activity**
  - Associée à une interface utilisateur
    - Cycle de vie (démarrée, en pause, arrêtée, détruite…)
      - Peut démarrer d'autres activités, peut émettre des évènements(intentions, intent)
    - Une configuration de type XML, permissions, librairies,
- **BroadcastReceiver**
  - Réception des évènements systèmes ou d'une autre application
  - Émission et réception d'intentions (intent)
- **Service**
  - Pas d'interface, un service à rendre, en tâche de fond
  - Intention de servir, langage commun aux clients et au service
- **ContentProvider**
  - Données rendues persistantes (pour d'autres applications)
  - Un fichier, base SQLite

- **Intent**
  - Notifications,
- **IntentFilter**
  - Critères de sélection, éligibilité, mécanisme de résolution

Activity, Service, ContentProvider, BroadcastReceiver



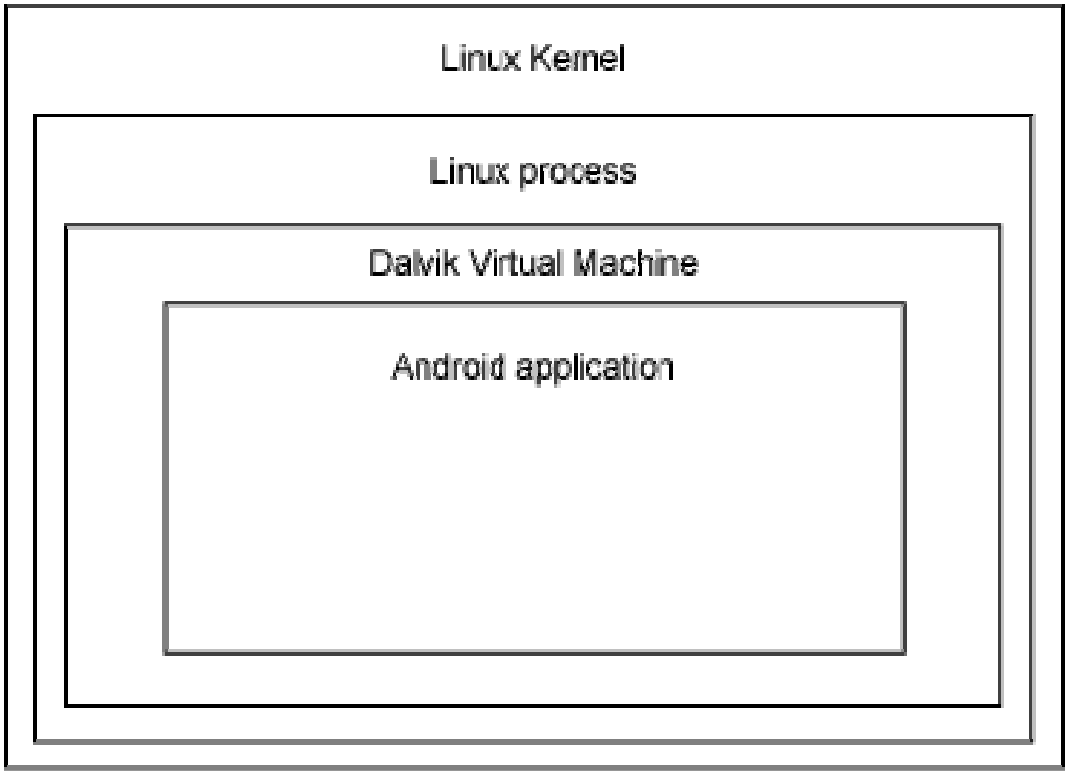
- Source : https://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf

Application

- **Une application peut être constituée de plusieurs écrans,**
- **A chaque écran lui correspond une activité,**
- **Une activité hérite et redéfinit certaines méthodes**
  - **onCreate, -> …, onPause, -> …, onStop, ->…onDestroy, -> …**
- **Android se charge des appels de ces méthodes**
  - **Un état de l'activité est donc induit par l'exécution de ces méthodes**
- **-> Inversion de contrôle**
  - (cf. M.Fowler) http://martinfowler.com/articles/injection.html
  - **Android impose un cycle de vie, un état de l'activité**

Architecture, vocabulaire

- **Source : http://developer.android.com/guide/topics/fundamentals.html**



- **Application**
  - **Activity**
  - **Service**
  - **ContentProvider**
  - **BroadcastReceiver**

