CAI / SMB116

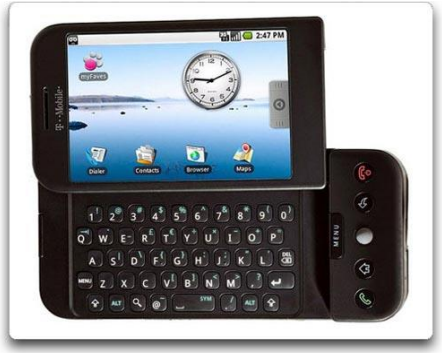
Conception et développement pour systèmes mobiles

Présentation d'Android

- Système d'exploitation développé par **l'Open Handset Alliance**
- Basé sur Kernel **Linux** (License Apache 2.0)
- Android est aussi un framework (SDK avec code ouvert)
- Possibilité de créer des versions personnalisées d'Android
- Possibilité de créer ses propres applications en réutilisant des composants existants

La genèse d'Android – Historique (1)

- Associé à **Google** mais son histoire commence bien avant
- Début du développement en 2003 par société éponyme
- Acquisition par Google en 2005 (un Google phone en projet ?)
- Le premier téléphone Google le G1 Phone sort en 2008
- Publication du SDK Android 1.0 pour le G1 Phone.



La genèse d'Android – Historique (2)

- A l'origine chaque constructeur de téléphone créait son propre Système
- Impossible de créer une application compatible avec tous les appareils
- En 2007 sortie de l'Iphone. IOS est moderne, ergonomique, en avance sur la concurrence.
- Réplique de la concurrence: 35 entreprises dont Google constituent le consortium « Open Handset Aliance »
- But du OHA: créer un Système open source pour révolutionner le marché et contrer Windows mobile mais surtout IOS.
- Publication du SDK Android 1.0 pour le G1 Phone en 2008

La genèse d'Android – Historique (3)

- **2008 :** la version 1.0 du SDK d'Android est publiée et le **G1 Phone** arrive sur le marché
- **2009 :** Apparition de nombreux équipements basés sur Android (20 nouveaux appareils) et de nouvelles versions du système sont publiées.
- **2010 :** Les appareils sous Android sont les plus vendus juste après les Black Berry (plus de 60 appareils utilisent Android 2.2)
- **2011 :** Android est le numéro 1 en nombre de nouvelles machines (Téléphone et Tablettes) vendues.
- **2017 : Android est utilisé sur 80% des smartphones dans le monde**
  http://www.journaldugeek.com/2017/02/17/android-ios-dominent-marche-smartphones/

La genèse d'Android – Android 1.0

- **Octobre 2008 :** Android 1.0 utilisée par G1 Phone (Fabriqué par HTC):
  - Pas de clavier virtuel ni gestion du multitouch
  - Apparition des Widgets
  - Android Market (ancêtre du Play Store)
  - Innovation : La barre de notification

La genèse d'Android – Android 1.x

- **Février 2009 :** Android 1.1 « Banana Bread » :
  - Correction des nombreux bugs de la 1.0
  - Mise à jour via OTA (Over-The-Air) par téléchargement en Wifi ou 3G
- **Avril 2009 :** Android 1.5 « Cupcake » :
  - Amélioration interface graphique
  - Arrivée du clavier tactile et du presse papier
  - Capture et publication de vidéos
- **Septembre 2009 :** Android 1.6 « Donut » :
  - Réseau CDMA (standard pour la 3G)
  - Ecran portrait / paysage, nouvelles définitions jusqu'à la HD (1280x720)
  - Amélioration interface APN

La genèse d'Android – Android 2.x

- **Octobre 2009 :** Android 2.0 « Eclair » + Maj vers 2.1 :
  - Amélioration clavier tactile
  - Fonds d'écran animés
  - Bluetooth 2.1
  - Google Map
  - Nouvelles fonctionnalités pour APN
  - Speech To Text et Synthèse vocale
- **Mai 2010 :** Android 2.2 « Froyo » , 2.3 « Gingerbread » :
  - Nombreuses améliorations pour le « Nexus One »
  - Flash Player, Ecran 320 PPI, Voip, NFC, Système de fichier ext4
  - Outils pour le développement en natif

La genèse d'Android – Android 3.x

- **Février 2011 :** Android 3.x « Honeycomp » réservé aux tablettes:
  - Boutons tactiles pour retour et menu
  - Evolution du multitâche
  - Redimensionnement des widgets
  - Support du format son FLAC
  - Prise en charge tablette 7''
  - Prise en charge processeurs **Qualcomm**

La genèse d'Android – Android 4.x

- **Octobre 2011 :** Android 4.0 « Ice Cream Sandwich »
  - Dictionnaire associé au clavier virtuel
  - Amélioration application photo
  - Capture d'écran native
  - Consultation statistique sur trafic de données
  - Ecran verrouillage « Cadenas »
  - Mise à jour Navigateur Web
  - Prise en charge enregistrement vidéo Full HD 1080P
  - Chiffrement des données
- **Juin 2012 :** Android 4.1, 4.2, 4.3 « Jelly Bean» utilisé par la tablette Nexus 7. 4.4 « Kitkat »
  - Accélération vitesse du rafraichissement
  - Assistant Google Now
  - Fonction Swipe : utilisation du clavier sans lever le doigt
  - Multi-utilisateur : plusieurs comptes pour des usages différents
  - Photosphere, Open GL ES 3.0, Réduction de la consommation

La genèse d'Android – Android 5.x / 6.x / 7.x

- **Octobre 2014 :** Android 5.0, 5.0.1, 5.0.2 et 5.1 «Lollipop»
  - Material Design (https://graphism.fr/comprendre-le-material-design-de-google/)
  - Amélioration sécurité du noyau
  - Economie d'énergie par amélioration de la veille
  - Abandon de la machine Dalvik au profit d'ART (Android Runtime)
- **Octobre 2015 :** Android 6.0 « Marshmallow»
  - Amélioration intégration de l'assistant Google now
  - Amélioration autonomie
  - Nouveau système de permissions et de nombreuses autres évolutions…
- **Août 2016 :** Android 7.0 « Nougat» (https://www.android.com/intl/fr_fr/versions/nougat-7-0/)
  - Multi-fenêtrage
  - Modification écran paramètres
  - Amélioration du multitâche
  - Blocage de numéros
  - Outils pour la réalité virtuelle

La genèse d'Android – Android 8.0 Oreo

- **Août 2017 :** Android 8.0 « Oreo »
  - Mode Picture in Picture
  - Regroupement des notifications par type
  - Les API d'auto-remplissage pour l'authentification
  - Simplification de l'accès à des polices personnalisées pour les applications
  - Icônes adaptatives
  - Connectivité: Sony LDAC, NAN (Neighborhood Aware Networking)
  - Amélioration des WebView et les API de Java 8

(http://www.frandroid.com/android/419174_android-o-developer-preview-quelles-sont-les-nouveautes)

La genèse d'Android – Android 9.0 Pie

- **Août 2018 :** Android 9.0 « Pie »
  - Fonction pour l'accessibilité: Menu adapté, Amplification son, Lecture de texte sélectionné
  - Economie d'énergie: contrôle applications trop gourmandes,
  - Digital Wellbeing: plages horaires avec limitation utilisation des applis et notifications
  - Multi utilisateurs
  - Améliorations pour lecture média (audio, vidéo, casques)
  - Gestion notification (autorisations), proposition de réponses
  - Nouvelles fonctions pour la sécurité

  (https://www.android.com/versions/pie-9-0/)

La genèse d'Android – Android 10.0 (Q)

- **3 septembre 2019 :** Android 10 « Q » (plus de dessert mais une lettre)
  - UI adapté pour les écrans pliables (redimensionnement automatique Plié/Déplié)
  - Accessibilité : transcription automatique en texte de tous les flux audio
  - WIFI : amélioration performances, mode P2P
  - Support de nouvelles caméras (Monochrome)
  - Améliorations codecs pour lecture média (audio, vidéo)
  - Panel de configuration : possibilité de modifier config sans quitter une application
  - Nouvelles fonctions pour la sécurité

(https://developer.android.com/about/versions/10/features)

La genèse d'Android – Android 11

- **8 septembre 2020 :** Android 11
  - Capture vidéo d'écran
  - Bulles de messagerie
  - Nouvelles fonctions pour la sécurité

(https://developer.android.com/about/versions/11/features)

La genèse d'Android – Android 12

- **18 février 2021 :** Android 12
  - Evolution UI
  - Support de plus de formats audio et vidéo
  - Amélioration vie privée (notification quand une application utilise le micro)
  - Son surround

