



# Développement d'Application Android

Mr BELLO Daouda Adébayo

Ingénieur en développement d'application

# Objectif du cours

- Comprendre l'architecture du système Android
- Comprendre l'organisation d'une application Android
- Développer et déployer des applications natives Android
- Connaître les bonnes pratiques du développement d'application Android

# Référence

- L'art du développement Android par **Mark L. Murphy**
- Professional NFC application development for android by **Vedat**
- <https://fr.wikipedia.org/wiki/Android>
- Android developer : <http://developer.android.com/index.html>

# Contrainte

- Tenir compte du matériel :
- Un processeur faible; Une RAM limitée;
- Plusieurs type de résolutions de l'écran;
- Des coûts élevés de transferts de données;
- Des connexions réseau moins stables;
- Efficacité : optimiser votre code afin qu'il soit rapide et réactif.

La performance; La réactivité; La sécurité; La transparence;

# Pourquoi Android

Plusieurs langages :



Multi OS



# Pourquoi Android

Plusieurs outils :



Déploiement facile (multi plateforme)



# Historique et Définition

- Android est un système d'exploitation OPEN SOURCE pour terminaux mobiles (smartphones, PDA, tablet, ...)
- Conçu à la base par une startup (Android) et rachetée par Google en 2005
- Pour la promotion de ce système Google a fédéré autour de lui une trentaine de partenaires réunis au sein de l'Open Handset Alliance (OHA)
- C'est aujourd'hui le système d'exploitation mobile le plus utilisé à travers le monde

# ANDROID VERSIONS LIST: A COMPLETE HISTORY & FEATURES



Cupcake  
1.5



Donut  
1.6



Eclair  
2.0/2.1



Froyo  
2.2



Gingerbread  
2.3



Honeycomb  
3.0/3.1



Ice Cream Sandwich  
4.0



Jelly Bean  
4.1/4.2/4.3



KitKat  
4.4



Lollipop  
5.0



Marshmallow  
6.0



Nougat  
7.0



Oreo  
8.0



Pie  
9.0



android  
10



android

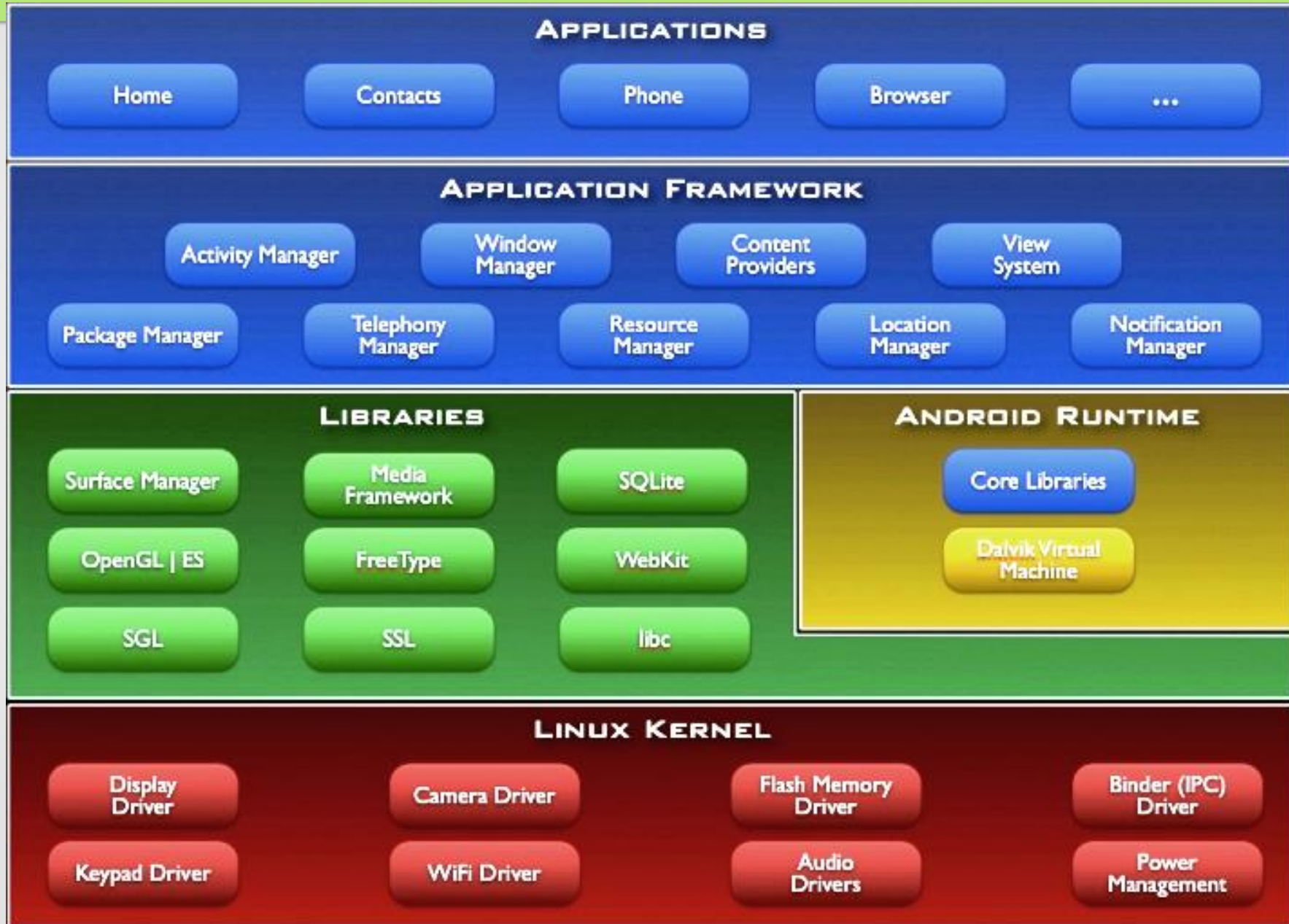


# Plateforme Android

Le système d'exploitation Android est basé sur Linux. Au plus bas niveau de ce système se trouve un noyau Linux destiné à la gestion du matériel comme :

- Drivers de ces terminaux,
- La gestion de la mémoire,
- La gestion des processus
- L'accès au réseau
- ...