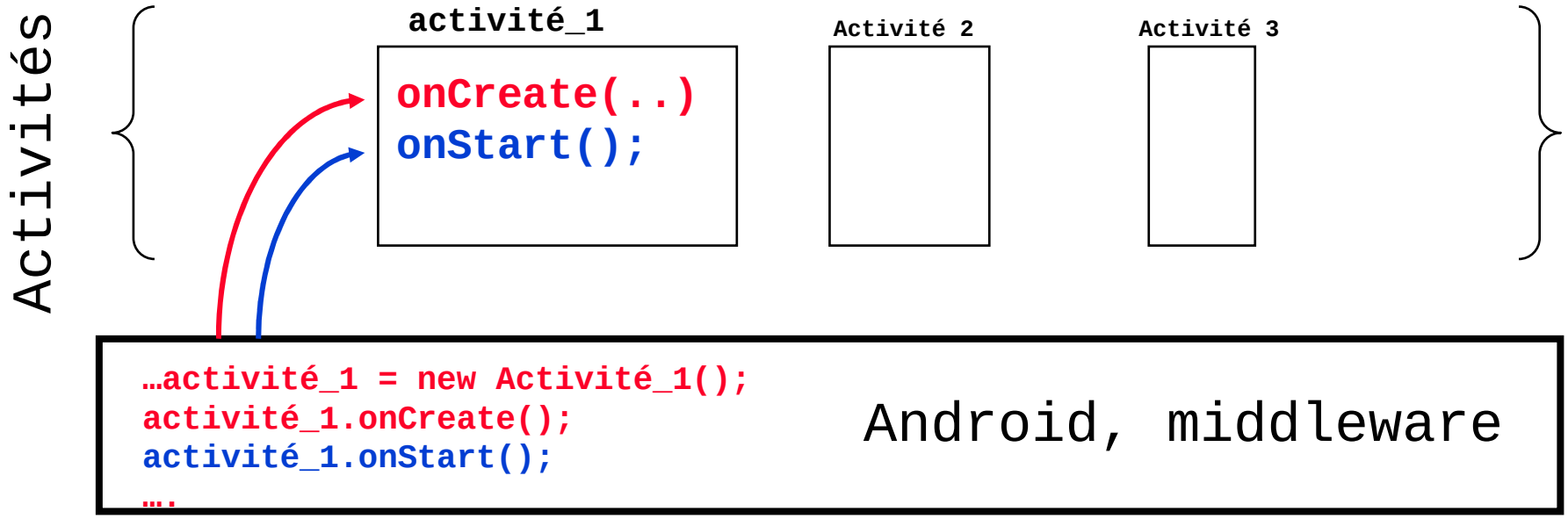
public class android.app.Activity

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

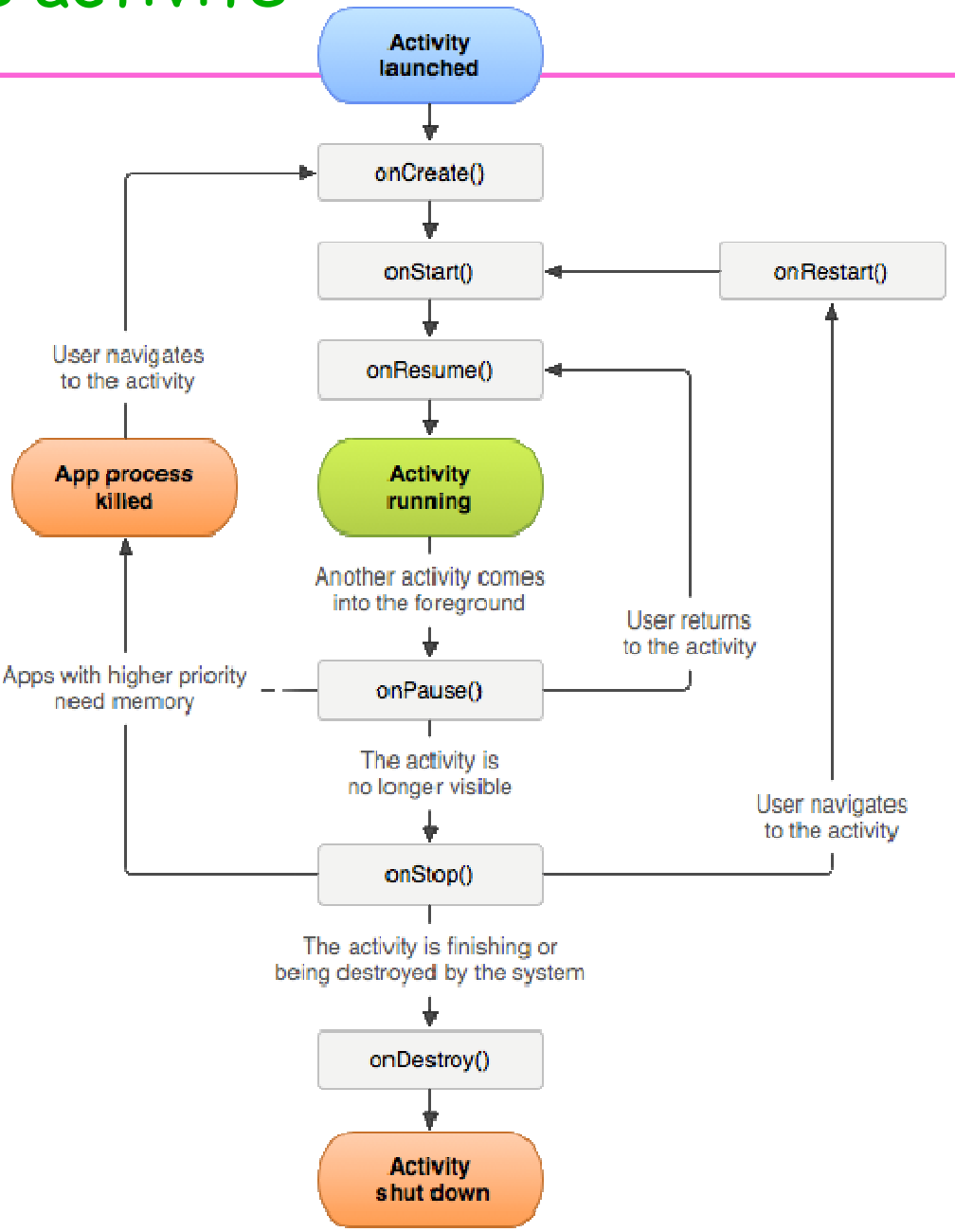



```
public class Activity extends ApplicationContext {
      protected void onCreate(Bundle savedInstanceState);
      protected void onStart();
       …..}
```

