



- 1) Comprendre les éléments de base pour le développement sous d'Android**
- 2) Pouvoir créer une première application simple**



**Android :**

**Le quoi,**

**Le pourquoi,**

**Le contexte,**

**...**

# Introduction : Téléphonie mobile



- **SmartPhone [wikipedia]**

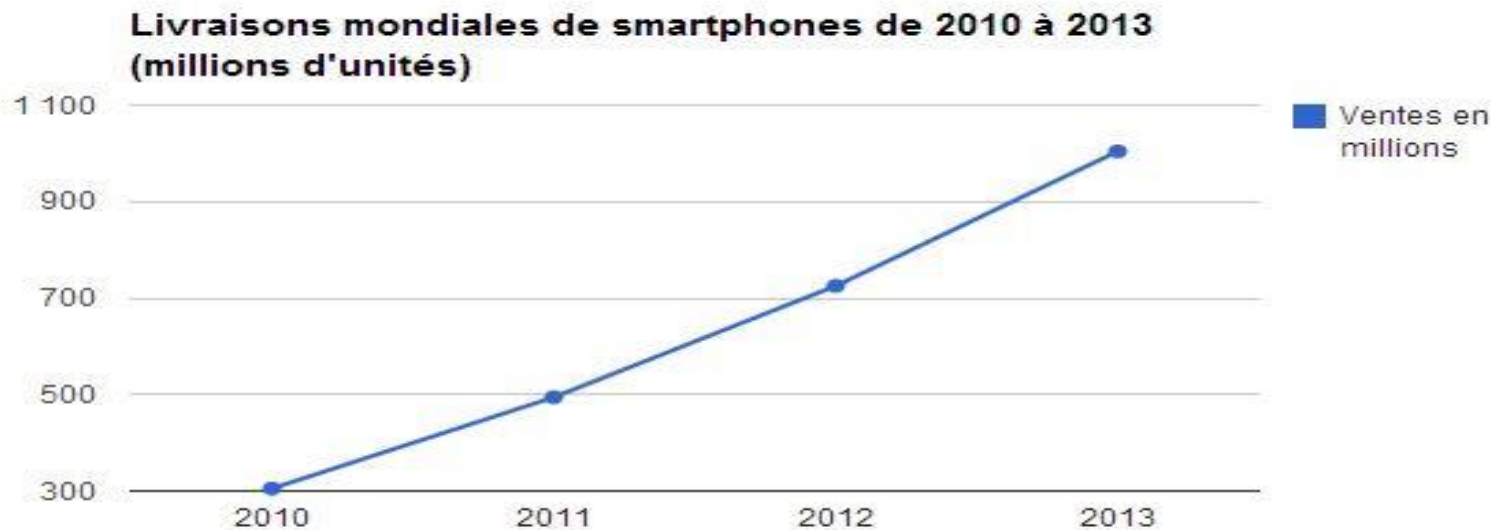
- « Un smartphone, ordiphone ou téléphone intelligent, est un téléphone mobile disposant aussi des fonctions d'un assistant numérique personnel
- La saisie des données se fait par le biais d'un écran tactile ou d'un clavier
- Il fournit des fonctionnalités basiques comme : l'agenda, le calendrier, la navigation sur le web, la consultation de courrier électronique, de messagerie instantanée, le GPS, etc ».



# Introduction : Téléphonie mobile



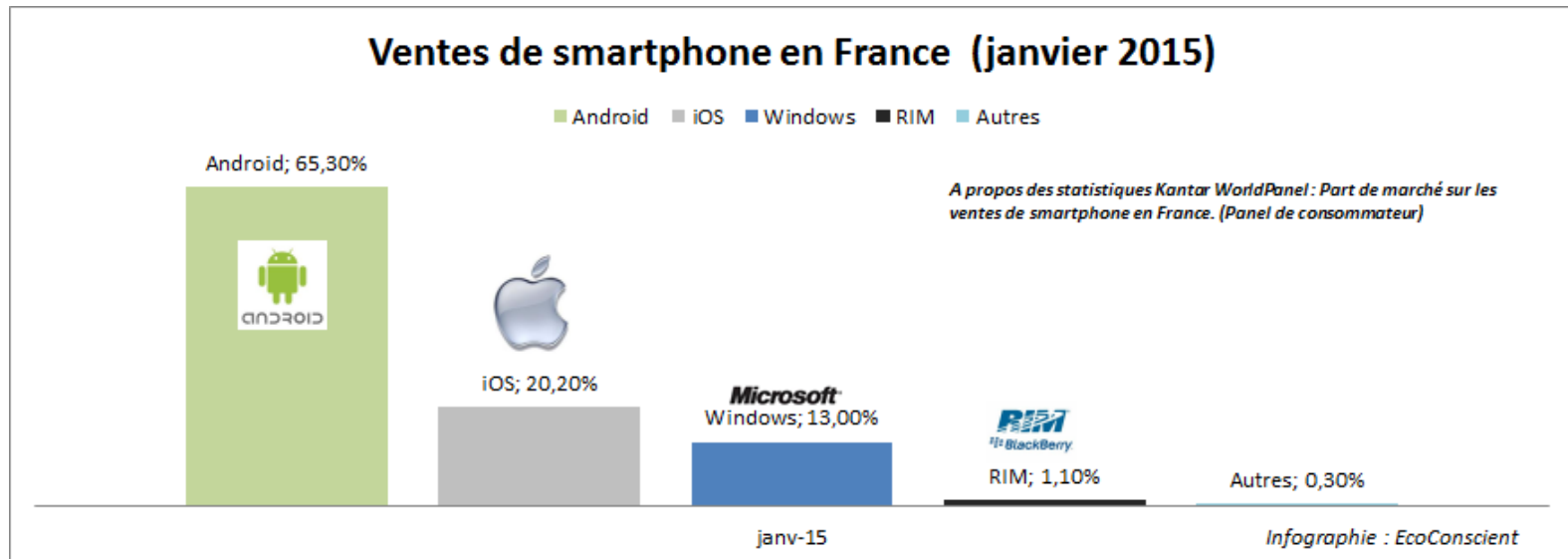
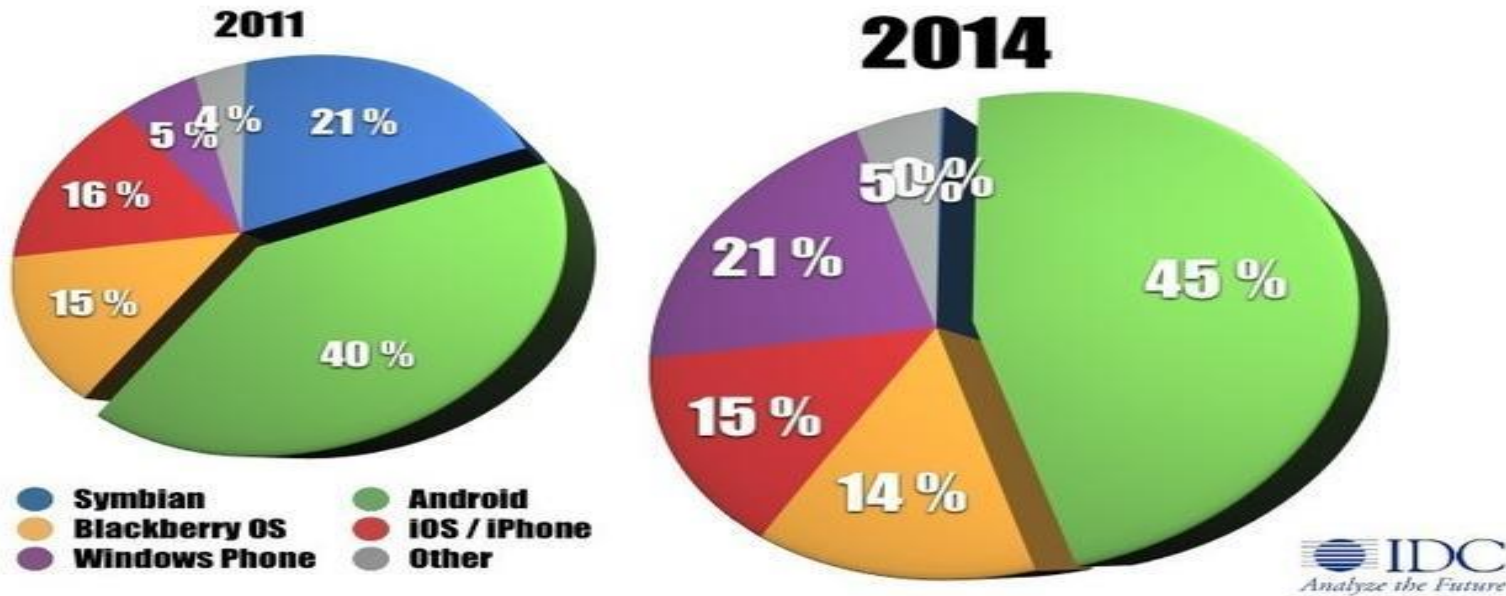
- Vente de "terminaux mobiles" évolués
  - Un téléphone mobile sur deux vendu dans le monde est un Smartphone
  - En 2013, les ventes mondiales de Smartphones ont atteint en volume un peu plus d'un milliard d'unités



Source IDC - via [ZDNet.fr/chiffres-cles](http://ZDNet.fr/chiffres-cles)

- On estime que 20 milliards d'applications ont été téléchargées en 2014 contre 2,3 en 2009

# Introduction : parts de marché des OS



# Introduction : Les principaux OS mobiles



- **Android** (google, ...)
- **Iphone OS** (Apple) sur des téléphones iPhone et sur les tablettes d'Apple
- **Windows Mobile** (Microsoft) ; système propriétaire
- **Symbian** (Nokia) ; récemment passé en open source
- **BlackBerry OS**. Présent sur tous les téléphones de la marque RIM (Research In Motion) ;
- **Palm Web OS** (successeur de Palm Os)
- **LiMo** (Linux Mobile), système ouvert basé sur Linux (<http://www.tizenassociation.org/en/>)
- **MeeGo**, Intel et Nokia (<https://meego.com/>)
- **Bada**, Samsung (<http://www.bada.com/whatisbada/index.html>)
- Etc.