# Plateforme et composante



# Android pour tous

## Développeurs :

- Pas besoin de licence
- Simple et intuitifs
- Modulables

## Constructeurs :

- Tous les constructeurs peuvent utiliser Android
- Un ensemble de services sont déjà disponibles dans le core
- API disponible pour les accès de bas niveau

# Programmation en Android

Android n'est pas un langage de programmation

Pour développer sous Android, il existe deux possibilités :

- Développement native (**Java** ou **Kotlin**)
- Développement hybride

# Programmation en Android

- ❑ Sources Java (ou Kotlin) compilés pour une machine virtuelle appelée «ART », amélioration de l'ancienne machine « Dalvik » (versions 4.4).
- ❑ Fichiers appelés **ressources** :
  - format XML : interface, textes. . .
  - format PNG : icônes, images. .
- ❑ Manifeste = description du contenu du logiciel
  - fichiers présents dans l'archive
  - demandes d'autorisations
  - signature des fichiers, durée de validité,

# Programmation en Android

Tout cet ensemble est géré à l'aide d'un **IDE** (environnement de développement) appelé **Android Studio** qui s'appuie sur un ensemble logiciel (bibliothèques, outils) appelé **SDK Android**

# Android et Java

- Le SDK Android est développé en Java → □ Permet de développer des applications avec un haut niveau d'abstraction
- Android a sa propre machine virtuelle (DVM)
- Ne supporte pas toutes les fonctionnalités de la JRE
- Une application Android ne peut pas s'exécuter sur une machine virtuelle Java
- Une application Java (native) ne peut pas s'exécuter sous Android
- Android dispose de sa propre machine virtuelle

# Android et Kotlin

- C'est un langage de programmation concurrent et symbiotique de Java : un programme Kotlin est compilé dans le même langage machine que Java, un programme Kotlin peut utiliser les classes Java et réciproquement. On peut mélanger des sources Java et Kotlin dans une même application
- Kotlin est promu par Google parce qu'il permet de développer des programmes plus sains. Par exemple, Kotlin oblige à valider chaque appel de méthode sur des variables objets pouvant valoir null, ce qui évite les NullPointerException.



# Vérification des variables objet

En Java :

```
void getNomComplet(Personne p) {  
    return p.getPrenom()+" "+p.getNom();  
}  
private Personne p1 = getPersonne(); // retourne null  
System...println(getNomComplet(p1)); // NullPointerException
```

En Kotlin :

```
fun getNomComplet(p: Personne): String {  
    return p.prenom+" "+p.nom  
}  
var p1: Personne = getPersonne() // retour null interdit  
println(getNomComplet(p1)) // ne se compile pas
```

# Android et développement hybride

Android supporte le développement hybride

- Titanium
- Phonegap
- Neomad

# Dans ce cours

## Framework et langage

- Android SDK
- Java
- XML

## Outils

- Android Studio

# L'environnement Android Studio

• Depuis mi-2015, il faut utiliser Android Studio (IDE IntelliJ ayant subi l'intégration de fonctionnalités de développement Android).

# Avantage d'Android Studio

- meilleur intégration du SDK dans Android Studio
- puissance de l'IDE IntelliJ

# Inconvénient Android Studio

- .Lourdeur de l'IDE IntelliJ
- .Moins d'outils standalone (gestion des émulateurs, du SDK)
- .Obligation de s'habituer à un autre IDE
- .Nouvelle architecture des répertoires
- .Meilleure gestion des dépendances avec gradle

# Android SDK

Outil de développement d'application Android

Fournit un environnement unifié permettant de développer « facilement » des applications Android

Mise à jour automatique via le SDKManager Prise en charge de toutes les versions d'Android Plusieurs Outils et API

- Android xx
- Google API xx
- Outils d'administration, de débogage ...



# Installation de Android Studio



### Default Settings

#### Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location:  [Edit](#)