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

Le cycle de vie d'une activité

- Séquencement des appels imposé par le système



Activité son état

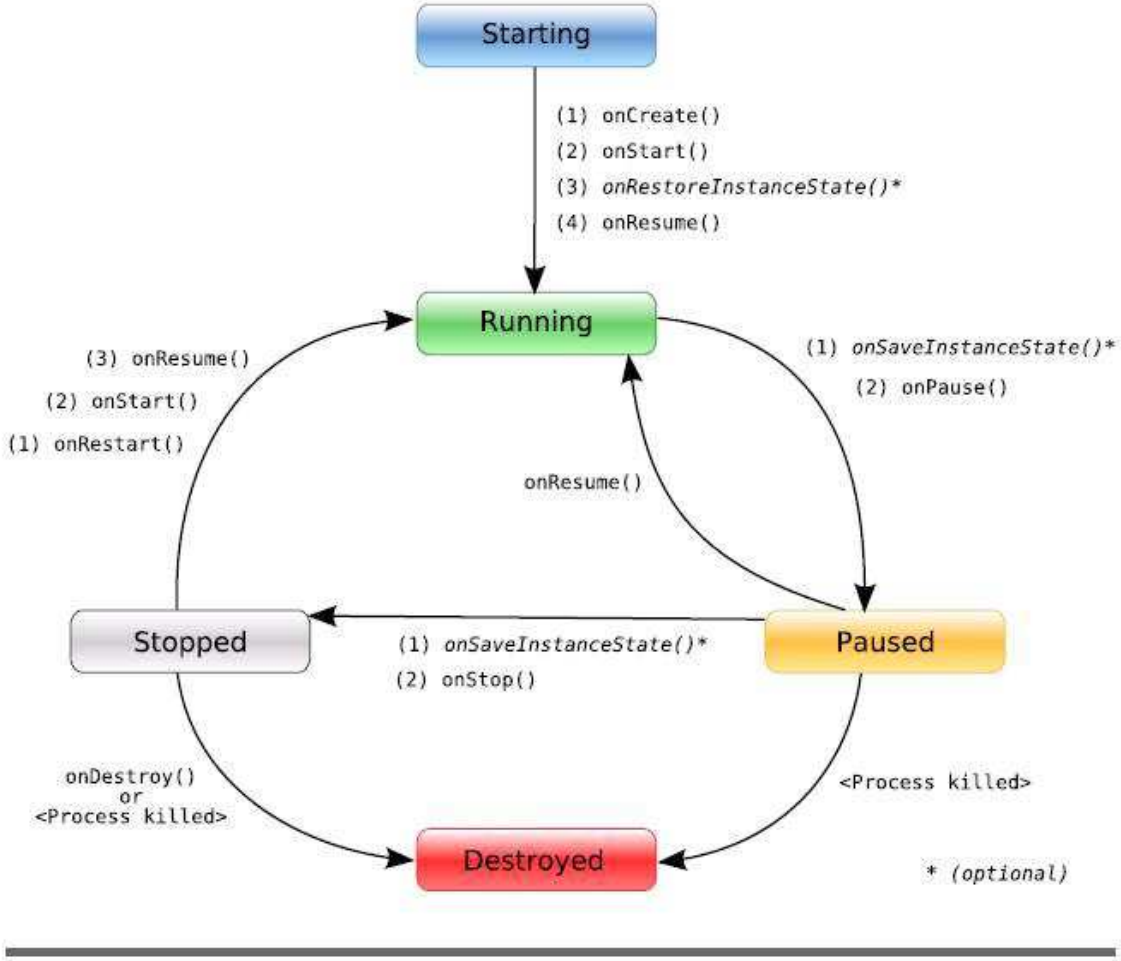
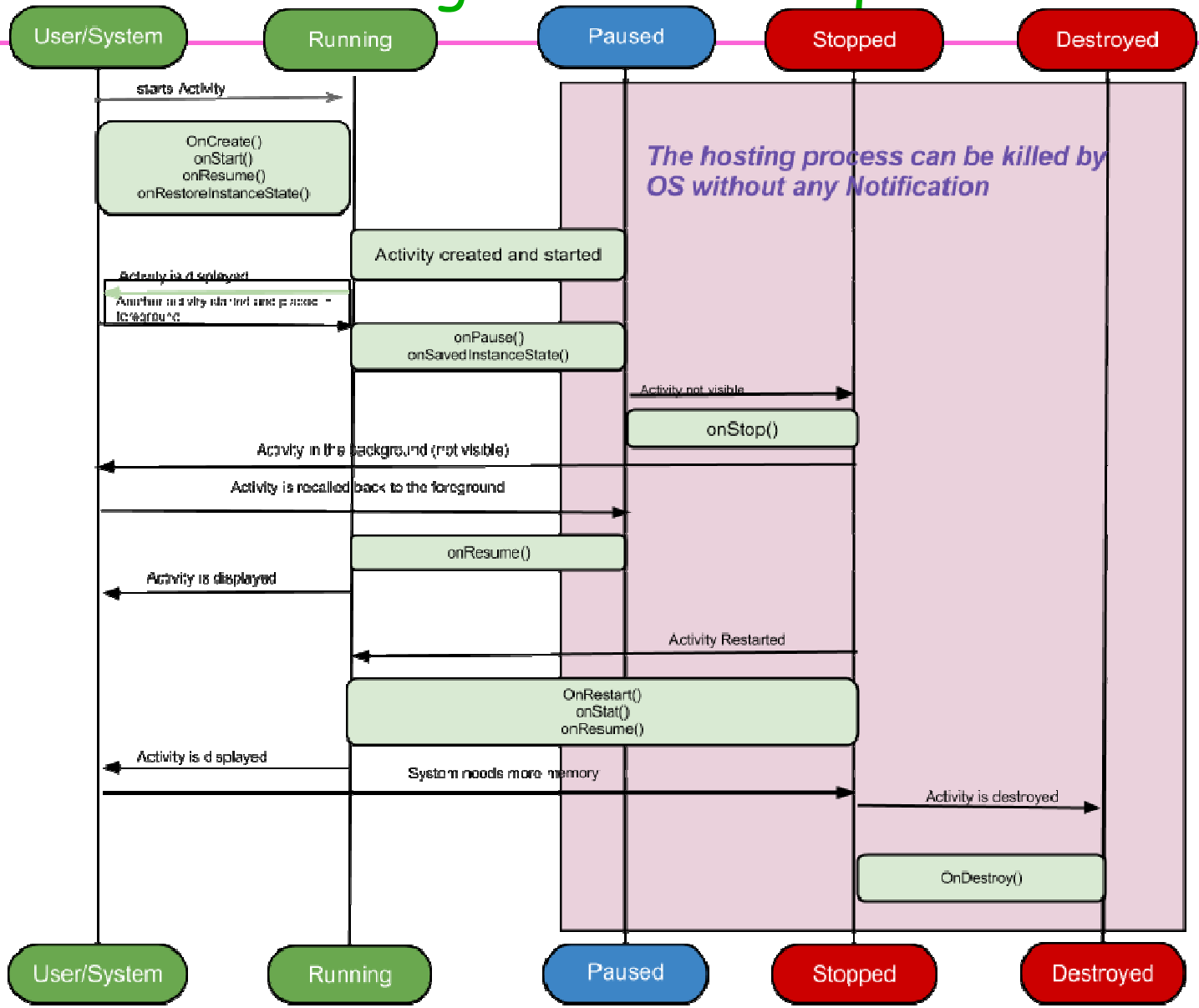


Figure 2.3: Life cycle of an Android activity

- http://inandroid.in/archives/tag/activity-lifecycle
- **Attention une application dans l'état Paused ou Stopped peut être supprimé par Android**

En diagramme de séquences



- https://thamilandroid.files.wordpress.com/2013/04/activitylifecycle1.png

Un exemple: une très petite IHM

Une « IHM »

- **Un bouton, un écouteur, un clic et l'heure est affichée !**

1. **En approche *traditionnelle***
   - **Tout est codé en Java IHM comprise**

2. **En approche *déclarative***
   - **Usage :**
     - **d'XML pour la configuration,**
     - **de java pour l'utilisation**

Activity Un Click et l'heure est actualisée

```
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 buttonNow;

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

    setContentView(buttonNow);         // <- le bouton devient la vue, (l'écran)
  }

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

Est-ce une vue apparentée swing
Et ici un MVC à lui tout seul … ? discussion

Activity-bis Un Click et l'heure est actualisée

- **Approche « déclarative »,**
  - **res/layout/activity_now.xml**
  - **Balise XML <Button** *attribut* **android:onClick**
  - **Le fichier res/layout/activity_now.xml**

