Architectures Mobiles

Tarak Chaari

Maître – assistant à l'ISECS

Votre interlocuteur

- Tarak CHAARI
- Maître – Assistant à l'Institut Supérieur d'Electronique et de Communication de Sfax
- Thèse soutenue fin septembre 2007 à l'INSA de Lyon
- Recherche (ReDCAD – ENIS)
  - l'adaptation dans les systèmes d'information pervasifs
  - gestion sémantique de la qualité de service
- Enseignement (ISECS – ENIS)
  - Ingénierie des systèmes d'information distribués

Présentation générale de l'intervention

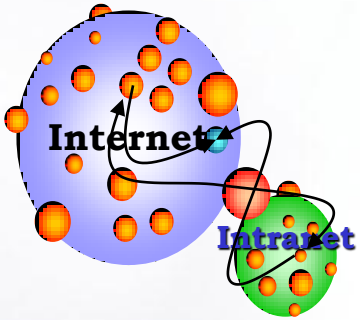
- Initiation à la programmation Android
- Prérequis: POO en Java & un peu d'XML

- Objectifs
  - Présentation des enjeux des architectures mobiles
  - Avoir une idée sur l'architecture Android
  - Avoir une idée sur la programmation Mobile sous Android

Introduction

Constat

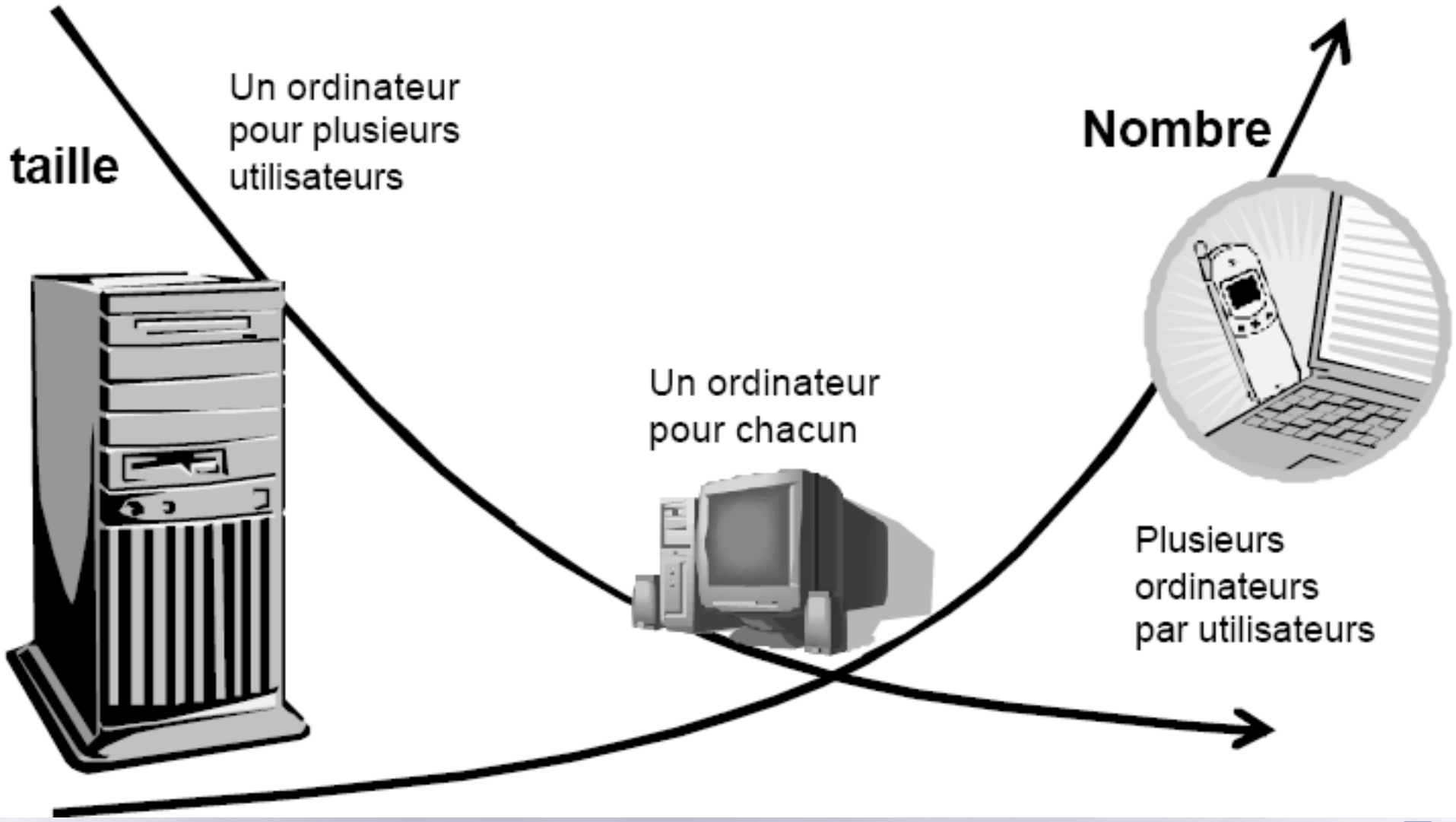
- Explosion du marché des terminaux mobiles
  - Faible coût
  - Marché grande consommation
  - Evolution technologique exponentielle



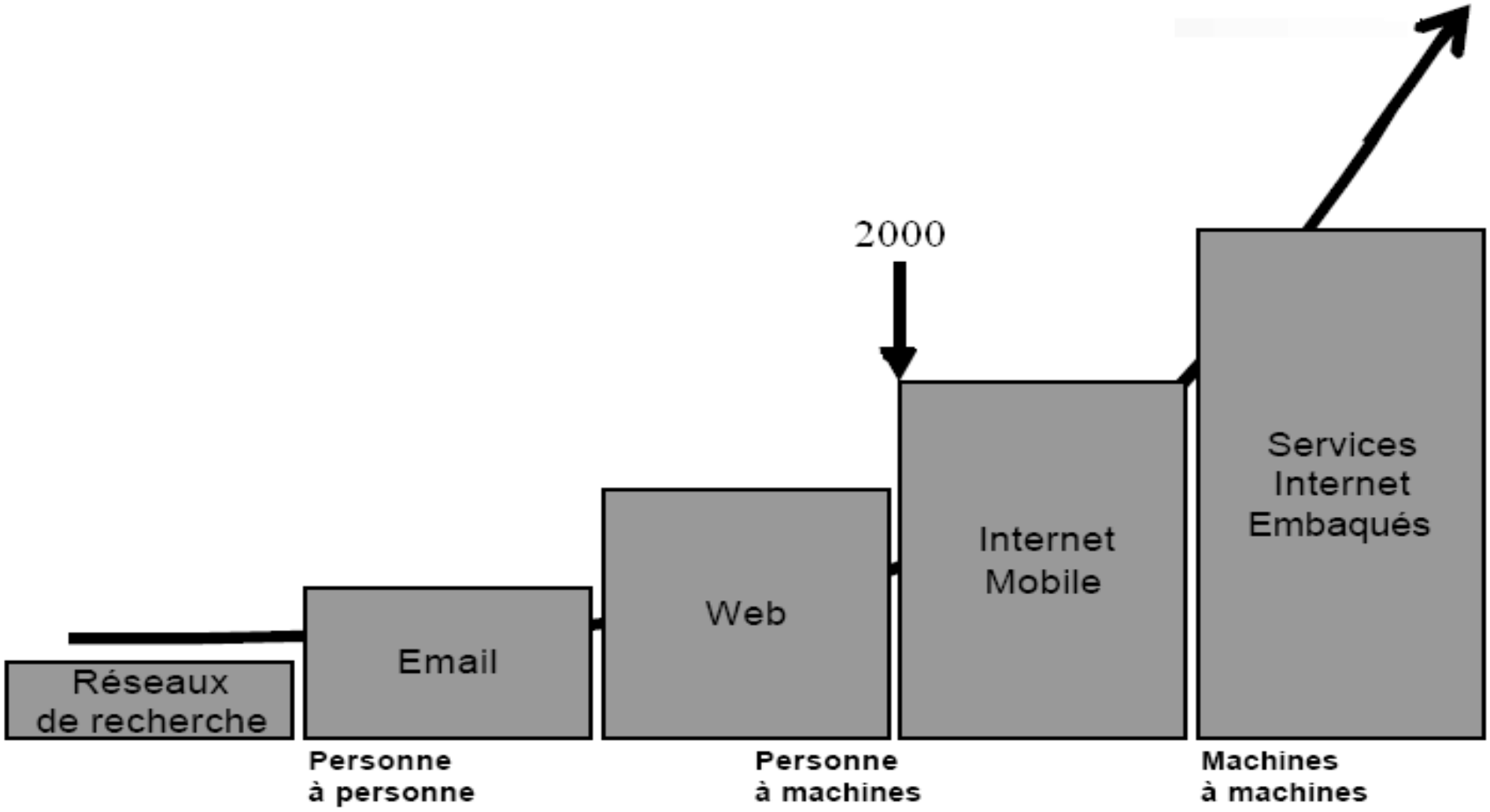
- AVANT :
  - Affichage limité
    - 100x60 pour les écrans N/B
    - 640x240 pour les terminaux avec écrans couleurs
  - Faible mémoire
    - Ne dépasse pas 1Mo de mémoire morte et vive réunies
  - Faibles capacités de calcul
  - Moyens de communication limités
- MAINTENANT :
  - Hautes résolutions, GPS, WIFI, extension mémoire…

➔ L'évolution technologique dépasse le marché de développement

L’ordinateur : une tendance claire



Évolution de l'utilisation d'internet



Enjeux

Fournir l'information partout et à tout moment

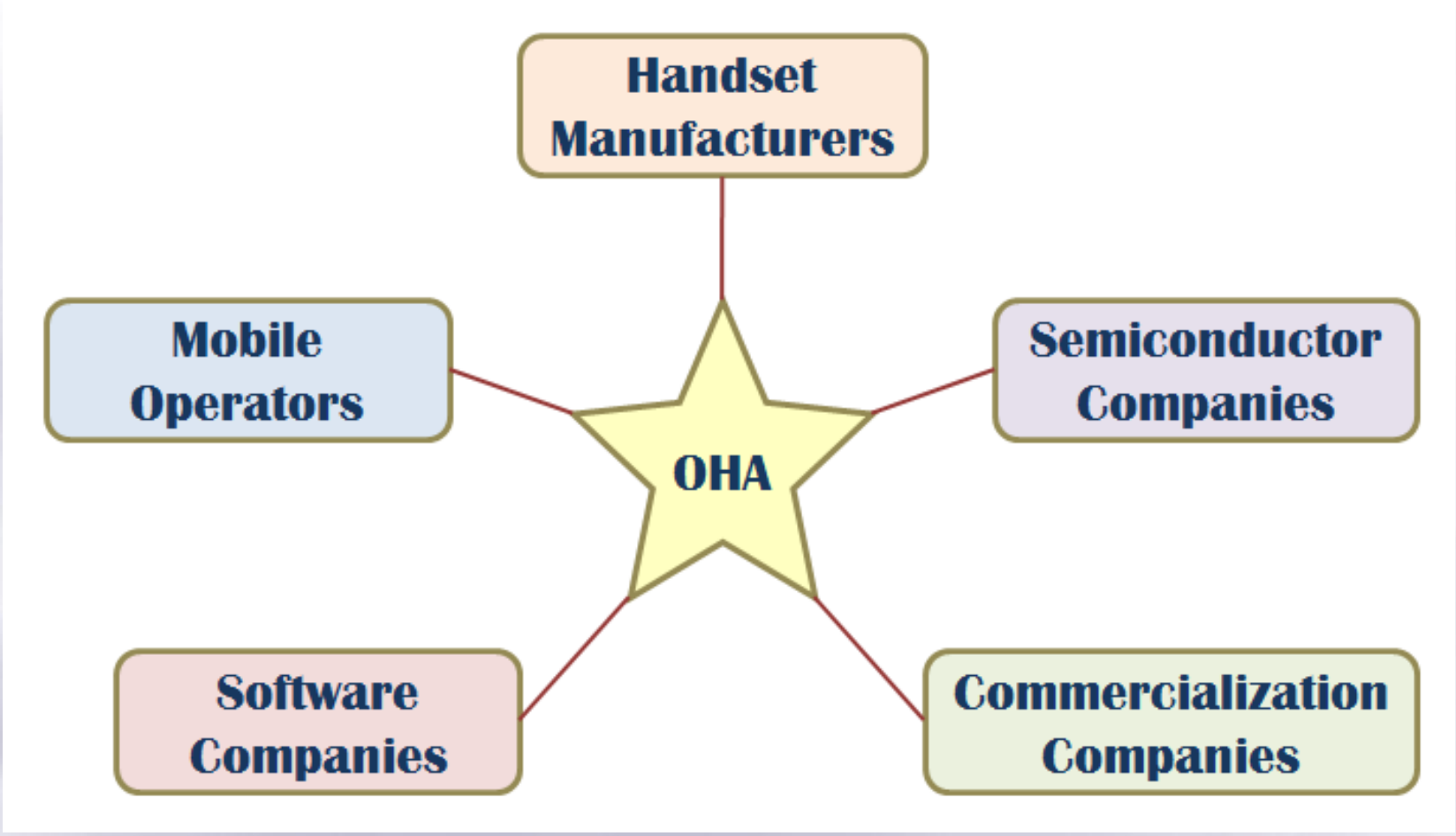
- Téléphone mobile
- PALM
- PDA
- Tablettes
- eBook / cartable électronique

Origines

- Google? Pas tout à fait, c'est une PME américaine qui s'appelle Android créé en 2003 et rachetée par Google en 2005
- Google voulait s'introduire dans le marché mobile (pas réussi au début)
- L'objectif est d'avoir un OS mobile qui peut apercevoir la situation de l'utilisateur (notamment l'emplacement géographique)
- En 2007, Apple dévoile l'Iphone (un gouffre de technologie apparait)

L'influence de l'Open Handset Alliance (OHA) (1/2)

- En novembre 2007, l'OHA a été créé par 35 entreprises dont Google



L’influence de l’Open Handset Alliance (OHA) (2/2)

- L’objectif est d’avoir des standards open source pour les appareils mobiles
- Et pourquoi pas un OS mobile open source pour concurrencer les OS propriétaires iOS et Windows Mobile (à l’image de Linux par rapport à MacOS et Windows)?
- Android est ainsi né de l’OHA qui a attiré des dixaines d’autres entreprises y compris Samsung, LG, HTC, Asus… (plus de 80 à l’heure actuelle)

Philosophie et avantages d'android (1/2)

- **Open Source**
- **Gratuit (même pour les constructeurs)**
  - 25$ pour publier des applications android via le market (play store)
  - publier autant d'application qu'on veut à vie pour 25 $
- **Facile pour le développement**
  - API très complète basé sur JDK 1.6 (+ d'autres classes complémentaires)
  - API simple de haut niveau (2 lignes de code pour envoyer un SMS)
- **Facile à vendre (via le play store)**

Philosophie et avantages d'android (2/2)

Flexibilité

- s'adapte à différentes architectures matérielles (avec clavier, tactile, téléphones, tablettes et même micro-ondes)
- distribution intelligente
  - Exemple : application nécessitant bluetooth → ne sera visible qu'à partir des terminaux avec bluetooth)

Ingénieux

- réutiliser le maximum de l'existant et combiner les composants :
  - Exemple : utiliser le répertoire téléphonique
  - Combiner les composants (Appareil photo + GPS par exemple)
- Collecte de contributions d'amateurs (avantage de l'open source)

Contraintes du développement mobile (1/2)

- Peu de mémoire (8 fois moins qu'un PC en général)
- Respecter l'ordre de priorité des tâches
  - Les applications doivent interagir avec le système sans le bloquer
  - Un SMS ou un appel est toujours plus prioritaire que les autres applications
- Respecter des contraintes de sécurité plus strictes
  - risque d'abimer le matériel est plus fort
- Autonomie (des jeux/applications usent toute la batterie en 30 min)

Contraintes du développement mobile (2/2)

- **Puissance de calcul limité (risque de surchauffe)**
- **Taille de l'écran**
  - **Réduite voire variable**
  - **L'interface doit d'adapter à plusieurs écrans (risque de perdre des utilisateurs)**
- **Nature de l'écran**
  - **avec stylet → petits composants**
  - **sensitif → composants plus gros**
- **Hétérogénéité (langues, composants matériels, versions de l'OS…)**
- **Utilisateur plus exigeant (ce ne sont plus les mêmes besoins qu'avec un Nokia 3310)**

Java : un choix stratégique

- **Portabilité (la plus grande motivation)**
- **Partager le même langage et la même conception depuis les serveurs jusqu'aux terminaux**
- **Programmes compacts et portables**
- **Ateliers et Outils homogènes**
- **Interaction plus évoluée que les applications WEB**
- **Gestion complète des ressources matérielles**

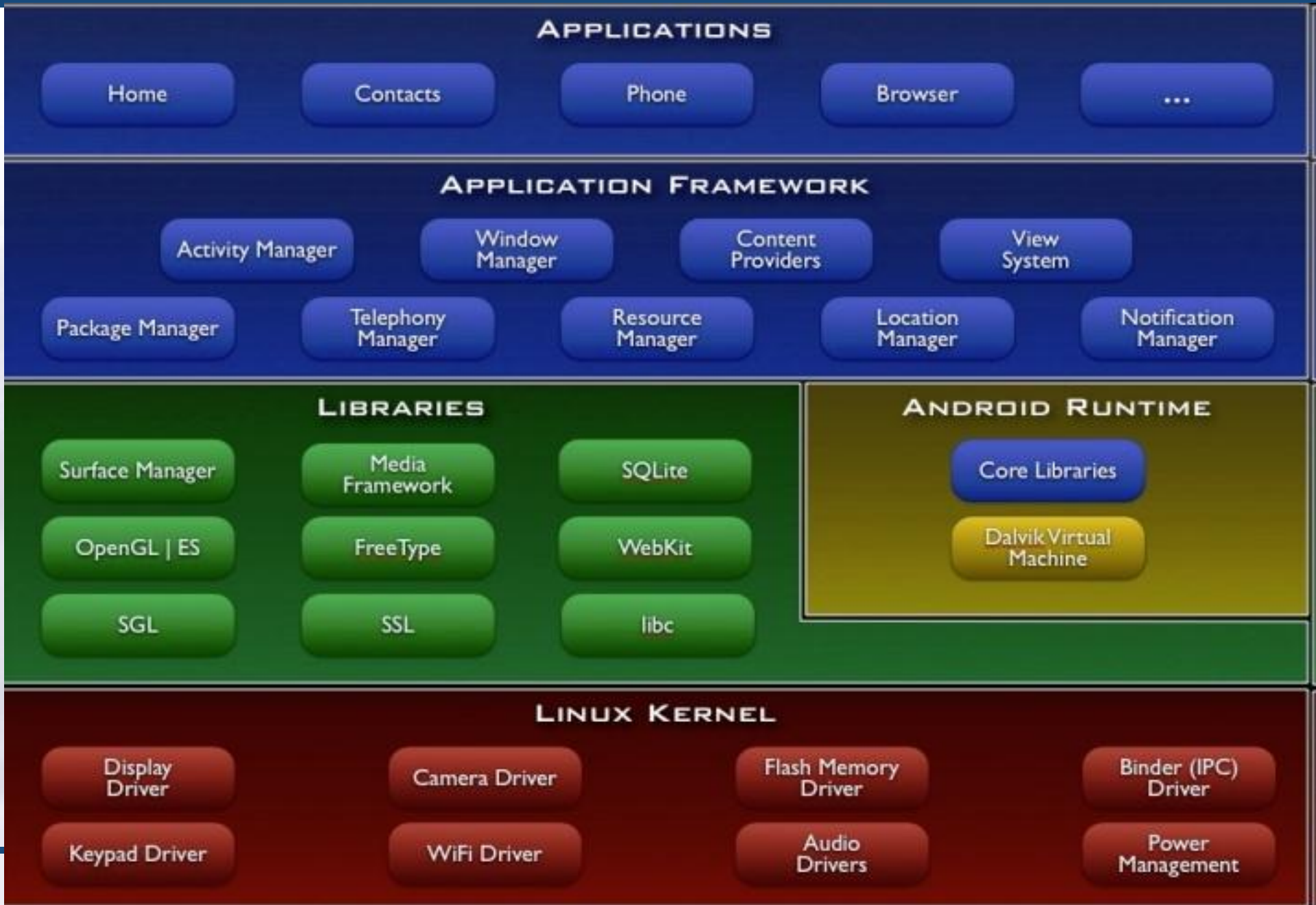
Les technologies actuelles JAVA

- Coté serveur : J2EE
- Développement standard (PC) : J2SE
- Coté clients légers : J2ME → Android
- Pour les systèmes embarqués : EmbeddedJava
- Pour les cartes à puces : JavaCard