**SDK Platforms** | SDK Tools | SDK Update Sites

Each Android SDK platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
<b>Android API 27</b>			
<input checked="" type="checkbox"/> Android SDK Platform 27	27	1	Installed
<input type="checkbox"/> Sources for Android 27	27	1	Not installed
<input type="checkbox"/> Android TV Intel x86 Atom System Image	27	2	Not installed
<input type="checkbox"/> Google APIs Intel x86 Atom System Image	27	2	Not installed
<input type="checkbox"/> Google Play Intel x86 Atom System Image	27	2	Not installed
<input type="checkbox"/> <b>Android 8.0 (Oreo)</b>			
<input type="checkbox"/> <b>Android 7.1.1 (Nougat)</b>			
<b>Android 7.0 (Nougat)</b>			
<input type="checkbox"/> Google APIs	24	1	Not installed
<input checked="" type="checkbox"/> Android SDK Platform 24	24	2	Installed
<input type="checkbox"/> Sources for Android 24	24	1	Not installed
<input type="checkbox"/> Android TV Intel x86 Atom System Image	24	15	Not installed
<input type="checkbox"/> ARM 64 v8a System Image	24	7	Not installed
<input type="checkbox"/> ARM EABI v7a System Image	24	7	Not installed
<input type="checkbox"/> Intel x86 Atom System Image	24	8	Not installed
<input checked="" type="checkbox"/> Intel x86 Atom_64 System Image	24	8	Installed
<input type="checkbox"/> Google APIs ARM 64 v8a System Image	24	20	Not installed
<input type="checkbox"/> Google APIs ARM EABI v7a System Image	24	20	Not installed
<input type="checkbox"/> Google APIs Intel x86 Atom System Image	24	20	Not installed

Show Package Details

Help

OK

Cancel

Apply

# Dossiers du SDK

Le gestionnaire installe plusieurs Go dans différents sous-dossiers :

- SDK Tools : indispensable, contient le gestionnaire,
- SDK Platform-tools : indispensable, contient adb,
- SDK Platform : indispensable, contient les librairies,
- System images : pour créer des AVD,
- Android Support : divers outils pour créer des applications,
- Exemples et sources.

# Compilation en Android

- La compilation sous Android avec le plugin ADT est automatisée.
- Android SDK utilise aussi ANT pour automatiser la compilation.
- Android 2.2 Froyo intègre le compilateur JIT(Just in Time compiler) stable.

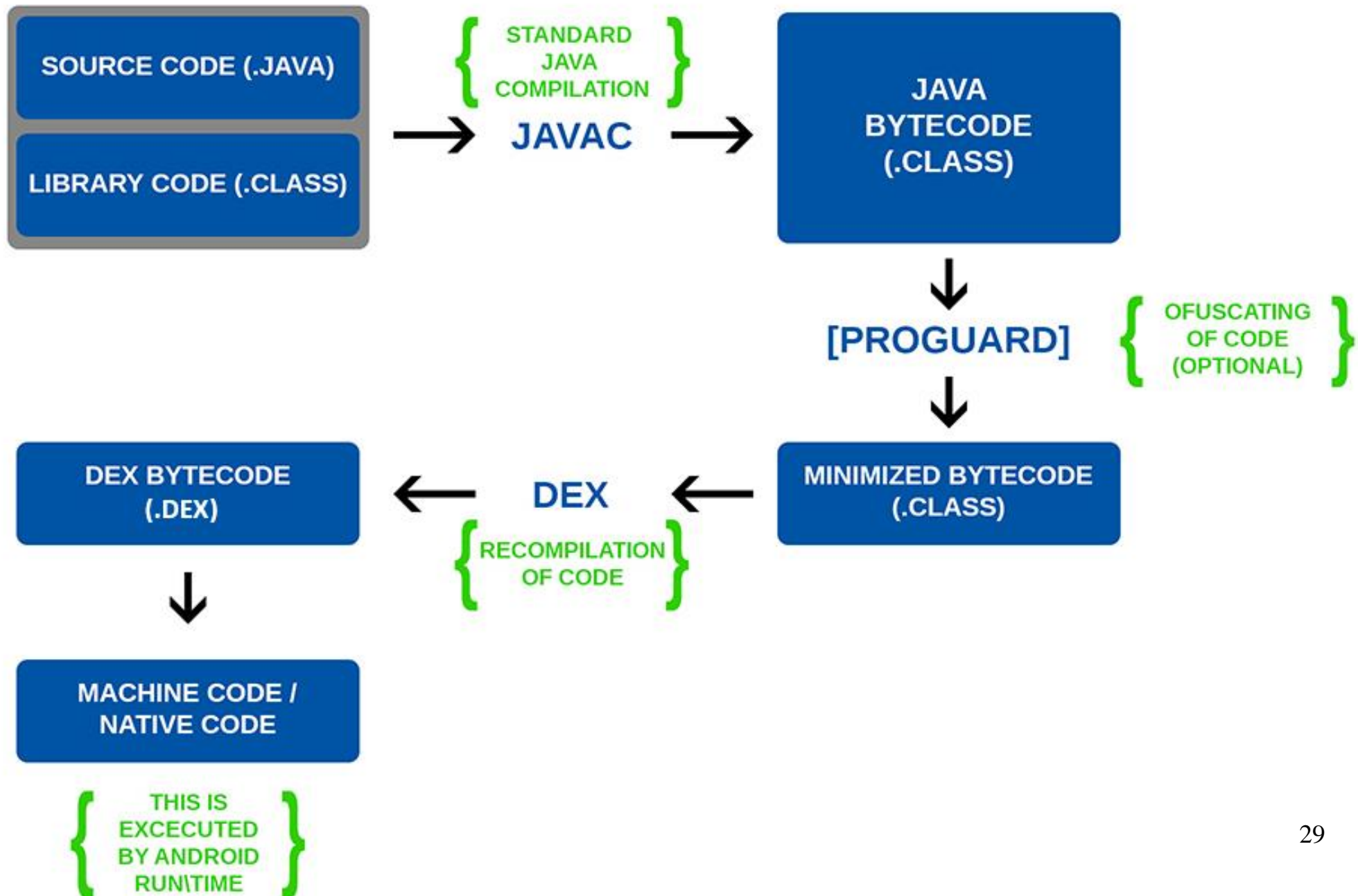
# Dalvik et ART

.[Dalvik] est le nom de la machine virtuelle open-source utilisée sur les systèmes Android. Cette machine virtuelle exécute des fichiers .dex, plus ramassés que les .class classiques. Ce format évite par exemple la duplication des String constantes. La machine virtuelle utilise elle-même moins d'espace mémoire et l'adressage des constantes se fait par un pointeur de 32 bits.