```
<Button
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/buttonNowId"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Now"
    android:onClick="onClickButtonNow" />
```

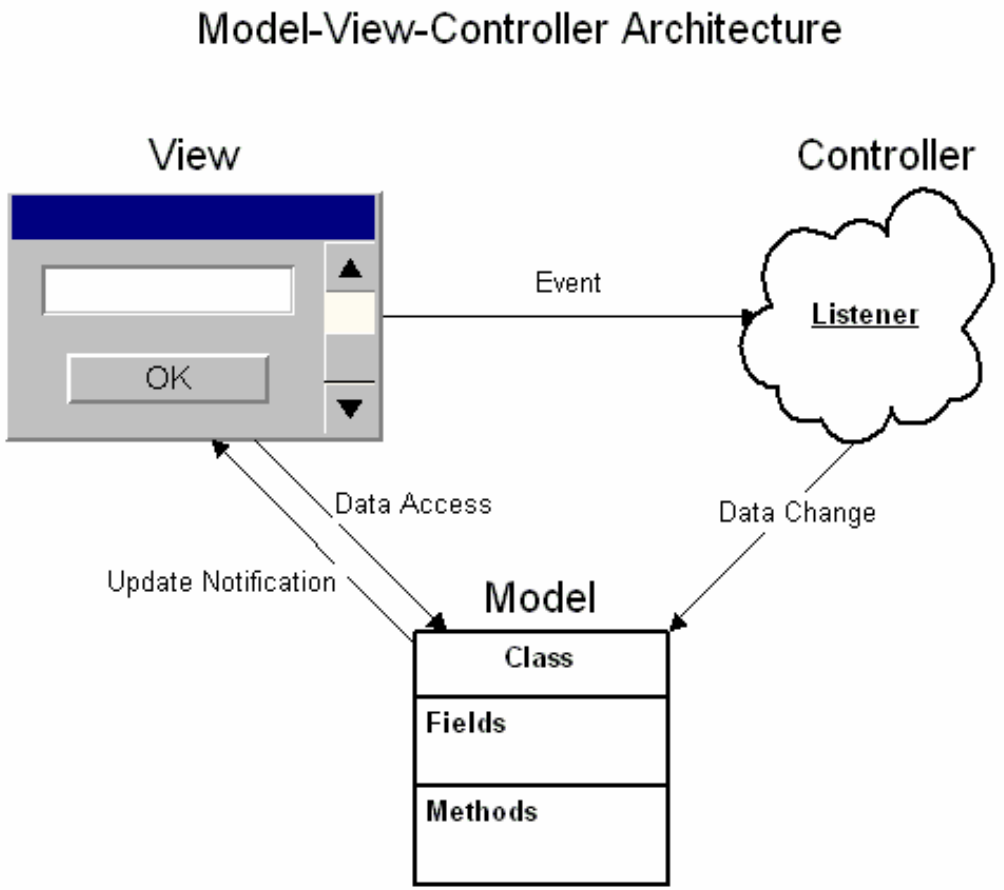
- ***onClickButtonNow: le nom de la méthode déclenchée à chaque clic***
  - ***public void onClickButtonNow(View v){ …..}***

L'activity devient (version déclarative)

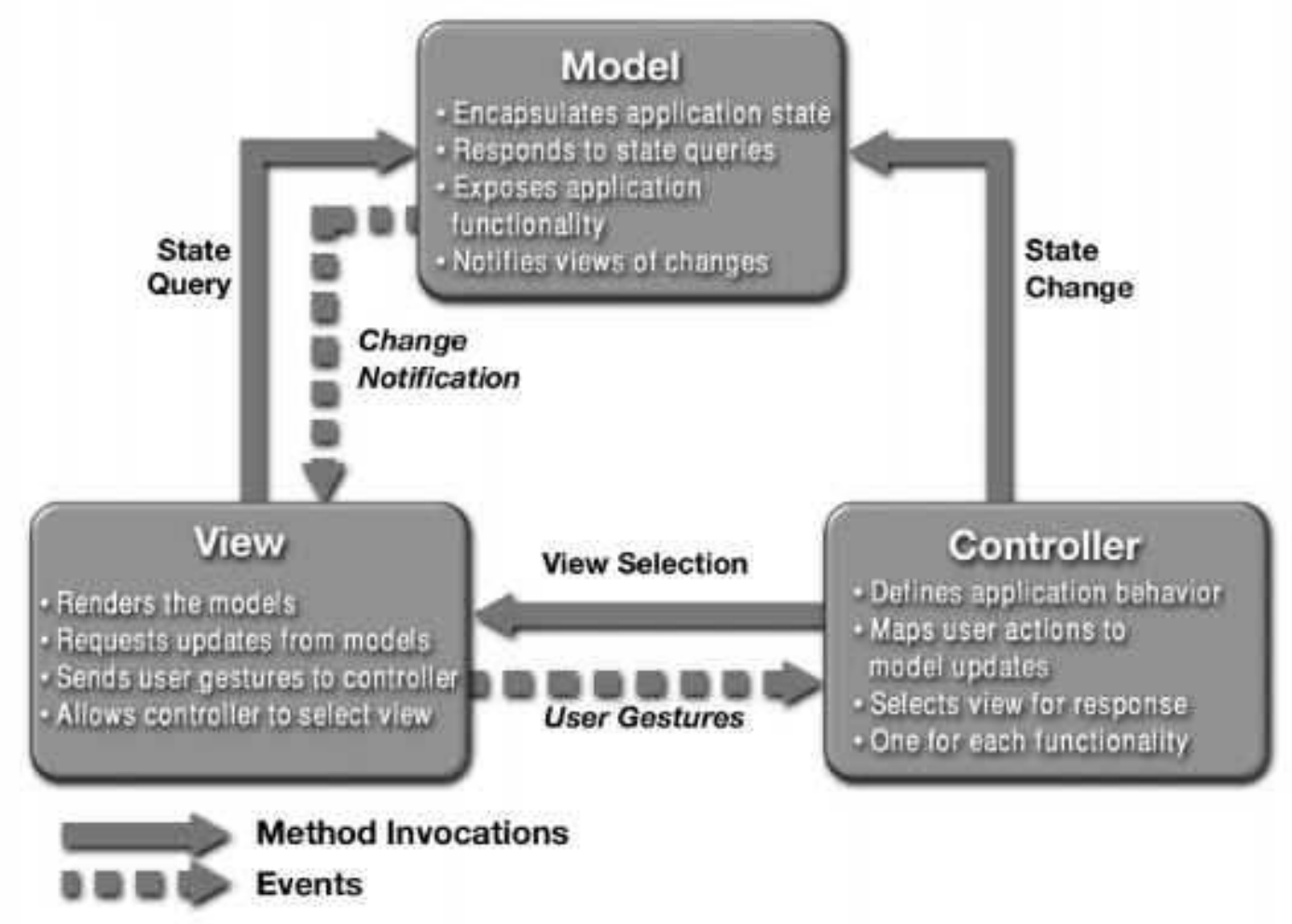
```
public class NowActivity extends Activity {
  private Button buttonNow;

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

    // Vue issue du fichier XML, le fichier R.java est généré automatiquement
    // R.java représente un accès en java aux ressources décrite en XML
    setContentView(R.layout.activity_now);

    // Button, identifié par un Id, est issu du fichier XML
    buttonNow = (Button)findViewById(R.id.buttonNowId);
  }

  // l'attribut XML onClick a ce nom de méthode comme valeur
  public void onClickButtonNow(View view) {
    buttonNow.setText(new Date().toString());
  }