Architecture d'Android

Une architecture complexe pour vous rendre la vie plus facile

Architecture à couches



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 à la couche matérielle par le noyau Linux

Le noyau android

Noyau standard Linux assurant :

- ✓ Gestion de mémoire et de processus
- ✓ Gestion de la pile réseau
- ✓ Gestion des pilotes des capteurs
- ✓ Gestion des permissions d’accès

Avec quelques retouches pour le matériel du mobile :

- ✓ Alarm
- ✓ Process Binder
- ✓ Power Management
- ✓ Low Memory Killer
- ✓ Kernel Debugger
- ✓ Logger

Gestion de l'énergie avec le noyau android (1/2)

Gestion de l'énergie

✓Problème

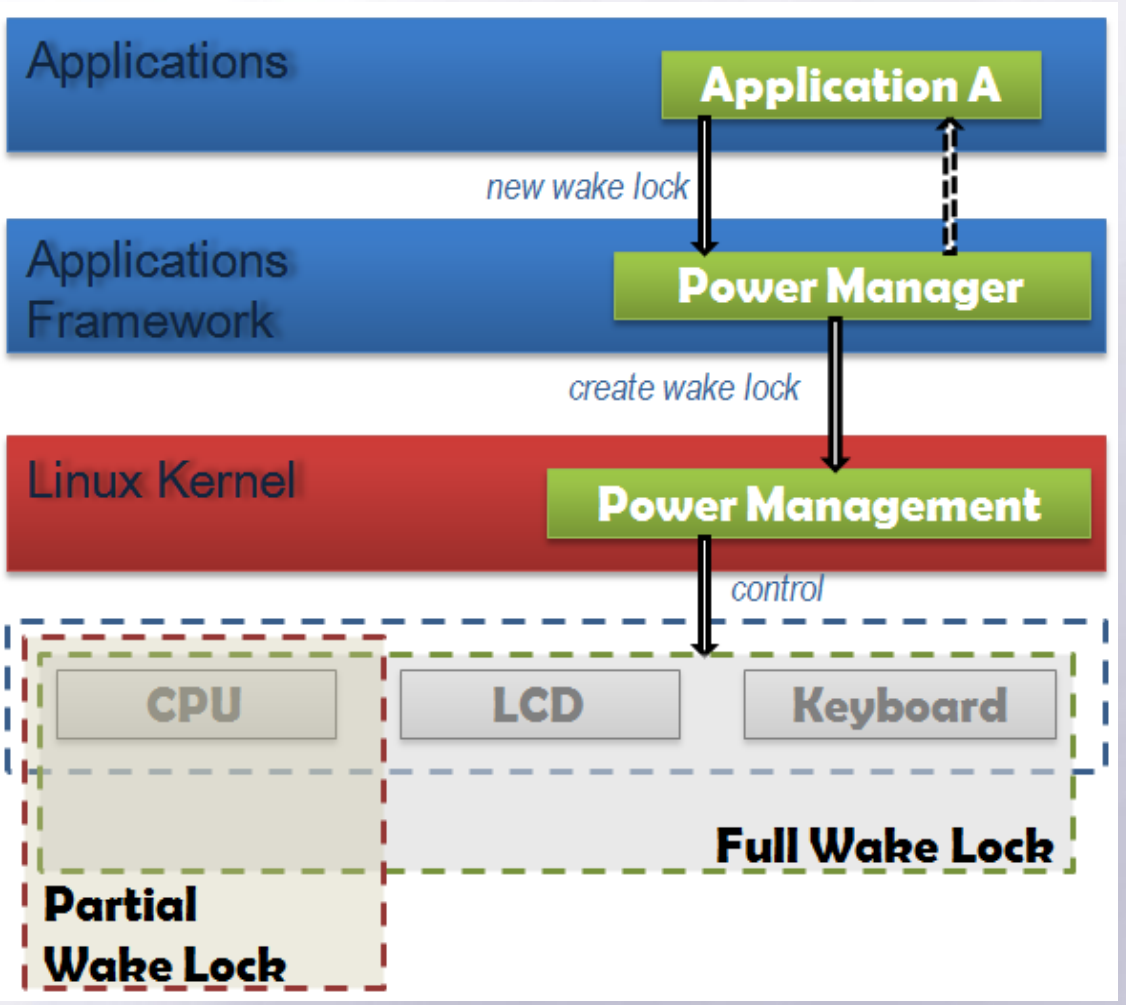
- Limite de la capacité des batteries et sa dégradation continue avec leur utilisation

✓Propriétés du module de gestion d'énergie (PM)

- PM est basé sur la gestion standard de l'énergie dans un noyau linux
- PM ajoute des règles un peu plus agressives pour préserver l'énergie
- Principe de la course à la mise en veille (Race to idle)
- Les couches supérieures demande d'activer les composants matériels à travers des signaux appelés "Wake Locks".
- Support de plusieurs types de "Wake Locks".

Gestion de l'énergie avec le noyau android (2/2)

Fonctionnement du module de gestion d'énergie



- ✓If there are no active wake locks, CPU will be turned off.
- ✓If there are no partial wake locks, screen and keyboard will be turned off.

Environnement d'exécution Android (1/3)

Core Libraries

- ✓ Fournit les fonctionnalités offertes par les librairies de bases du langage Java
- ✓ APIs
  - Structures de données
  - Utilitiés
  - Gestion d'accès aux fichiers
  - Accès au réseau
  - Composants graphiques
  - Etc

Environnement d'exécution Android (2/3)

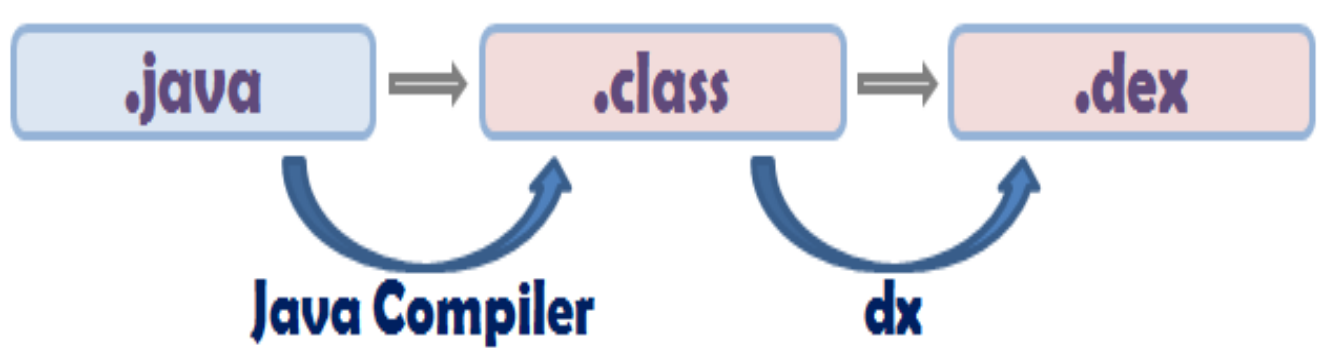
Dalvik Virtual Machine

- ✓ Fournit un environnement sur lequel toute application android est exécutée
  - ➢ Chaque application Android est exécutée dans son propre processus et dans sa propre instance de Dalvik VM.
  - ➢ Dalvik VM a été réalisé d'une façon efficace que la grande majorité des terminaux puissent exécuter plusieurs instances de la machine virtuelle sans problèmes.
- ✓ Machine virtuelle basée sur des registres et non sur une pile

Environnement d'exécution Android (3/3)

Dalvik Virtual Machine

- ✓ Execute des fichiers avec un format spécial (.dex)
  - ➢ Le format .dex est optimisé pour consommer le moins de mémoire
  - ➢ Compilation



- ✓ Utilise le noyau Linux pour:
  - ➢ Threading (gestion des processus)
  - ➢ Gestion de la mémoire physique du système

Librairies (1/2)

- Développées en C/C++
- Fournissent des fonctions de base pour les applications Android (développées en Java)

Librairies (2/2)

Suite de la liste des librairies principales

- ✓System C Library (Bionic)
- ✓Media Libraries
- ✓Surface Manager (Surface Flinger)
- ✓Audio Manager (Audio Flinger)
- ✓3D Libraries (OpenGL)
- ✓LibWebCore (WebKit)
- ✓Gestion de la persistance (SQLite)

Bionic

- ✓Custom libc implementation optimized for embedded use
- ✓Problem with GNU libc

License	The authors want to keep GPL out of user-space.
Size	Libc will load in each process, so it needs to be small.
Speed	Limited CPU power means it needs to be fast.

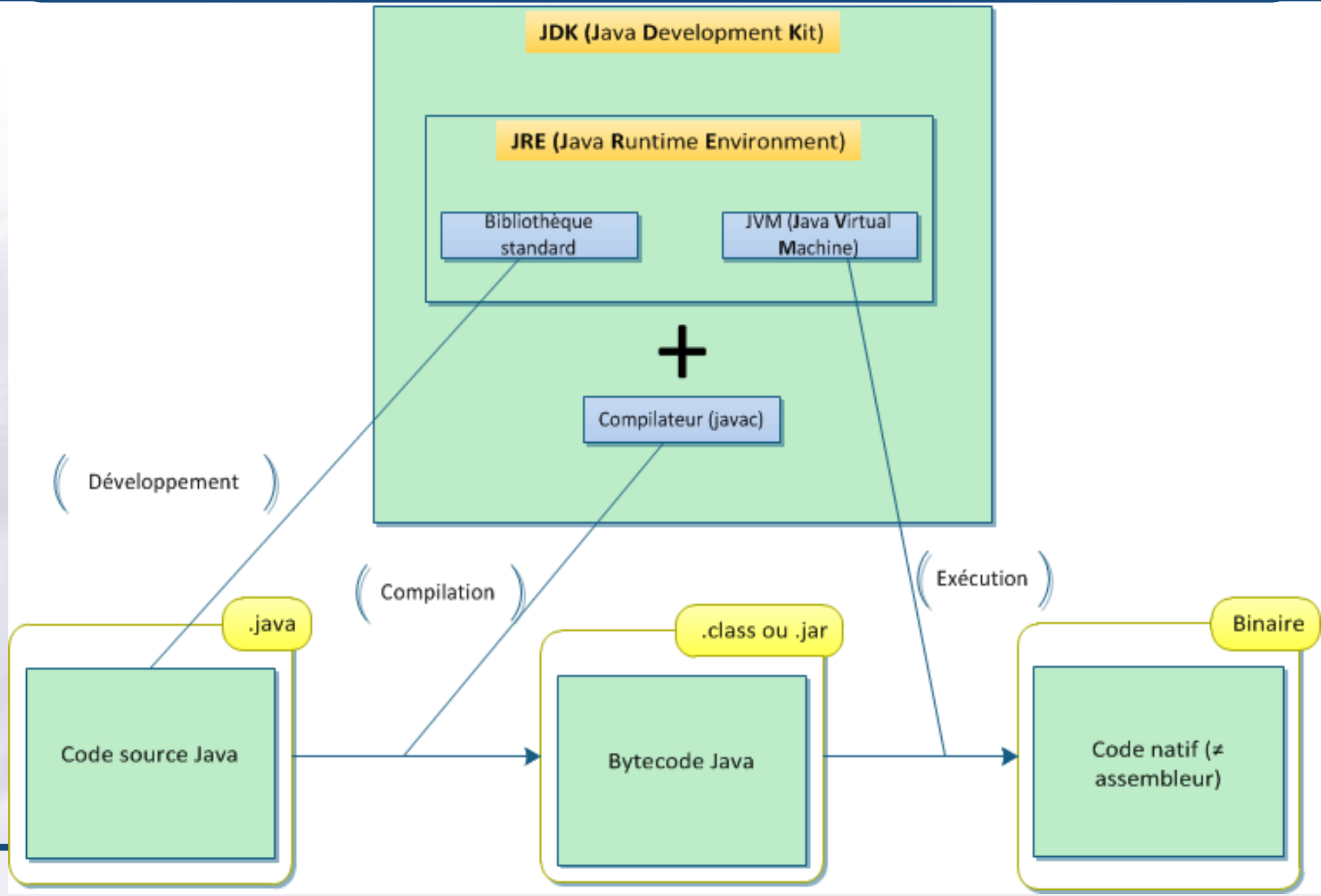
Framework des applications

- **Composants Java de base pour les autres applications**
- **View System: composants graphiques (Boutons, zones texte…)**
- **Resource Manager: accès aux ressources (images, icones, traductions de texte, dispositions sur l'écran…)**
- **Content provider: Partage de données entre applications**
- **Activity Manager: Gestion du cycle de vie des applications**

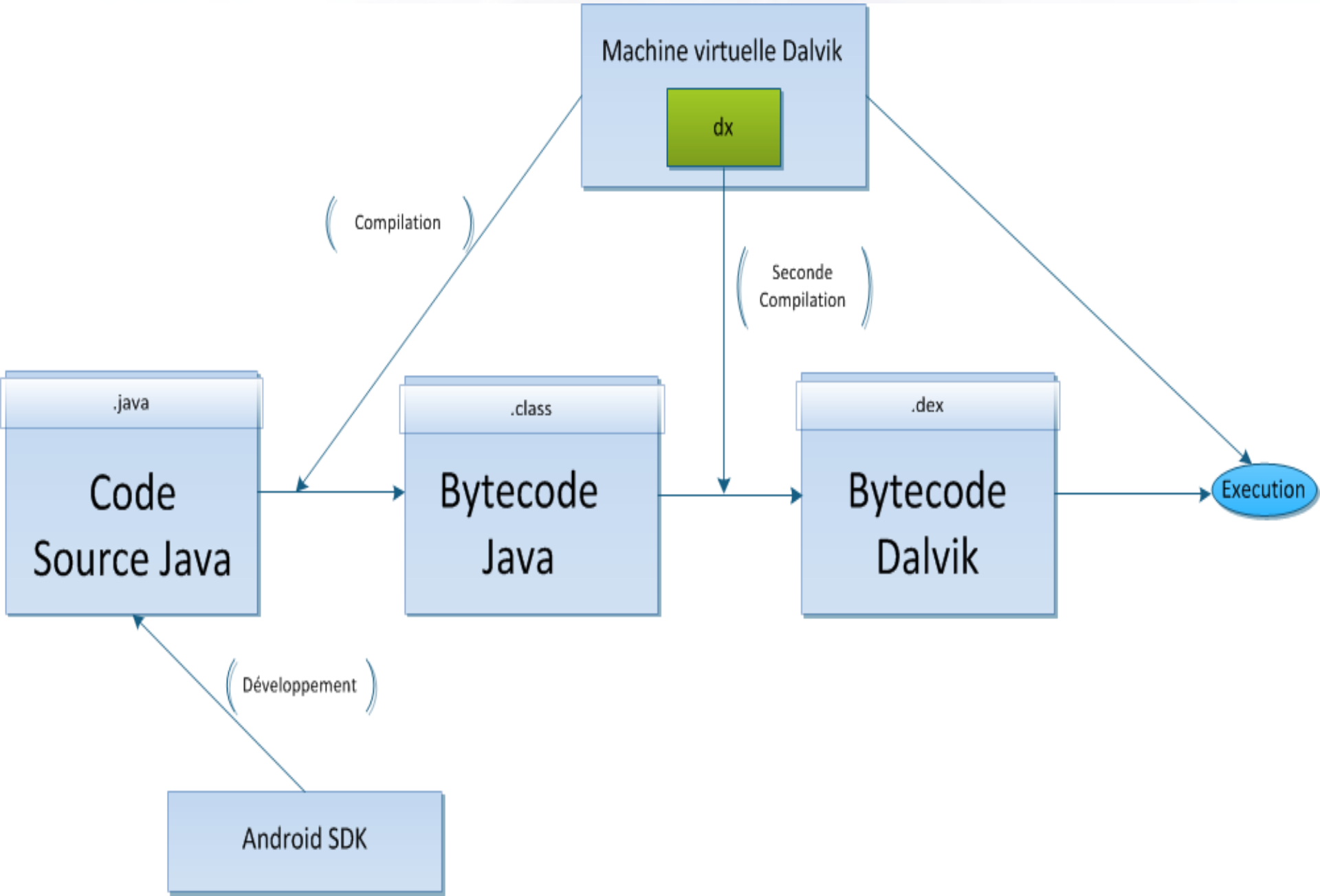
Couche « Applications »

- **Applications de base pour utiliser le téléphone**
  - **Répertoire téléphonique**
  - **Application de téléphone**
  - **Navigateur**
  - **Paramètres du téléphone**
  - **Lanceur d’applications**

Etapes de développement (JVM)



Etapes de développement (Dalvink VM)