# Introduction : Les principaux OS mobiles



	<b>Android</b>	<b>Bada</b>	<b>BlackBerry OS</b>	<b>iOS</b>	<b>Symbian OS</b>	<b>Windows Phone</b>
<b>Appareils compatibles</b>	HTC, Samsung Galaxy, Motorola	Samsung Waves 3	BlackBerry torch, bold, curve...	iPhone, iPod, iPad	N8, N9	Windows Phone, Nokia lumia 710
<b>Dernière version</b>	4.4.4	2.0.5	10	7.1.2	Nokia Belle (Symbian^3)	8.10.12397.895
<b>Date de sortie</b>	23 juin 2014	15 mars 2012	30 janv. 2013	30 juin 2014	1 août 2011	12 juin 2014
<b>Open source</b>	✓	✗	✗	✗	✓	✗
<b>Mis à jour</b>	🔊🔊✗ 12 sept. 2014 23:01:54	29 juin 2014 02:10:59	29 juin 2014 02:10:59	12 sept. 2014 23:01:43	29 juin 2014 02:10:59	12 sept. 2014 23:04:56
<b>Platforms available for the SDK</b>						
<b>Windows</b>	✓	✓	✓	✗	✓	✓
<b>Mac OS</b>	✓	✗	✗	✓	✗	✗
<b>Linux</b>	✓	✗	✗	✗	✗	✗
<b>Détails techniques</b>						
<b>Support d'Adobe Flash</b>	✓ Intégré directement dans les applications	✓ FlashLite 3.1 (Flash9 - AS2)	✗	✗	✓	✓
<b>Market</b>						
<b>Place de marché</b>	Google Play	Samsung Apps	BlackBerry App World	App Store	OVI Store	Windows Phone Marketplace
<b>Nombre d'applications</b>	800 000 +	3000 (Q1 2011)	70 000 +	1000000 +	> 30 000	> 9 000
<b>Langage de développement</b>	Java	C++	Java	Objective-C	Java - C/Qt - Python	Visual Basic / Visual C#
<b>Environnement compatible</b>	Windows Mac OS Linux	Windows	Windows	Mac OS	Windows	Windows
<b>Prix du SDK</b>	Gratuit	Gratuit	Gratuit	99\$/an	Gratuit	Gratuit
<b>Wikipédia</b>	<a href="http://wikipedia.org/">wikipedia.org/...</a>	<a href="http://wikipedia.org/">wikipedia.org/...</a>	<a href="http://wikipedia.org/">wikipedia.org/...</a>	<a href="http://wikipedia.org/">wikipedia.org/...</a>	<a href="http://wikipedia.org/">wikipedia.org/...</a>	<a href="http://wikipedia.org/">wikipedia.org/...</a>

Source : <http://socialcompare.com/>

# Applications mobiles



- Une application mobile s'exécute sur un support matériel **mobile**:
  - **Ressources** limitées
    - Batterie (énergie), interface graphique, CPU, périphériques d'IO, ...
  - Périphériques très **divers**
    - De très élémentaire au très évolué
  - **Utilisation** ubiquitaire
    - Ubiquité géographique
    - Ubiquité des utilisateurs
  - Utilisation basée sur l'interaction avec l'utilisateur via une interface mobile



# Applications mobiles



- **Ubiquité des supports**

- Adaptation au contexte, aux situations et aux utilisateurs
  - Rôle de l'infrastructure (matériel + OS+ langage) :
    - Capteurs intégrés (localisation, température , vitesse, ...)
  - Rôle du développeur (niveau applicatif) :
    - Applications sensibles au contexte

- **Ressources limitées**

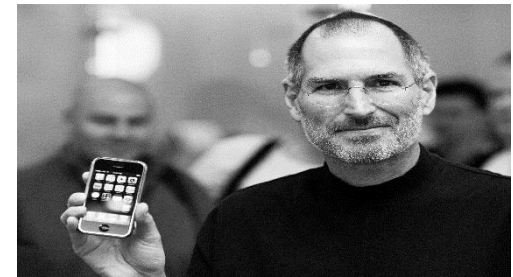
- Adaptation au support matériel
  - Rôle de l'infrastructure (OS + langage) :
    - Mode d'interaction adapté (tactile), gestion des processus, gestion des évènements et des priorités, etc.
  - Rôle du développeur (niveau applicatif):
    - Applications adaptées aux ressources (écrans tactiles, clavier limitées, batterie, ...)

# La plate-forme Android : Historique



- **L'iphone d'Apple**

- A bouleversé le paysage des systèmes d'exploitation mobiles par :
  - Son ergonomie et les capacités du matériel
  - Les usages proposés
  - Les possibilités offertes avec l'Apple Store



- **Handset Alliance**

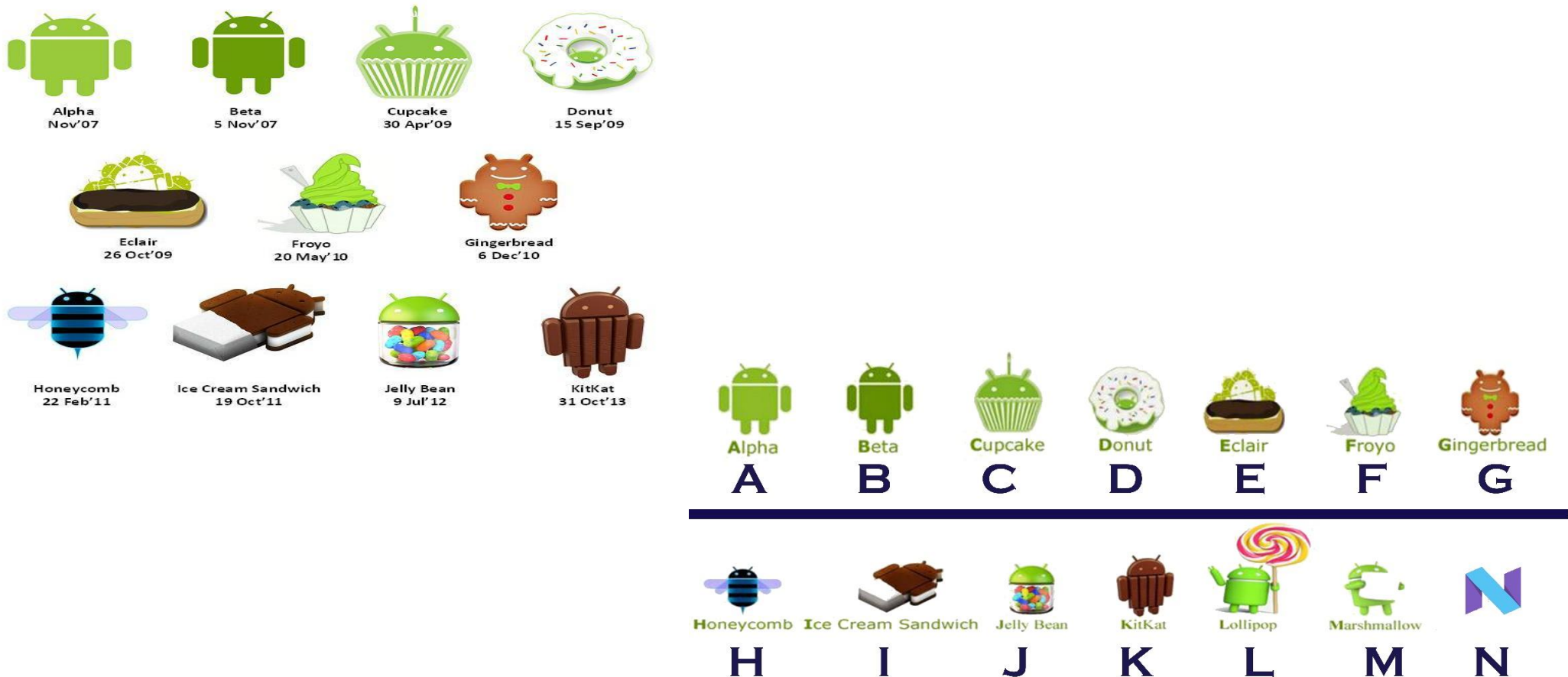
- Est une coalition qui a vu le jour fin 2007
- A pour objectif de créer et de promouvoir le système Android comme système ouvert et gratuit dans le monde du mobile
  - Google est l'acteur majeur
- Adresse web : **<http://www.openhandsetalliance.com>**

# La plate-forme Android : Historique



## • Les versions de la plate-forme

- Première version d'Android en septembre 2008, 1.1 (février 2009), 1.5 (Avril 2009), 1.6 (septembre 2009), 2.0 (octobre 2009), 2.0.1 (Octobre 2009) ...