(https://developer.android.com/about/versions/12/features)

La genèse d'Android – Android 12

- **15 août 2022 :** Android 13 (Tiramisu) La toute dernière version.
  - Sélecteur de média qui améliore la confidentialité (choix des médias accessibles)
  - Autorisation nécessaire pour les notifications
  - Prise en charge de Bluetooth LE Audio

  (https://developer.android.com/about/versions/13/features)

L’environnement de développement (1)

- **Java :** codes écrits en Java puis compilés en bytecode dans une première phase d’où besoin d’un JDK (Java Development Kit)

- **Android Studio:** Ide basé sur IntelliJ adapté pour développer des applications Android

L'environnement de développement (2)

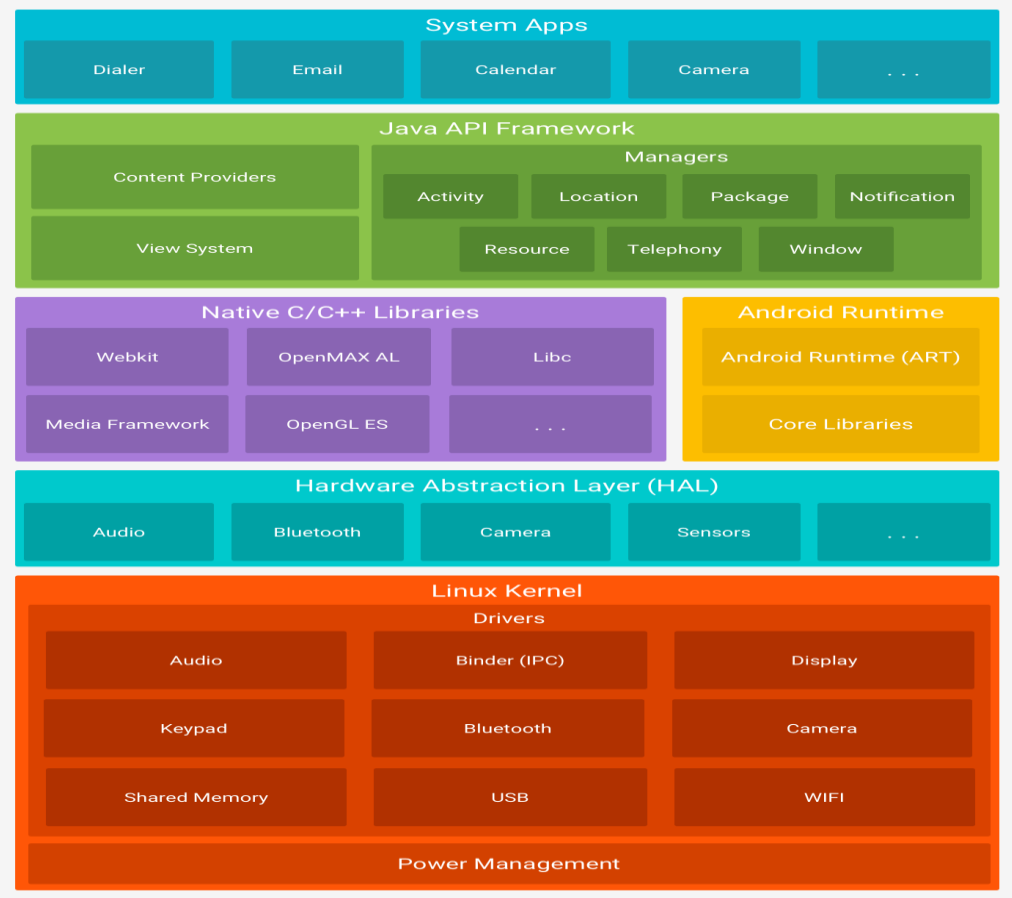
- **SDK Android:** Les outils nécessaires pour créer une application Android:
  - **L' AAPT** - Android Asset Packaging Tool : pour créer et analyser les archives apk
  - **L'ADB** - Android Debug Bridge : pour transférer une application sur un appareil Android ou un émulateur l'exécuter et la déboguer
  - **Le DDMS** - Dalvik Debug Monitor Service : Outil utilisé pour déboguer et monitorer les applications
    - Observer les threads en cours d'exécution
    - Consulter le logcat
    - lister les processus en cours d'exécution sur l'appareil
    - Simuler l'envoi de messages et d'appels
    - Simuler une localisation
    - Simuler des mesures comme si elles étaient fournies par les différents capteurs.
  - La **documentation** et des exemples d'utilisation des différentes API

L'environnement de développement (3)

- **Emulateur :** outil permettant de simuler un téléphone ou une tablette qui fonctionne sous Android. Il permet de tester et mettre au point une application durant son développement.
- **ART (Android RunTime) :** Machine virtuelle pour l'exécution d'applications Android (5.0). Utilise AOT compilation (Ahead of Time) par opposition à la Machine virtuelle Dalvik et sa compilation JIT (Just In Time). Amélioration du GC https://medium.com/android-news/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924
- **NDK (Native Development Kit) :** Outil pour le développement natif sous Android en langage C ou C++. Accès aux composants natifs en utilisant **JNI**

Architecture de la plateforme Android (1)

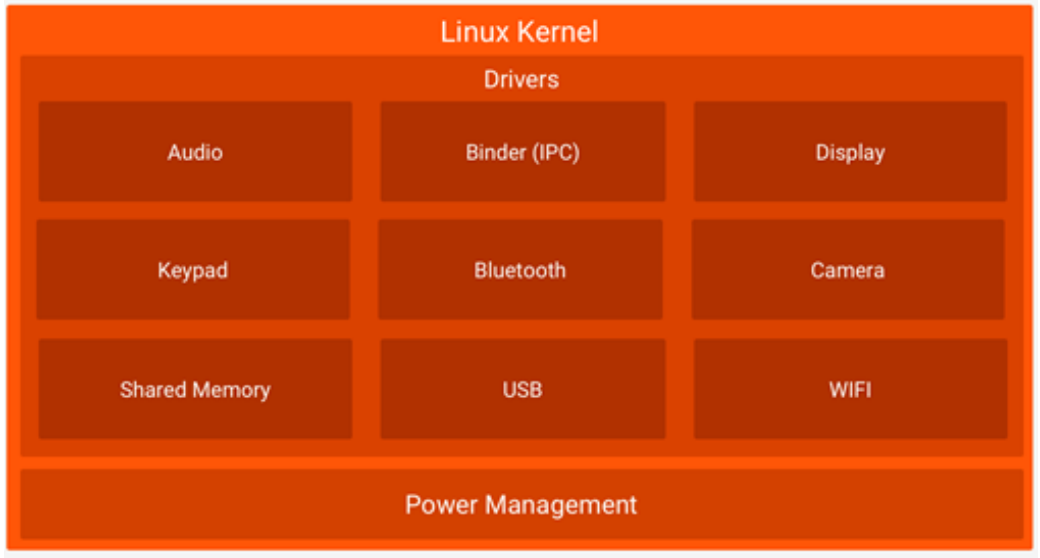
- **Android est basé sur Linux et est utilisable sur un grand nombre d'appareils et de définitions d'écrans.**



Architecture de la plateforme Android (2)

Le Noyau Linux

- **Linux** est la base de la plateforme Android.
- **ART** utilise les primitives Linux pour:
  - La gestion de la mémoire
  - L'accès au système de fichiers
  - La gestion des processus

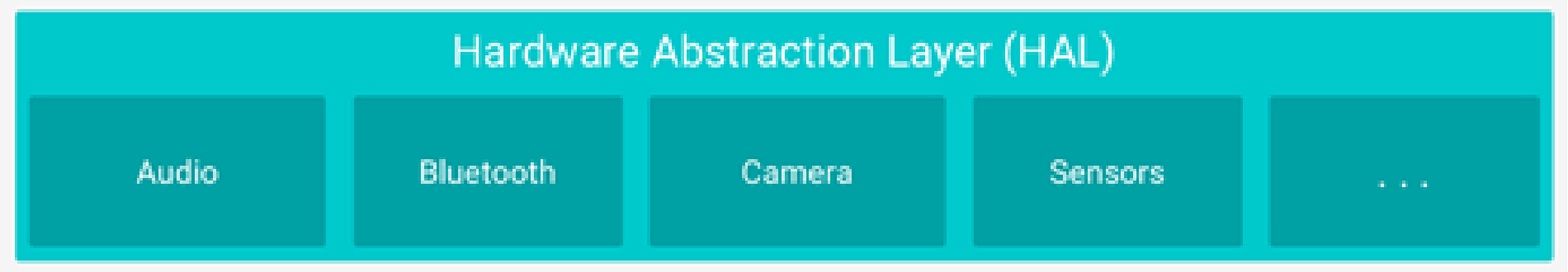


- **Avantages:**
  - Réutiliser les fonctionnalités de Sécurité
  - Facilité d'écriture de pilote par les fabricants de périphériques