.[Dalvik] n'est pas compatible avec une JVM du type Java SE ou même Java ME. La librairie d'accès est donc redéfinie entièrement par Google.

.[A partir de Lollipop, Android dispose d'ART qui compile l'application au moment du déploiement (Ahead-of-time compilation).

# ANDROID COMPILATION PROCESS



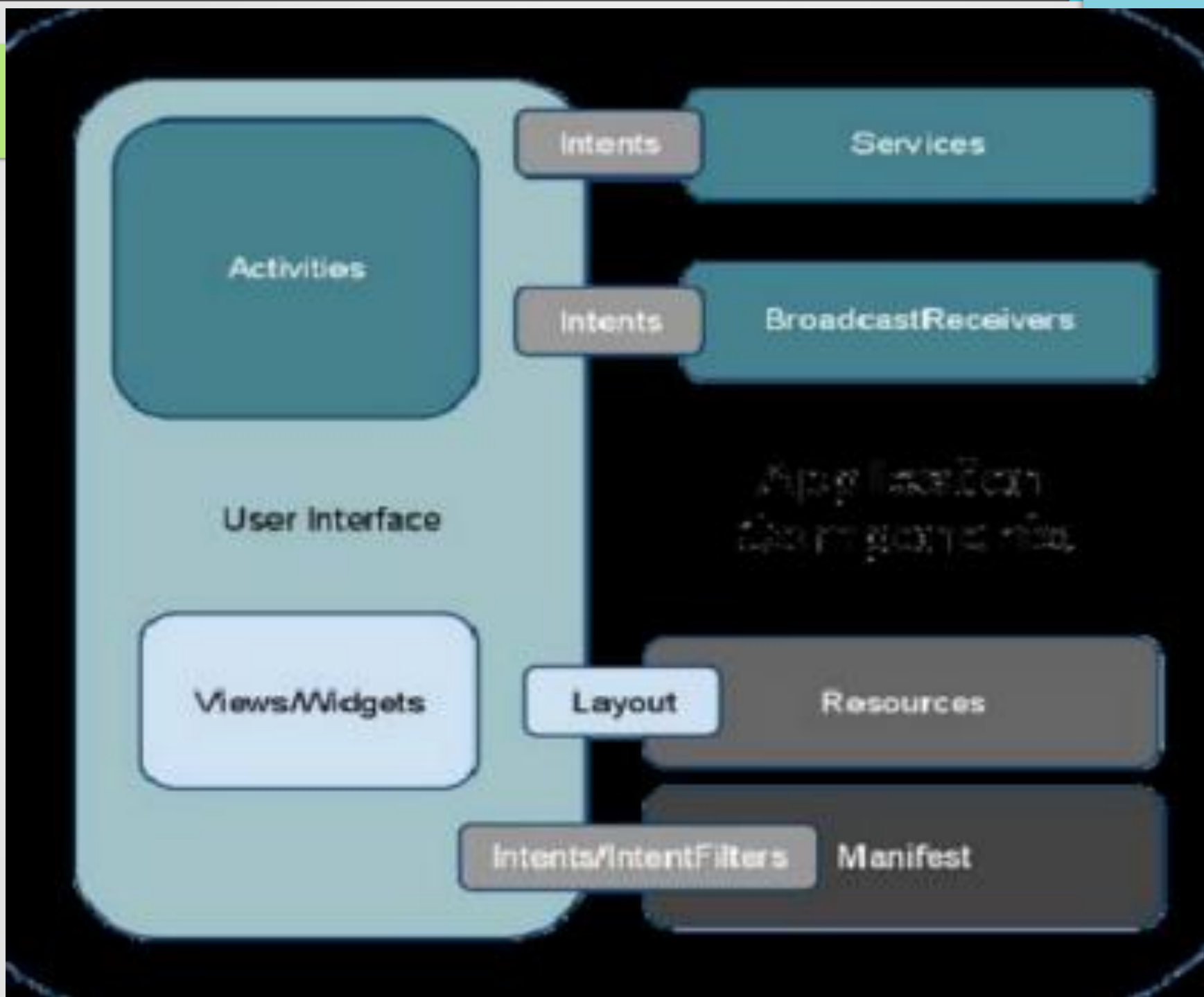
# Composante d'une application

## Android

Les applications Android sont constitués de composants à couplage

Les composantes sont liées par un Manifest qui décrit chacun d'eux ainsi que les interactions entre elles.