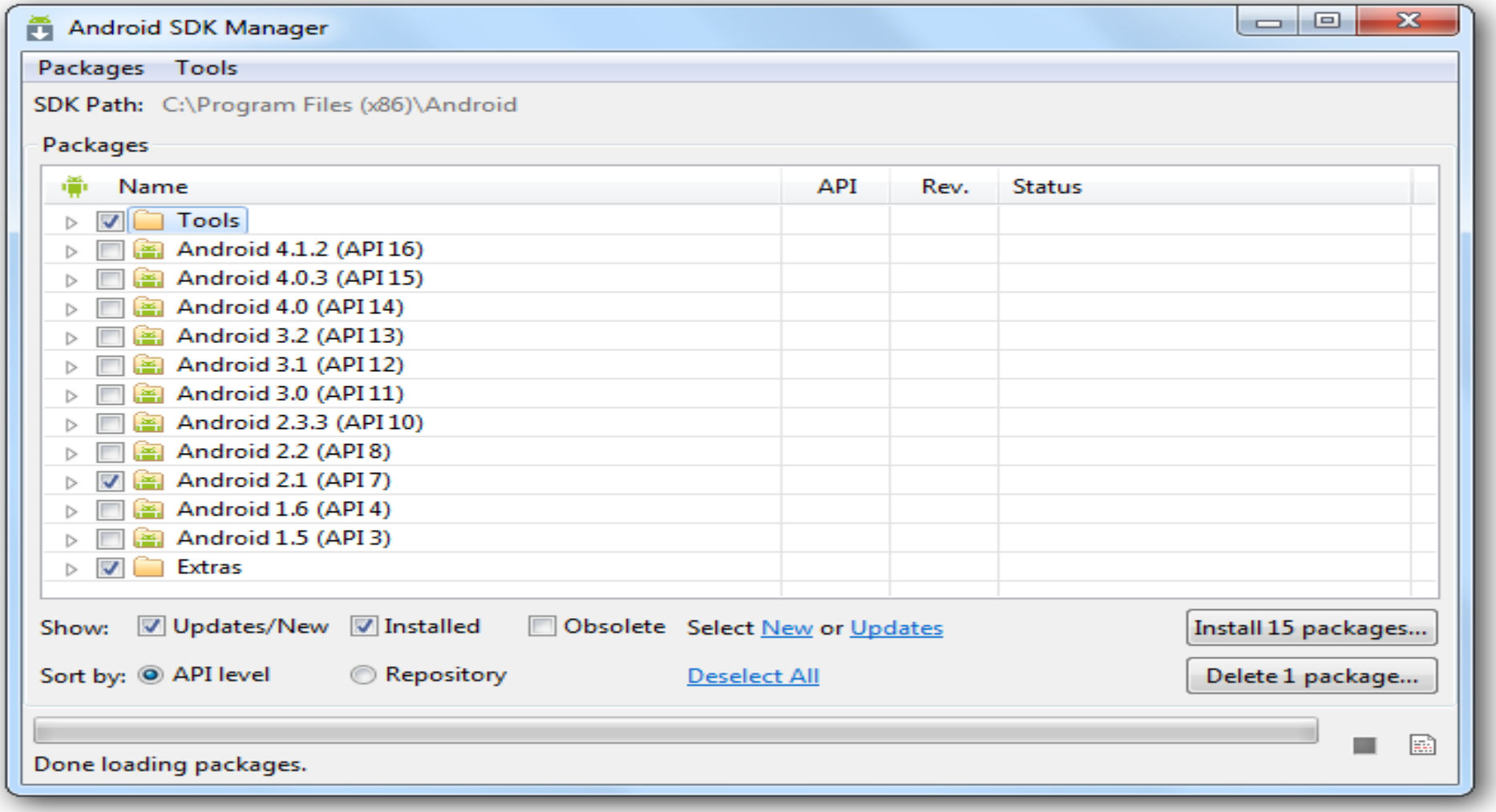


Outils Android

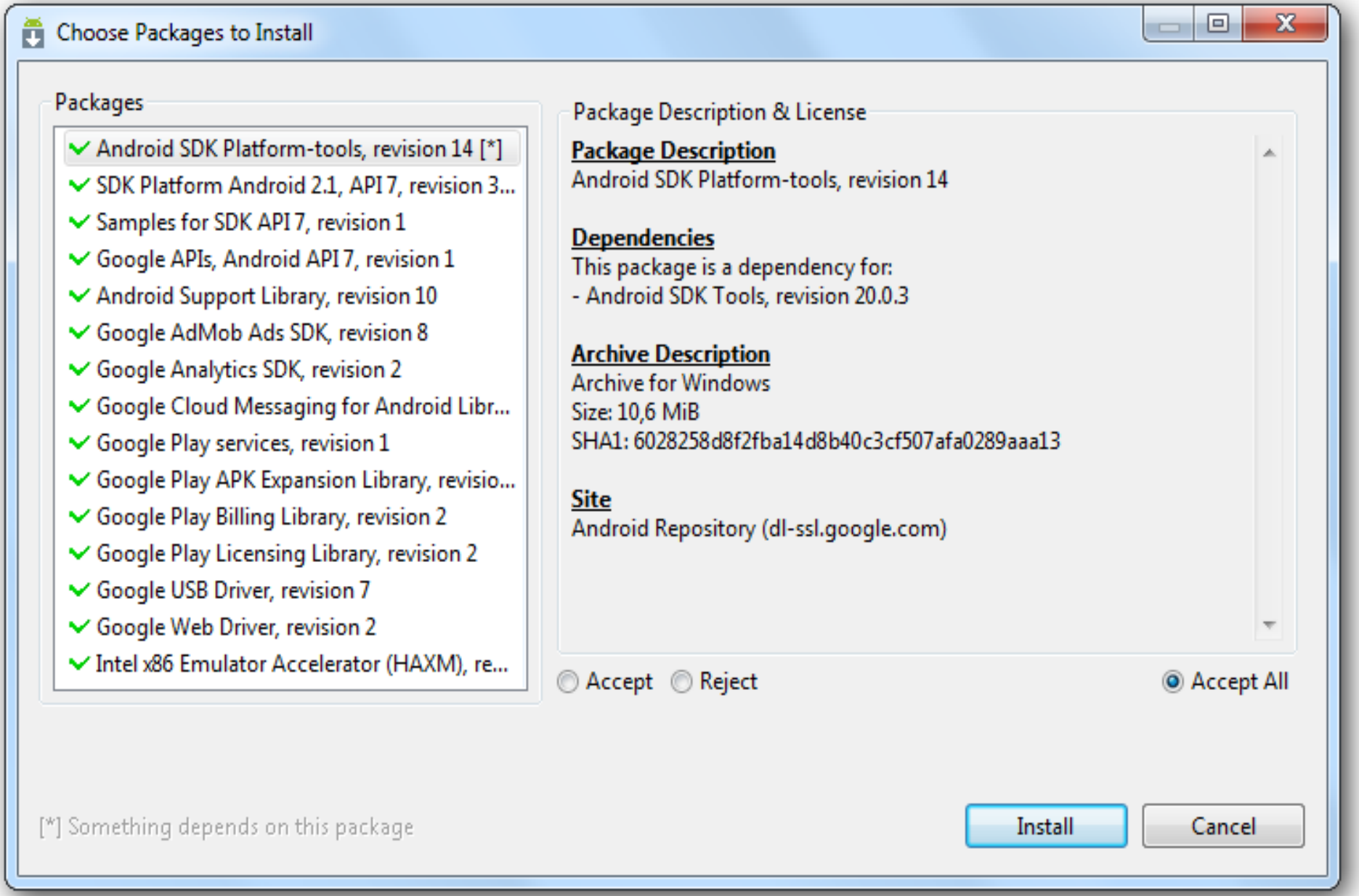
Le outils fournis pour le développement Android ont fait son succès

Prérequis (1/2)

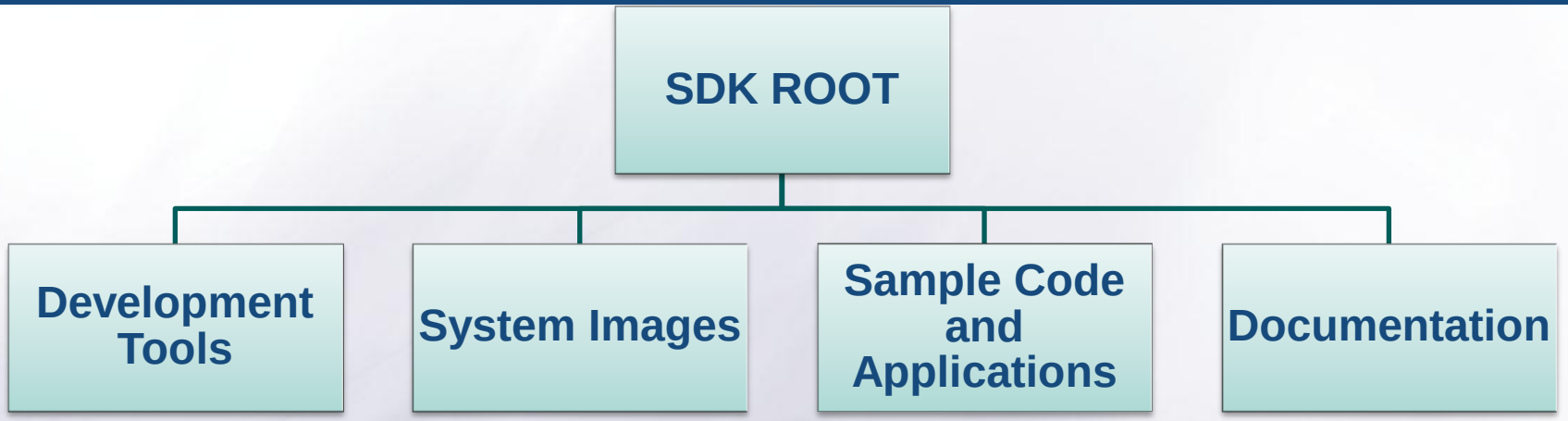
- **JDK** http://www.oracle.com/technetwork/java/javase/downloads
- **Android SDK** http://developer.android.com/sdk/index.html



Prérequis (2/2)



Contenu de “Android SDK”



Répertoire	Description
Development Tools	Un ensemble d'outils pour le développement et le déboggage des applications et pour la création des interfaces utilisateurs
System Images	Images système d'Android
Sample Code and Applications	Des exemples d'applications Android
Documentation	JavaDoc locale

Contenu de “Android SDK” (Cont)

Development Tools

- ✓ **Hierarchy Viewer**
  - Un outil permettant le déboguage et l’optimisation des interfaces utilisateurs
  - Cet outil offre une représentation visuelle des dispositions hiérarchique des composants graphiques

- ✓ **Android Debug Bridge (adb)**
  - Un outil qui permet d’installer et de déboguer les applications développées
  - Ceci peut se faire de la même façon que sur un vrai terminal que sur un émulateur
  - Il offre un ensemble de commandes pour gérer le transfert d’applications entre l’espace de développement et le terminal ou l’émulateur
  - Le déboguage se fait à travers un Log (journal) maintenu sur le terminal ou l’émulateur (adb logcat)

Contenu de “Android SDK” (Cont)

Development Tools (Cont)

- 9-patch: Un éditeur graphique pour la création d’images redimentionnables ou dédiées à une résolution donnée

- Android Asset Packaging Tool (aapt)
  - C’est l’outil de finalisation du développement qui permet de créer des paquetages d’application (Android Package Files .apk) contenant les fichiers compilés et les ressources (images, fichiers de configuration…)

- Dalvik Debug Monitor Service (ddms) : un outils qui permet de gérer les processus sur l’émulateur ou le terminal
  - Tuer des processus, sélectionner un processus à déboguer,faire des copiers d’écrans…

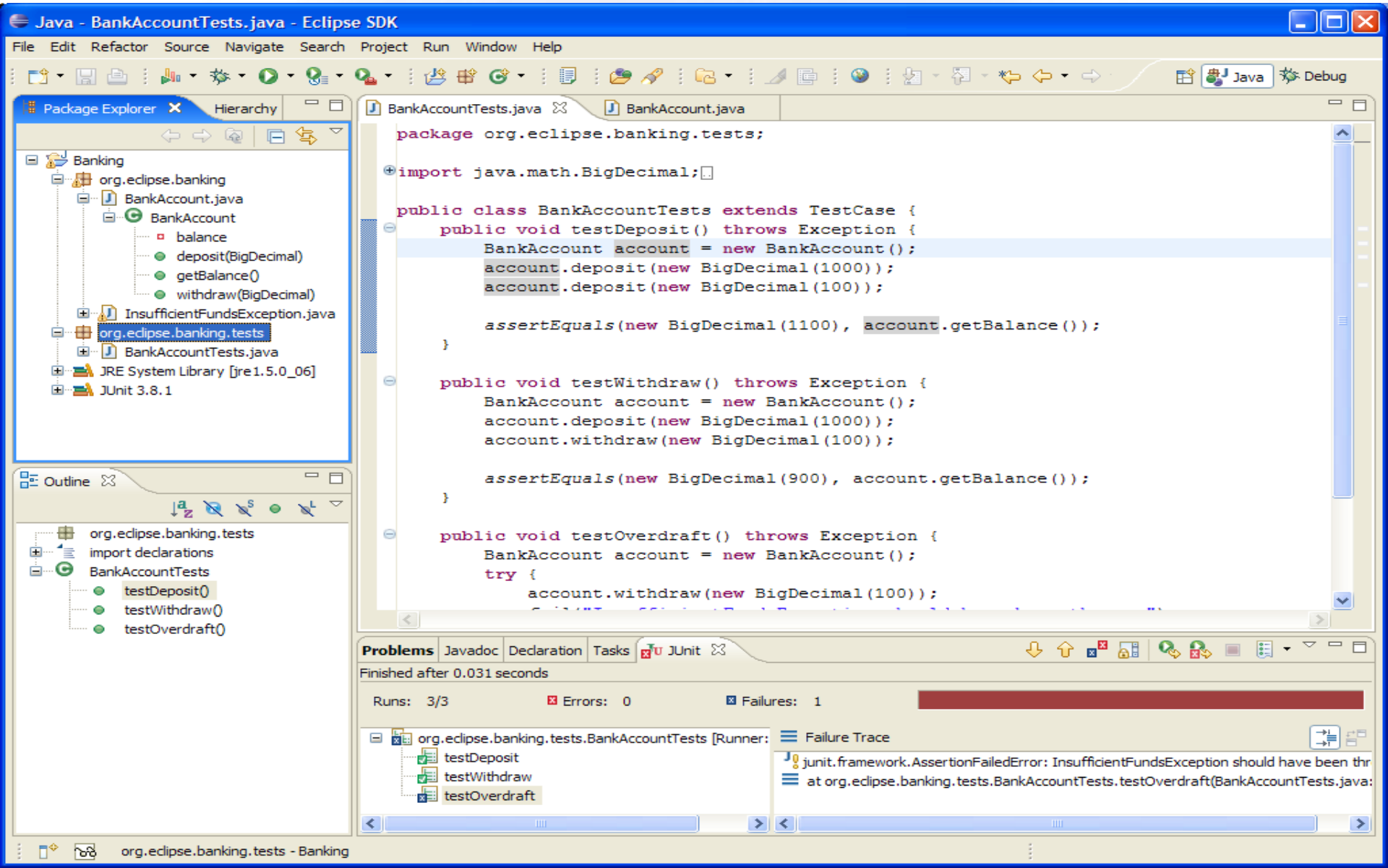
Contenu de “Android SDK” (fin)

Development Tools (Cont)

- ✓ **Android Interface Description Language (aidl)**
  - ➢ Un langage qui permet de définir les interfaces déchanges inter-processus
  - ➢ Ce sont des interfaces Java
- ✓ **sqlite3**
  - ➢ Un outil permettant d’accéder aux bases de données SQLite des applications Android
- ✓ **Traceview**
  - ➢ Analyse statistiques sur les applications (mémoire utilisée, charge processeur…)
- ➢ **dx**
  - ➢ L’outil qui permet de transformer les .class en bytcode Dalvik

IDE de développement

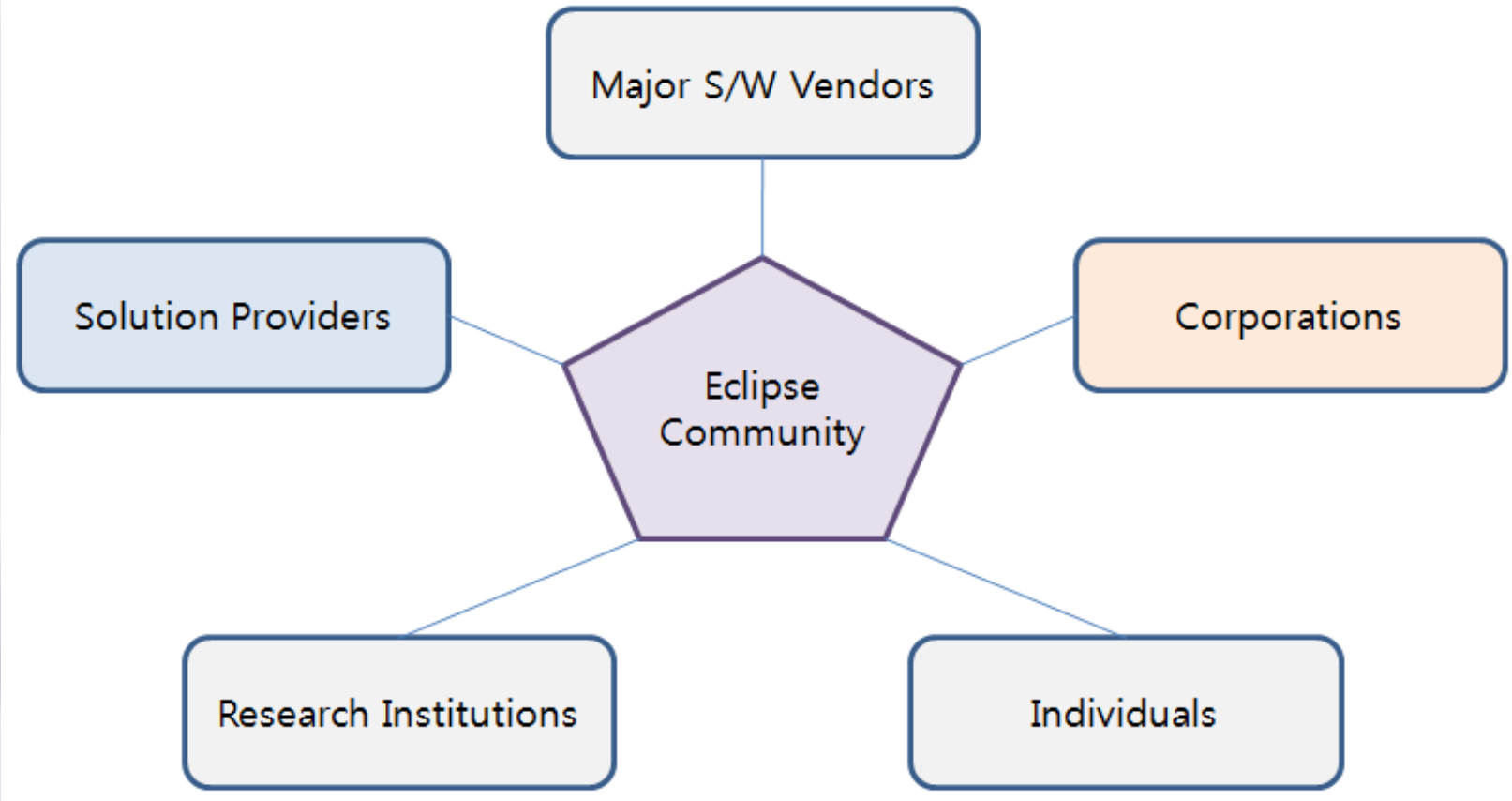
Pour ne pas déstabiliser les développeurs: un éditeur habituel



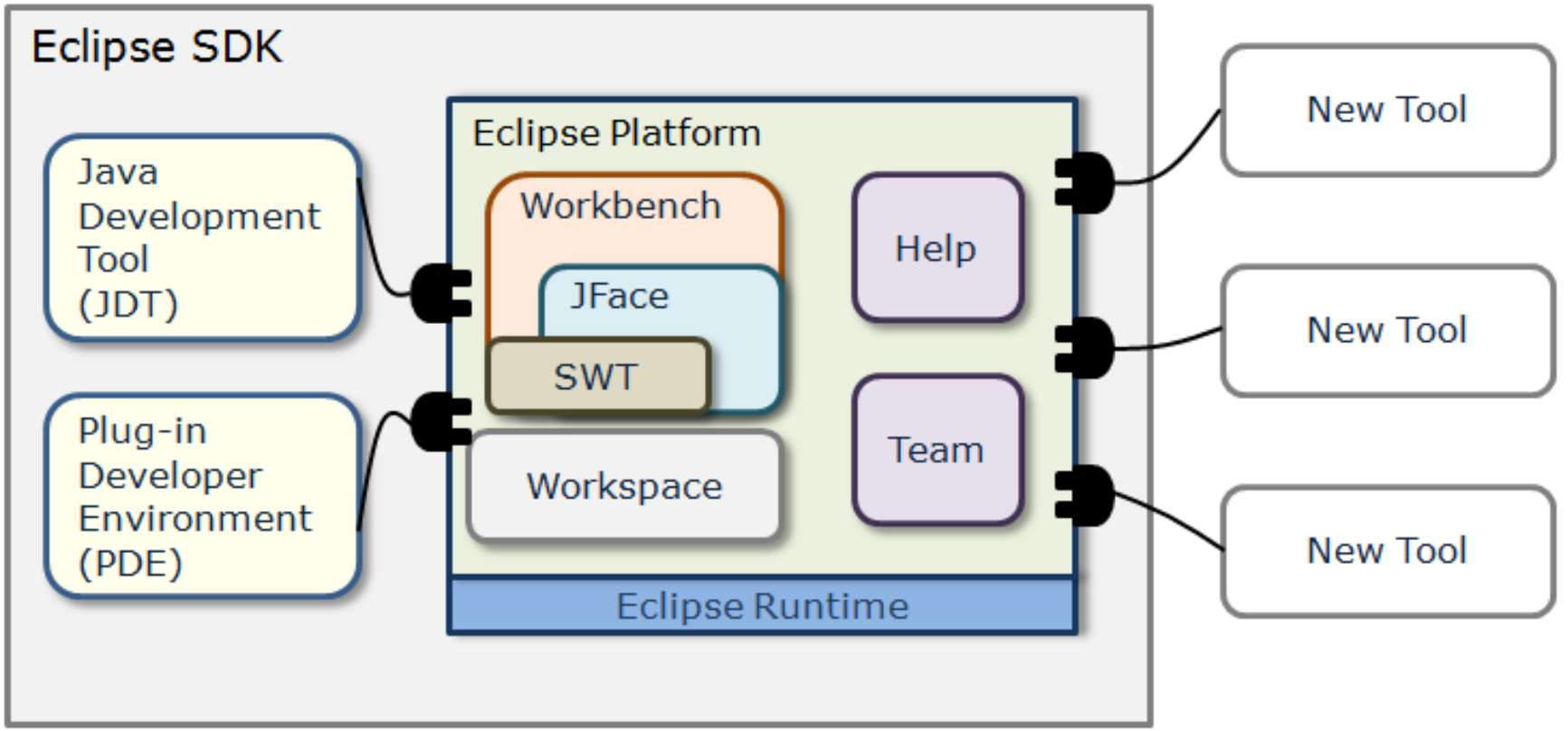
IDE de développement (cont)