Architecture de la plateforme Android (3)

HAL : Hardware Abstraction Layer

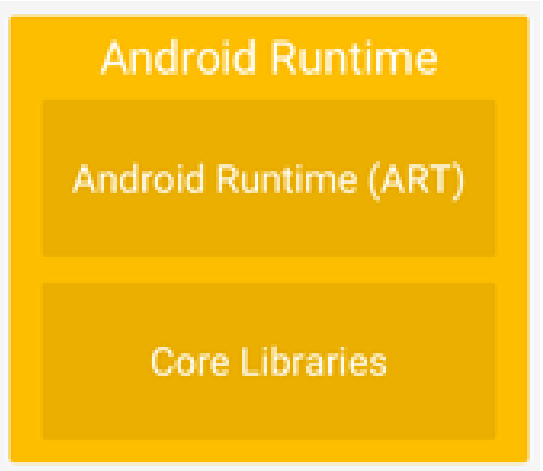
- **HAL :** couche pour permettre l'utilisation des périphériques hardware (son, Bluetooth, caméra, divers capteurs, etc.) par les Api du Framework Java pour Android.



Architecture de la plateforme Android (4)

ART : Android RunTime

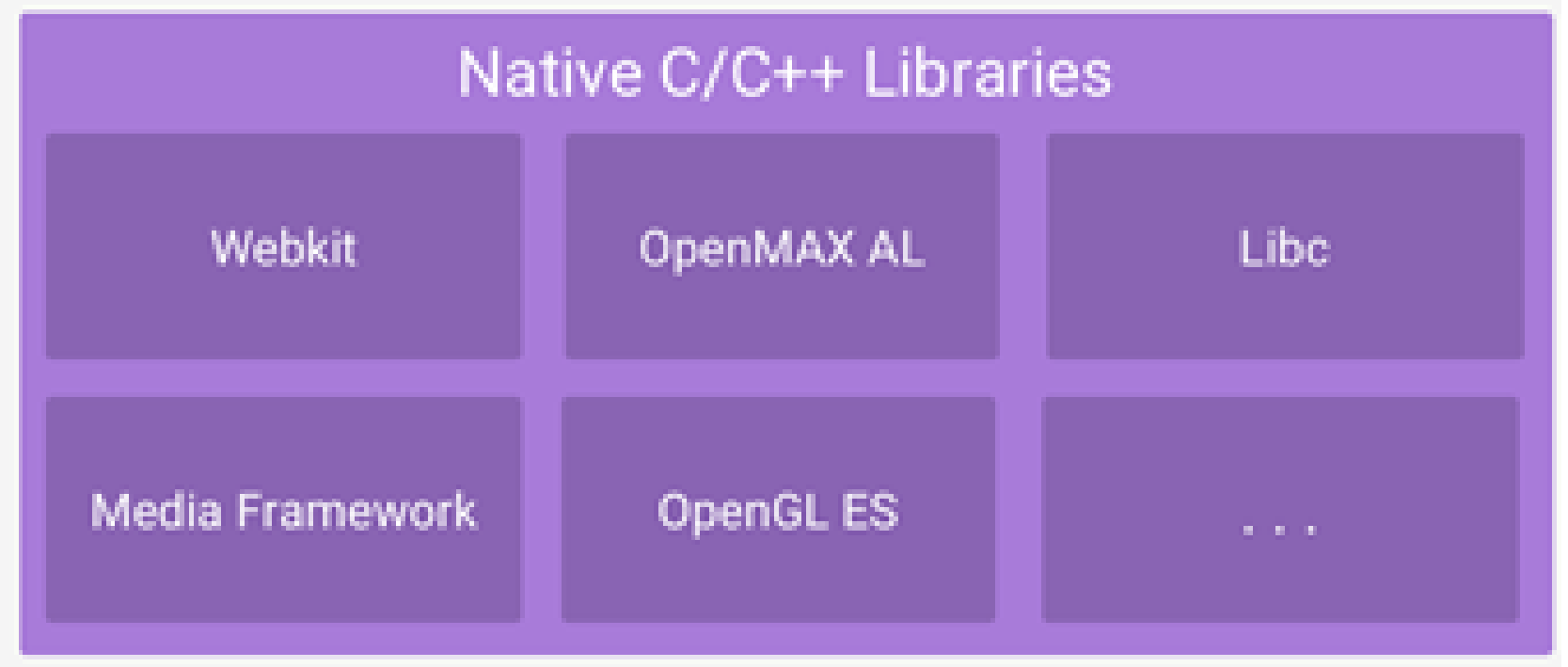
- **ART :** environnement d'exécution d'une application Android depuis la 5.0
- Chaque application s'exécute dans sa **« Sandbox »** son propre processus Linux.
- Exécution de fichiers de bytecode DEX créés à partir de sources Java compilés puis convertis
- Apports principaux de ART:
  - Compilation AOT (Ahead-Of-Time) et JIT (Just In Time) https://source.android.com/devices/tech/dalvik/jit-compiler
  - Ramasse miette optimisé
  - Amélioration possibilités de débogage et de profilage



Architecture de la plateforme Android (5)

Bibliothèques Natives C/C++

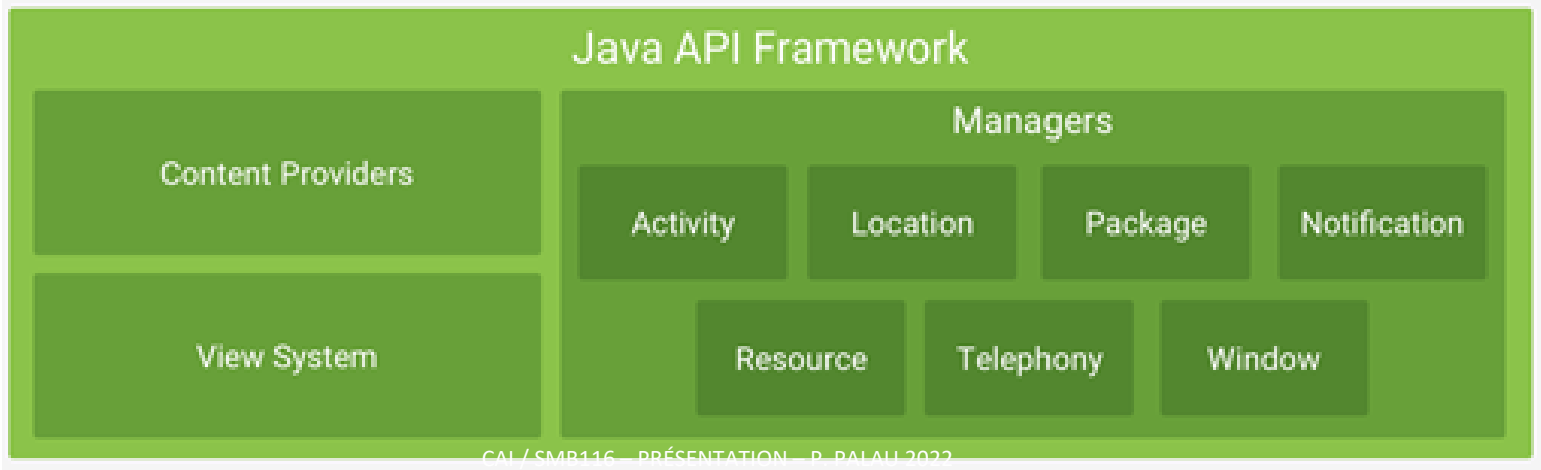
- Ensemble de bibliothèques natives écrites en C/C++ exploitées par ART et les Api du framework Java.