# La plate-forme Android : Caractéristiques



- **Elle est innovante**
  - Toutes les dernières technologies de téléphonie y sont intégrées : écran tactile, accéléromètre, GPS, appareil photo numérique, etc.
- **Elle est accessible**
  - En tant que développeur, il n'y a pas besoin de matériel spécifique
  - Utilisation d'un émulateur
  - Pas d'apprentissage d'un langage spécifique. Le développement se fait en *Java*
- **Elle est ouverte**
  - Elle est fournie sous licence *open source*, permettant aux développeurs et constructeurs de consulter les sources et d'effectuer les modifications qu'ils souhaitent
    - Utilisation de la *licence Apache* ce qui permet la redistribution du code sous forme libre ou non et d'en faire un usage commercial

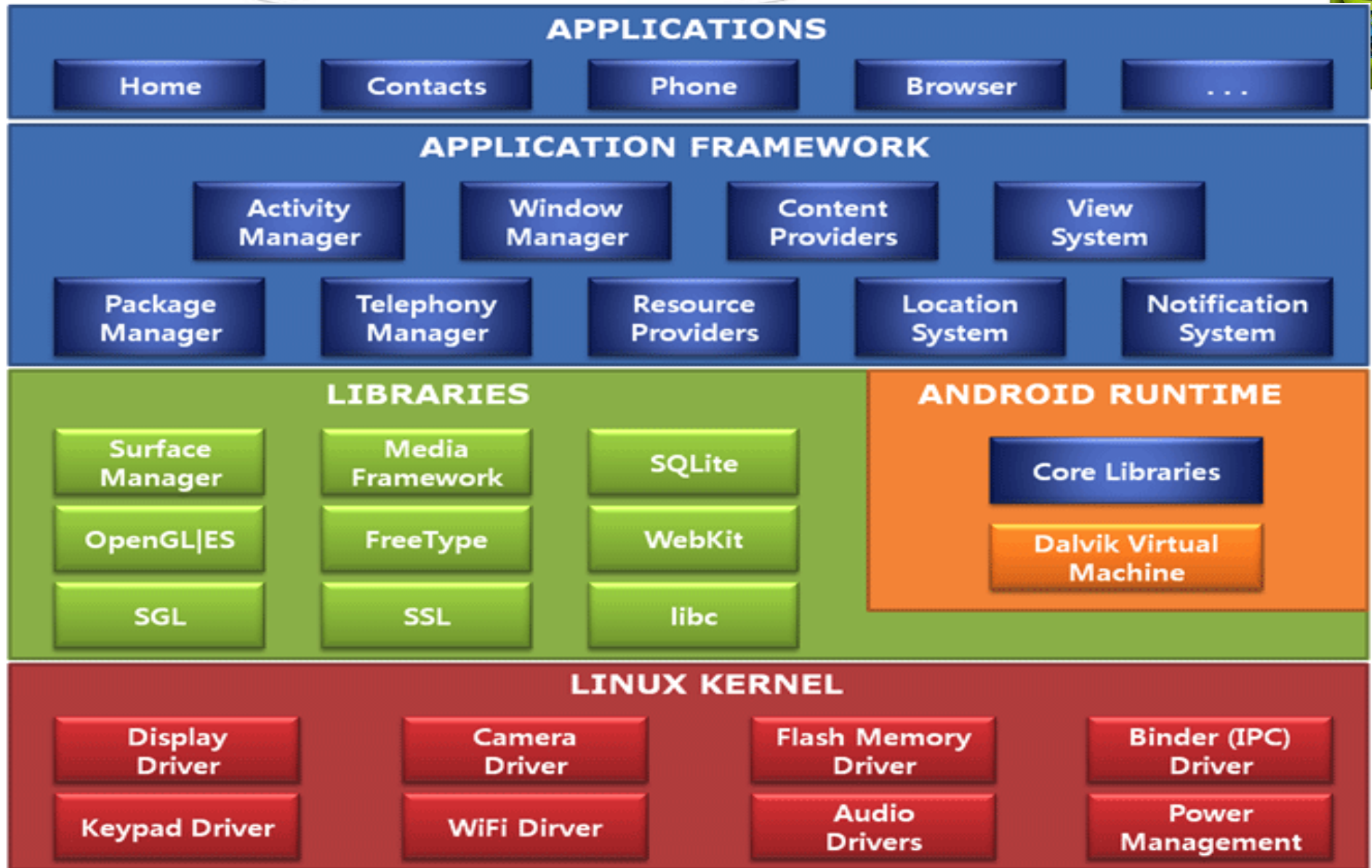
# La plate-forme Android : Architecture



- **Android est conçue pour des appareils mobiles au sens large**
  - Téléphones mobiles, tablettes, ordinateurs portables, bornes interactives, baladeurs, Téléviseurs, machine à laver, interaction usagers voiture, ...
- **La plate-forme Android est composée de différentes couches**
  - **Un noyau Linux** permettant des caractéristiques multitâches
  - **Des bibliothèques** graphiques, multimédias
  - **Une machine virtuelle** Java open-source : la Davik Virtual Machine
    - Il existe un framework natif permettant le développement en C/C++ NDK (Native Development Kit)
  - **Un framework applicatif** proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu...
  - **Des applications** dont un navigateur web, une gestion des contacts, un calendrier...



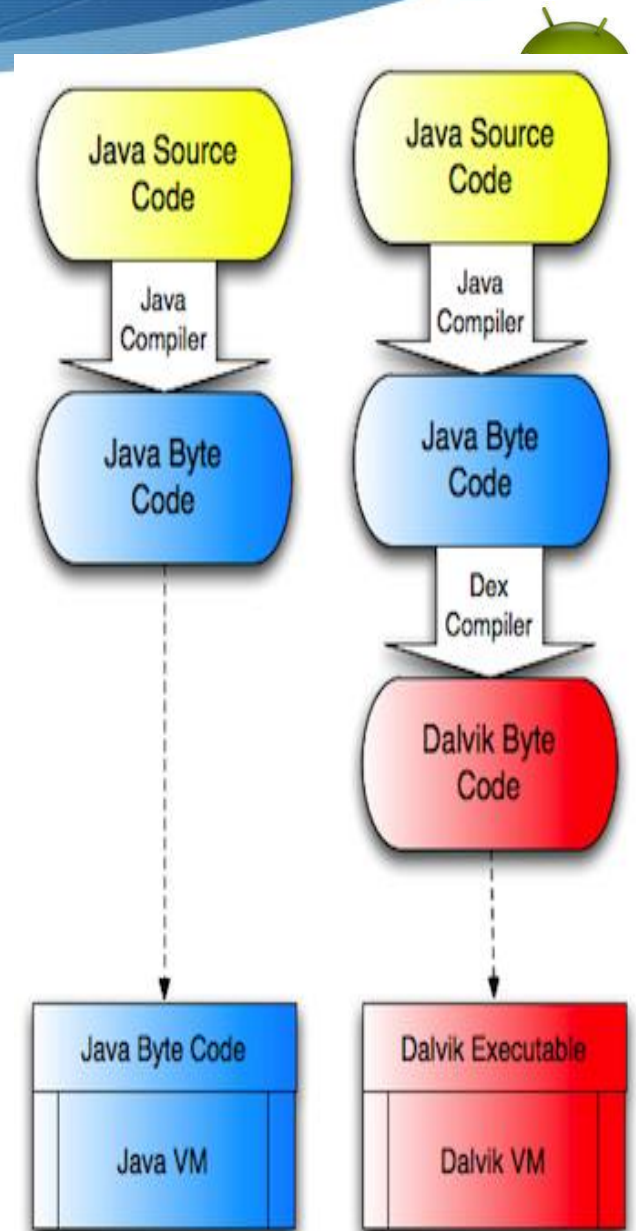
# La plate-forme Android : Architecture





# La plate-forme Android : kit de développement

- **Machine virtuelle "Dalvik"**
  - Offre l'avantage de toute machine virtuelle
    - Couche d'abstraction entre le développeur d'applications et des implémentations matérielles particulières
  - La VM Dalvik n'est pas une VM Java
    - Tient compte des contraintes de CPU et mémoire
    - Exécute des fichiers .dex (*Dalvik Executable*) optimisé
    - Les applications sont totalement indépendantes ("*sandbox*")