Eclipse

- ✓Un IDE open source project développé par une grande communauté



Eclipse Platform



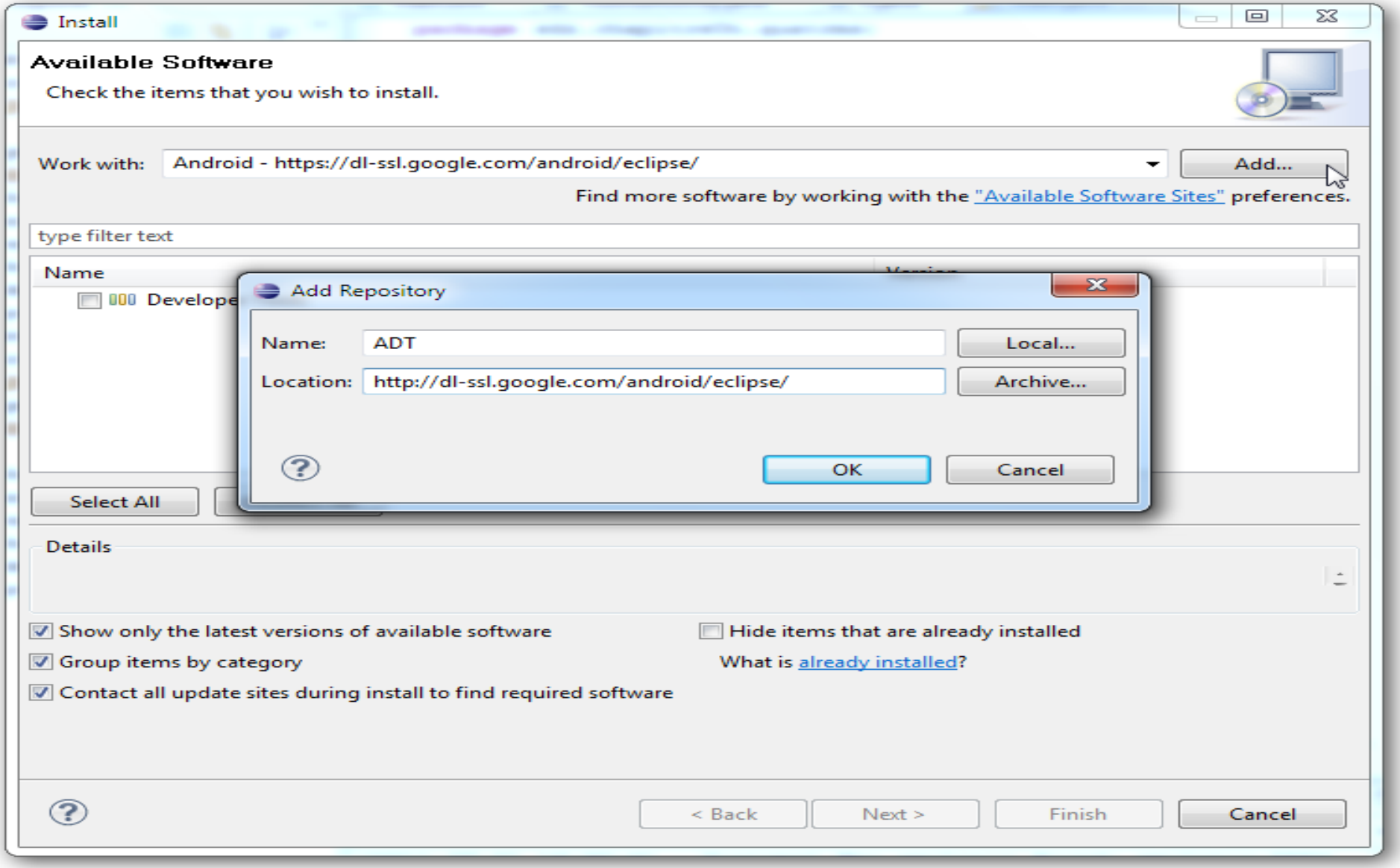
- ✓Eclipse est basé sur une architecture ouverte permettant d'ajouter des plug-ins de développement offrants de nouvelles fonctionnalités

IDE de développement (cont)

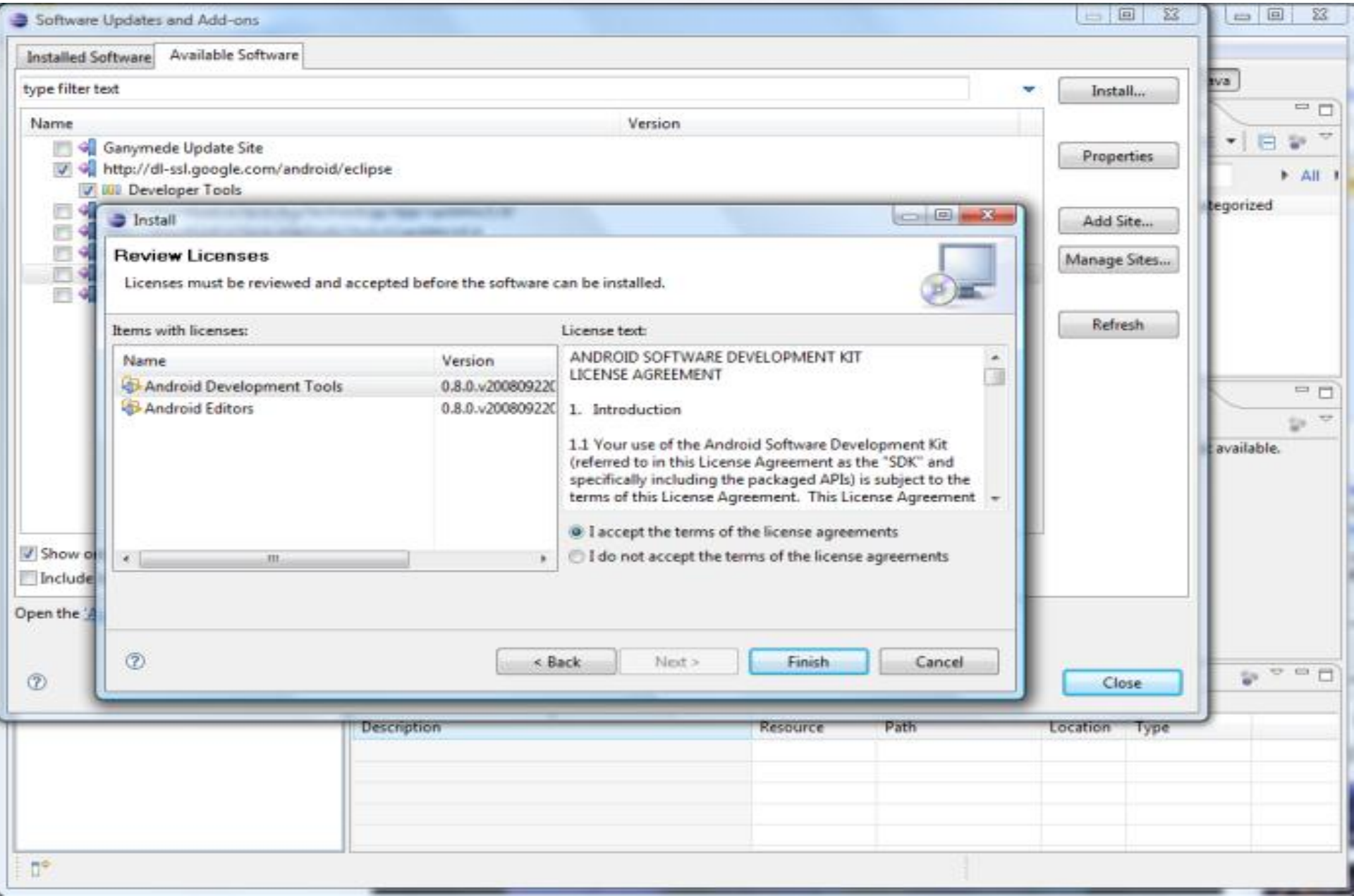
Exemples de plug-ins:

Plug-in	Function
JDT	provides the capability to create, edit, navigate, build, and debug projects that use Java as a programming language
CDT	provides the capability to create, edit, navigate, build, and debug projects that use C and/or C++ as a programming language
UML2	provides the capability to create UML models
...	...

Le plugin ADT - Android Development Tools

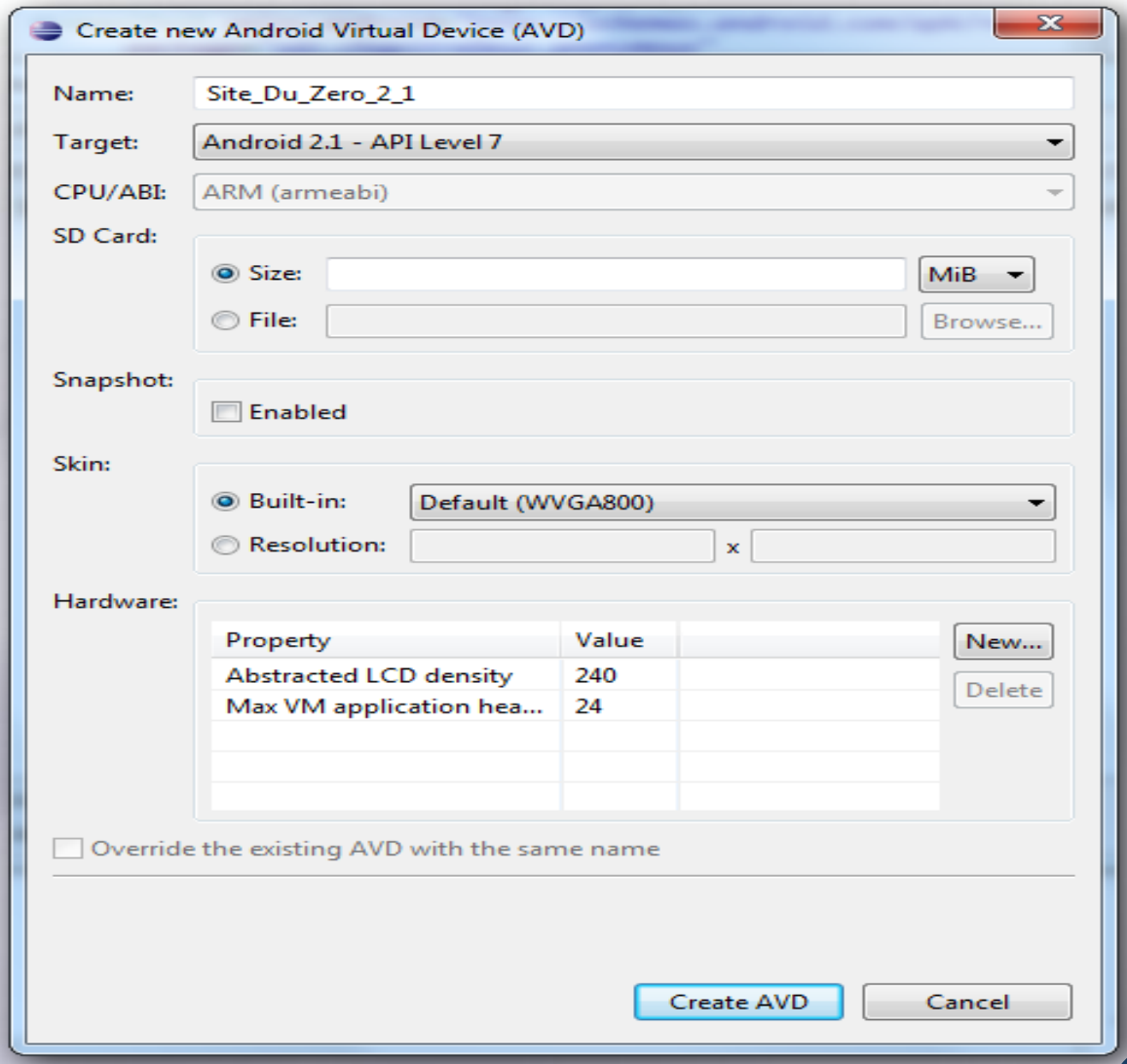


Installation - ADT



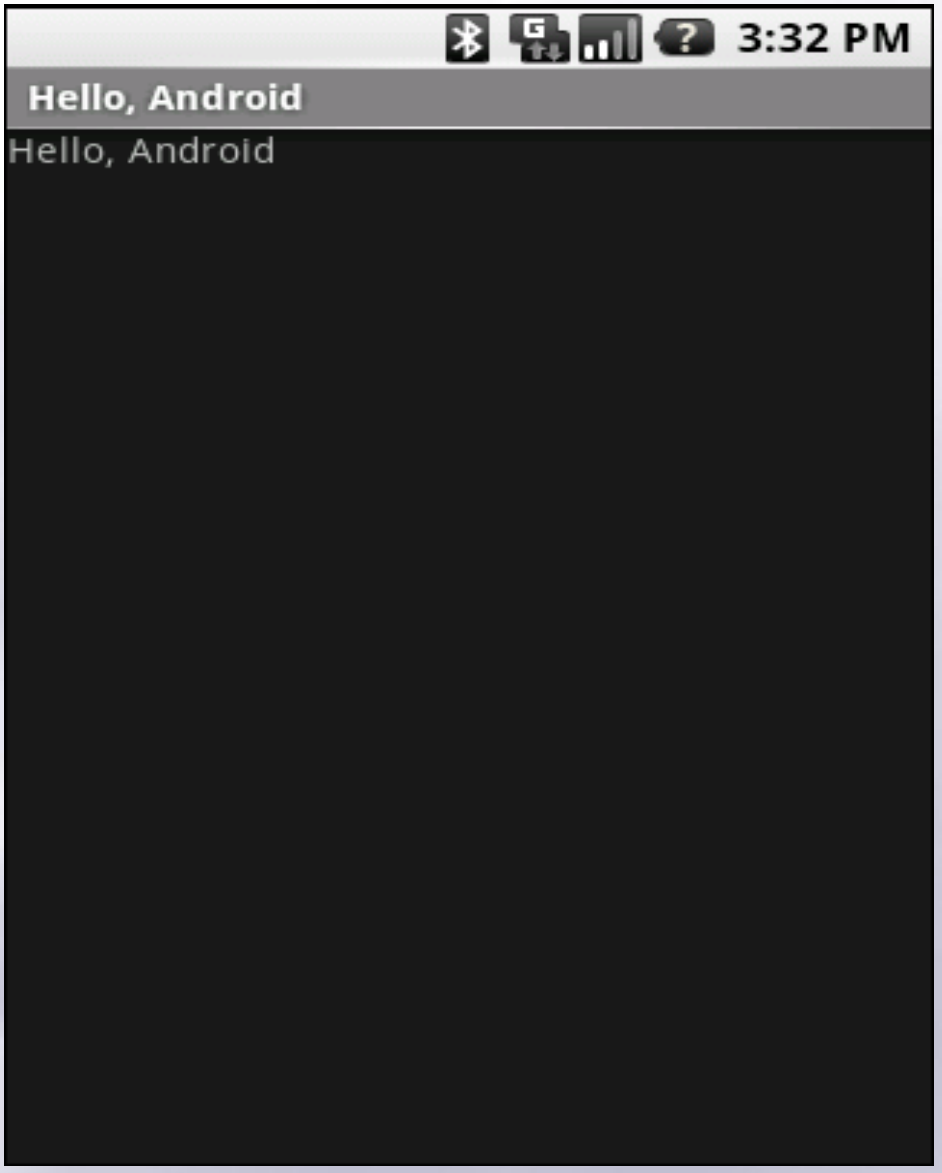
Un émulateur (1/3)

- Pour essayer les applications développées (pas obligatoire)



Un émulateur (2/3)

Un émulateur (3/3)



Pour le test sur le terminal, ne pas oublier d'installer les pilotes USB !

Composants principaux d'applications Android

Le développement d'applications (graphiques) sur Android est basé sur la notion d'activités

Mots-clés

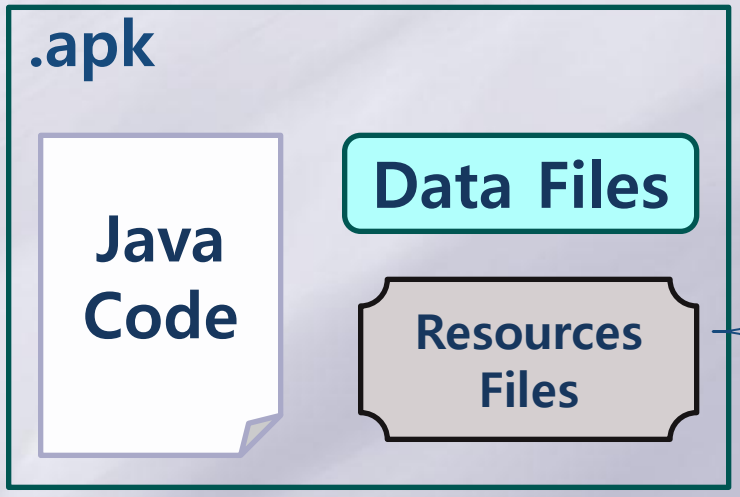
- Applications,
- Communication, évènements, intentions,
- Services en tâche de fond,
- Persistance.

Une Application hérite le la classe Activity

- Surcharge de certaines méthodes
  - Du déjà vu : MIDlet, Applet, Servlet,…
- Le cycle de vie est imposé par le framework
  - Déjà vu : pour une Applet init() puis …

Contenu d'une application Andoid

- Les applications android sont développées en Java
- Une application android (.apk) et construite par l'outil aapt



- res/layout: declaration layout files
- res/drawable: intended for drawing
- res/anim: bitmaps, animations for transitions
- res/values: externalized values
  - strings, colors, styles, etc
- res/xml: general XML files used at runtime
- res/raw: binary files (e.g. sound)

Composants principaux d'applications

- Les applications Androids n'ont pas un point d'entrée unique (comme la méthode main())
- Les applications Android sur des composants que le système pourrait instancier quand il le faut
- Ces composants principaux sont :

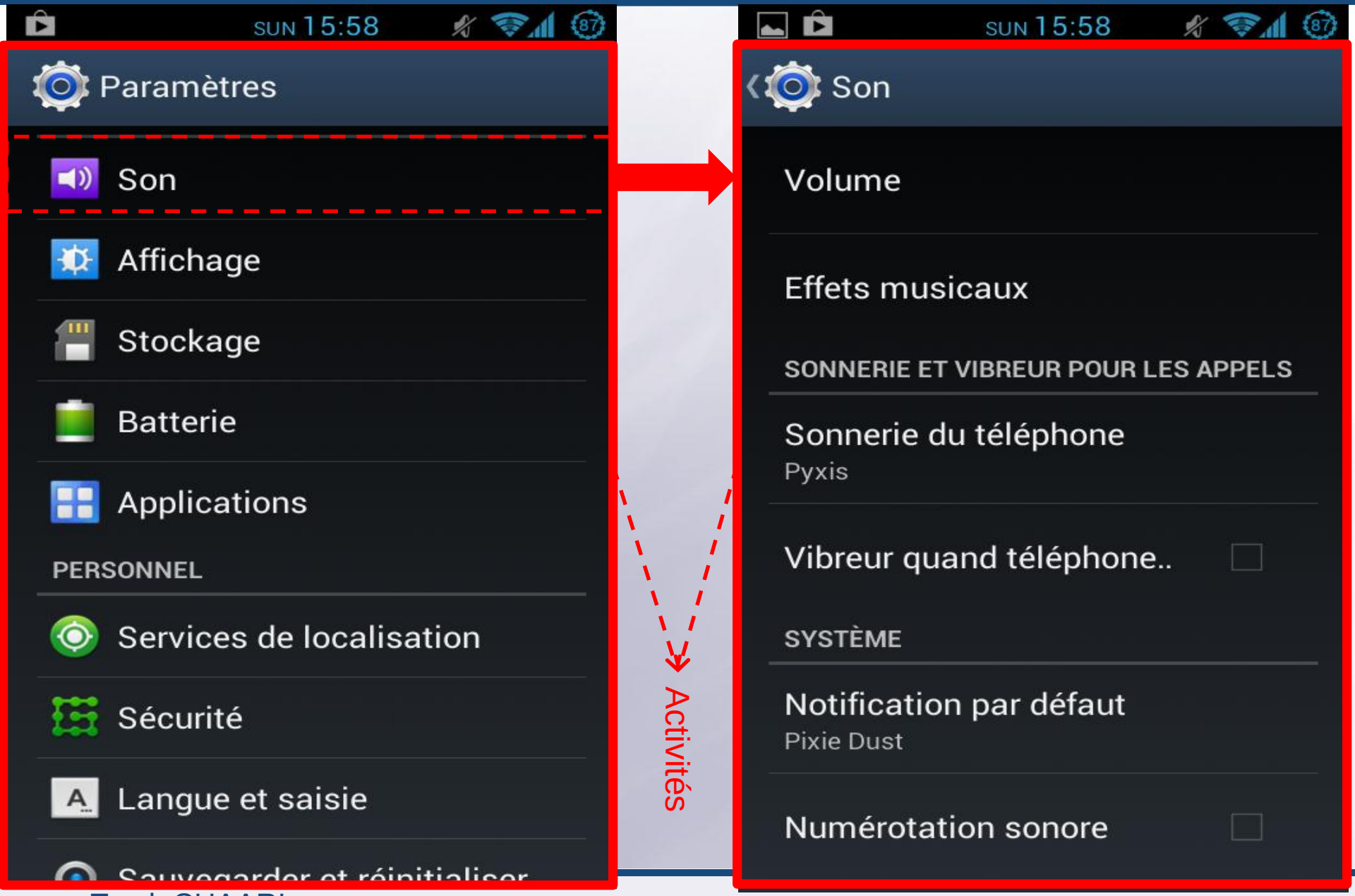
Components	Description
Activity	UI component typically corresponding to one screen
Service	Background process without UI
Broadcast Receiver	Component that responds to broadcast Intents
Content Provider	Component that enables applications to share data

- Ces composants doivent être décalrés dans un fichier xml (appelé manifest) associé à l'application