Architecture de la plateforme Android (6)

API Java

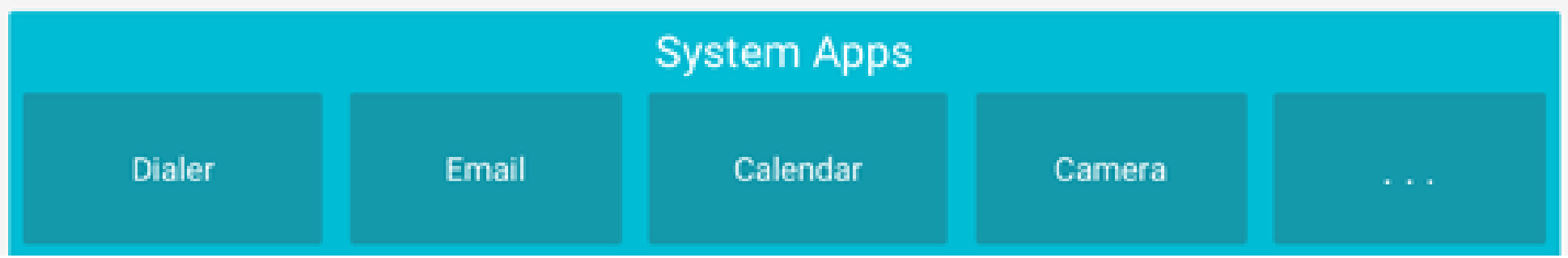
- Permet l'accès aux fonctionnalités du Fmk Android à partir des codes Java
  - **View System APIs :** pour créer des interfaces graphiques (Composants, Grilles, Layouts)
  - **Resource Manager APIs :** pour accéder aux ressources autres que le code (I18n, images etc.)
  - **Notification Manager :** pour permettre aux applications de créer des messages d'alerte (notification)
  - **Activity Manager :** fonctionnalités pour la gestion de la pile des « Activités »
  - **Content Providers :** fonctionnalités pour permettre aux applications de mettre à disposition leurs propres données ou d'accéder aux données des autres applications. *(exemple : Accès à la liste des contacts)*



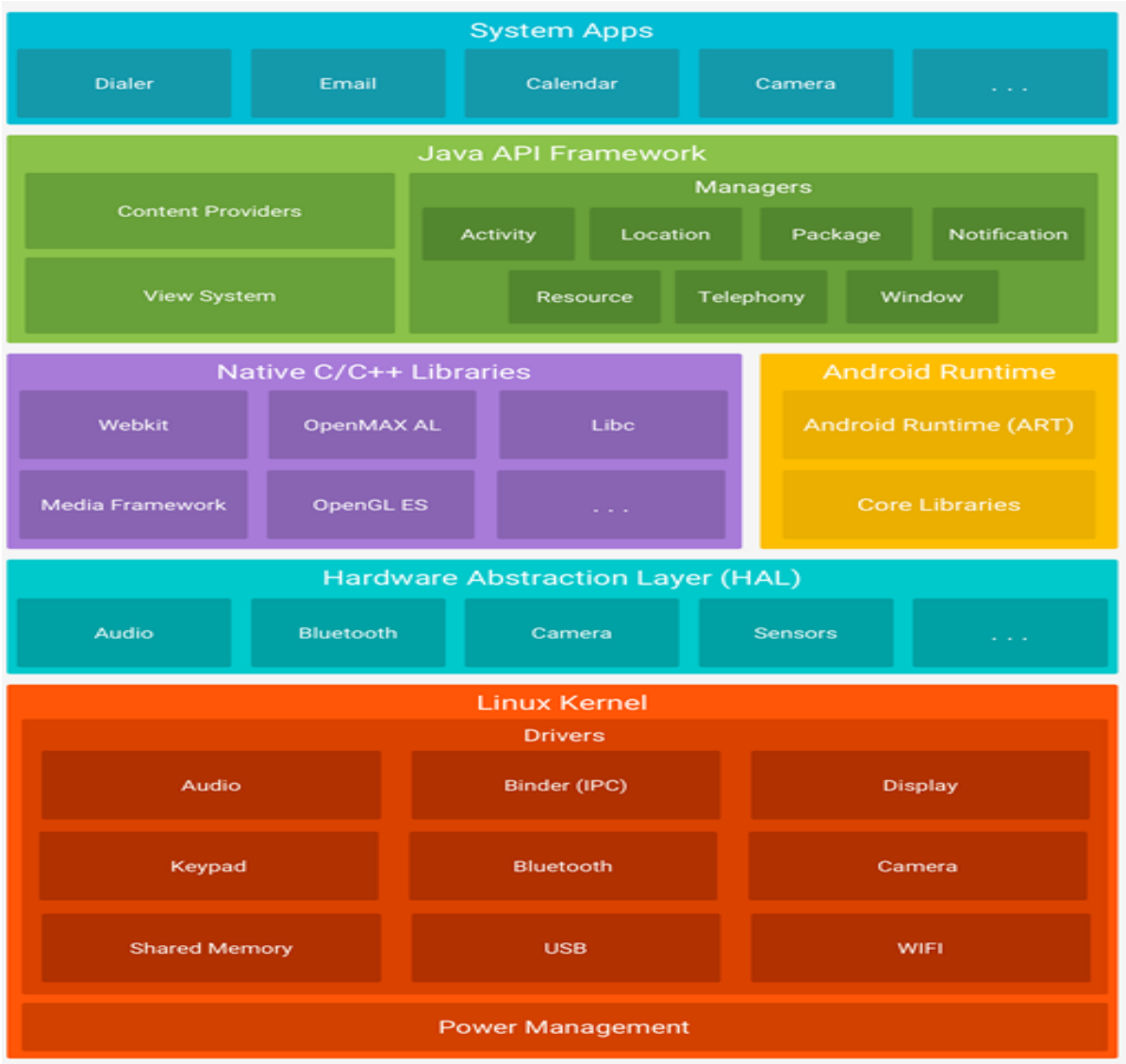
Architecture de la plateforme Android (7)

Applications systèmes

- Applications livrées avec Android:
- Types d'applications: gestion contacts, navigateur Web, communication par SMS etc.
- Applications identiques à celles créés par des développeurs
- Exploitables pour développer de nouvelles applications (réutilisation de fonctionnalités)



Architecture de la plateforme Android



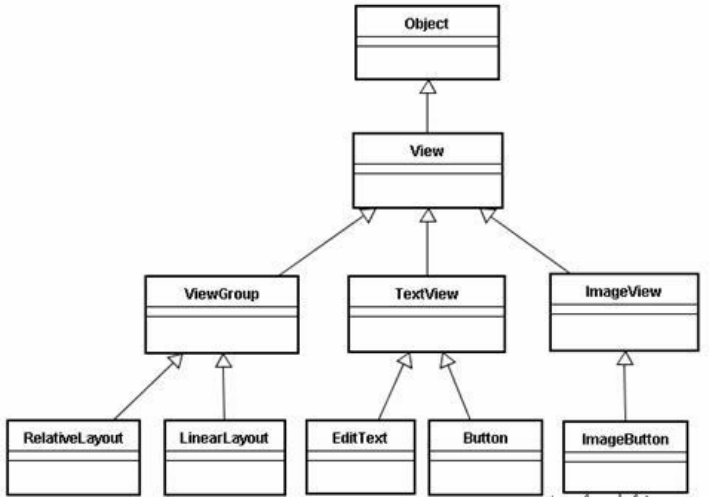
Les composants d'une application Android

- **View :** élément de l'interface graphique
- **Activity :** gestion des interactions entre utilisateur et l'application en utilisant la vue (associée à une IHM)
- **BroadcoastReceiver :** récepteur de messages de notifications
- **Service :** une tâche qui s'exécute en arrière plan sans IHM
- **ContentProvider :** persistance et mise à dispo des données d'une application pour d'autres applications
- **Intent et IntentFilter :** Un intent est un messages utilisés par les applications pour communiquer avec le système ou d'autres applications et services. Un IntentFilter permet de sélectionner les types de messages à recevoir.
- **Fragment :** comportement d'une portion de l'IHM d'une activité
- **Loader :** met à disposition des méthodes pour lire des données des ContentProvider ou d'autres sources.
- **AppWidget :** applications miniatures avec leur IHM (embarquable).
- **Processes and Threads :** une application s'exécute dans son propre Processus mais peut créer des Threads.