# La plate-forme Android : kit de développement



- **Le SDK Android est composé de plusieurs éléments :**
  - Des API (Application Programming Interface)
  - Un certain nombre d'exemples illustrant les possibilités du SDK
  - De la documentation
  - Des outils - parmi lesquels un **émulateur**
- **Le SDK Android est disponible sur le site de Google : <http://developer.android.com>**
- **Autres briques logicielles :**
  - ADT : Android Development Tools Plugin
    - Outil s'intégrant directement à Eclipse
    - Propose des interfaces et des assistants pour la création et le débogage des applications Android
  - Android Studio : <https://developer.android.com/studio/index.html>



# **Cours1-Partie 2**

## **Présentation des différents composants d'une application Android**



# Fichier de configuration Android (manifest)



- **Qu'est que c'est ?**
  - Une application Android est un assemblage de composants liées grâce à un fichier de configuration
  - Décrit entre autres :
    - Le point d'entrée de l'application : quel code doit être exécuté au démarrage de l'application
    - Quels composants constituent ce programme : les activités, les services, ...
    - Les permissions nécessaires à l'exécution du programme
- **Comment ?**
  - Fichier XML : `AndroidManifest.xml`

# Fichier de configuration Android

- **Exemple**



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="andro.jf"
    android:versionCode="1"
    android:versionName="1.0">
<application android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity android:name=".Main"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <service>...</service>
    <receiver>...</receiver>
    <provider>...</provider>
</application>
</manifest>
```

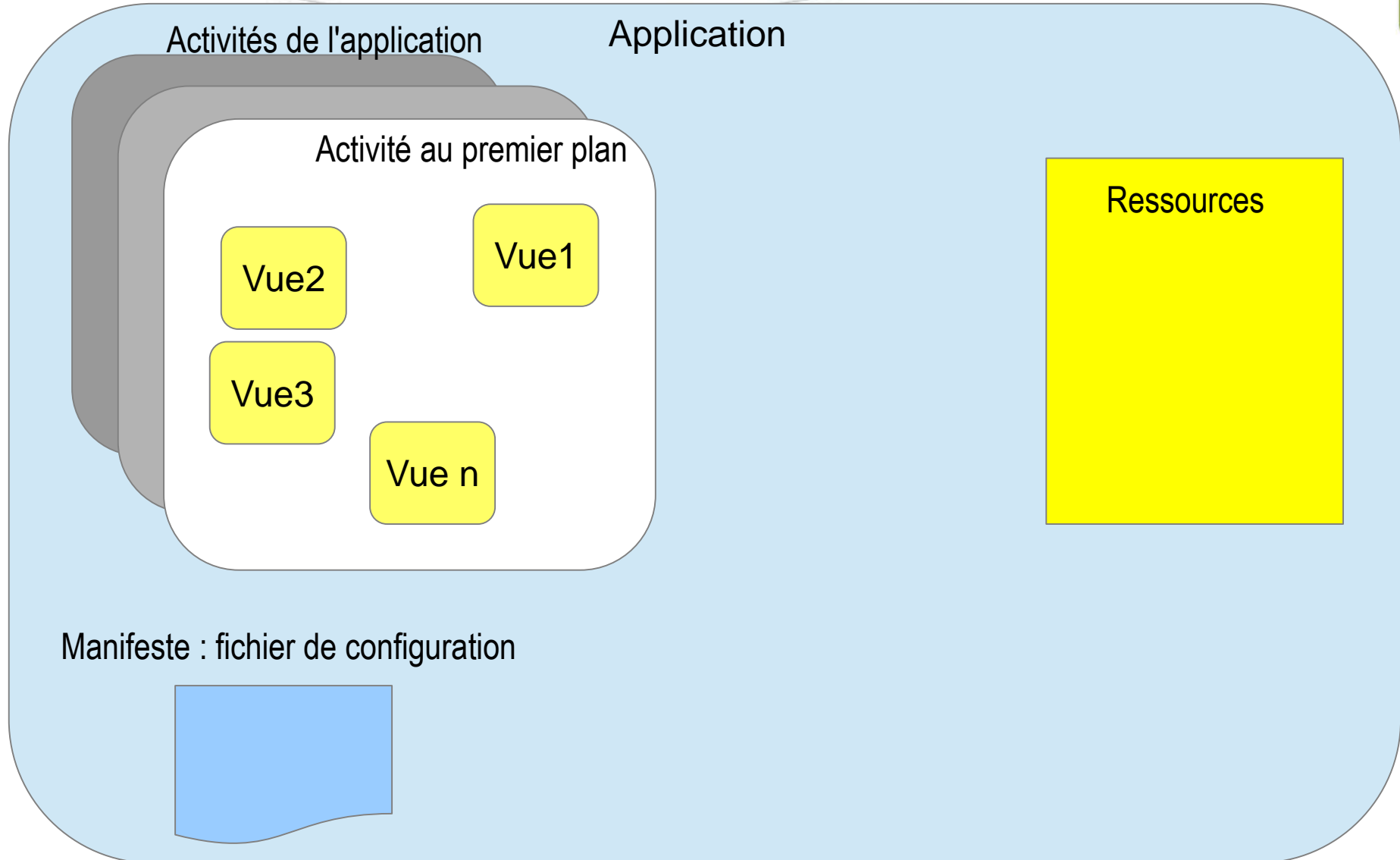
# Composants d'une application Android



- Les composants peuvent être classés en éléments applicatifs et éléments d'interaction
  - **Eléments applicatifs**
    - Activité
    - Service
    - Fournisseur de contenu
    - Gadget (widget)
  - **Eléments d'interaction**
    - Objet Intent
    - Récepteur d'Intents
    - Notification



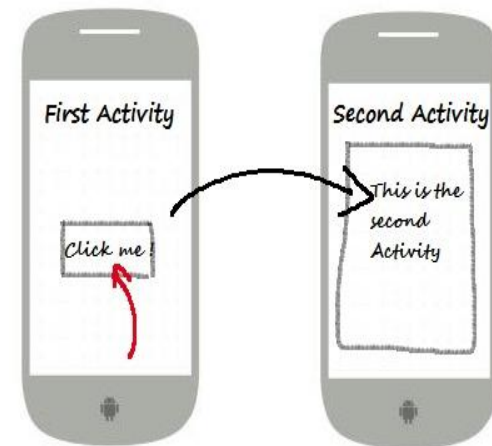
# Composants d'une application Android



# Activité

- **Présentation**

- Elle correspond à la partie présentation de l'application : *correspond à un écran*
  - Représente le bloc de base d'une application
- Fonctionne par le biais de *vues* qui affichent des interfaces graphiques et répondent aux actions utilisateur
  - Elle est composée d'une hiérarchie de vues contenant elles-mêmes d'autres vues
    - Un formulaire d'ajout de contacts ou encore un plan Google Maps sur le lequel on peut ajouter de l'information
- Une application comportant plusieurs écrans, possédera donc autant d'activités



# Activité



- **Utilisation**

- Une activité est composée de deux volets :
  - Sa logique métier et la gestion de son cycle de vie
    - Implémentés en Java dans une classe héritant de *Activity*
  - Son interface utilisateur
    - Deux façons alternatives pour sa définition:
      - Programmative : dans le code de l'activité
      - Déclarative : dans un fichier XML

# Activité

- **Logique métier d'une activité : Squelette minimal**



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

public class ActiviteBasic extends Activity {

    //méthode onCreate appelée à la création de l'activité
    public void onCreate(Bundle etatSauvegarde){
        super.onCreate(etatSauvegarde);
    }
}
```

# Activité



- La balise `<activity>` déclare une activité
  - Les paramètres généralement utilisés sont :
    - `name` qui désigne la classe de l'activité
    - `label` qui désigne le nom sous lequel elle apparaîtra sur le terminal
    - `icon` qui désigne l'icône à afficher sur le terminal
  - Structure

```
<application ...>
<activity android:name=".ClasseDeLActivite"
          android:label="nom_de_l_activite"
          android:icon="@drawable/nom_du_fichier_icone">

  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
</application>
```

# Activité



- **Cycle de vie d'une activité**

- Les états principaux d'une activité sont les suivants :
  - Active (active)
    - Activité visible qui détient le focus utilisateur et attend les entrées utilisateur
    - Appel à la méthode *onResume()*
  - Suspendue (Paused)
    - Activité au moins en partie visible à l'écran mais qui ne détient pas le focus
    - Appel à la méthode *onPause()* pour entrer dans cet état
  - Arrêté (stopped)
    - Activité non visible
    - Appel de la méthode *onStop()*