# Composante d'une application Android

1. **Activities** : Couche de présentation de l'application;
2. **Services** : les composants qui tournent en arrière plan;
3. **Content providers** : Partage de contenus entre applications;
4. **Intents** : Framework de communication interapplications.
5. **Broadcast receivers** : Consommateurs des messages diffusés par les intents.
6. **Widgets** : Composant d'application visuels;
7. **Notifications** : Framework de notifications aux utilisateurs



# Type de Projet Android

Il existe trois catégories de projets sous Android:

**Application Android:** Type primaire des applications Android destiné à être exécuté directement sur un terminal

**Test Projects:** Projet de test d'une application Android

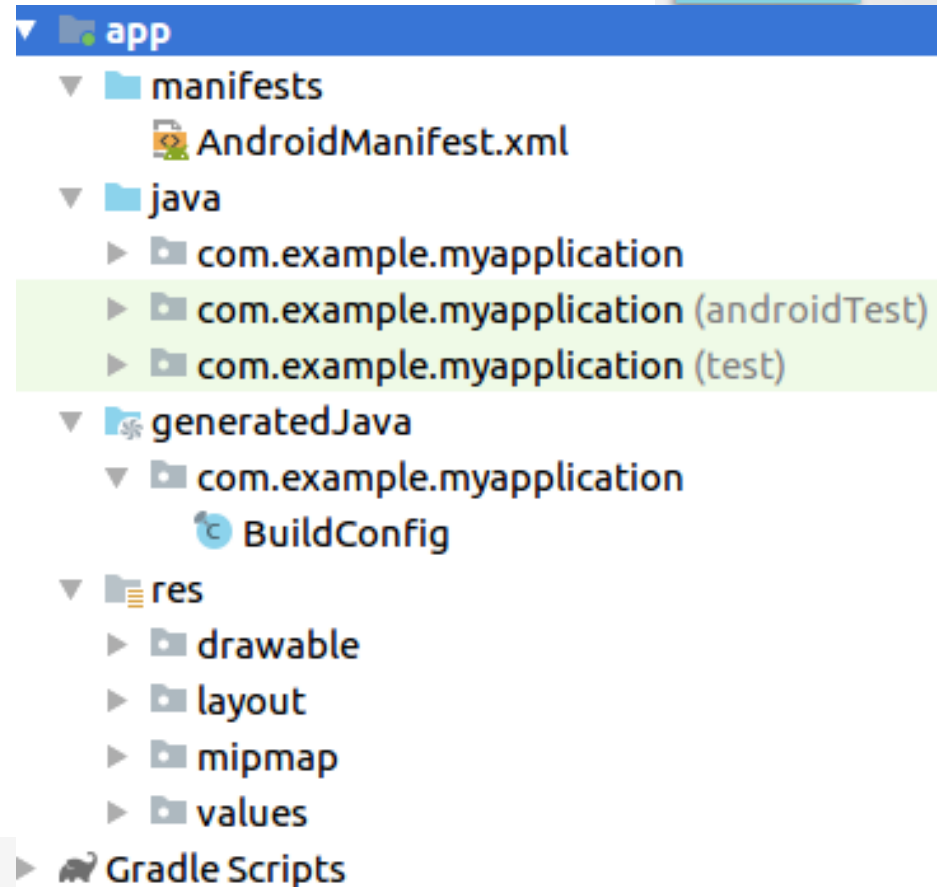
**Library project:** Projet de types bibliothèques, équivalents à une API exposant certaines fonctionnalités pouvant être réutilisés par d'autres applications.

# Type de Projet Android

- 1. Application de premier plan:** c'est une application qui est utilisable uniquement lorsqu'elle est visible et mise en suspens lorsqu'elle ne l'est pas;
- 2. Application d'arrière plan (Services):** N'interagit pas avec l'utilisateur, elle s'exécute en tâche de fond.
- 3. Intermittente :** c'est une application qui présente une certaine interactivité mais effectue l'essentiel de sa tâche en arrière plan. Ces applications notifient l'utilisateur lorsque cela est nécessaire;
- 4. Widget :** ces applications représentées sous forme d'un widget de l'écran d'accueil;

# Structure d'une application Android

```
+-- app/  
|   +-- build/  
|   +-- build.gradle  
|   |-- src/  
|       +-- androidTest/  
|       +-- main/  
|           |   +-- AndroidManifest.xml  
|           |   +-- java/  
|           |       |-- res/  
|           |-- test/  
+-- build/  
+-- build.gradle  
|-- gradle/
```



# Android Manifest

Chaque projet Android contient un fichier au format XML nommé manifeste (AndroidManifest.xml).

Le manifeste permet de définir la structure et les métadonnées d'une application, ses composants et ses prérequis.

Il contient des balises pour chacun des composants qui constituent l'application (Activities, Services, Content Providers et Broadcast Receivers)

# Android Manifest

Le manifeste fournit aussi des attributs permettant de spécifier les métadonnées d'une application (Exemple son icône, son thème) .

Il contient également les paramètres de sécurité, les tests unitaires et la définition des prérequis matériels et de plateforme.

Le fichier manifeste (AndroidManifest.xml) se trouve à la racine du projet.

# Android Manifest

The screenshot displays the Android Studio interface for editing the AndroidManifest.xml file of an application named 'ClientRestAndroid Manifest'. The main window is titled 'Android Manifest' and contains the following sections:

- Manifest General Attributes:** This section defines general information about the AndroidManifest.xml. It includes several input fields:
  - Package:** com.edou.gmail.android.rest (with a 'Browse...' button)
  - Version code:** 1
  - Version name:** 1.0 (with a 'Browse...' button)
  - Shared user id:** (with a 'Browse...' button)
  - Shared user label:** (with a 'Browse...' button)
  - Install location:** (with a dropdown arrow)
- Manifest Extras:** This section allows for adding, removing, or reordering extras. It features a toolbar with icons for 'U' (Uses Sdk), 'S' (Screens), 'P' (Permissions), 'C' (Categories), 'U' (Uses Permissions), 'P' (Permissions), 'L' (Languages), and 'Az' (Languages). Below the toolbar, there is a list containing one item: 'Uses Sdk'. To the right of the list are buttons for 'Add...', 'Remove...', 'Up', and 'Down'.
- Exporting:** This section provides instructions for exporting the application for distribution. It states: 'To export the application for distribution, you have the following options:' followed by two bullet points:
  - [Use the Export Wizard](#) to export and sign an APK
  - [Export an unsigned APK](#) and sign it manually

At the bottom of the window, there is a navigation bar with tabs for 'Manifest', 'Application', 'Permissions', 'Instrumentation', and 'AndroidManifest.xml'. The status bar at the very bottom shows '151M of 317M' and a trash icon.