Les composants d'une application Android

View (1)

- Elément de l'interface graphique: bouton, liste, champ texte, navigateur, conteneur, layout etc…

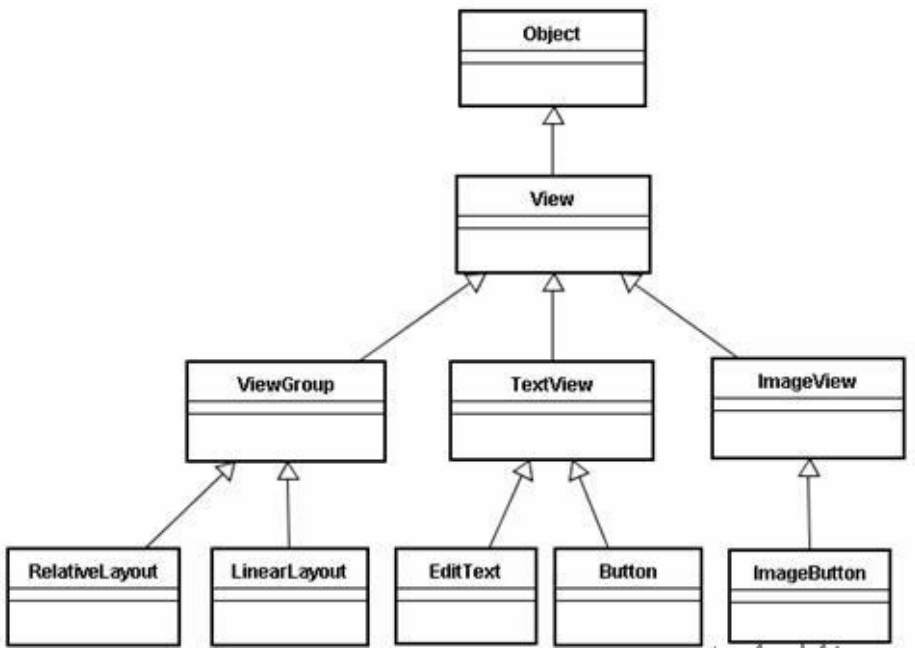


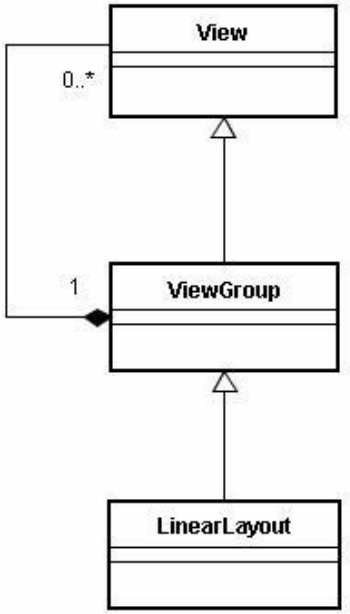
Que manque-t-il à ce modèle pour pouvoir créer une IHM comportant par exemple:

un bouton et un EditText ?

Les composants d'une application Android

View (2)



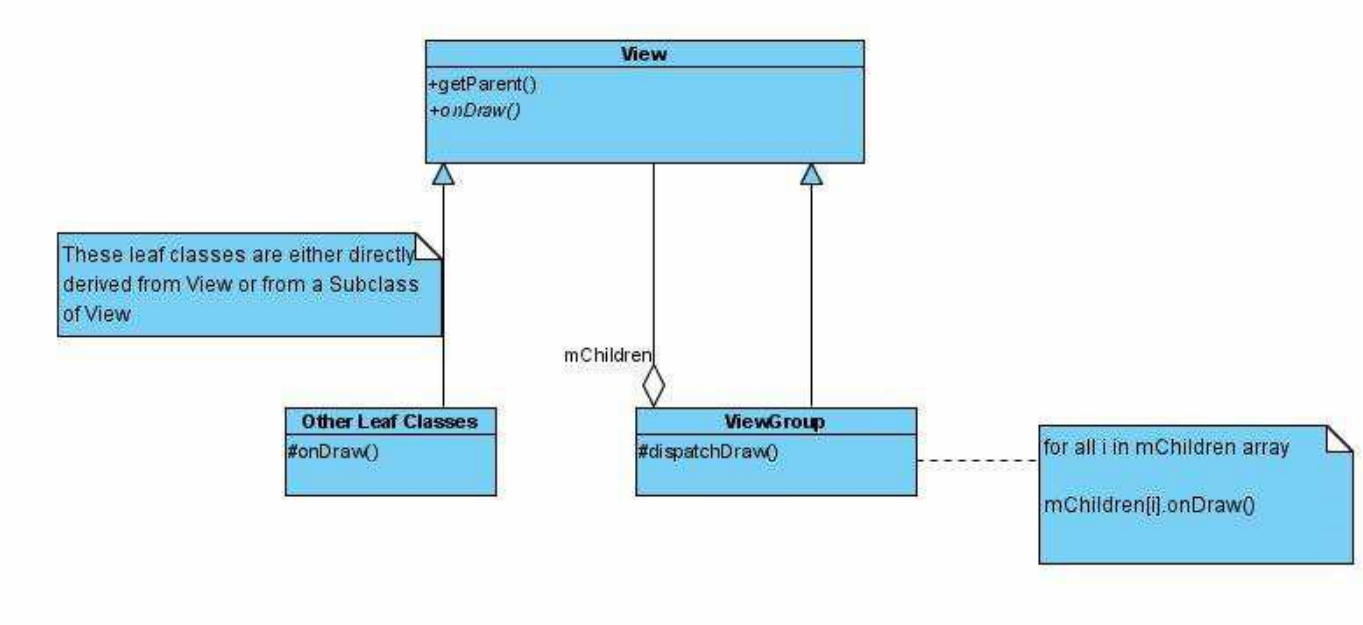


Si les ViewGroup sont composés de View notre problème est résolu.

Les composants d'une application Android

View (3)

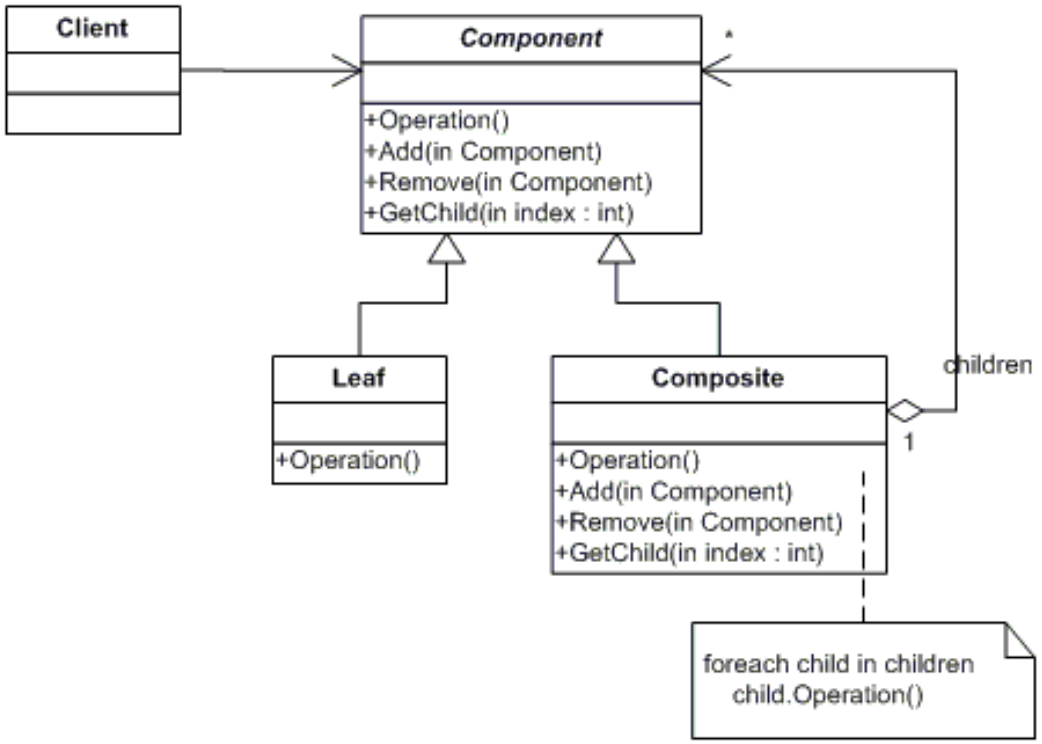
- Un **Ecran Android** est une **ViewGroup** contenant des **View** qui peuvent être **composites ViewGroup** ou **élémentaires**



Il s'agit d'une utilisation du Design Pattern: « Composite »

View (4) : Le patron « Composite »

- Une **utilisation du patron « Composite »**



Les composants d'une application Android

Activity (1)

- Brique fondamentale des applications
- Associée à une View qu'elle crée (1 IHM = 1 Activité)
- Point d'entrée des interactions entre l'utilisateur et l'application
- Une activité a un cycle de vie géré par l'Activity Manager d'Android.
- Une activité peut démarrer d'autres activités de son application ou d'une autre
- La navigation de l'utilisateur dans une application est un enchainement d'activités
- Une activité peut utiliser des services, des managers, des data providers (fournis par Android, par d'autres applications ou conçu spécialement)

Les composants d'une application Android

Activity (2) : Cycle de vie

- Les méthodes de la classe Activity sont invoquées par l'Activity Manager d'Android. Le cycle de vie est imposé par le framework

```
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();
 }
```

Ce mode de fonctionnement ne correspondrait il pas à un design pattern ?