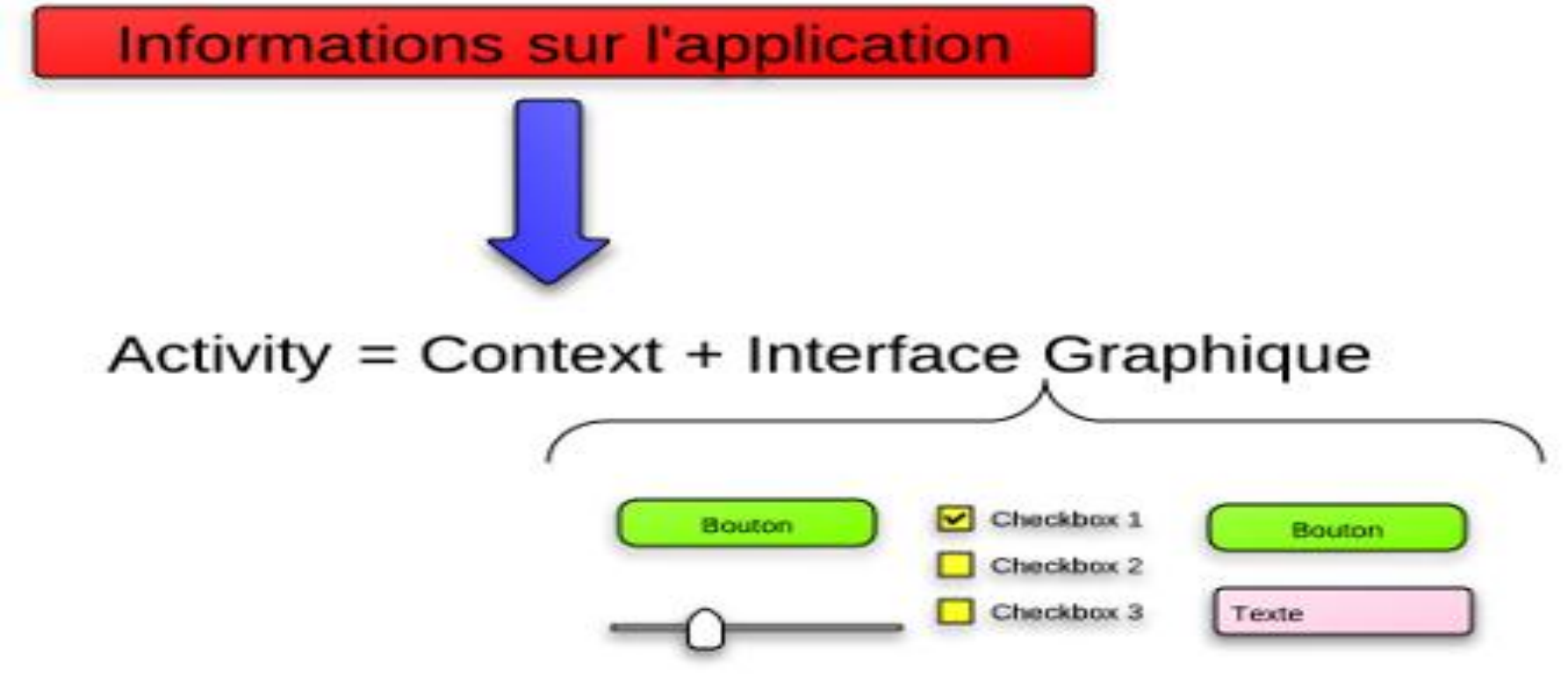
Applications Android

- une application graphique est un assemblage de fenêtres entre lesquelles il est possible de naviguer
- Ces différentes fenêtres sont appelées des activités
- comme une activité remplie tout l'écran, alors votre application ne peut en afficher qu'une à la fois
- Une activité est un ensemble de composants appelés vues (View)
- Elle contient aussi des informations sur l'application appelé contexte (Context)

Exemple d'une application Android : Paramètres

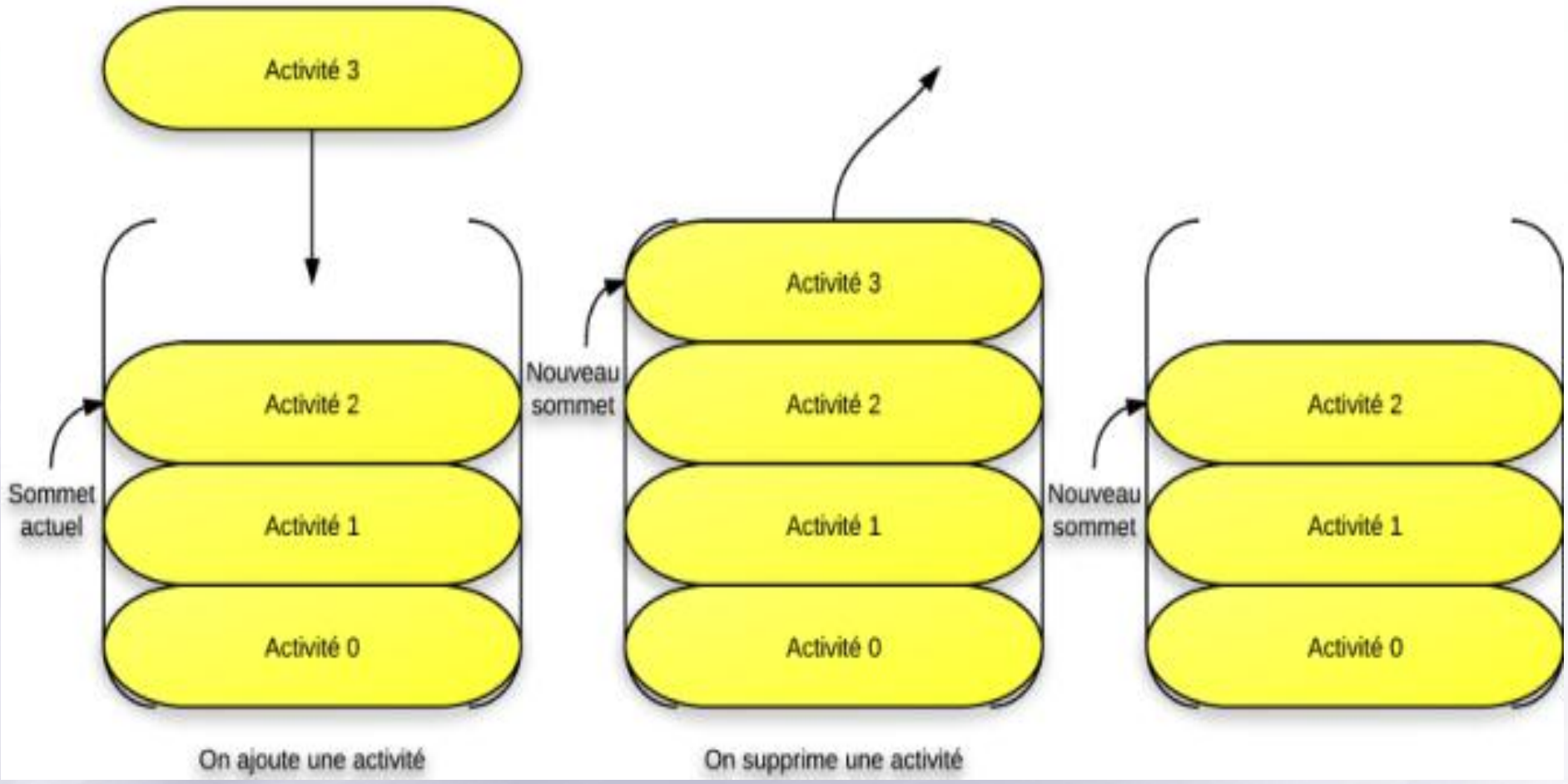


Activité



- Interface graphique + Contexte d'exécution
- En inversion de contrôle (callback)
  - Méthodes onStart()…. Déclenchées par le middleware
- Cycle de vie contrôlé par le système d'exploitation

Pile d'activités gérée par le système d'exploitation



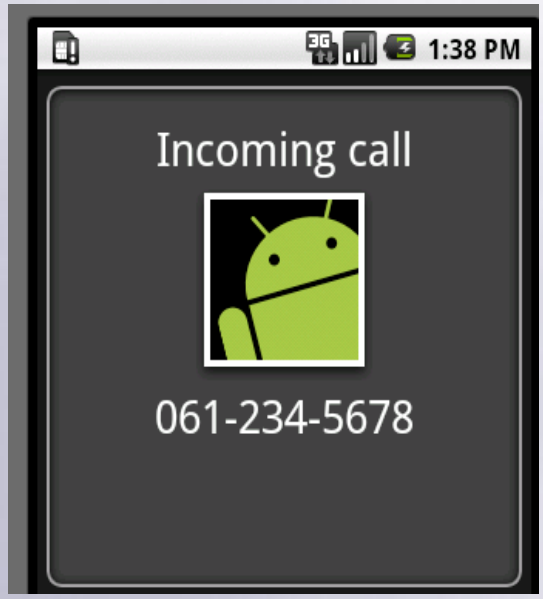
- **En tache de fond (onPause) :** Empiler(l'activité_courante);
- **Activité au 1er plan (onResume) :** activité_courante = Dépiler();

OnPause -> onResume (1/2)

1)

OnPause

2)

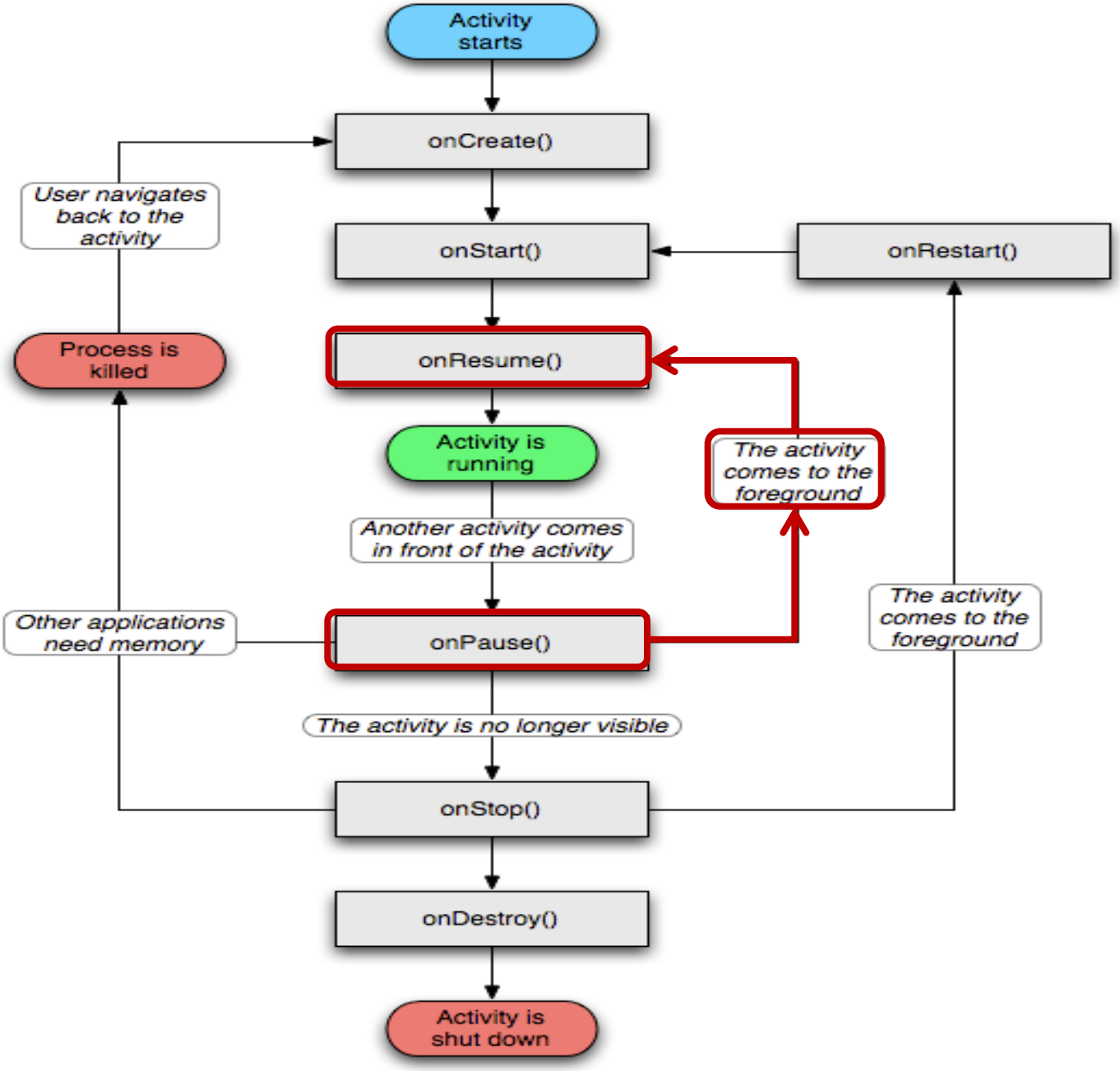


3)

OnResume

4)

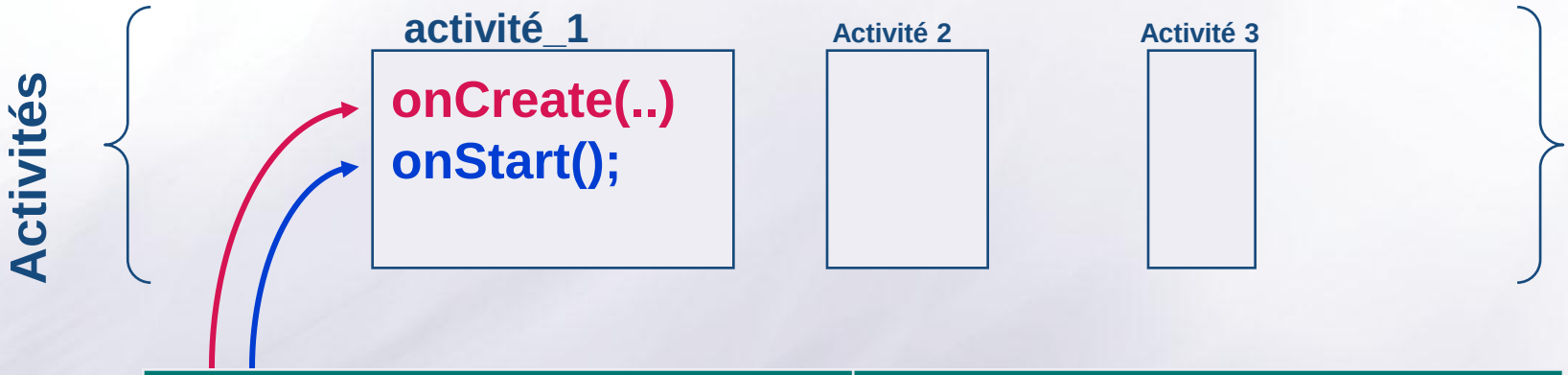
onPause → onResume (2/2)



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();
    }
```

Inversion de Contrôle (callback)… Rappel

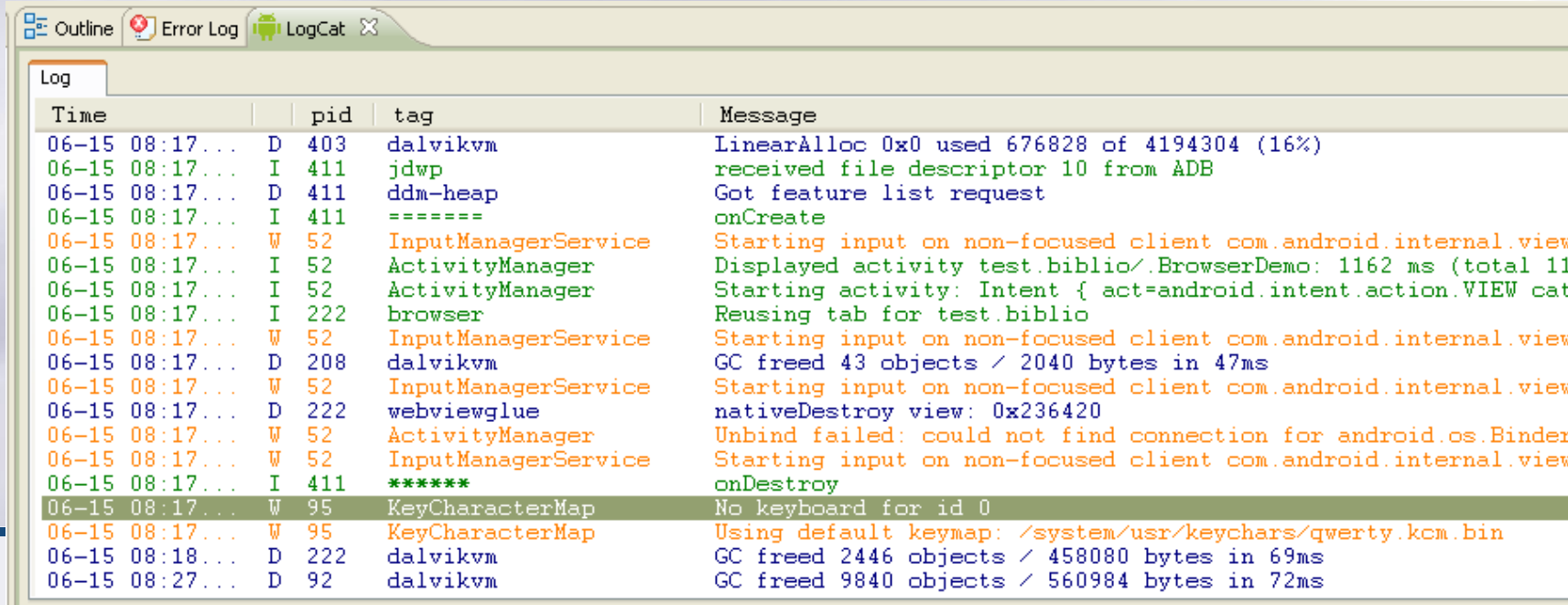


Développeur	Android OS
activité_1 = new Activité_1();	activité_1.onCreate();
startActivity(activité_1);	activité_1.onStart();

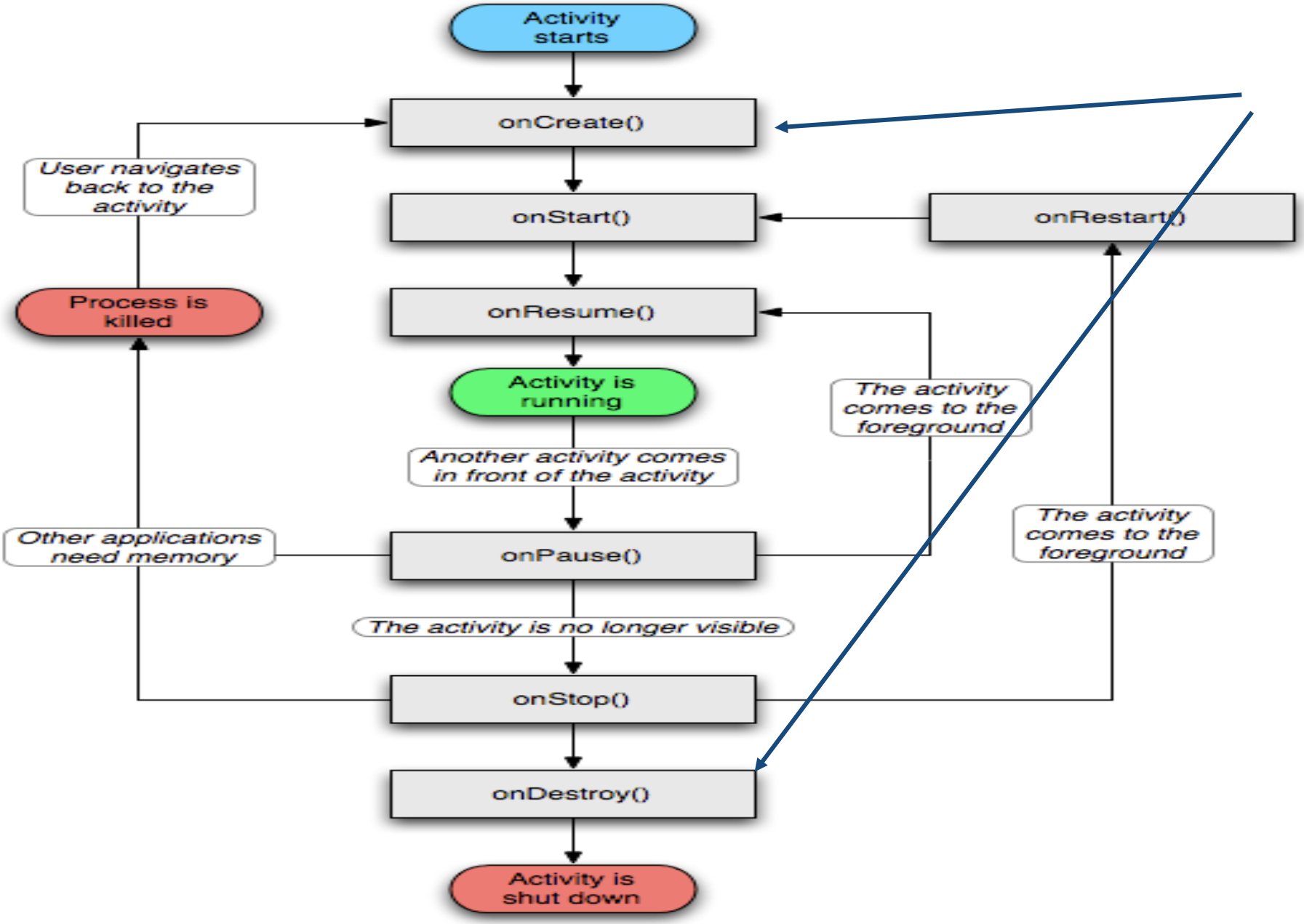
```
public class Activity extends ApplicationContext {
        protected void onCreate(Bundle savedInstanceState);
       protected void onStart();
        …..}
```

Démonstration, Activity dans tous ses états

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



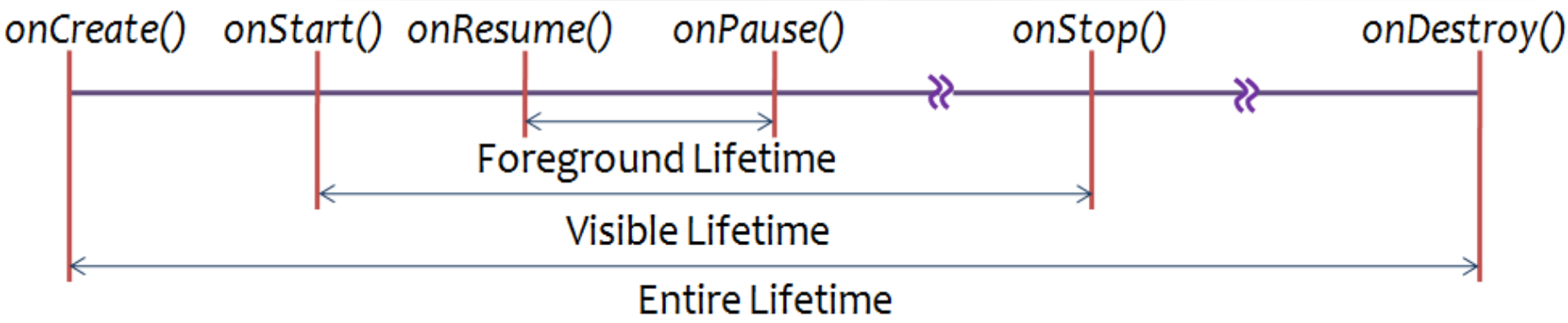
Emulateur (ou terminal) + LogCat : traces bien utiles



Démonstration, Activity dans tous ses états

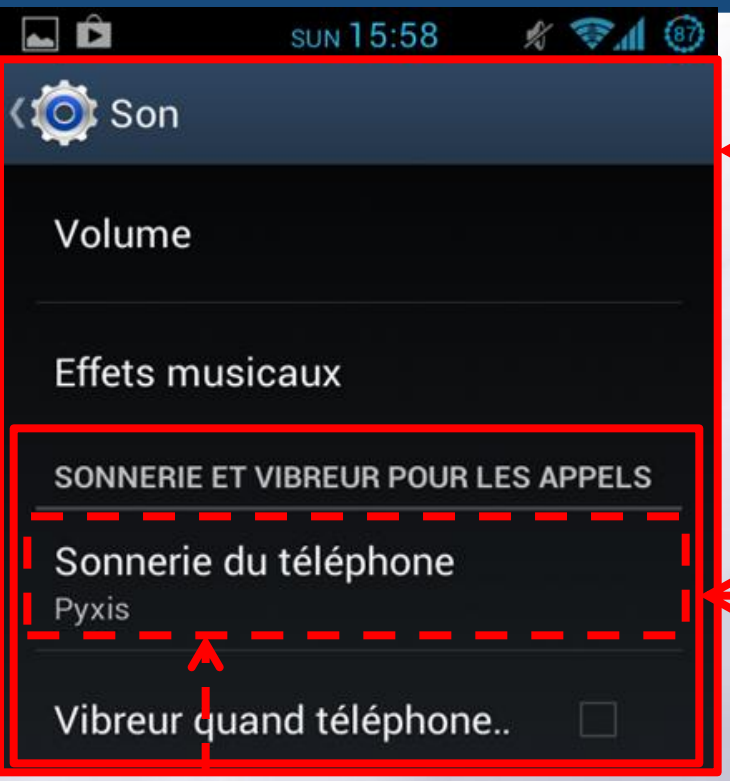
```
public class ActivityLifeCycle extends Activity {