# Activité (Activity)

Une activité décrit les actions que peuvent effectuer un utilisateur. Elles sont destinées à interagir avec l'utilisateur

Classe héritant de la classe Activity d'Android

Doit être déclarée dans le Manifest pour être visible par le système

Ne peut être instanciée directement, cette tâche se fait par le système

**Activity a = new Activity(); A ne pas faire !!!**

L'instanciation d'une activité se fait par les intentions

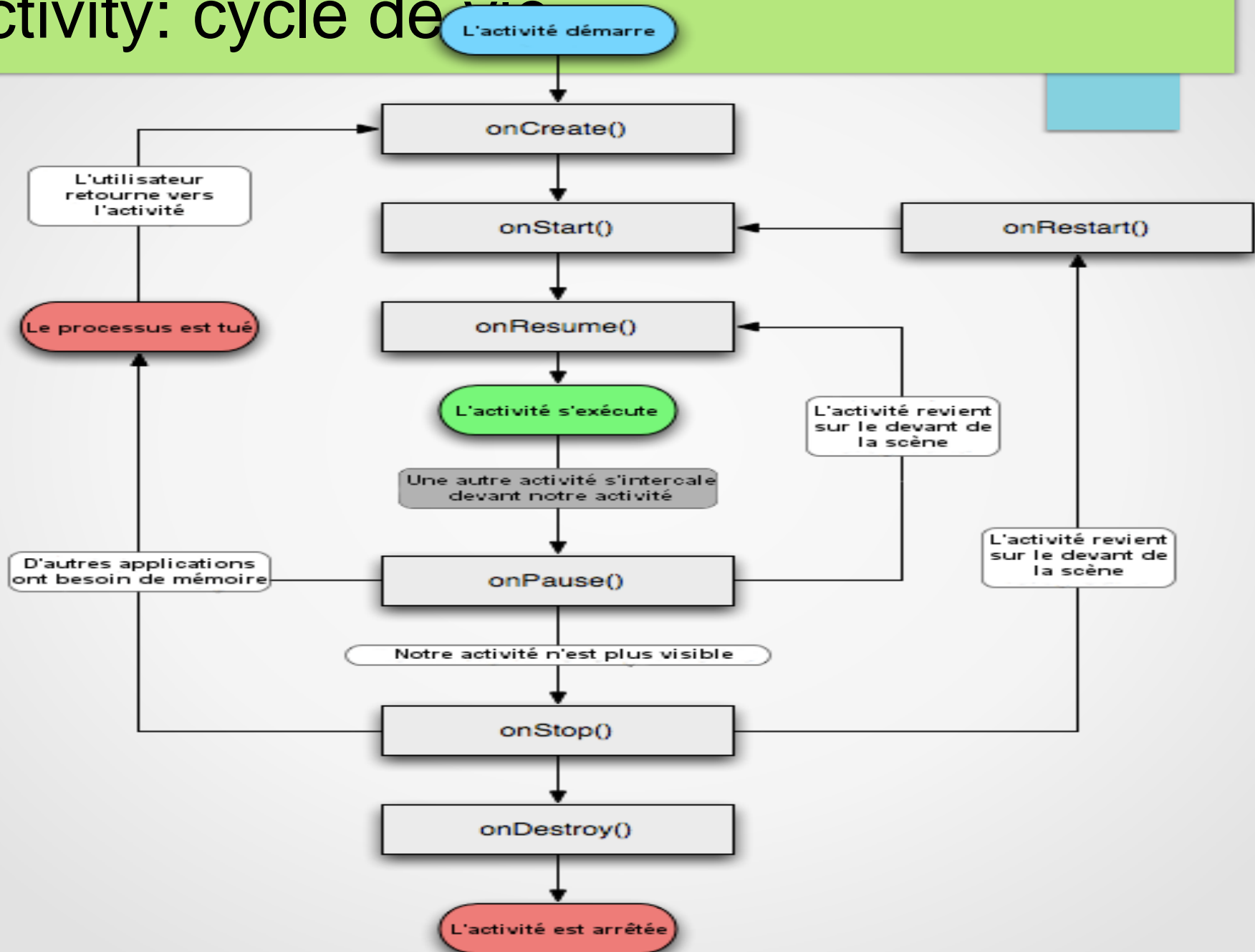
# Activity: cycle de vie

Une bonne compréhension du cycle de vie de l'Activity permet de garantir que votre application satisfait respecte les bonnes pratiques de développement Android.