Les composants d'une application Android

Activity (3) : Inverse Of Control

- Le design pattern : **« Inversion de contrôle »**
  - Le framework prend en charge l'exécution d'un programme principal
  - Il contrôle et coordonne l'activité de l'application
  - Des blocs de code sans dépendances développés pour l'application sont invoqués dans l'ordre défini par le framework
  - L'IOC est illustrée par le principe d'Hollywood : *« Ne nous appelez pas, c'est nous qui vous appellerons »*
  - L'IOC est réalisée entre l'application et le framework:

    C'est le framework qui appelle les composants de l'application et non le contraire

Activity (4) : Cycle de vie

- **4 états**
  - ***Active*** or ***Running*** **:** l'activité est affichée à l'écran et a le focus
  - ***Paused*** **:** l'activité est affichée à l'écran mais n'a plus le focus
  - ***Stopped*** **:** l'activité n'est plus visible, elle est masquée par une autre
  - ***Destroyed*** **:** fin de l'activité (à la demande de l'utilisateur ou du système qui réclame des ressources)

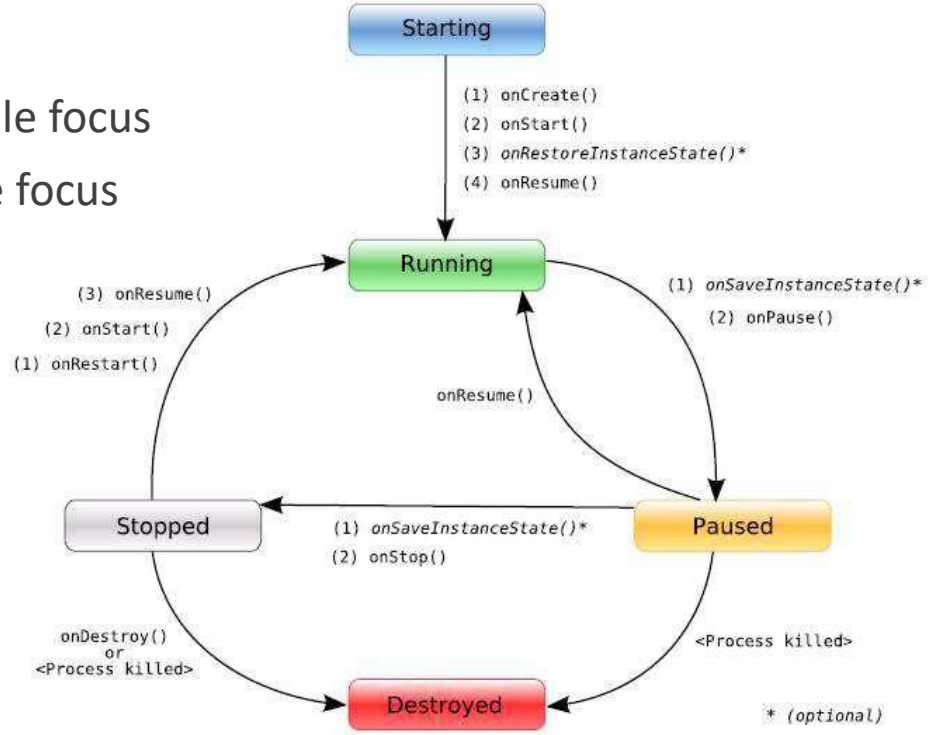


Figure 2.3: Life cycle of an Android activity

Les composants d'une application Android

Activity (5) : Cycle de vie

onCreate() **:** appelé lorsque l'activité est créé

onRestart() **:** appelé lorsque l'activité a été arrêtée, juste avant de redémarrer

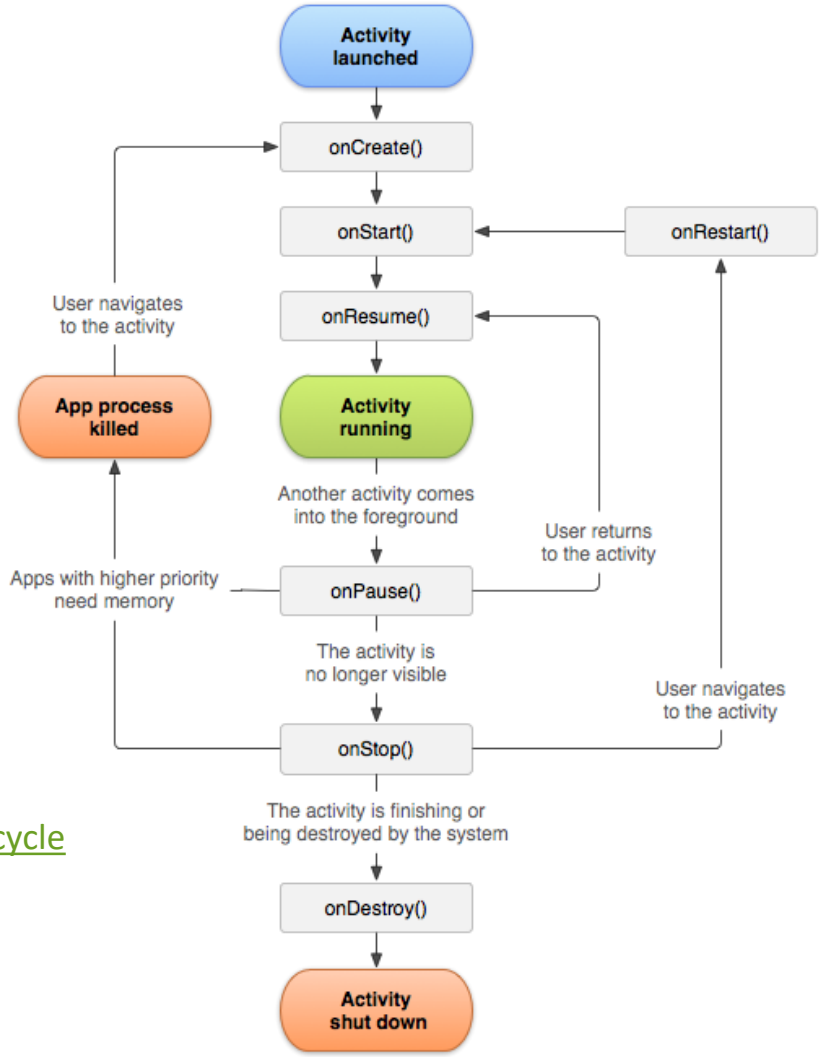
onStart() **:** appelé immédiatement après *onRestart()*, suivi par *OnResume()*

onResume() **:** appelé lorsque l'activité est prête à interagir avec l'utilisateur

onPause() **:** appelé lorsque une autre activité passe au premier plan

onStop() **:** appelé quand l'activité n'est plus visible par l'utilisateur

onDestroy() **:** appelé juste avant la destruction de l'activité. (le système a besoin de ressources ou la méthode finish a été appelé par l'application.



https://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle

Les composants d'une application Android

BroadcastReceiver

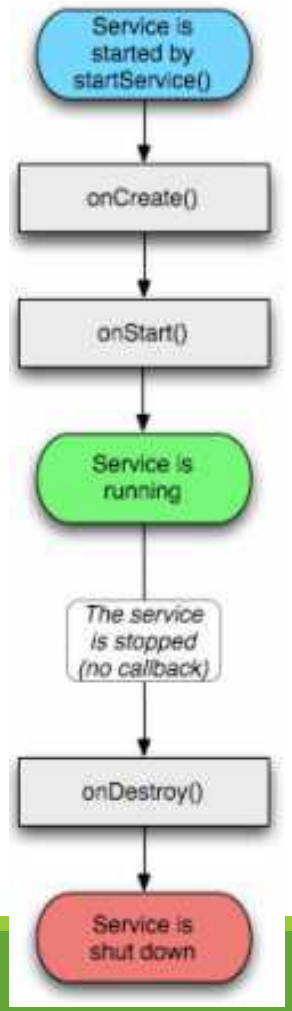
- **BroadcastReceiver :** un récepteur de messages
  - Les applications Android envoient et reçoivent des messages
  - Le système Android se charge de poster et de router les messages
  - Les messages permettent aux activités et aux applications de communiquer
  - Concept :
    - Le BroadcastReceiver s'abonne à un type de message
    - Lorsqu'un message de ce type est « Posté », le BroadcastReceiver le reçoit par une invocation de sa méthode *OnReceive()*

Ce mode de fonctionnement ne correspondrait-il pas encore à un design pattern ?