# Activité

- Cycle de vie d'une activité

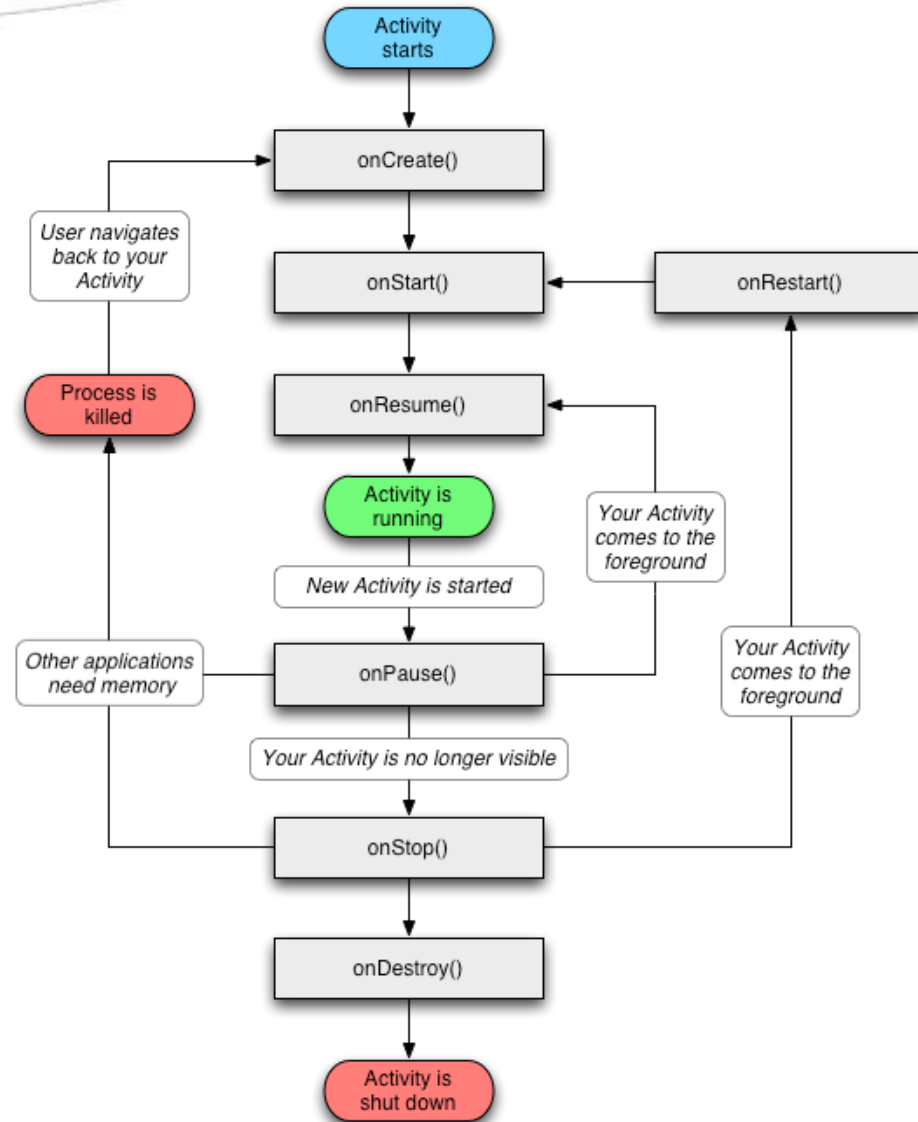


```
public class Main extends Activity {  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.acceuil); }  
  
    protected void onDestroy() {  
        super.onDestroy(); }  
  
    protected void onPause() {  
        super.onPause(); }  
  
    protected void onResume() {  
        super.onResume(); }  
  
    protected void onStart() {  
        super.onStart(); }  
  
    protected void onStop() {  
        super.onStop(); } } }
```

# Activité



- Cycle de vie d'une activité



# Les interfaces d'applications : Les Vues

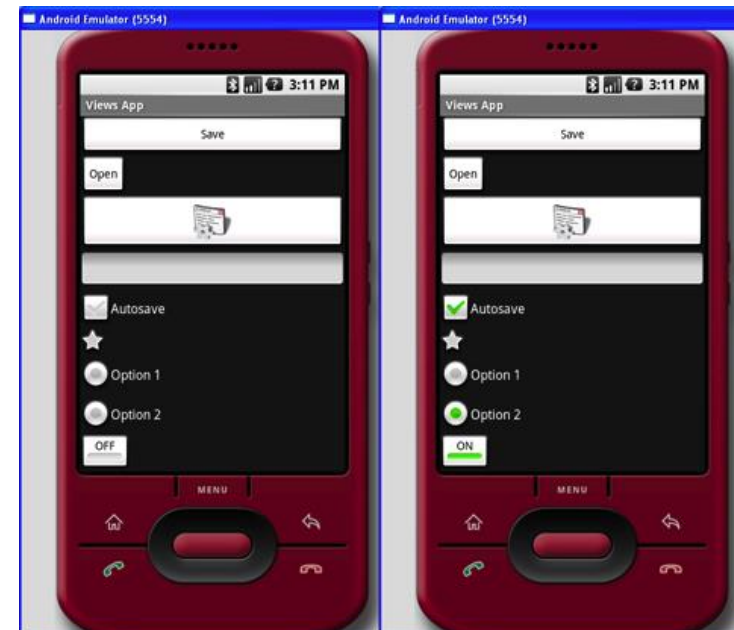


- **Présentation**

- Sont les briques de construction de l'interface graphique d'une activité Android

- **Utilisation**

- Les vues sont soit prédéfinies par la plateforme -textes, boutons, ... ou créées comme des éléments personnalisés
- Chaque écran Android contient un arbre d'éléments de type *View*
- Les vues peuvent être disposées dans une activité (objet *Activity*) et donc à l'écran soit par une description XML, soit par un morceau de code Java



# Les interfaces d'applications Android



- Tous les composants graphiques (boutons, images, cases à cocher, etc.) d'Android héritent de *la classe View*
- Android permet de regrouper plusieurs vues dans une structure arborescente à l'aide de *la classe ViewGroup*
  - Cette structure peut regrouper d'autres éléments de la classe ViewGroup → un arborescence
- L'utilisation et le positionnement des vues dans une activité se fera via des *gabarits de vues*

# Les interfaces d'applications Android



- **Positionnement des vues avec les gabarits**
  - Un gabarit, layout ou mise en page, est une extension de la classe **ViewGroup**
  - Est un **conteneur** qui aide à positionner les objets (vues, gabarits, etc.)
  - Les gabarits peuvent être imbriqués les uns dans les autres
- **Quelques types de gabarits**
  - **LinearLayout**
    - Permet d'aligner de gauche à droite ou de haut en bas les éléments qui y seront incorporés
    - En modifiant la propriété **ORIENTATION**, il est possible de modifier le sens de d'affichage des éléments
      - **Horizontal** : affichage de gauche à droite
      - **Vertical** : affichage de haut en bas

# Les interfaces d'applications Android



- **Quelques types de gabarits**

- **RelativeLayout**

- Ses enfants sont positionnés les uns par rapport aux autres
  - Le premier enfant servant de référence aux autres

- **TableLayout**

- Permet de positionner les vues en lignes et colonnes à l'instar d'un tableau



# Les interfaces d'applications Android



- **Les propriétés communes à tous les types de gabarit**
  - layout\_weight : comportement pour le remplissage en largeur
  - layout\_height : comportement pour le remplissage en hauteur
  - Ces propriétés peuvent être exprimées en :
    - Une unité de mesure spécifiant une taille précise ou relative
      - Taille précise : le même nombre de pixels quelle que soit la taille de l'écran
      - Taille relative en DIP ou en SP: permettent un ajustement automatique des éléments
        - sont à privilégier
        - Elles permettent de s'adapter plus aisément à différentes tailles d'écran
        - Rendent les applications plus portables

# Les interfaces d'applications Android



- **Les propriétés communes à tous les types de gabarit**
  - layout\_weight
  - layout\_height
  - Ces propriétés peuvent être exprimées en :
    - Une unité de mesures spécifiant une taille précise ou relative
    - Valeurs prédéfinies
      - Les valeurs prédéfinies
        - fill\_parent
        - spécifie que le gabarit doit prendre toute la place disponible sur la largeur/hauteur
        - wrap\_content
        - spécifie que le gabarit ne prendra que la place qui lui est nécessaire en largeur/hauteur

# Création d'une interface utilisateur



- **Deux possibilités pour créer une interface**
  - Directement dans le code : instancier les vues dans le code
  - La création en deux étapes en séparant la présentation de la logique fonctionnelle de l'application
    - Définition de l'interface utilisateur (gabarit, etc.) de façon déclarative dans un fichier XML
    - Définition de la logique utilisateur (comportement de l'interface) dans une classe d'activité