1. L'état de chaque Activity est déterminé par sa position dans la pile des Activities:
2. Active : Quand l'Activity est au sommet de la pile;
3. En pause : Si l'Activity est visible sans avoir le focus;
4. Arrêtée: Lorsque l'Activity n'est plus visible;
5. Inactive : Quand elle est tuée ou qu'elle n'est pas démarrée;



# Activity: cycle de vie



# Activité / Création

```
public class ExampleActivity extends
    Activity {

    @Override

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

        // The activity is being created.

    }

    @Override

    protected void onStart() {

        super.onStart();

        // The activity is about to become visible.

    }
}
```

# Activité / Création

```
@Override
protected void onResume () {
    super.onResume ();
    // The activity has become visible (it is now
    "resumed").
}
@Override
protected void onPause () {
    super.onPause ();
    Another activity is taking focus (this activity is
    about to be
"paused").
}
}
```

# Contexte (context)

- Interface fournissant des informations globales sur l'environnement de l'application:
- C'est une classe abstraite implémentée par le système Android
- Il permet d'accéder aux principales ressources de l'application
- Obtenir le contexte courant d'une application
- Context c = getApplicationContext(): Contexte global de l'application
- Context c = Activity.this / getContext(): Contexte d'une activité ou
- un service

# Resolution des Intents

Il existe deux principaux types d'intentions :

- **Explicite** : Spécifie les composants qui précisent la classe exacte qui doit être exécutée (setComponant (nom) ou setClass(context, class))
- **Implicite** : Ne spécifie pas le composant mais fournit assez d'informations permettant au système de déterminer les composants nécessaires correspondant à cette action.

# Initialisations des Intents

```
Intent i = new Intent();
```

```
Intent i = new Intent(action:String)
```

```
Intent i = new Intent(action:String, uri:Uri)
```

```
Intent i = new Intent (context:Context,  
class:Class<?>)
```

# Les paramètres des Intents

Les intentions peuvent être utilisées pour transiter des données entre deux activités

`putExtra(nom:String, valeur:Object)`

`getxxxExtra(nom:String)`

dépend du type de données: `Int`, `String`, `StringArray`

# Action sur les Intents

`addCategory(category: String)`

`setDataAndType(uri:Uri, mime:String)`



# Quelques valeurs prédéfinis

## Actions

- `android.intent.action.CALL` appel téléphonique
- `android.intent.action.EDIT` affichage de données pour édition par l'utilisateur
- `android.intent.action.MAIN` activité principale d'une application
- `android.intent.action.VIEW` affichage de données
- `android.intent.action.WEB_SEARCH` recherche sur le WEB

# Quelques valeurs prédéfinis

## Catégories

- `android.intent.category.DEFAULT` activité pouvant être lancée explicitement
- `android.intent.category.BROWSABLE` peut afficher une information désignée par un lien
- `android.intent.category.LAUNCHER` activité proposée au lancement par Android
- `android.intent.category.TAB` activité associée dans un onglet d'interface (TabHost)

# Lancement d'une activité Intent

Lancer explicitement une activité Intent

```
demarre = new Intent(context,  
nomActiviteALancer.class);
```

```
startActivity(demarre);
```



# Première application

# Objectif

- Création d'une application « HelloWorld » avec un assistant,
- Tour du propriétaire,
- Exécution de l'application,
- Mise sous forme d'un paquet.

Android Studio contient un assistant de création d'applications  
**L'apparence change d'une version à l'autre du SDK.**

# Choix de la version

Chaque version d'Android, dénotée par son API level, ex: 25, apporte des améliorations et supprime des dispositifs obsolètes

Toute application exige un certain niveau d'API :

**Minimum SDK** : il faut au moins cette API car on utilise certaines classes et méthodes absentes des précédentes APIs,

Il y a aussi deux notions à connaître :

- **Target SDK** : l'application sera testée et marchera correctement jusqu'à ce niveau d'API,
- **Compile With** : c'est le niveau maximal de fonctionnalités qu'on se limite à employer. Si on fait appel à quelque chose de plus récent que ce niveau, le logiciel ne se compilera pas.



## Target Android Devices

### Select the form factors your app will run on

Different platforms may require separate SDKs

Phone and Tablet

Minimum SDK

Lower API levels target more devices, but have fewer features available.

By targeting API 15 and later, your app will run on approximately **96.2%** of the devices that are active on the Google Play Store.

[Help me choose](#)

Wear

Minimum SDK

TV

Minimum SDK

Android Auto

Glass

Minimum SDK

Previous

Next

Cancel

Finish

# Choix du type d'application

Create New Project

Add an activity to Mobile

Add No Activity

Blank Activity

Empty Activity

Fullscreen Activity

Google AdMob Ads Activity

Google Maps Activity

Previous Next Cancel Finish



# Points à configurer

L'assistant demande ensuite plusieurs informations :

- Nom de l'application, ex : HelloWorld,
- Nom de la classe principale : MainActivity,
- Nom du layout de la classe principale : activity\_main2,
- Nom du layout du menu principal : menu\_main.
- Tout peut être renommé ultérieurement, voir refactor/rename.

Le package du logiciel a été défini dans le premier écran.

# Noms des packages et classes

Create New Project

Customize the Activity

Creates a new blank activity with an app bar.