  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");
    }
}
```

Activity : visibilité pour l'utilisateur et états

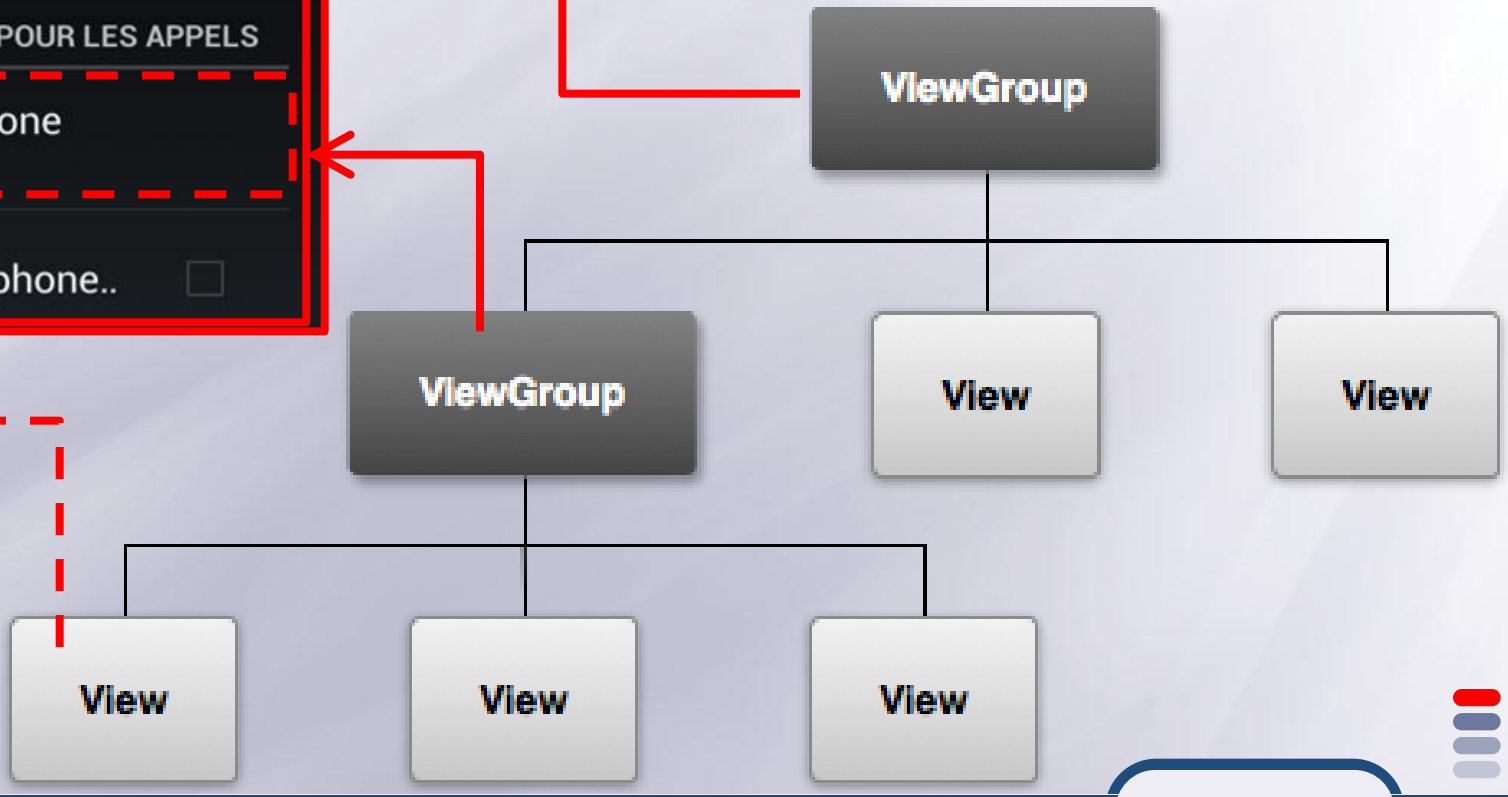


State	Description
Running	• L'activité est en premier plan de l'écran (A la tête de la pile des activités courante).
Paused	• L'activité a perdu le Focus puisqu'une autre vient de s'afficher au premier plan.
Stopped	• L'activité est arrêté mais elle est tjr présente en arrière plan

Une activité est composée d'un ensemble de View



Ceci est généralement défini en XML (détails plus tard)



Déclaration des activités dans le manifest de l'application

Package principal

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="isecs.cours.android.exemples"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:icon="@drawable/android_logo" android:label="@string/app_name">
        <activity android:name=".ActivityLifeCycle" android:label="@string/app_name"/>
    </application>
</manifest>
```

Permission pour réaliser des connexions internet

Nom de l'activité

ressources

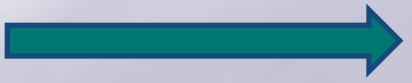
les Ressources Android

Les resources

- Un répertoire res est automatiquement créé lors de la création d'un projet android (en plus d'un répertoire src)

- /res
  - anim
  - drawable
    - xhdpi
    - hdpi
    - mdpi
    - ldpi
  - layout
  - values
    - arrays.xml
    - colors.xml
    - strings.xml
  - xml
  - raw

R.java



```
/* 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 cs454.demo;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    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;
    }
}
```

Types de ressources

Type	Description
Dessin et image (res/drawable)	Images, icones et fichiers XMlL de dessin simple (rectangles, cercles…)
Mise en page graphique (res/layout)	Disposition des composants dans les activités
Menu (res/menu)	Les fichiers XML pour pouvoir constituer des menus.
Donnée brute (res/raw)	Données diverses au format brut. Par exemple de la musique ou des fichiers HTML…
Différentes variables (res/values)	On y trouve entre autre des variables standards, comme des chaines de caractères, des dimensions, des couleurs, etc.

Les sous types de ressources

- **Dans chaque type de ressources on peut avoir des sous types (langue, taille de l'écran…)**
- **Ces sous typessont spécifiés par des quantificateur séparés par des "-"**
  - res/<type_de_ressource>[<-quantificateur 1>…<-quantificateur N>]
- **Exemples de quantificateurs :**
  - **Langues : fr,en, fr-rFR, fr-rCA…**
  - **Taille de l'écran : small, normal, large, xlarge**
  - **résolution de l'écran : ldpi (160 dpi), mdpi (160 dpi), hdpi (240 dpi), xhdpi (320 dpi)**
  - **orientation de l'écran : land (paysage), port (portrait)**
- **Les sous types les plus utilisés :**

  res/drawable-xhdpi, res/drawable-hdpi, res/drawable-ldpi, res/drawable-mdpi, res/layout-land, res/layout, res/values-fr-rFR ...

Les ressources (2/2)

- Les ressources sont utilisées de la manière suivante:
  - R.type_ressource.nom_ressource qui est de type int.
  - Il s'agit de l'identifiant de la ressource.
- On peut alors utiliser cet identifiant ou récupérer l'instance de la ressource en utilisant la classe Resources:
  - Resources res = getResources();
  - String hw = res.getString(R.string.hello);
  - XXX o = res.getXXX(id);
- Une méthode spécifique pour les objets graphiques permet de les récupérer à partir de leur id, ce qui permet d'agir sur ces instances même si elles ont été créées via leur définition XML:
  - TextView texte = (TextView)findViewById(R.id.le_texte);
  - texte.setText("Here we go !");

Référencement entre ressources

- On peut utiliser les ressources comme valeurs d'attributs dans d'autres ressources sous forme XML. Cette possibilité est très utilisée dans les mises en page par exemple.
- La notation pour faire référence à une autre ressource est la suivante :
  - attribute="@[package_name:]resource_type/resource_identifier"
- En voici l'utilisation au sein d'un exemple qui crée une mise en page sous forme de table (d'une ligne et de deux colonnes) dont les cellules sont remplies par des chaînes de caractères :

```
<?xml version="1.0" encoding="utf-8"?>
   <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1">
   <TableRow>
            <TextView android:text="@string/table_cellule_gauche" />
            <TextView android:text="@string/table_cellule_droite" />
   </TableRow>
</TableLayout>
```

Les chaînes

- Les chaînes constantes de l'application sont situées dans res/values/strings.xml.
- L'externalisation des chaînes permettra de réaliser l'internationalisation de l'application.
- Voici un exemple:

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

- La récupération de la chaîne se fait via le code:
  - Resources res = getResources();
  - String hw = res.getString(R.string.hello);

Internationalisation

- **Le système de ressources permet de gérer facilement l'internationalisation d'une application.**
  - Il suffit de créer des répertoires values-XX où XX est le code de la langue que l'on souhaite implanter.
  - On place alors dans ce sous répertoire le fichier xml strings.xml contenant les chaines traduites associées aux même clefs que dans values/strings.xml.
  - Android chargera le fichier de ressources approprié en fonction de la langue du système.
  - On obtient par exemple pour les langues es (espagnol) et fr (français) l'arborescence:

```
MyProject/
  res/
    values/
        strings.xml
    values-es/
        strings.xml
    values-fr/
        strings.xml
```

Autres valeurs simples

- Plusieurs fichies xml peuvent être placés dans res/values.
  - Cela permet de définir des chaines, des couleurs, des tableaux,...
- L'assistant de création permet de créer de nouveaux fichiers de ressources contenant des valeurs simples, comme par exemple un tableau de chaines:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <string-array name="test">
            <item>it1</item>
            <item>it2</item>
        </string-array>
    </resources>
```

Les couleurs

- Une couleur définit une valeur RVB (rouge, vert et bleu) et une transparence.
  - Cette couleur peut être utilisée à de multiples endroits comme pour définir la couleur d'un texte.
  - Il existe différents formats dont la syntaxe globale est la suivante :
    - <color name=nom_couleur>valeur_de_la_couleur</color>
- Exemple de déclaration d'une couleur

```
<resources>
    <color name="bleu_transparent">#50F00FF</color>
</resources>
```

  - Utilisation des couleurs (exemple avec la déclaration de la couleur bleu_transparent):
    - Java : R.color.bleu_transparent
    - XML : @[package:]color/bleu_transparent

Les dimensions

- Les dimensions sont la plupart du temps référencées dans les styles et les mises en page.
- Les unités prises en charge par Android : px (pixels), in (pouces), mm (millimètres), pt (points), dp (density-independant pixel), sp (scale-independant pixel).
  - dp : une unité relative se basant sur une taille physique de l'écran de 160 dpi.
    - Avec cette unité, 1 dp est égal à 1 pixel sur un écran de 160 pixels
    - Si la taille de l'écran est différente de 160 pixels, les vues s'adapteront selon le ratio entre la taille en pixels de l'écran de l'utilisateur et la référence des 160 pixels;
  - sp : fonctionne de la même manière que dp. Ils sont aussi fonction de la taille de polices spécifiée par l'utilisateur.

Les dimensions : Exemple

- Exemple de fichier de ressources de dimensions

```
<resources><dimen name="taille_texte">5sp</dimen></resources>
```

- Utilisation des dimensions :
  - Java : R.dimen.un_nom
    - Resources.getDimen(R.dimen.taille_texte);
  - XML : @dimen/un_nom

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
  android:textSize="@dimen/taille_texte"/>
```

Les images

- Android prend en charge les fichiers bitmap sous différents formats avec une préférence pour le format PNG.
  - Java : R.drawable.image1
  - XML : @[package:]drawable/image1

- Concernant les résolutions, chacune des catégories (low, medium, high) dispose d'un répertoire où enregistrer les ressources.

Les animations

- **La déclaration d'animations contient des possibilités de rotation, de fondu, de translation et de changements de taille. Les animations sont placées dans un ou plusieurs fichiers dans le répertoire res/anim.**
- Voici les quatre types d'animations proposées :

<alpha> : permet de faire un fondu ;

<scale> : définit un facteur d'échelle de départ et d'arrivée en X et Y ;

<translate> : permet de déplacer l'élément ;

<rotate> : propose une rotation avec un point pivot et un angle en degrés.

- **Android propose également des attributs permettant de nombreux réglages comme la durée, la répétition, etc.**
- **Exemple d'animation de type fondu avec la transparence qui passe de 0 à 1 :**

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="100" />
```

Les interfaces graphiques Android

Tout est basé sur XML

Réalisation d’interface utilisateur sous Android

UI construction can be done in three ways:

- Programmatic, like hand-coded Java desktop GUI construction
- Declarative hand-written, like Java web UI construction
  - XML
- Declarative with a GUI builder, like .NET UI construction
  - GUI builder generates the XML

Par programmation

```
package cs454.demo;

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

public class AndroidDemo extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

// Activity is a subclass of context, so the TextView takes this as a parameter
        TextView tv = new TextView(this);
        tv.setText("Hello, CS454");
        setContentView(tv);
    }
}
```

Par déclaration manuelle XML

main.xml Layout File:

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

strings.xml resource file:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello Again, CS454!</string>
    <string name="app_name">CS454 AndroidDemo</string>
</resources>
```

Par déclaration manuelle XML (suite)

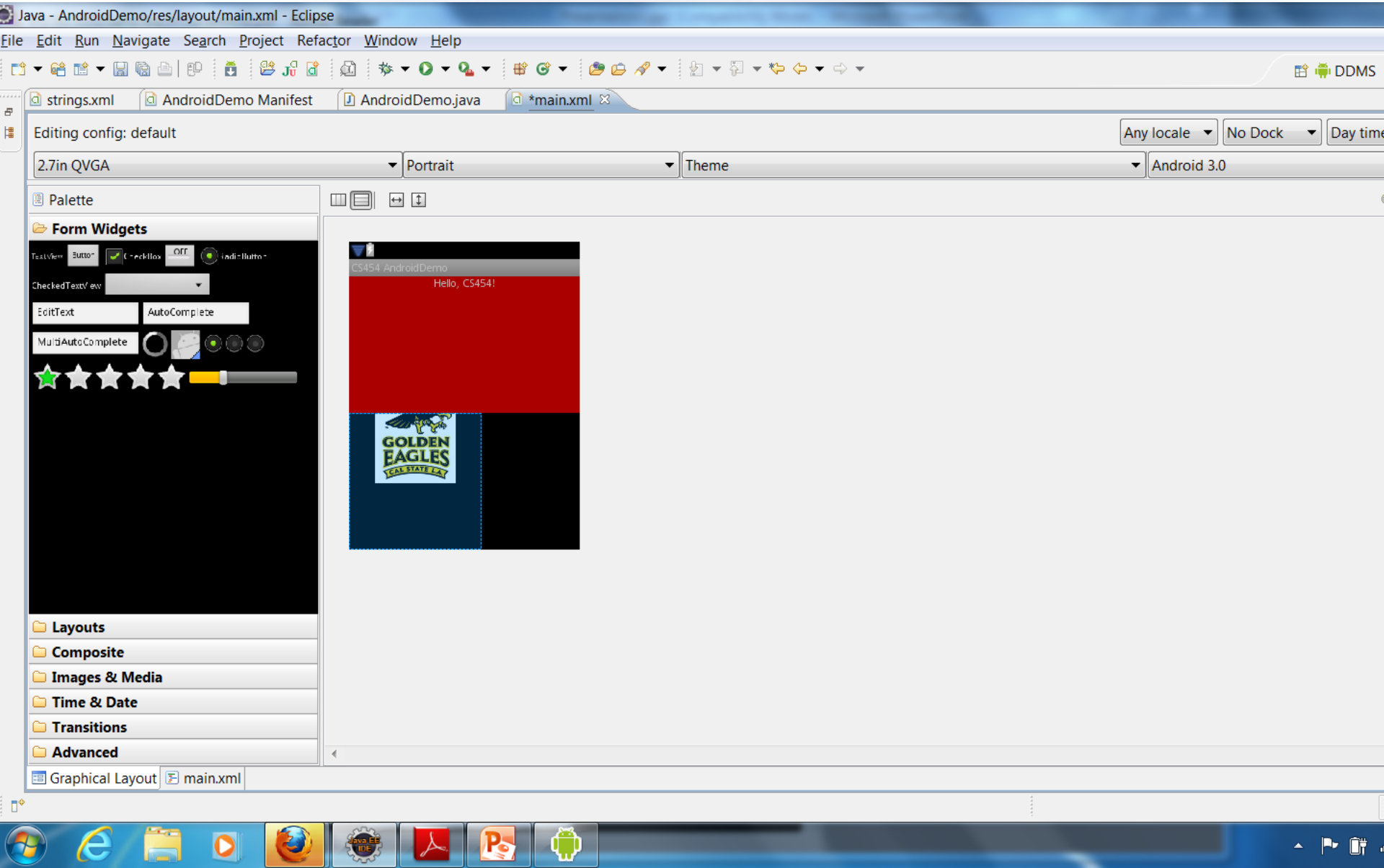
Java class:

```
package cs454.demo;
import android.app.Activity;
import android.os.Bundle;

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

What's R?