Les composants d'une application Android

Service

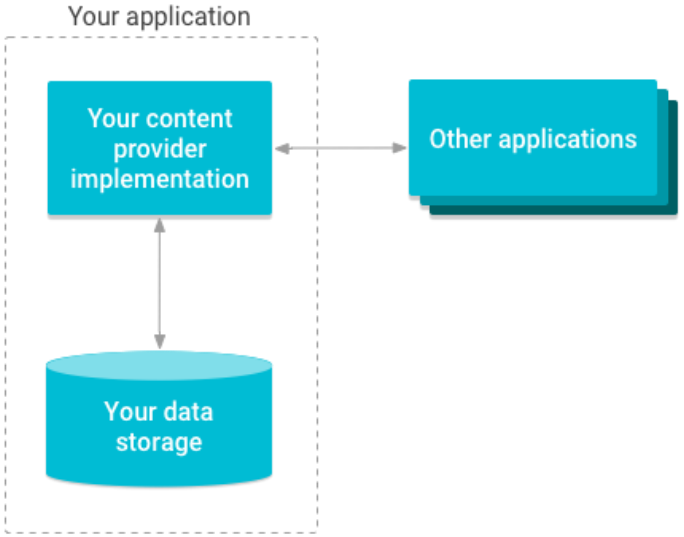
- Tâche qui s'exécute en arrière-plan sans IHM
- Fonctionne même si l'utilisateur bascule entre plusieurs applications
  Exemple: Ecouter de la musique tout en consultant ses mails
- Disponible pour être utilisable à partir d'une activité ou d'un autre service

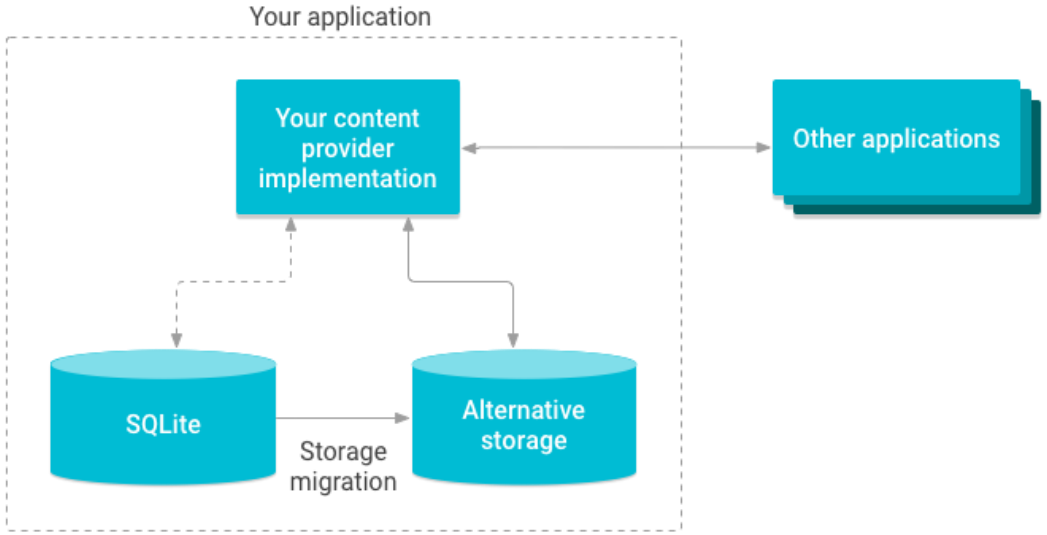


Les composants d'une application Android

ContentProvider

- **ContentProvider:** un fournisseur de contenu
  - Assure la persistance des données pour une application
  - Permet à une application de mettre à disposition ses données (contrôle d'autorisation possible)
  - Permet d'accéder aux données d'autres applications
  - Masque la stratégie de stockage des données.





Ce mode de fonctionnement ne correspondrait il pas à nouveau à un design pattern ?

Les composants d'une application Android

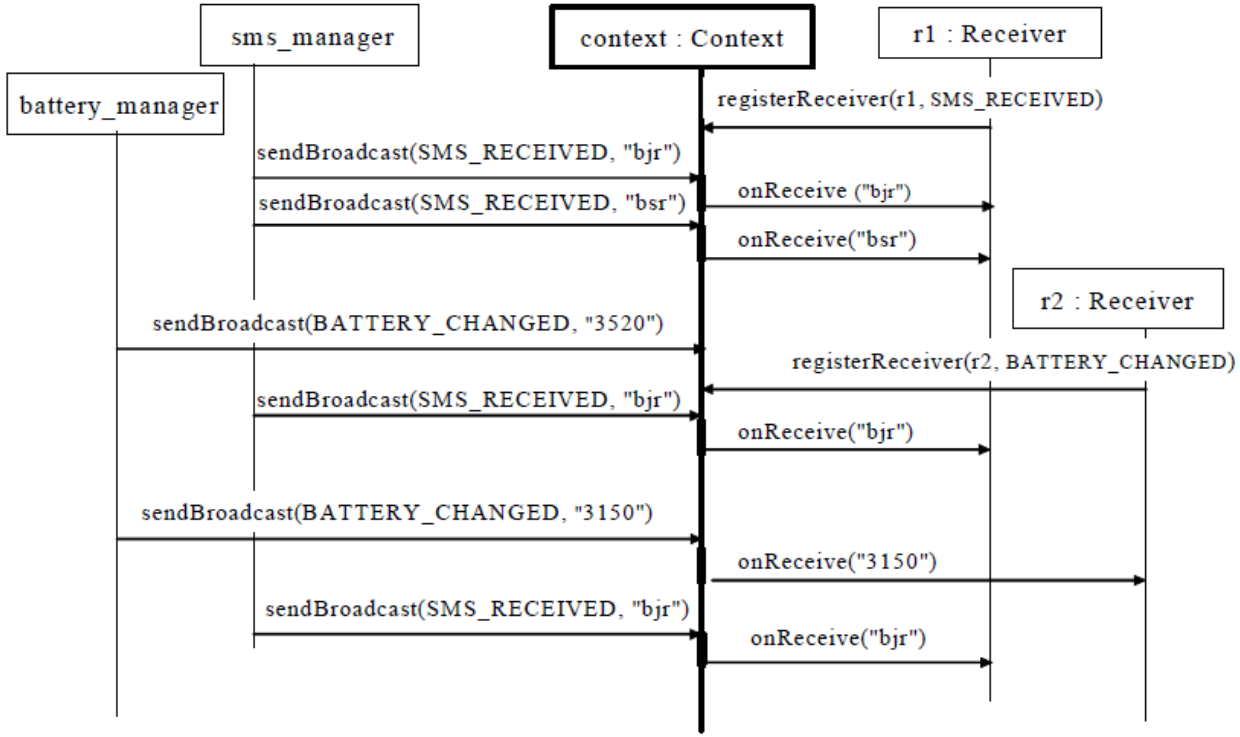
Intent et IntentFilter

- **Intent:** « Intention » message pour demander une action:
  - à une activité
  - à une autre application
  - au système Android

- **Les 3 cas principaux d'utilisation des Intent:**
  - Démarrer une activité
  - Démarrer un service
  - Créer et propager un message en Broadcast

- **2 types d'Intent:**
  - **Intent explicite:** Défini avec le nom de l'activité ou du service à démarrer
  - **Intent implicite:** Défini avec le nom du type d'action à réaliser, un composant enregistré pour réaliser cette action reçoit l'Intent et réalise l'action.