# Définition de l'interface en XML



- Les fichiers de définition d'interface en XML sont enregistrés dans le dossier res/layout du projet
- Chaque fichier XML définissant une interface graphique est associé à un identifiant unique généré automatiquement qui peut être référencé dans le code de l'application
  - Exemple : R.layout.monLayout

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

<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width = "fill_perent"
    android:layout_height="fill_parent"
    >
    <TextView
    android:layout_width="fill_perent"
    android:layout_height="wrap_content"
    android:id="@+id/monText" />
</LinearLayout>
```

# Association entre activité et interface



- Une interface est affichée par l'intermédiaire d'une activité
- Le chargement du contenu de l'interface s'effectue à l'instanciation de l'activité
  - Redéfinition de la méthode *onCreate()* de l'activité pour y spécifier la définition de l'interface à afficher via la méthode
  - Affichage de l'interface par la méthode *setContentView()*
    - Prend en paramètre un identifiant qui spécifie quelle ressource de type interface doit être chargée et affichée

# Association entre activité et interface



- **Spécifier une vue pour l'activité**

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

public class Main extends Activity {

    @override
    public void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

    }
}
```



# Création une interface dans le code source (sans définition XML)



- **Exemple**

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

public class Main extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        TextView monTextView = new TextView(this);

        setContentView(monTextView);

        monTextView.setText(" Notre premier cours Android");

    }

}
```

# Utilisation des gabarits



- Pour intégrer plus d'une vue à une activité : réunir tous ces vues dans un gabarit de vues
  - Détail technique : la méthode `SetContentView()` n'accepte qu'un seul objet graphique (une vue) comme paramètre
  - Directement dans le code

```
public class Main extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        LinearLayout monLinearLayout = new LinearLayout(this)  
        monLinearLayout.setOrientation(LinearLayout.VERTICAL);  
  
        TextView monTextView1 = new TextView(this);  
        monTextView1.setText(" Notre premier cours Android");  
        TextView monTextView2 = new TextView(this);  
        monTextView2.setText(" vivement le premier TP");  
  
        monLinearLayout.addView(monTextView1);  
        monLinearLayout.addView(monTextView2);  
  
        setContentView(monLinearLayout);    }  
}
```

# Utilisation des gabarits



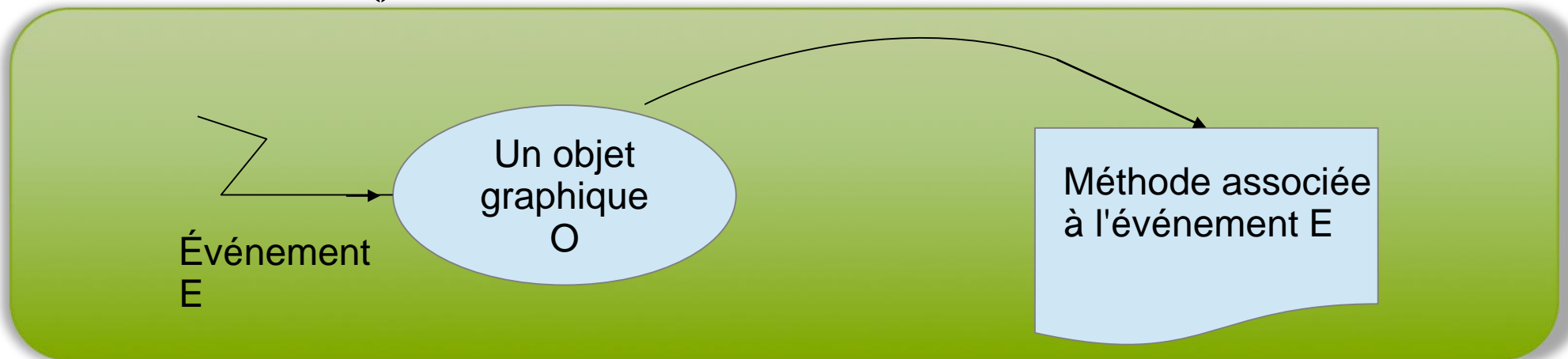
- Pour intégrer plus d'une vue à une activité : réunir tous ces vues dans un gabarit de vues
  - De manière programmatique : Directement dans le code
  - De manière déclarative : dans un fichier XML

```
<?xml version="1.0" encoding = "utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width = "fill_perent"
android:layout_height="fill_parent" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monText"
    android:text="Texte en haut à droite"
    android:gravity="topright" >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monText"
    android:text="Texte en bas au centre"
    android:gravity="bottomlcenter_horizontal"/>
</LinearLayout>
```

# Gestion des événements



- Sous Android, toutes les actions de l'utilisateur sont perçues comme un événement
- Les événements sont interceptés par les éléments d'une interface en utilisant des écouteurs (listeners)
  - Association entre un événement et une méthode à appeler en cas d'apparition de cet événement
    - Exemple : pour un événement `OnClick`, la méthode associée est `OnClick()`



- **Exemple**



## Insertion d'un bouton dans l'interface

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

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width ="fill_perent"
android:layout_height="fill_parent"
android:gravity="center_vertical | center_horizontal"
>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/monBouton"
    android:text="Cliquez ici !"
    >
</Button>
</LinearLayout>
```

# Gestion des événements

- **Exemple**

## création d'un écouteur sur un bouton

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class Main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ((Button) findViewById(R.id.monBouton)).
            setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                    Toast.makeText(Main.this, "Bouton cliqué !", Toast.LENGTH_LONG).show();
                }
            });
    }
}
```







# Les ressources

# Les ressources



- **Présentation**

- Sont des fichiers externes ne contenant pas d'instructions qui sont utilisés par le code
  - Les fichiers images JPEG et PNG, les fichiers XML...

- **Utilisation**

- L'**externalisation** des ressources permet une meilleure gestion de ces ressources ainsi qu'une maintenance plus aisée
- Les ressources de l'application sont déposées dans le *répertoire res* du projet
  - Android crée *une classe nommée R* utilisée pour référer aux ressources dans le code
- Toutes les ressources sont placées, converties ou non, dans **un fichier de type APK** qui constituera le programme distribuable de l'application

# Les ressources

Type de ressources	Répertoire associé	Description
Valeurs simples	res/values	définitions en XML de valeurs : chaînes, tableaux, valeurs numériques
Drawables	res/drawable	Des ressources images
Layouts	res/layout	description en XML des interfaces
Animations	res/anim	description en XML d'animations
Ressources XML	res/xml	Fichier XML qui peuvent être lus et convertis à l'exécution par la méthode <code>ressources.getXML</code>
Ressources brutes	res/raw	tous les autres types de ressources : fichiers texte, vidéo, son. Fichiers à ajouter sous leurs formats d'origine.

# Les ressources



- **Création de ressources**

- Les ressources de type valeur (entiers, booléens, chaînes de caractères, etc. et des tableaux) peuvent être décrites dans des fichiers xml :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="coulfond">      #AA7B03          </color>
<integer name="limite">      567            </integer>
<integer-array name="codes_postaux">
    <item>34100</item>
    ...
    <item>30000</item>
</integer-array>
<string name "nom_de_mon_application" > mon premier exemple Android </string>
<string-array name="planetes">
    <item>Mercure</item>
    ...
    <item>Venus</item>
</string-array>
<bool name="actif">          true            </bool>
<dimen name "taille">      55px          </dimen>
</resources>
```

# Les ressources



- **Utilisation des ressources**

- Les ressources peuvent être utilisées dans les fichiers XML ou dans le code java
- Utilisation des ressources dans le code Java
  - Les ressources peuvent être utilisées via leurs identifiants : utilisation de *la classe statique R* automatiquement générée

```
android.R.type_ressource.nom_ressource
```

# Les ressources



- Utilisation des ressources

```
public final class R {  
  
    public static final class string {  
        public static final int invitation = 0x7f040001;  
        public static final int texte_titre_ecran = 0x7f040002;  
    };  
  
    public static final class layout {  
        public static final int ecran_de_demarrage= 0x7f030001;  
        public static final int ecran_principal= 0x7f030002;  
    };  
  
    public static final class drawable {  
        public static final int image_android = 0x7f020000;  
    };  
};
```

Utilisation de la ressource dans  
le code Java



Android.R.string.invitation



# Les ressources



- Utilisation des ressources

- Les ressources peuvent être utilisées dans les fichiers XML ou dans le code java
- Utilisation des ressources dans le code Java
  - Les ressources peuvent être utilisées via leurs identifiants : utilisation de *la classe statique R* automatiquement générée
  - Les ressources peuvent être utilisées en récupérant 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:

```
TextView texte = (TextView) findViewById(R.id.le_texte);  
texte.setText("Here we go !");
```

# Les ressources



- **Utilisation de ressources**

- Référencement d'une ressource dans un fichier XML
  - On référence une ressource dans un fichier XML par