```
package cs454.demo;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    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;
    }
}
```

Par décalaration XML avec assistant graphique



Gestion des évènements

From the code file for the activity:

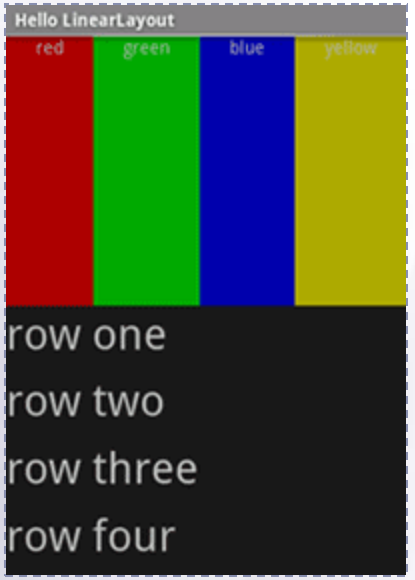
```
Button ok = (Button) findViewById(R.id.button1);
ok.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
                CharSequence s = et.getText();
                tv.setText("Welcome, " + s);
        }
});
```

Elements and layouts

Linear Layout

- Shows nested View elements

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



Elements and layouts

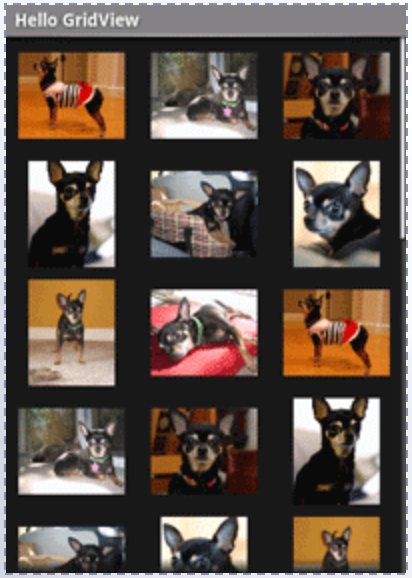
Table Layout

- Like the HTML `div` tag

```
/* table.xml */
<?xml version="1.0" encoding="utf-8"?>
<TableLayout android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:stretchColumns="1">
   <TableRow>
      <TextView android:layout_column="1"
         android:text="Open..."
         android:padding="3dip" />
      <TextView android:text="Ctrl-O"
         android:gravity="right"
         android:padding="3dip" />
   </TableRow>
</TableLayout>
```

Grid View

```
/* grid.xml */
<?xml version="1.0" encoding="utf-8"?>
<GridView
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```



Grid View

```
/* GridExample.java */
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.grid);

    GridView gridview = (GridView) findViewById(R.id.gridview);
    gridview.setAdapter(new AdapterForGridView(this));

    gridview.setOnItemClickListener(
        new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View v,
                                                    int pos, long id) {
              Toast.makeText(
                    GridPrimer.this, "" + pos, Toast.LENGTH_SHORT).show();
        }});
}
```

Grid View

```
/* AdapterForGridView.java */
public class AdapterForGridView extends BaseAdapter {
    private Context mContext;
    public AdapterForGridView(Context c) { mContext = c; }
    public int getCount() { return mThumbIDs.length; }
    public Object getItem(int position) { return null;}
    public long getItemId(int position) { return 0; }

    // bad getView implementation
    public View getView(int pos, View convertView, ViewGroup parent) {
        ImageView imageView = new ImageView(mContext);
        imageView.setImageResource(mThumbIDs[pos]);
        return imageView;
    }
    private Integer[] mThumbIDs =
                    { R.drawable.img1, R.drawable.img2 /*...*/ };
}
```

Tab Layout

```
/* tab.xml */
<?xml version="1.0" encoding="utf-8"?>
<TabHost android:id="@android:id/tabhost"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <LinearLayout android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
        <TabWidget android:id="@android:id/tabs"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"/>
        <FrameLayout
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"/>
   </LinearLayout>
</TabHost>
```

Tab Layout

```
/* selector1.xml */
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <!- Tab is selected -->
   <item android:drawable="@drawable/ic_tab_1_selected"
         android:state_selected="true" />
   <!- Tab not selected -->
   <item android:drawable="@drawable/ic_tab_1_not_selected" />
</selector>
```

```
/* selector2.xml */
```

```
/* selector3.xml */
```

Tab Layout

```
/* Tab1.java */
public class Tab1 extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView textview = new TextView(this);
        textview.setText("This is the Artists tab");
        setContentView(textview);
    }
}
```

```
/* Tab2.java */
```

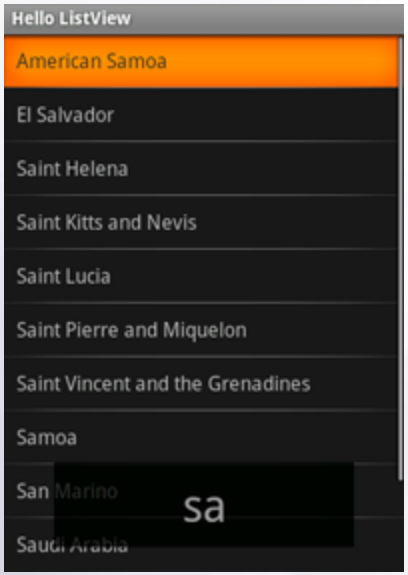
```
/* Tab3.java */
```

Tab Layout

```
/* TabExample.java */
public class TabExample extends TabActivity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.tab);
      TabHost tabHost = getTabHost();
      //--- tab 1 ---
      Intent intent = new Intent().setClass(this, Tab1.class);
      TabHost.TabSpec spec = tabHost.newTabSpec("tab1").setIndicator(
         "Artists", getResources().getDrawable(R.drawable.selector1))
      .setContent(intent);
      tabHost.addTab(spec);
      //--- tab 1 ---
      tabHost.setCurrentTab(2);
}
```

List View

```
/* list_item.xml */
<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp" />
```

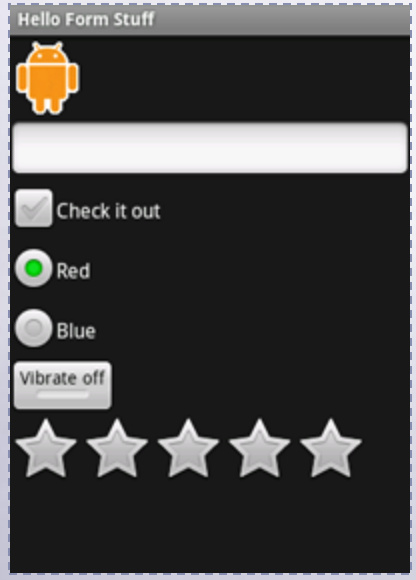


List View

```
/* ListViewExample.java */
public class ListViewExample extends ListActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(this,
                                        R.layout.list_item, COUNTRIES));
        ListView lv = getListView();
        lv.setTextFilterEnabled(true);
        lv.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View view,
                                                int position, long id) {
                Toast.makeText(getApplicationContext(),
                 ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
        }});
}
```

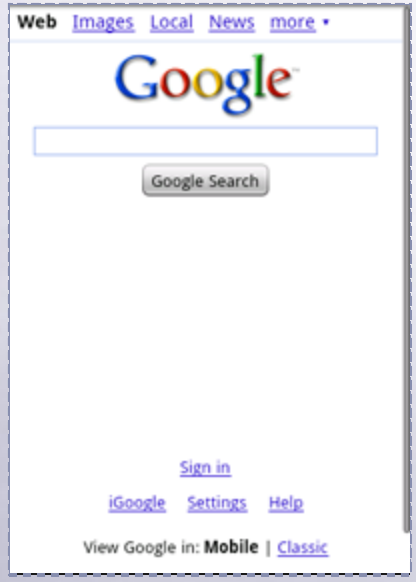
Elements and layouts

- Button
- ImageButton
- EditText
- CheckBox
- RadioButton
- ToggleButton
- RatingBar



Elements and layouts

- DatePicker
- TimePicker
- Spinner
- AutoComplete
- Gallery
- MapView
- WebView



Events

- Event Handler
  - Hardware buttons
- Event Listener
  - Touch screen

Events

- KeyEvent is sent to callback methods
  - onKeyUp(), onKeyDown(), onKeyLongpress()
  - onTrackballEvent(), onTouchEvent()

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_CAMERA) {
          return true; // consumes the event
    }
    return super.onKeyDown(keyCode, event);
}
```

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) { /* ... */ }
});
```

Events

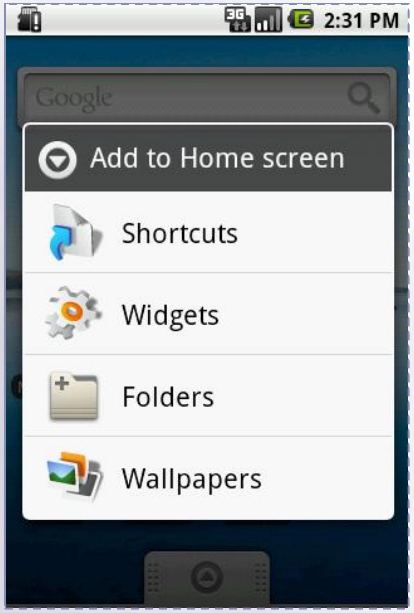
```
public class TouchExample extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) { /*...*/ }
        });
        button.setOnLongClickListener(new OnLongClickListener() {
            public boolean onLongClick(View v) {
                // ...
                return true;
            }
        });
    }
}
```

Menus

- Options Menu: MENU button, tied to an Activity
- Context Menu: View LongPress
- Submenu

```
public void onCreate(Bundle savedInstanceState) {
   registerForContextMenu((View)findViewById(/*...*/));
}
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuInfo menuInfo){
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, MENU_SMS, 0, "SMS");
    menu.add(0, MENU_EMAIL, 0, "Email");
}
public boolean onContextItemSelected(MenuItem item) {
    switch(item.getItemId()) { case MENU_SMS: /*...*/ }
}
```

```
    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, MENU_ADD, 0, "Add")
            .setIcon(R.drawable.icon);
        menu.add(0, MENU_WALLPAPER, 0, "Wallpaper");
        return super.onCreateOptionsMenu(menu);
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch(item.getItemId()) {
            case MENU_ADD: //... ; return true;
            case MENU_WALLPAPER: //... ; return true;
            default: return false;
        }
    }
```



les intents

Aperçu sur les Intents (les intentions)

- Un intent est un message simple envoyé à d'autres applications

- Chaque Intent est composé de:
  - ✓Une action à effectuer (MAIN, VIEW, EDIT, PICK, DELETE, DIAL, etc)
  - ✓Contenu concerné (identifié par un URI)

- Exemples

```
startActivity(new Intent(Intent.VIEW_ACTION, Uri.parse("http://www..google.fr"));

startActivity(new Intent(Intent.VIEW_ACTION, Uri.parse("geo:47.480843,8.211293"));

startActivity(new Intent(Intent.EDIT_ACTION,Uri.parse("content://contacts/people/1"));
```

Aperçu sur les Intent Filters

- Un intent filter permet de spécifier une capacité de répondre à un un type d'intents.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
    <application . . . >
        <activity android:name="com.example.project.FreneticActivity"
                  android:icon="@drawable/small_pic.png"
                  android:label="@string/freneticLabel"
                  . . .  >
            <intent-filter . . . >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter . . . >
                <action android:name="com.example.project.BOUNCE" />
                <data android:mimeType="image/jpeg" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        . . .
    </application>
</manifest>
```

- Le premier intent filter indique que l'activité sera exécutable (1ère activité lors de l'exécution)
- Le second filtre déclare une action que l'activity peut réaliser sur un type particulier de données

Principe des Intents

- Les Intents permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications.
- Le but des Intents est de déléguer une action à un autre composant, une autre application ou une autre activité de l'application courante.
- Un objet Intent contient les information suivantes:
  - le nom du composant ciblé (facultatif)
  - l'action à réaliser, sous forme de chaîne de caractères
  - les données: contenu MIME et URI
  - des données supplémentaires sous forme de paires clé/valeur
  - une catégorie pour cibler un type d'application
  - des drapeaux (informations supplémentaires)
- On peut envoyer des Intents informatifs pour faire passer des messages.
- On peut aussi envoyer des Intents servant à lancer une nouvelle activité.

Fonctionnement des intents

- Les Intents ont essentiellement trois utilisations :
  - Démarrer une activité au sein de l'application courante ou de solliciter d'autres applications et d'envoyer des informations.
  - Démarrer une activité au sein d'une même application est utilisé pour la navigation entre écrans d'une interface graphique et l'appel d'une boîte de dialogue.
    - C'est le seul cas où on doit démarrer une activité en mode explicite.
  - Solliciter une autre application pour répondre à un besoin (qui peut être l'exécution d'une action ou bien la transmission d'informations).
    - On se contenterde transmettre son intention au système qui, lui, va se charger trouver l'application et le composant le plus approprié puis démarrer ce dernier et lui transmettre l'Intent correspondant.

Fonctionnement des intents

- Les Intents ont aussi d'autres utilisations, dont le démarrage d'un service. Le mécanisme relatif aux objets Intent et leur utilisation sont en effet indispensables pour les applications fonctionnant en arrière plan (telles que les services) afin de recevoir des actions à effectuer et de pouvoir communiquer avec d'autres applications.
- Il est aussi possible de vouloir diffuser un objet Intent à plusieurs applications (par exemple pour informer l'ensemble des applications ouvertes que la batterie est défaillante).
  - les applications peuvent mettre en place un filtre permettant de ne conserver que les Intents que l'application juge nécessaires.

Naviguer entre écrans

- Une application est souvent composée de plusieurs écrans qui s'enchaînent les uns à la suite des autres en fonction de l'utilisateur et chaque écran est représenté par une activité définissant son interface utilisateur et sa logique.
  - La principale utilisation d'un Intent est le démarrage de ces activités (une à la fois) permettant cet enchaînement.
  - De façon plus générale, chaque composant de l'application nécessitera l'emploi d'un Intent pour être démarré.

- Il existe deux méthodes pour démarrer une activité, en fonction de la logique de l'interface : parfois on a besoin de savoir comment s'est déroulée l'activité (et obtenir un retour lors de son arrêt), parfois non.

Intents pour une nouvelle activité

- Il y a plusieurs façons de créer l'objet de type Intent qui permettra de lancer une nouvelle activité.

- Si l'on passe la main à une activité interne à l'application, on peut créer l'Intent et passer la classe de l'activité ciblée par l'Intent:

```
Intent login = new Intent(this, GiveLogin.class);
startActivity(login);
```

- Le premier paramètre de construction de l'Intent est en fait le contexte de l'application.
  - Dans certain cas, il ne faut pas mettre this mais faire appel à getApplicationContext() si l'objet manipulant l'Intent n'hérite pas de Context.

Intents pour une nouvelle activité (suite)

- S'il s'agit de passer la main à une autre application, on donne au constructeur de l'Intent les données et l'URI cible: le SE est chargé de trouver une application pouvant répondre à l'Intent.

```
Button b = (Button)findViewById(R.id.Button01);
b.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
  Uri telnumber = Uri.parse("tel:0248484000");
  Intent call = new Intent(Intent.ACTION_DIAL, telnumber);
  startActivity(call);}});
```

- Sans oublier de déclarer la nouvelle activité dans le Manifest.

Retour d'une activité (1/4)

- Lorsque le bouton retour est pressé, l'activité courante prend fin et revient à l'activité précédente.
  - Cela permet par exemple de terminer son appel téléphonique et de revenir à l'activité ayant initié l'appel.
- Au sein d'une application, une activité peut vouloir récupérer un code de retour de l'activité "enfant".
  - On utilise pour cela la méthode startActivityForResult qui envoie un code de retour à l'activité enfant.

Retour d'une activité (2/4)

- Lorsque l'activité parent reprend la main, il devient possible de filtrer le code de retour dans la méthode onActivityResult pour savoir si l'on revient ou pas de l'activité enfant.

- L'appel d'un Intent devient donc:

```
Intent login = new Intent(getApplicationContext(), GivePhoneNumber.class);
startActivityForResult(login,48);}
```

- Le filtrage dans la classe parente permet de savoir qui avait appelé cette activité enfant:

```
public void onCreate(Bundle savedInstanceState) {
protected void onActivityResult(int requestCode, int resultCode, Intent data){
  if (requestCode == 48)
    Toast.makeText(this, "Code de requête récupéré (je sais d'où je viens)",
     Toast.LENGTH_LONG).show();}
```

Retour d'une activité (3/4)

- Il est aussi possible de définir un résultat d'activité, avant d'appeler explicitement la fin d'une activité avec la méthode finish().
  - Dans ce cas, la méthode setResult permet d'enregistrer un code de retour qu'il sera aussi possible de filtrer dans l'activité parente.
- Dans l'activité enfant, on met donc:

```
Button finish = (Button)findViewById(R.id.finish);
 finish.setOnClickListener(new OnClickListener() {
 @Override
 public void onClick(View v) {
  setResult(50);
  finish();}});
```

Retour d'une activité (4/4)

- Et la classe parente peut filtrer ainsi:

```
protected void onActivityResult(int requestCode, int resultCode, Intent
    data){
  if (requestCode == 48)
    Toast.makeText(this, "Code de requête récupéré (je sais d'ou je
    viens)", Toast.LENGTH_LONG).show();
  if (resultCode == 50)
    Toast.makeText(this, "Code de retour ok (on m'a renvoyé le bon code)",
    Toast.LENGTH_LONG).show();}
```

Solliciter d'autres applications (1/4)

- On utilise des objets Intent pour démarrer des activités (via le nom du type d'activité) au sein d'une même application

- L'envoi d'un Intent permet également de demander à un composant d'une autre application de traiter l'action souhaitée.
  - C'est le système qui décide alors de l'application à utiliser
  - Pour décider du composant le plus approprié, le système se base sur les informations qu'on spécifie dans l'objet Intent : action, données, catégorie, etc.

Solliciter d'autres applications (2/4)

- Ainsi on exprime l'intention au système et le système se chargera de la résoudre en proposant le composant de l'application le plus approprié.
  - Ce mécanisme permet d'éviter les dépendances vers des applications puisque l'association entre l'application et le composant nécessaire se fait au moment de l'exécution et non de la compilation.

- Cette utilisation d'Intent est implicite puisque le système doit résoudre celle-ci en fonction de son environnement.
  - Il recourt à des filtres comme points d'entrée pour distribuer les intents aux composants les plus appropriés.
  - Les informations qui sont utilisées pour la résolution sont : l'action, les données (l'URI et le type de contenu MIME) et la catégorie.
  - Les autres données ne jouent pas de rôle dans la résolution et la distribution des Intent aux applications.

Solliciter d'autres applications (3/4)

- Exemple : composer un numéro de téléphone.
  - On doit utiliser le type d'action ACTION_DIAL permettant de demander au système de composer un numéro.
  - Pour pouvoir demander au système de réaliser cette action, on crée un nouvel objet Intent dont le type d'action et les valeurs complémentaires sont spécifiées dans le constructeur ; et cela sans préciser le type de la classe ciblée.
  - Puis on démarre une nouvelle activité en spécifiant cet objet Intent. À la charge du système, en fonction des filtres d'Intents déclarés par les applications, de déterminer à quel composant d'une quelconque application envoyer cette demande d'action.
  - Pour appeler un numéro de téléphone, l'intention à envoyer au système sera composée d'une action et d'un URI comportant le numéro à appeler.

Solliciter d'autres applications (4/4)

- Le système déterminera quelle application ouvrir (par défaut l'interface de composition de numéro)
- **Code de l'exemple:**

```
Uri uri = Uri.parse("tel:74111111");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);
```

- Un URI (Uniform ressource Identifier) est un identifiant unique permettant d'identifier une ressource de façon non ambigüe sur un réseau.

Ajouter des informations

- Les Intents permettent de transporter des informations à destination de l'activité cible.
  - On appelle ces informations des Extra
  - Les méthodes permettant de les manipuler sont getExtra et putExtra.

- Lorsqu'on prépare un Intent et que l'on souhaite ajouter une information de type "clé -> valeur", on procède ainsi:

```
Intent callactivity2 = new Intent(getApplicationContext(), Activity2.class);
callactivity2.putExtra("login", "jfl");
startActivity(callactivity2);
```

- Du côté de l'activité recevant l'Intent, on récupère l'information de la manière suivante:

```
Bundle extras = getIntent().getExtras();
String s = new String(extras.getString("login"));
```

Types d'Intent: actions

- Le premier paramètre de construction de l'Intent est le type de l'action véhiculé par cet Intent.
  - Ces types d'actions peuvent être les actions natives du système ou des actions définies par le développeur.

- Plusieurs actions natives existent par défaut sur Android. La plus courante est l'action Intent.ACTION_VIEW qui permet d'appeler une application pour visualiser un contenu dont on donne l'URI.

Types d'Intent: actions (suite)

- Voici un exemple d'envoi d'email.
  - Dans ce cas où l'on utilise l'action native, il faut ajouter des informations supplémentaires à l'aide de putExtra:

```
Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
String[] recipients = new String[]{"my@email.com", "",};
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, recipients);
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Test");
emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, "Message");
emailIntent.setType("text/plain");
startActivity(Intent.createChooser(emailIntent, "Send mail..."));
finish();
```

- Pour définir une action personnelle, il suffit de créer une chaîne unique:

```
Intent monIntent = new Intent("andro.jf.nom_du_message");
```

Types d'Intent: catégories

- Les catégories d'Intent permettent de grouper les applications par grands types de fonctionnalités (clients emails, navigateurs, players de musique, etc...).

- Par exemple, on trouve les catégories suivantes qui permettent de lancer:
  - DEFAULT: catégorie par défaut
  - BROWSABLE: une activité qui peut être invoquée depuis un clic sur un navigateur web, ce qui permet d'implémenter des nouveaux types de lien, e.g. foo://truc
  - APP_MARKET: une activité qui permet de parcourir l'Android market de télécharger des applications
  - APP_MUSIC: une activité qui permet de parcourir et jouer de la musique

Diffuser des informations

- Il est aussi possible d'utiliser un objet Intent pour diffuser (broadcaster) un message à but informatif.
  - Ainsi, toutes les applications pourront capturer ce message et récupérer l'information.
  - La méthode putExtra permet d'enregistrer une paire clé/valeur dans l'Intent.