}
```

Discussion MVC ?

- **Démonstration**
- **Discussion MVC ?**
  - **Modèle ?**
  - **Vue ?**
  - **Contrôleur ?**



MVC discussion suite



- http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/introduction/summary/index.html

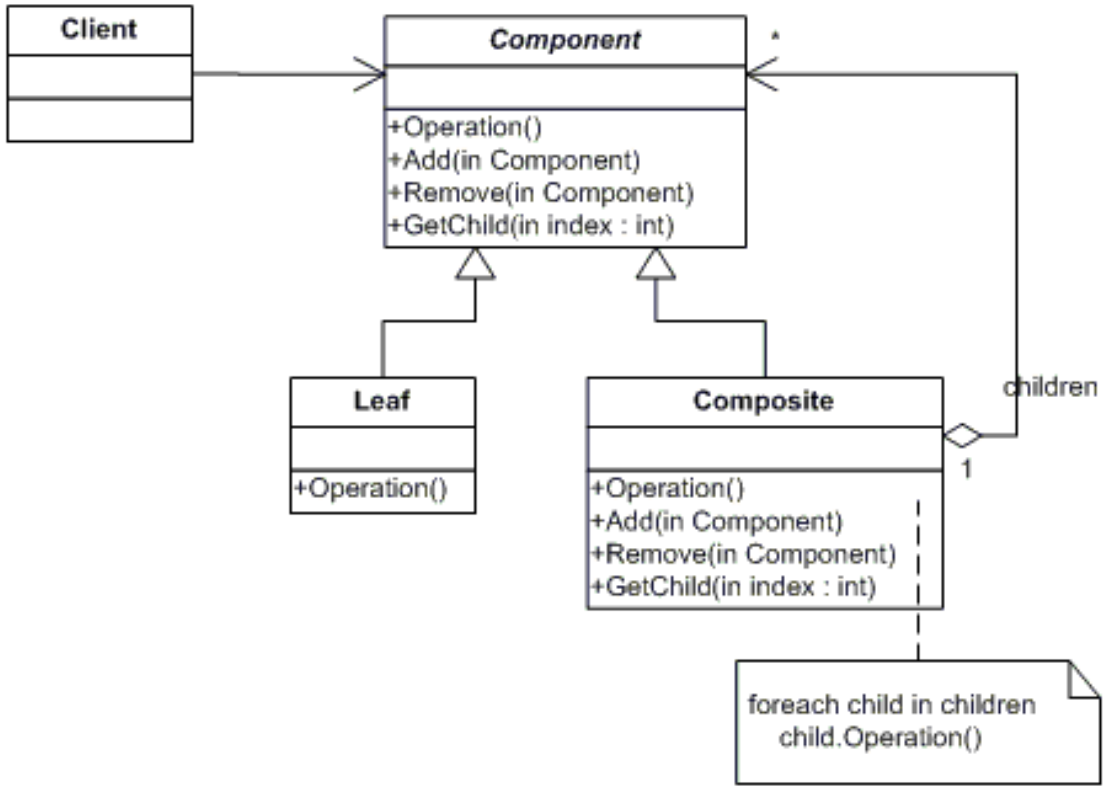
Retour sur les API

- **Android: un ensemble d'API**
  - **Activity Manager**
  - **View** System listes, boutons,… navigateur (WebView)

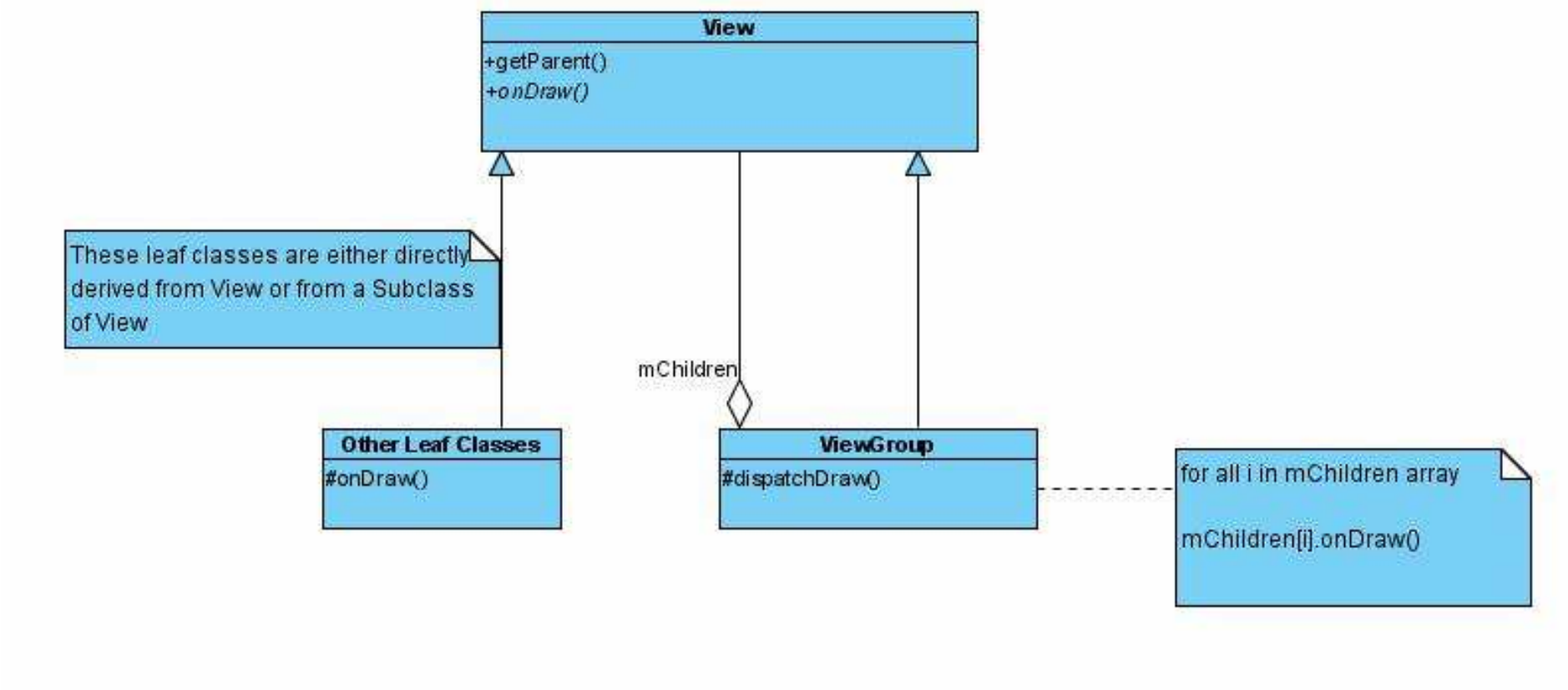
  - http://developer.android.com/guide/basics/what-is-android.html

View System

- **Un usage du patron composite**