`"@[paquetage:]type/identificateur"`

- Exemple

`@string/nom_de_mon_application`

- fait référence à une chaîne décrite dans un fichier XML placé dans le répertoire `res/values`

`<string name "nom_de_mon_application" > mon premier exemple Android </string>`

# Les ressources



- Utilisation des ressources
  - Ressources référencées par d'autres ressources
    - Les ressources définies peuvent être utilisées comme valeurs d'attributs dans d'autres ressources sous forme XML

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

<TableLayout xmlns:android="http://schema.android.com/apk/res/android"

android:layout_width = "fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="1">

<TableRow>
  <TextView
    android:text="@string/table_contenu_cellule_gauche" />

  <TextView
    android:text="@string/table_contenu_cellule_droite" />
</TableRow>
</TableLayout>
```

Référencement  
d'une autre  
ressource

# Autres composants



- **Service**

- Est un composant qui fonctionne en tâche de fond, de manière invisible
- Ses principales utilisations sont la mise à jour de sources de données ainsi que d'activités visibles et le déclenchement de notifications

- **Gadget**

- Est un composant graphique qui s'installe sur le bureau Android
- Exemples :
  - Le calendrier qui affiche de l'information

- **Fournisseur de contenu**

- Permet de gérer et de partager des informations
- Un même fournisseur permet d'accéder à des données au sein d'une application et entre applications



# **Cours 1 – Partie 3**

## **Communication entre composants de l'application**

# Composants d'interactions



- Permettent l'interaction :
  - Entre les différents composants, entre les applications installées sur l'appareil, avec l'utilisateur
- **Les composants**
  - **L'objet Intent**
    - Permet à une application de demander l'exécution d'action
  - **Récepteur d'Intents**
    - Permet à une application d'être à l'écoute des objets Intent qui lui sont destinés
  - **Notification**
    - Signale une information à l'utilisateur sans interrompre ses actions en cours
  - **Intent-Filter**

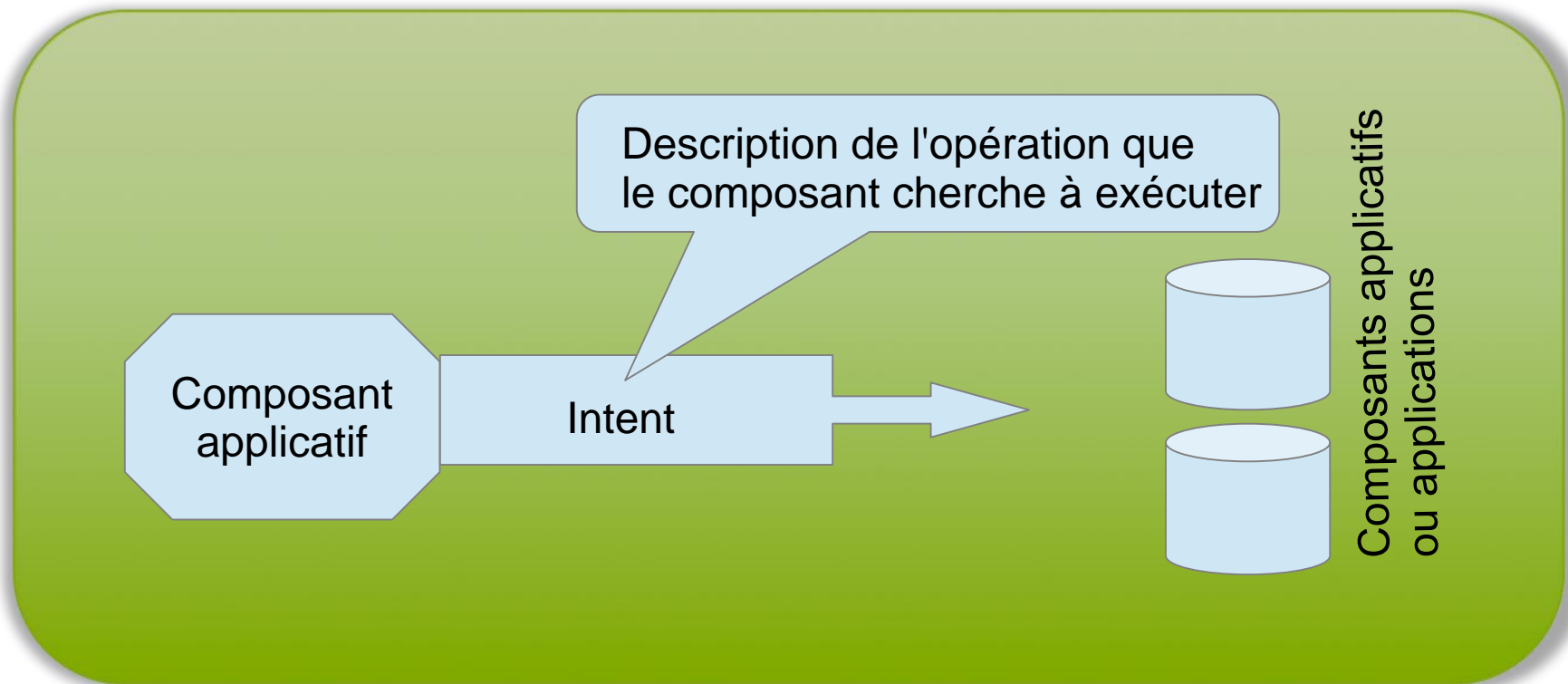


# Objets Intent



- La communication entre les composants d'applications Android se fait via l'expression d'intention
- Une intention d'action est une description abstraite d'une opération à effectuer
  - Exprimer ce que l'opération demandée doit faire
- Les intentions (souhaits) peuvent être envoyés aux composants d'une même application (activité, service, etc.) ou aux autres applications
- Les intentions sont des objets instances de la classe Intent

# Objets Intent



# Principe de fonctionnement



- Les objets Intent ont trois utilisations possibles
  - Démarrer une activité au sein de l'application courante
    - Utilisation
      - Navigation entre écrans d'une interface graphique
    - Démarrage explicite d'une activité : spécifier l'activité cible
  - Solliciter d'autres applications
    - Transmission de l'intention au système
    - Le système se charge de trouver l'application ou le composant le plus approprié
    - Le système démarre l'application ou le composant approprié en lui transmettant l'objet Intent en question
  - Envoyer des informations
    - Exemple : batterie défaillante

# Naviguer entre écran au sein d'une même application



- **Une application = un ou plusieurs écrans**
  - Enchaînement des écrans en fonction du déroulement de l'application
    - Un écran = activité définissant son interface et sa logique
  - Un Intent permet d'assurer cet enchaînement en démarrant ces activités, une à la fois
  - De manière générale chaque composant d'une application nécessite l'emploi d'un Intent pour être démarré

# Naviguer entre écrans



- **Démarrer une activité sans attendre de retour**

- Utilisation de la méthode `startActivity()`
  - Avec comme paramètre une instance de la classe `Intent`
    - spécifier le type de classe de l'activité à exécuter

```
Intent intent = new Intent (this, ActiviteDemarrer.class);  
startActivity(intent);
```

- Le constructeur de la classe `Intent` prend les paramètres suivants :
  - `Context PackageContext` : le contexte à partir duquel l'`Intent` est créé. Fait référence la plupart du temps à l'activité en cours → utilisation de `this`
  - `Class<?> cls` : un type de classe Java héritant de la classe `Activity` → l'activité enfant à démarrer

# Naviguer entre écrans



- **Démarrer une activité et obtenir un retour**
  - Utilisation de la méthode `startActivityForResult()`

```
...  
private static final int CODE_MON_ACTIVITE = 1;  
...  
Intent intent = new Intent(this, ClassSousActivite.class);  
  