```
Intent broadcast = new Intent("andro.jf.broadcast");
broadcast.putExtra("extra", "test");
sendBroadcast(broadcast);
```

- On peut récupérer les données à l'aide de la méthode getExtras dans l'objet Bundle qui est dans l'Intent:

```
Bundle extra = intent.getExtras();
String val = extra.getString("extra");
```

Recevoir et filtrer les Intents

- Etant donné la multitude de messages véhiculés par des Intents, chaque application doit pouvoir facilement "écouter" les Intents dont elle a besoin.
- Android dispose d'un système de filtres déclaratifs permettant de définir dans le Manifest des filtres.
- Un filtre peut utiliser plusieurs niveaux de filtrage:
  - action: identifie le nom de l'Intent. Pour éviter les collisions, il faut utiliser la convention de nommage de Java
  - category: permet de filtrer une catégorie d'action (DEFAULT, BROWSABLE, ...)
  - data: filtre sur les données du message par exemple en utilisant android:host pour filtrer un nom de domaine particulier

Filtrage d'un Intent par l'activité

- En déclarant un filtre au niveau du tag activity, l'application déclare les types de messages qu'elle sait gérer et qui l'invoquent.

```
<activity android:name=".Main" android:label="@string/app_name">
  <intent-filter>
    <action android:name="andro.jf.nom_du_message" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

- Ainsi, l'application répond à la sollicitation des Intents envoyés par:

```
Button autoinvoc = (Button)findViewById(R.id.autoinvoc);
autoinvoc.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
  Intent intent = new Intent("andro.jf.nom_du_message");
  startActivity(intent);}});
```

Utilisation de catégories pour le filtrage

- Les catégories d'Intent permettent de répondre à des événements dans un contexte particulier.
  - L'exemple le plus utilisé est l'interception d'un clic lors d'une navigation sur le web ou un protocole particulier est prévu.

- Pour par exemple répondre à un clic sur un lien http://www.isecs.rnu.tn, il faut construire un filtre d'Intent utilisant la catégorie définissant ce qui est "navigable" tout en combinant cette catégorie avec un type d'action signifiant que l'on souhaite "voir" la ressource.

Utilisation de catégories pour le filtrage (suite)

- Il faut en plus utiliser le tag data dans la construction du filtre:

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:scheme="http" android:host="www.isecs.rnu.tn"/>
</intent-filter>
```

•L'attribut host définit l'URI de l'autorité (on évite ainsi le fishing). L'attribut scheme définit la partie de gauche de l'URI.

•On peut ainsi lancer une activité lors d'un clic sur:

```
<a href="http://www.isimsf.rnu.tn">lien</a>
```

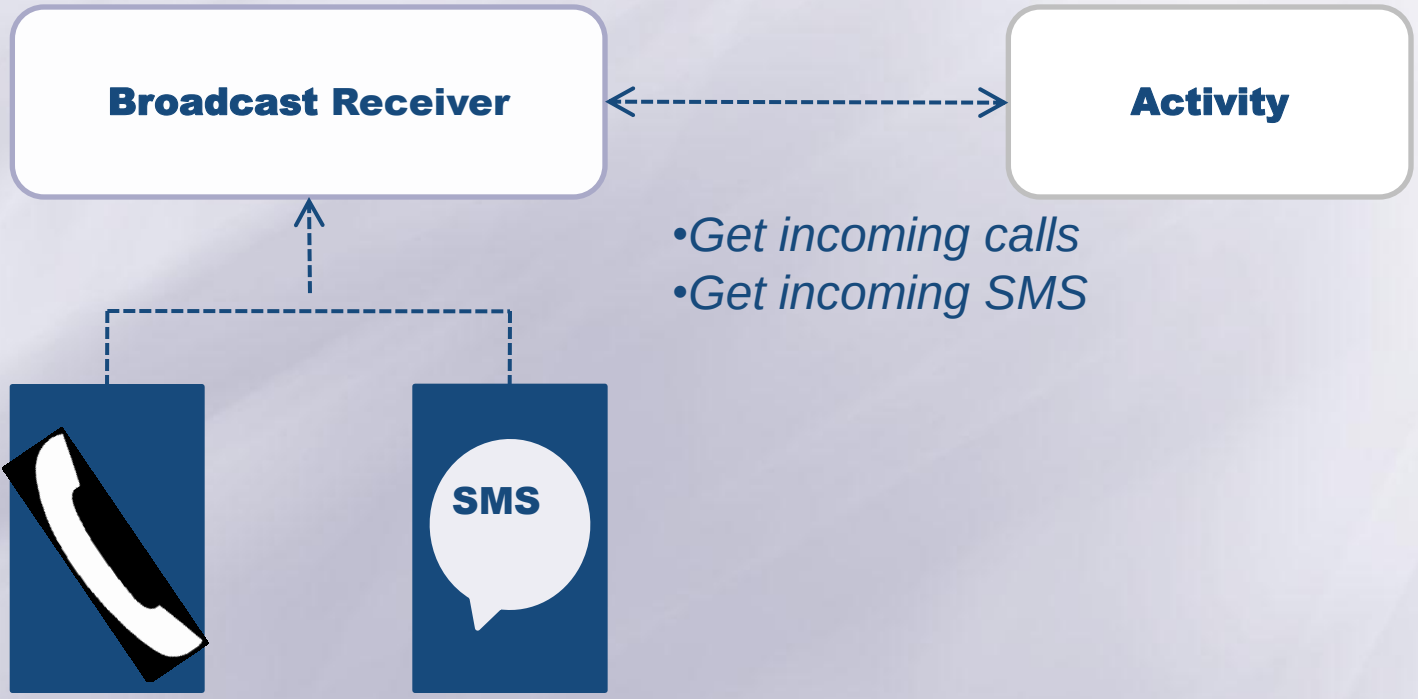
Les messages natifs

- Un certain nombre de messages sont diffusés par le SE:
  - ACTION_BOOT_COMPLETED: diffusé lorsque le système a fini son boot
  - ACTION_SHUTDOWN: diffusé lorsque le système est en cours d'extinction
  - ACTION_SCREEN_ON / OFF: allumage / extinction de l'écran
  - ACTION_POWER_CONNECTED / DISCONNECTED: connexion / perte de l'alimentation
  - ACTION_TIME_TICK: une notification envoyée toutes les minutes
  - ACTION_USER_PRESENT: notification reçue lorsque l'utilisateur déverrouille son téléphone
  - ...
- D'autres actions permettent de lancer des applications tierces pour déléguer un traitement:
  - ACTION_CALL (ANSWER, DIAL): passer/réceptionner/afficher un appel
  - ACTION_SEND: envoyer des données par SMS ou E-mail
  - ACTION_WEB_SEARCH: rechercher sur internet

Réception d'intents par un Broadcast Receiver

- Un "broadcast receiver" est un componsant qui reçoit et réagit à des événements appelés intentions (Intents)
  - ✓Beaucoup de ces évènements viennent du système

Par exemple, la batterie est faible, Réseaux wifi disponibles, téléchargement terminé…



Exemple d'un BroadcastReceiver

```
package com.tuto.android;


public class SMSReceiver extends BroadcastReceiver
{
 private final String          ACTION_RECEIVE_SMS  =
"android.provider.Telephony.SMS_RECEIVED";
 public void onReceive(Context context, Intent intent)
 {
  if (intent.getAction().equals(ACTION_RECEIVE_SMS))
  {
   Bundle bundle = intent.getExtras();
   if (bundle != null)
   {
    Object[] pdus = (Object[]) bundle.get("pdus");
    final SmsMessage[] messages = new SmsMessage[pdus.length];
```

Exemple d'un BroadcastReceiver

```
for (int i = 0; i < pdus.length; i++)
        {  messages[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
          Toast.makeText(context, "Expediteur : " + messages[i].getDisplayOriginatingAddress(), Toast.LENGTH_LONG).show();
    Toast.makeText(context, "Message : " + messages[i].getMessageBody(), Toast.LENGTH_LONG).show();
        }
```

- Dans le manifest
  - Déclaration du Broad castReceiver

```
<receiver
class="com.tuto.android.SMSReceiver"android:name="com.tuto.android.SMSReceiver">
    <intent-filter> <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

  - Permissions nécessaires

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

les Services et les contents providers

Composants principaux : Les services

- Un service est un composant qui n'a pas d'interface graphique
- Il tourne en tâche de fond pour une période généralement indéterminée

  Exemple: music player, Téléchargement, etc
- Un service Android hérite de android.app.Service
- La communication avec le service se fait à l'aide du langage AIDL (Android Interface Definition Language).

Cycle de vie d'un service Android

Deux façons d'utiliser un service

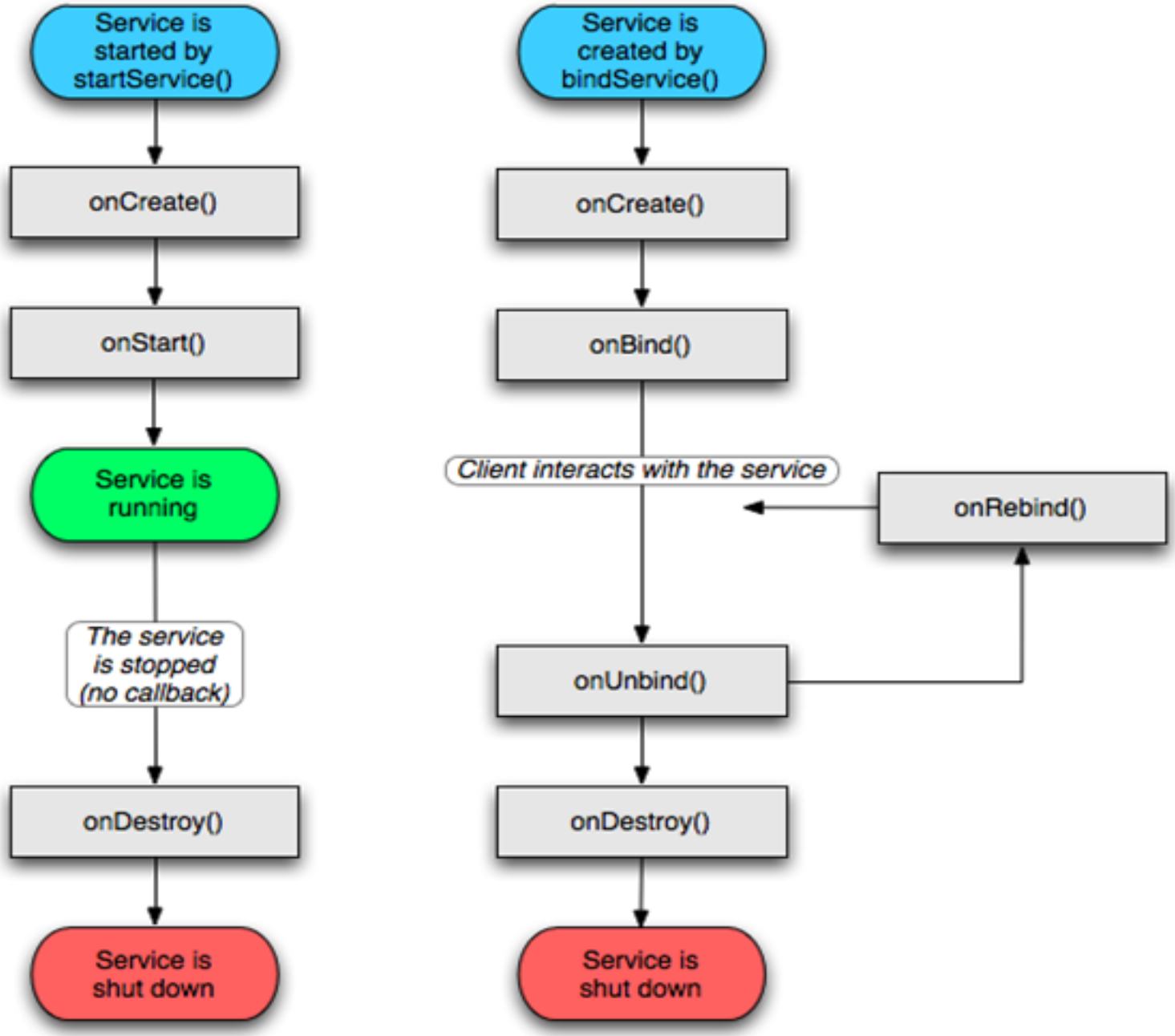
- Le service peut être démarré ou arrêté au besoin
  - Démarrer par Context.startService()
  - Arrêter par Context.stopService()

- Le service peut être géré par le système et appelé à travers une interface AIDL
  - Le serveur démarre le service en utilisant Context.bindService()
  - Le service peut être arrêté en utilisant Context.unbindService()
  - Les clients établissent des connexions au service pour l'utiliser

Déclaration obligatoire dans le manifest

```
<!--  Service Samples    -->
<service android:name=".app.LocalService" />
```

Cycle de vie d'un service Android (cont)



Exemple d'un service Android

```
package com.tuto.android;
public class MonPremierService extends Service
{
 private LocationManager locationMgr = null;
 private LocationListener onLocationChange          = new LocationListener()
 {
  public void onLocationChanged(Location location)
  {
   Toast.makeText(getBaseContext(),"coordonnées de votre téléphone : " + location.getLatitude()
+ " " + location.getLongitude(),   Toast.LENGTH_LONG).show();
  }
  };

 public void onCreate()
 {
  locationMgr = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
  locationMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10000, 0,
  onLocationChange);
  super.onCreate();
 }
```

Exemple d'un service Android (cont)

```
public void onDestroy()
 {
  super.onDestroy();
  locationMgr.removeUpdates(onLocationChange);
 }
}
```

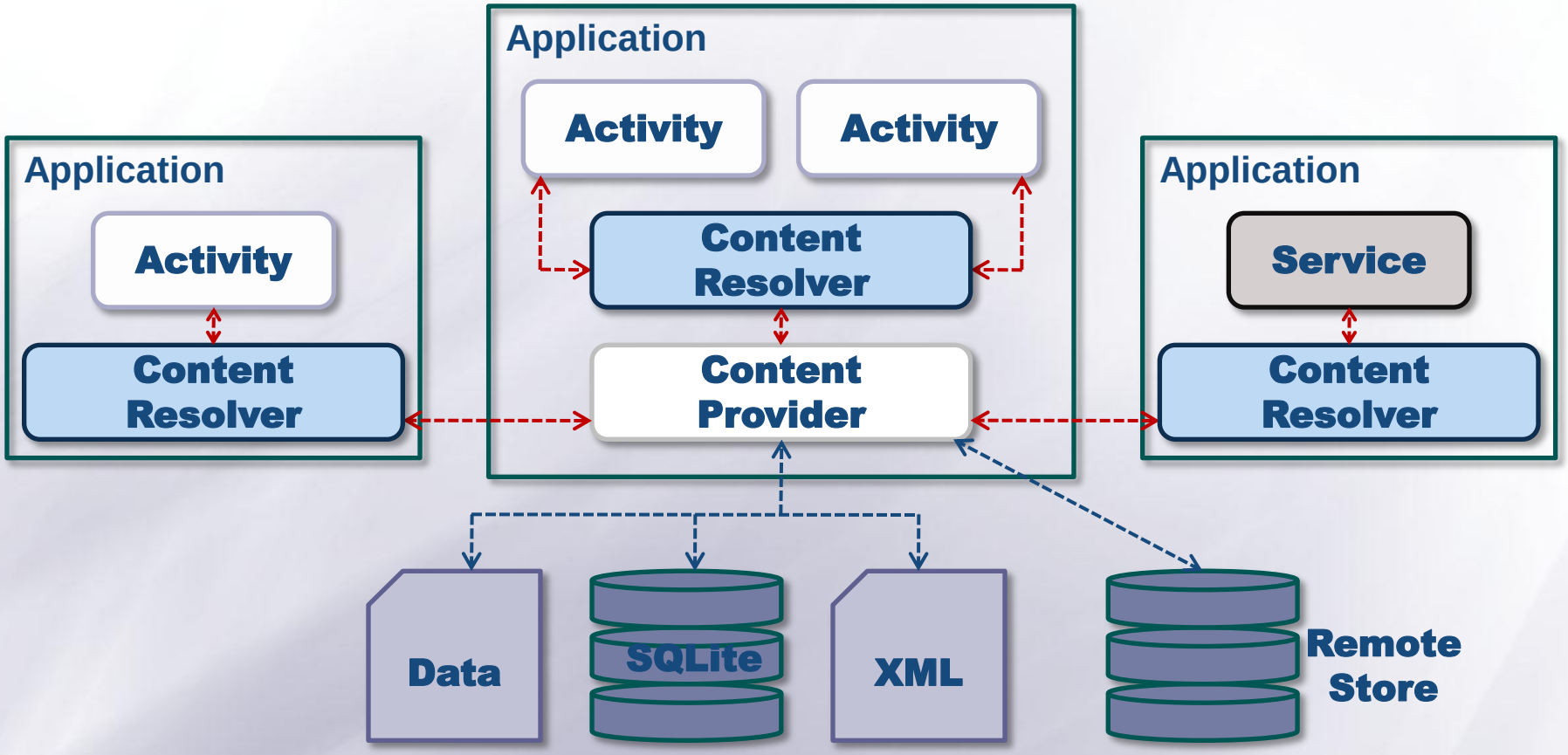
- **Dans le manifest**
  - **Déclaration du service**

```
<service android:name="com.tuto.android.MonPremierService"/>
```

  - **Permissions nécessaires**

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

Composants principaux : Content Provider



- Un “content provider” (fournisseur de contenu) permet de partager des données d’une application à d’autres applications.

Composants principaux : Content Provider (Cont)

- L'utilisation d'un "content provider" est le seul moyen pour partager des données entre des applications Android
- Un content provider est une extension de la classe de base "ContentProvider" qui doit implémenter un ensemble de méthodes standars (requête, ajout, supression et mise à jour)
- Les applications n'utilisent pas ces méthodes directement
  - ✓Elles doivent utiliser un ContentResolver pour intéragir avec les données
  - ✓Un ContentResolver peut interagir avec n'importe quel content provider.
- Le contenu (les données) est identifié par un URI

Ce support est disponible sur

www.redcad.org/members/tarak.chaari/cours/cours_android.pdf