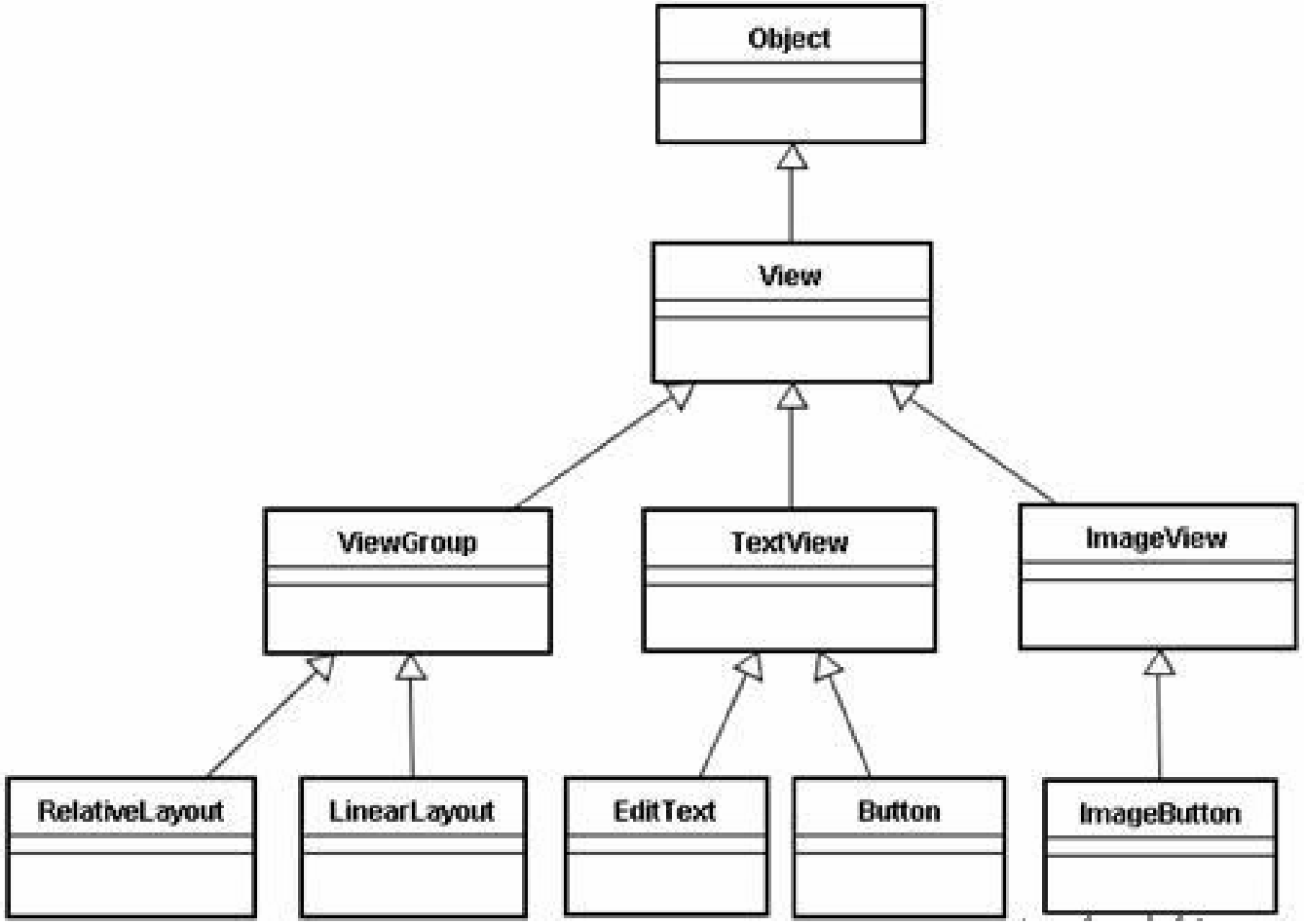


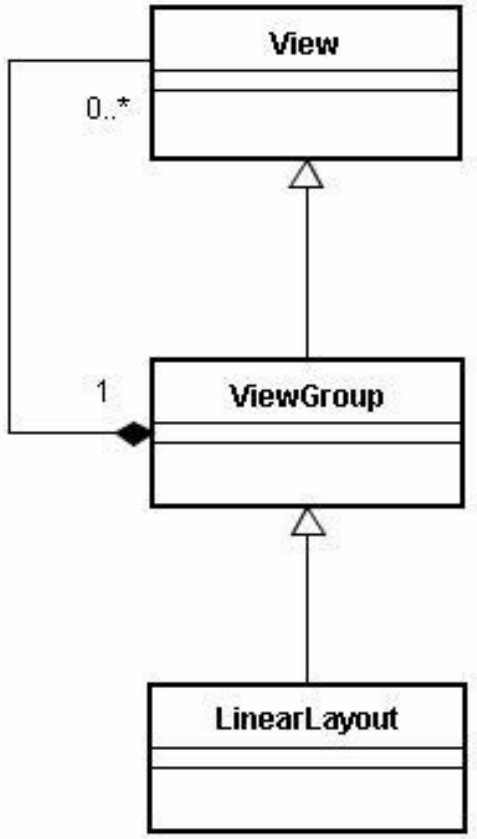
View (Component), ViewGroup (Composite)



Source: [Sheng-De Wang]
C'est bien un composite

Détaillé: Composite

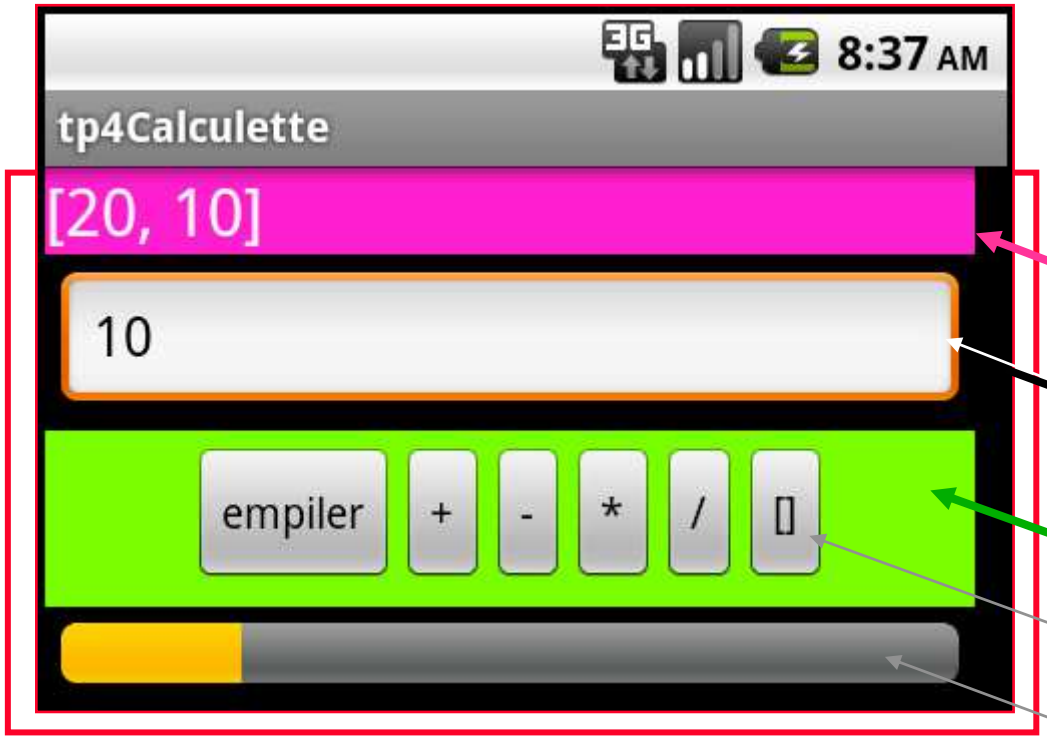




Source: [Sheng-De Wang]

Exemple précédent `setContentView(buttonNow);`

Une calculette: Layout, View , Button…



LinearLayout

TextView

EditText

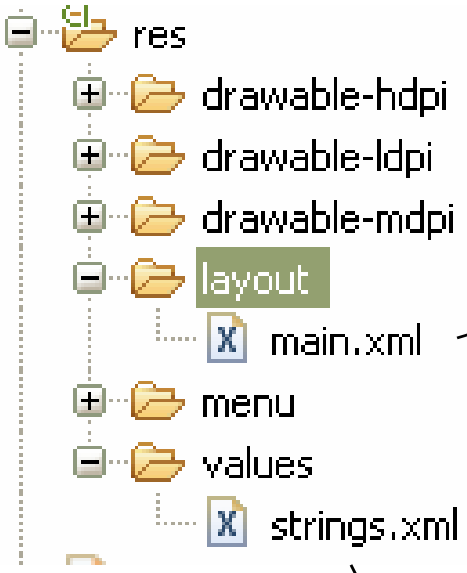
TableRow

Button

ProgressBar

- **Description de cette interface en XML**
  - **Fichier res/layout/main.xml**

Interface, IHM : Approche déclarative ./res/