Blank Activity

Activity Name: MainActivity

Layout Name: activity\_main

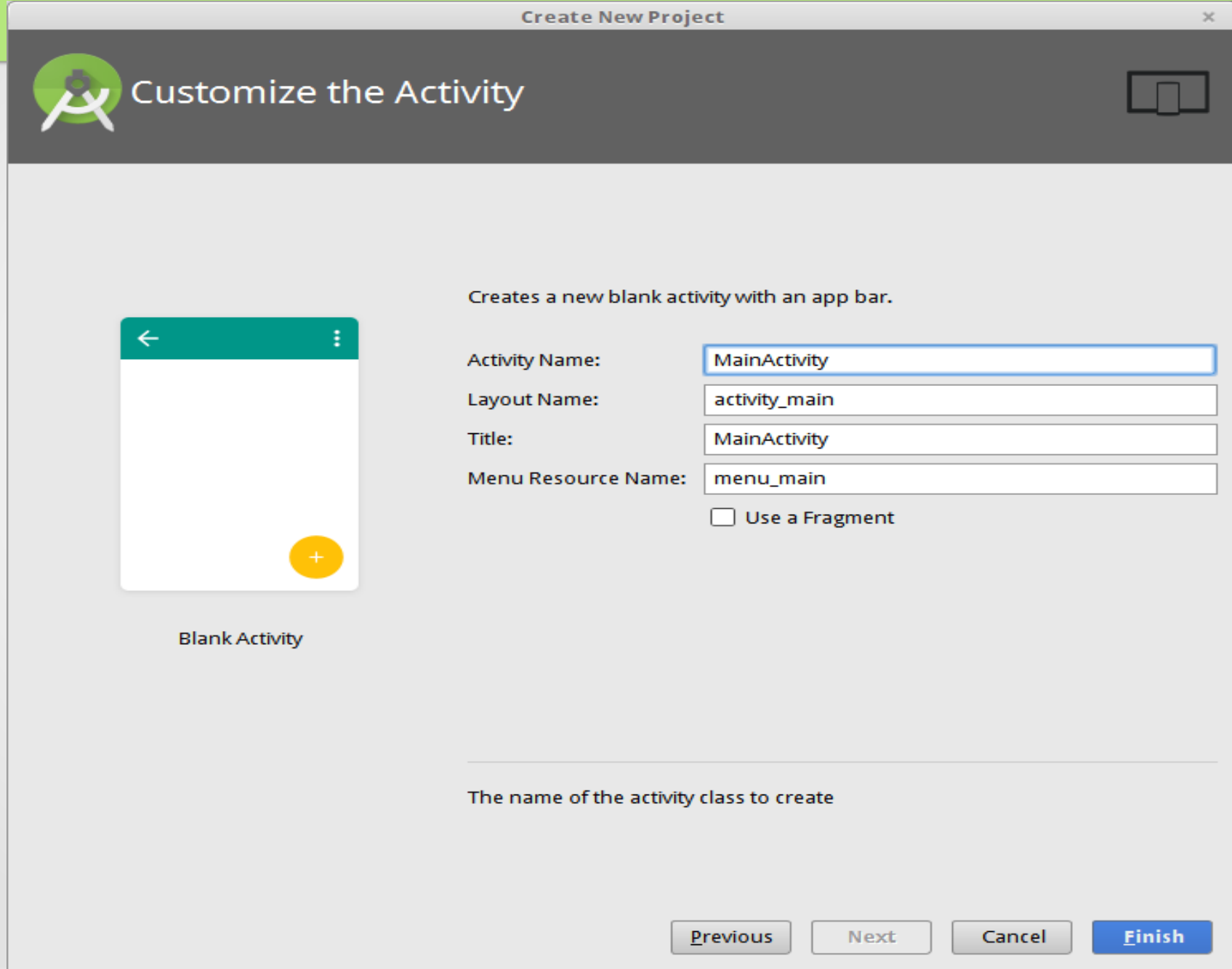
Title: MainActivity

Menu Resource Name: menu\_main

Use a Fragment

The name of the activity class to create

Previous Next Cancel Finish



# Résultat de l'assistant


L'assistant a créé de nombreux éléments visibles dans la colonne de gauche de l'IDE :

- manifests : description et liste des classes de l'application
- java : les sources, rangés par paquetage,
- res : ressources = fichiers XML et images de l'interface, il y a des sous-dossiers :
  - ❖ layout : interfaces (disposition des vues sur les écrans)
  - ❖ menu : menus contextuels ou d'application
  - ❖ mipmap et drawable : images, icônes de l'interface
  - ❖ values : valeurs de configuration, textes. . .
  - ❖ Gradle scripts : c'est l'outil de compilation du projet.

- app
  - manifests
    - AndroidManifest.xml
  - java
    - mementol.games.iutlan.fr.mementol
      - MainActivity
    - mementol.games.iutlan.fr.mementol (androidTest)
  - res
    - drawable
    - layout
      - activity\_main.xml
    - menu
      - menu\_main.xml
    - mipmap
      - ic\_launcher.png (5)
        - ic\_launcher.png (hdpi)
        - ic\_launcher.png (mdpi)
        - ic\_launcher.png (xhdpi)
        - ic\_launcher.png (xxhdpi)
        - ic\_launcher.png (xxxhdpi)
      - values
        - dims.xml (2)
        - strings.xml
        - styles.xml
  - Gradle Scripts
    - build.gradle (Project: MementOL)
    - build.gradle (Module: app)
    - gradle-wrapper.properties (Gradle Version)
    - proguard-rules.pro (ProGuard Rules for app)
    - gradle.properties (Project Properties)
    - settings.gradle (Project Settings)
    - local.properties (SDK Location)

# Éditeur du manifeste

MainActivity.java x strings.xml x Translations Editor x

+   Show only keys needing translations

Key	Default Value
action_set	Settings
app_name	MementOL