//représente l'identifiant de la requête qui sera utilisé pour  
//identifier l'activité renvoyant la valeur de retour  
  
startActivityForResult(intent, CODE_MON_ACTIVITE);
```

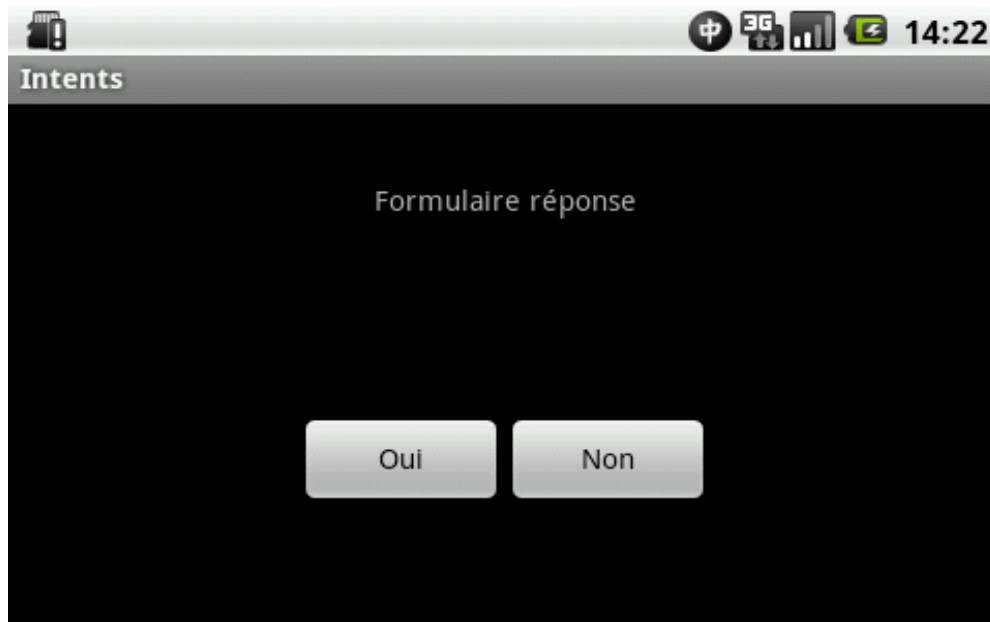


# Naviguer entre écrans



- **Démarrer une activité et obtenir un retour**

- Utilisation de la méthode `startActivityForResult()`
- Renvoyer une valeur de retour
  - Utilisation de la méthode `setResult()` de la classe `Activity`
    - A comme paramètre le code de retour
      - Valeurs par défaut : `RESULT_OK`, `RESULT_CANCELED`...
  - Exemple : retour d'une activité enfant représentant un formulaire avec deux boutons : OUI, NON



```
@Override
public void onClick(View v) {
    switch(v.getId()){
    case R.id.button1:
        setResult(RESULT_OK);
        finish();
        break;
    case R.id.button2:
        setResult(RESULT_CANCELED);
        finish();
        break;
    }
}
```

# Naviguer entre écrans



- **Démarrer une activité et obtenir un retour**

- Utilisation de la méthode `startActivityForResult()`
- Renvoyer une valeur de retour
- Récupérer la valeur de retour
  - Utilisation de la méthode *`OnActivityResult()`* de l'activité parent
    - Ses paramètres
      - *`Int requestCode`* : valeur identifiant quelle activité a appelé la méthode ; c'est la même valeur utilisée pour le paramètre de `StartActivityForResult`
      - *`Int resultCode`* : valeur de retour envoyée par l'activité enfant pour signaler son état à la fin de la transaction
      - *`Intent data`* : l'objet intent permettant d'échanger des données

# Naviguer entre écrans



- Démarrer une activité et obtenir un retour
  - Récupérer la valeur de retour

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){

//le code de requête est utilisé pour identifier l'activité enfant
switch (requestCode) {
case CODE_MON_ACTIVITE:
    switch(resultCode){
    case RESULT_OK:
        Toast.makeText(this, "Action validée", Toast.LENGTH_LONG).show();
        return;
    case RESULT_CANCELED:
        Toast.makeText(this, "Action annulée", Toast.LENGTH_LONG).show();
        return;
    default:
        //des instructions à faire
        return;
    }
default:
//instructions à faire
return;
}}
```

# Solliciter d'autres applications



- **Utilisation d'Intent implicite**

- Le destinataire de l'Intent n'est pas explicitement spécifié
- Le système doit trouver le destinataire approprié en se basant sur :
  - Les filtres
  - Les informations suivantes de l'Intent :
    - Le type d'action
      - ACTION\_DIAL, ACTION\_EDIT, ACTION\_CALL, ...
    - Les données spécifiées dans l'Intent
      - URI (Uniform Resource Identifier)
      - Son format dépend du type de l'action
      - Format général : schéma://hôte:port/chemin
      - Exemples de schéma : tel, www, market
      - le type de contenu MIME

Action	Définition
ACTION_ANSWER	Prendre en charge un appel entrant.
ACTION_CALL	Appeler un numéro de téléphone. Cette action lance une activité affichant l'interface pour composer un numéro puis appelle le numéro contenu dans l'URI spécifiée en paramètre.
ACTION_DELETE	Démarrer une activité permettant de supprimer une donnée identifiée par l'URI spécifiée en paramètre.
ACTION_DIAL	Afficher l'interface de composition des numéros. Celle-ci peut être pré-remplie par les données contenues dans l'URI spécifiée en paramètre.
ACTION_EDIT	Éditer une donnée.
ACTION_SEARCH	Démarrer une activité de recherche. L'expression de recherche de la pourra être spécifier dans la valeur du paramètre SearchManager.QUERY envoyé en extra de l'action.
ACTION_SEND	Envoyer des données texte ou binaire par courriel ou SMS. Les paramètres dépendront du type d'envoi.
ACTION_SENDTO	Lancer une activité capable d'envoyer un message au contact défini par l'URI spécifiée en paramètre.
ACTION_VIEW	Démarrer une action permettant de visualiser l'élément identifié par l'URI spécifiée en paramètre. C'est l'action la plus commune. Par défaut les adresses commençant par <i>http:</i> lanceront un navigateur web, celles commençant par <i>tel:</i> lanceront l'interface de composition de numéro et celles débutant par <i>geo:</i> lanceront Google Map.
ACTION_WEB_SEARCH	Effectuer une recherche sur Internet avec l'URI spécifiée en paramètre comme requête.

# Solliciter d'autres applications



- **Utilisation d'Intent implicite**

- Exemple

- Lancer une action permettant de composer un numéro de téléphone
      - Type d'action de l'Intent: ACTION\_DIAL
      - URI : le numéro à appeler

```
Uri uri = Uri.parse("tel:0612345678");  
  
Intent intent = new Intent(Intent.ACTION_DIAL, uri);  
  
startActivity(intent)
```



# Spécifier les permissions liées aux actions



- Pour certaines actions, il est nécessaire de spécifier dans le fichier de configuration les permissions nécessaires
  - Appel téléphonique, accès réseau, etc.
- Exemple

```
<manifest ...  
  
<uses-permission  
    android: name="android.permission.CALL_PHONE" />  
  
</manifest>
```

# Filtrer les actions



- Les applications Android peuvent spécifier la liste des actions prises en charge : qu'une application peut exécuter
  - Ce sont les Intent-filter → balise <intent-filter>
    - Utilisés par le système pour sélection l'activité qui peut satisfaire un Intent
  - Les éléments de Intent-filter
    - action : identifiant unique sous forme de chaîne de caractères
      - Les actions principales peuvent être:
      - ACTION\_MAIN: lancement de l'activité en tant que principale, sans entrées ni sorties
      - ACTION\_EDIT: modifie une valeur
      - ACTION\_VIEW: affiche une valeur

# Filtrer les actions



- Les applications Android peuvent spécifier la liste des actions prises en charge : qu'une application peut exécuter
  - Ce sont les Intent-filter → balise <intent-filter>
    - Action
    - category
      - Chaîne de caractères contenant des informations supplémentaires concernant le composant

# Filtrer les actions



- Les applications Android peuvent spécifier la liste des actions prises en charge : qu'une application peut exécuter
  - Ce sont les Intent-filter → balise `<intent-filter>`
    - category

Valeurs Constantes	Signification
CATEGORY_BROWSABLE	The target activity can be safely invoked by the browser to display data referenced by a link — for example, an image or an e-mail message
CATEGORY_GADGET	The activity can be embedded inside of another activity that hosts gadgets.
CATEGORY_HOME	The activity displays the home screen, the first screen the user sees when the device is turned on or when the Home button is pressed.
CATEGORY_LAUNCHER	The activity can be the initial activity of a task and is listed in the top-level application launcher.
CATEGORY_PREFERENC	The target activity is a preference panel.

# Filtrer les actions

- **Exemple**



```
<activity ...  
<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="demo" />  
</intent-filter>  
  
</activity>
```

# Réagir à la réception d'un Intent

- **Exemple**



```
@Override  
  
public void onCreate(Bundle savedInstanceState) {  
    ...  
  
    String data = getIntent().getDataString();  
    if(data != null)  
        // traiter l'intent : réagir  
    ...  
}
```

# Permissions



- Certaines opérations sont réalisables à condition d'en obtenir la permission
  - Opérations pouvant entraîner un surcoût (connexion, échange de données, envoi de SMS, etc.)
  - Utilisation de données personnelles (accès aux contacts, au compte Google, exploitation des informations linguistiques entre autres)
  - Accès au matériel du téléphone (accès à l'appareil photo, écriture sur la carte mémoire...)
- Pour utiliser les fonctionnalités liées à de telles permissions, il est nécessaire de déclarer leur utilisation dans le fichier de configuration
- À l'installation d'une application, l'utilisateur disposera d'un récapitulatif de toutes les permissions demandées pour que l'application fonctionne
  - Il pourra alors choisir de continuer ou d'interrompre l'installation en connaissance de cause.



# Permissions



- Pour autoriser une application à accéder à certaines ressources il faut lui en donner l'autorisation par une balise `<uses-permission>`
  - Exemple
    - **Accès aux données personnelles**

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

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

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

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

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

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