```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:weightSum="1" android:layout_
    <TextView android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_width="match_parent" android:layout_height="wrap_conte
        android:clickable="false" android:id="@+id/etatPile" android:text="@s
        android:background="#ff23cf" android:textColor="#F0F0F0"></TextView>

    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content" android:enabled="true"
        android:visibility="visible" android:clickable="false" android:id="@+
        android:text="10" android:layout_margin="5dip">
    </EditText>
    <TableRow android:id="@+id/tableRow1" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:background="#7cfc00" andr
        <Button android:layout_height="wrap_content" android:id="@+id/push"
            android:layout_width="wrap_content" android:text="@string/push"
            android:onClick="onClickPush"></Button>
        <Button android:layout_height="wrap_content"
```

```
<?xml version="1.0" encoding="utf-8" stan
<resources>
    <string name="hello">Tp4CalculetteAct
    <string name="app_name">tp4Calculette

    <string name="vide">[]</string>
    <string name="push">empiler</string>
    <string name="add"> + </string>
    <string name="div"> / </string>
    <string name="sub"> - </string>
```

- **Chaque composant possède un id (android:id= "@+id/push")**

Architecture: la calculette, un classique

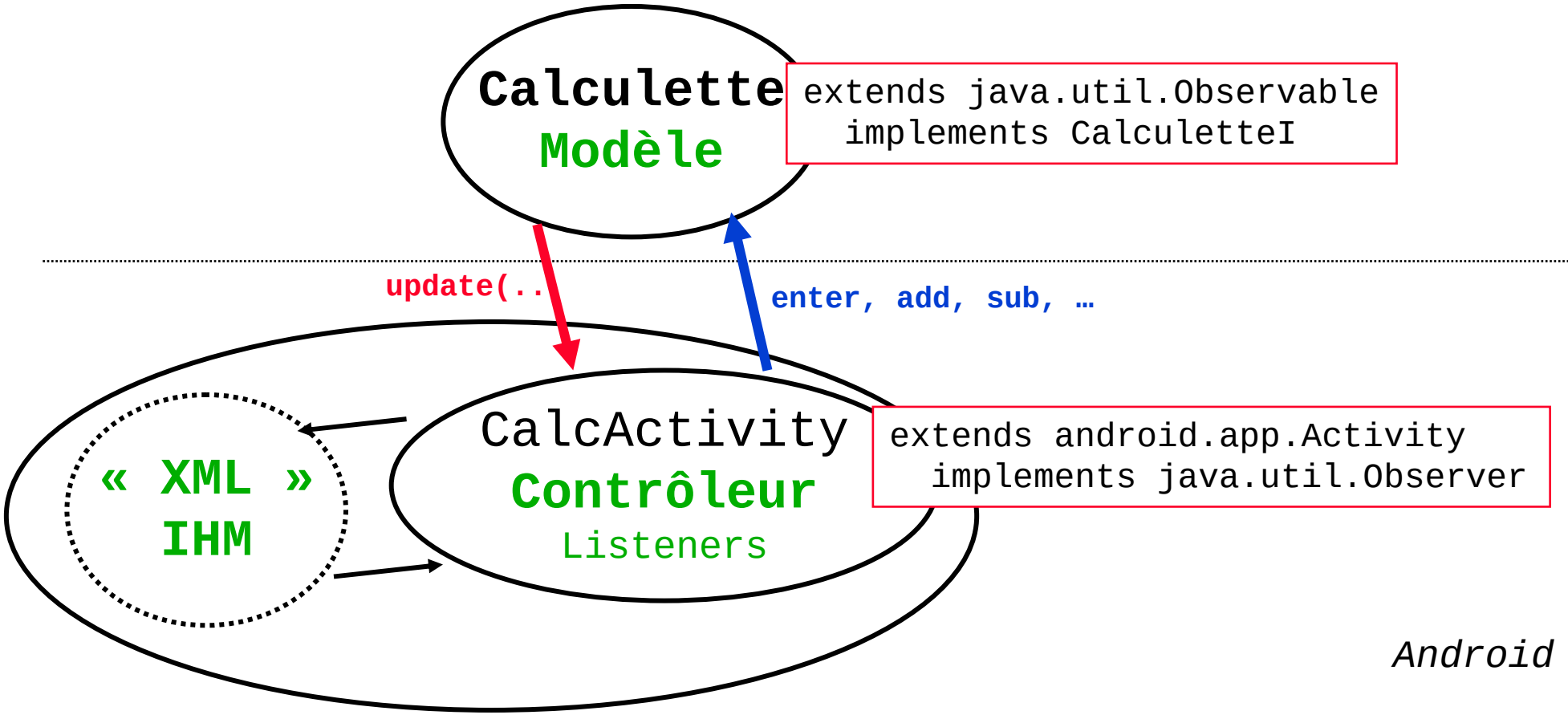
- **Le Modèle**
  - La calculette munie de ses opérations (+,-,/,*,…)
    - Les sources du modèle sont ici
      - http://douin.free.fr/tp4Calculette/
      - La calculette hérite de la classe java.util.Observable

- **La Vue**
  - L'IHM affichage, zone de saisie, boutons …

- **Le Contrôleur**
  - Le comportement de l'IHM

MVC nouvelle discussion

Classes déjà écrites



- **L'activity Android est une vue du Modèle Calculette** `(implements Observer)`
- **L'activity Android est le contrôleur de l'IHM décrite en XML (**`extends Activity)`

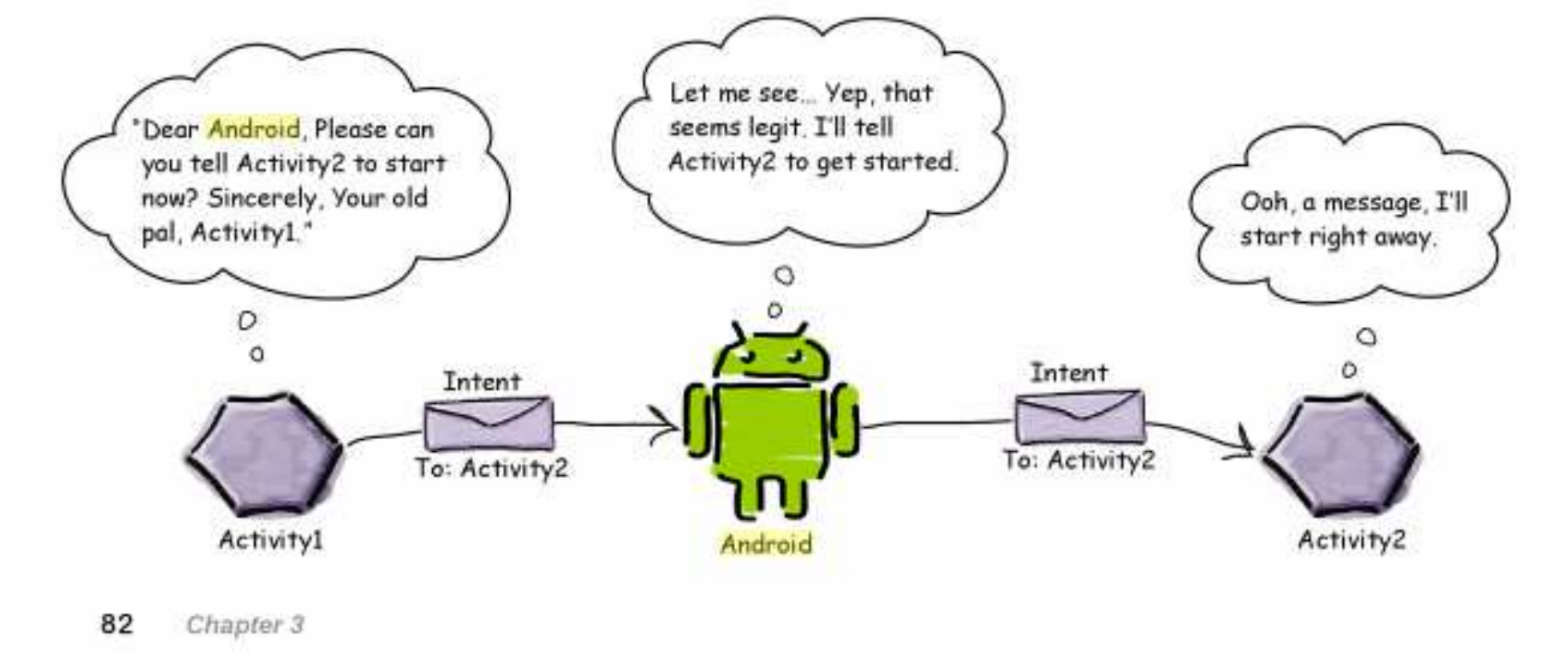
Résumé: Application, Activity …

- Un processus linux contient une application,
- Une application, peut contenir une ou plusieurs activités,
- Une activité se trouve dans un certain état, cf. cycle de vie
- Une activité possède sa vue (View)

Communication inter applications

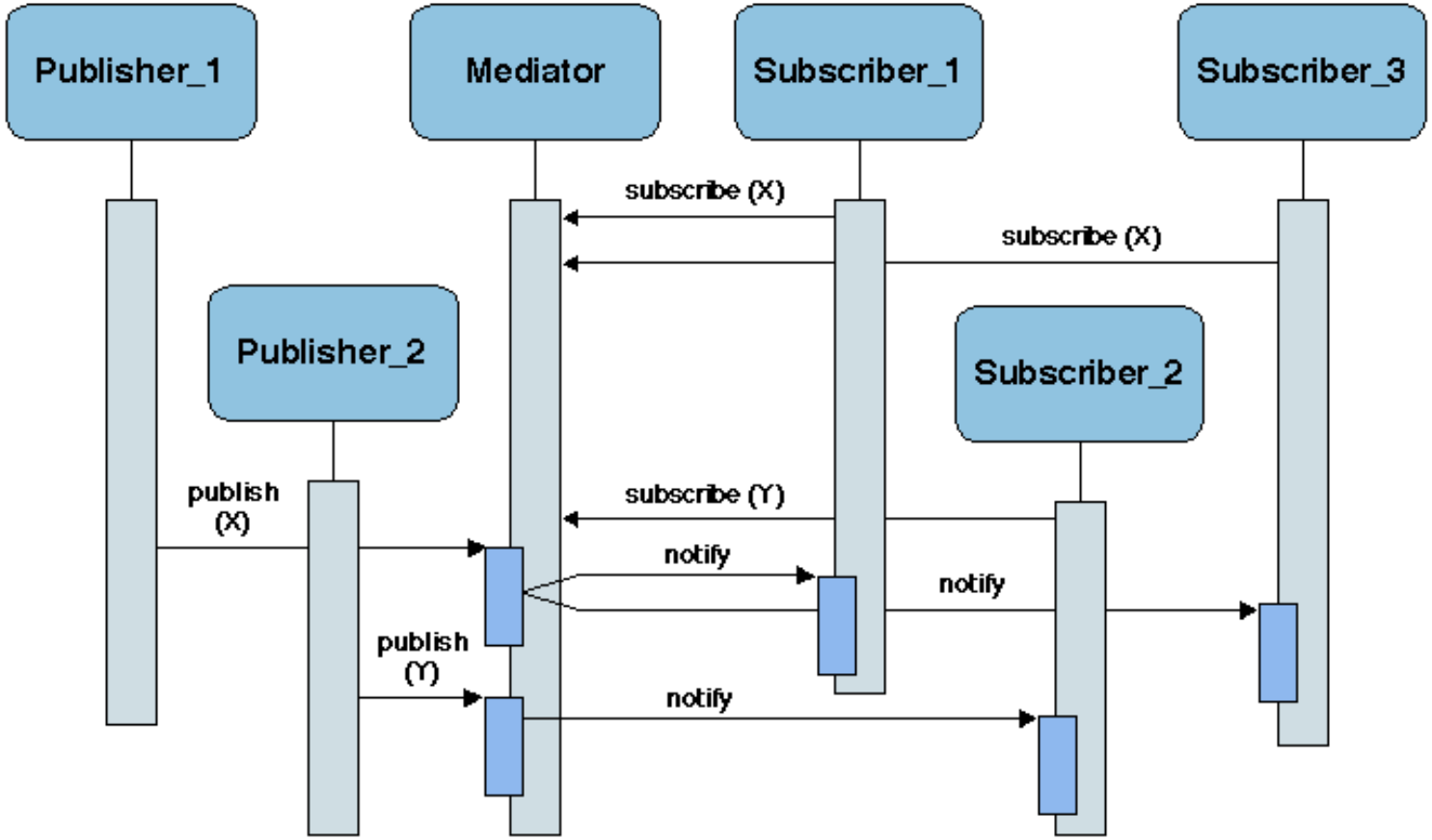
- **Intent, comme notification**
  - **Souscription/Publication**
  - **Appeler, déclencher une autre activité**
    - **Avec éventuellement des paramètres et des résultats attendus**
  - **L'intergiciel/ Android sélectionne la « bonne application »**
    - **Selon les critères exigés par le souscripteur**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

Head First Android O'Reilly



- https://books.google.fr/books?id=qkzrCQAAQBAJ&lpg=PR24&dq=head%20first%20android%20book%20android&hl=fr&pg=PA82#v=onepage&q=head%20first%20android%20book%20android&f=false

Patron Publish-Subscribe/Intent & Context



- 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, le souscripteur

- **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, le souscripteur

- **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);
}
```

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 ?

L'intent peut contenir des paramètres

- **Les extras, un *Bundle*, une *Map* !**
  - **Une table de couples <clé, valeur>, la clé est de type String**

```
Intent i = new Intent();
// i.setAction…
i.putExtra("fichier","hello.mp3");
i.putExtra("compteur",2);
```

Intent	putExtra (String name, double[] value) Add extended data to the intent.
Intent	putExtra (String name, int value) Add extended data to the intent.
Intent	putExtra (String name, CharSequence value) Add extended data to the intent.
Intent	putExtra (String name, char value) Add extended data to the intent.
Intent	putExtra (String name, Bundle value) Add extended data to the intent.
Intent	putExtra (String name, Parcelable[] value) Add extended data to the intent.
Intent	putExtra (String name, Serializable value) Add extended data to the intent.

Des paramètres à l'intention de

L'activité lit les paramètres transmis

- **Les extras, un *Bundle*, une *Map* !**
  - **Une table de couples <clé, valeur>, la clé est de type String**

```
Intent i = getIntent();
String f = i.getStringExtra("fichier");
```

double	getDoubleExtra (String name, double defaultValue) Retrieve extended data from the intent.
Bundle	getExtras () Retrieves a map of extended data from the intent.
int	getFlags () Retrieve any special flags associated with this intent.
float[]	getFloatArrayExtra (String name) Retrieve extended data from the intent.
float	getFloatExtra (String name, float defaultValue) Retrieve extended data from the intent.
int[]	getIntArrayExtra (String name) Retrieve extended data from the intent.
int	getIntExtra (String name, int defaultValue) Retrieve extended data from the intent.

Des paramètres reçus par l'activité sélectionnée

Conclusion

- **Ce n’est qu’une introduction …**