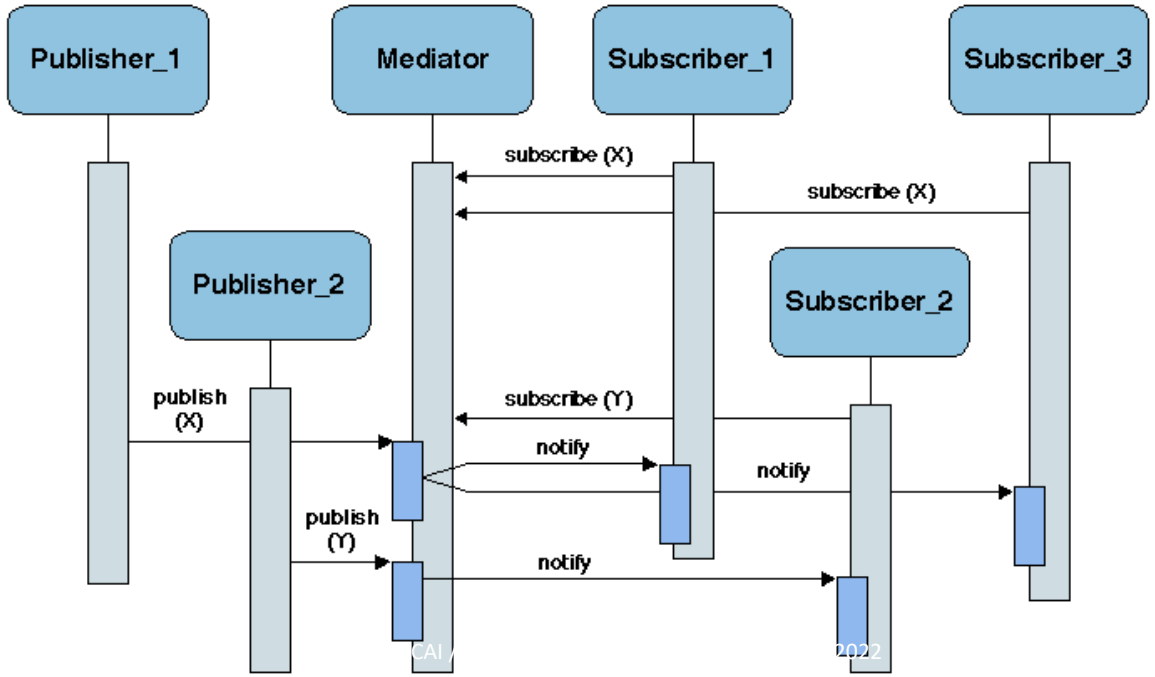
Intent Implicite



Context, BroadcastReceiver, Emetteur de messages en broadcast : encore un design pattern !

Intent Implicite : Le Pattern Publish Subscribe (1)

- Des **Subscriber** se sont abonnés à un type de message
- Un **Mediator** enregistre les abonnements et se charge de **notifier** les abonnés lorsque un message du type attendu est envoyé
- Des **Publisher** envoient un message par l'intermédiaire du **Mediator**
- La **Notification** aux **Subscriber** peur être effectuée selon un ordre particulier, la propagation de cette notification peut être stoppée par un **Subscriber**.



Les composants d'une application Android

Intent Implicite : Le Pattern Publish Subscribe (2)

- **Utilisation du design pattern Publish-Subscribe:**
  - Des **Subscriber** à certaines **Intentions** ➔ **BroadcastReceiver**
  - Des **Publisher** d'**intentions** ➔ **Activity** émettrice d'**Intent** avec **IntentFilter**
  - Un mécanisme de résolution pour obtenir la bonne activité (**Mediator**) ➔ **Context**

Les composants d'une application Android

Loader / AppWidget

- **Loader:**
  - Mise à disposition de méthodes pour lire les données d'un ContentProvider
  - Intérêt des **Loader** :
    - Chargement en asynchrone sans bloquer l'IHM (Thread autre que UIThread)
    - Optimisation par utilisation de caches (évite de faire plusieurs fois la même demande)
    - Possibilité d'observer la ressource lue pour détecter les changements. (Pattern Observateur)

    https://developer.android.com/guide/components/loaders.html

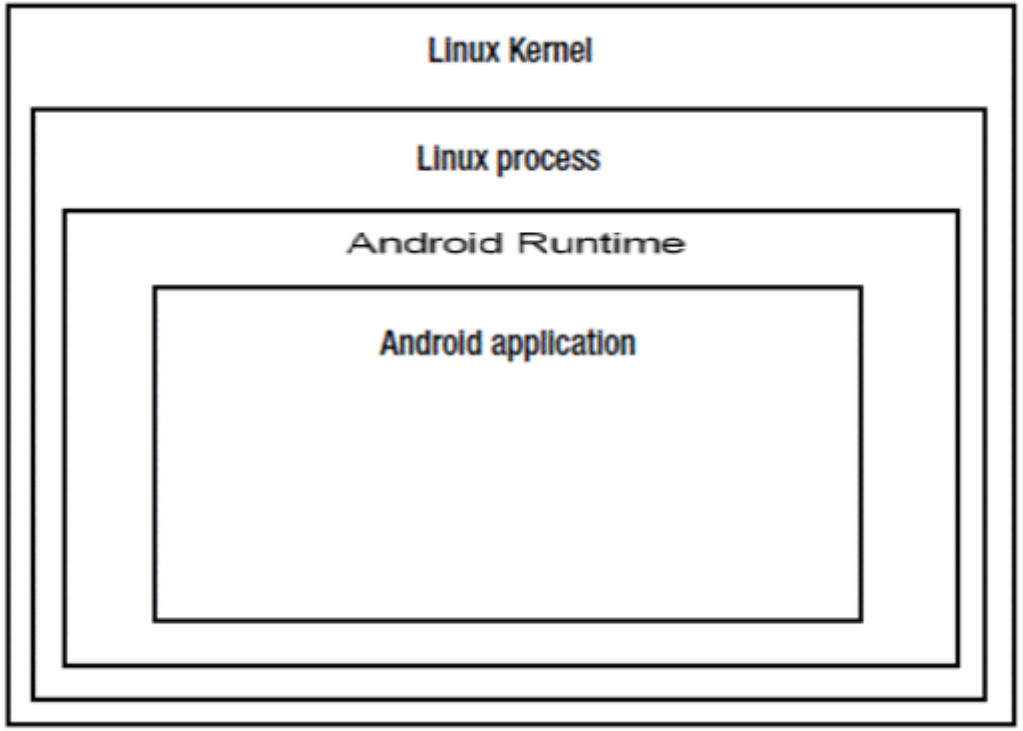
- **AppWidget:**
  - Applications miniatures avec leur IHM destinées à être embarquées dans d'autres applications

  Exemple : le widget météo d'un écran d'accueil

  https://developer.android.com/guide/topics/appwidgets/index.html

Processes et Threads

- Chaque **Application** s'exécute dans son **propre Processus**



Les composants d'une application Android

Processes et Threads

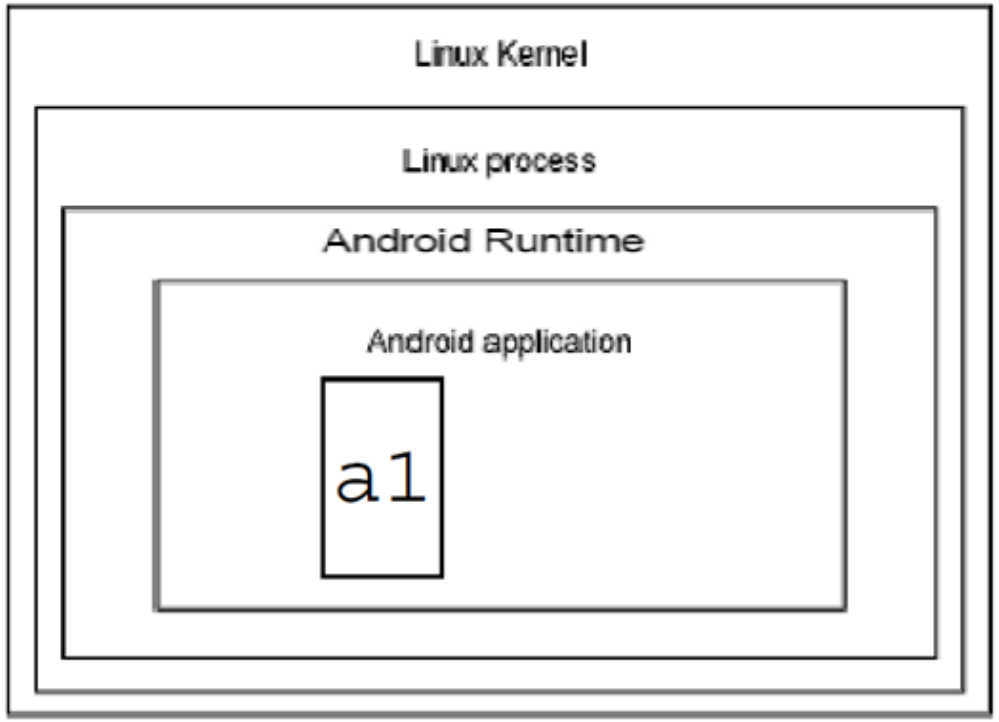
- Par défaut: tous les composants d'une application s'exécutent dans le **même Processus** et le **même Thread**
- Une application peut toutefois **lancer l'exécution de composant** dans d'**autres Processus** et même créer de **nouveaux Threads** dans n'importe lequel de **ses Processus**.
- Une seule activité est active à la fois : exécution dans **UIThread**
- Toute opération sur les composants graphiques doit se faire dans le **UIThread**

Les composants d'une application Android

Processes et Threads

- Un Processus, une DVM/ART, une Application, une Activité

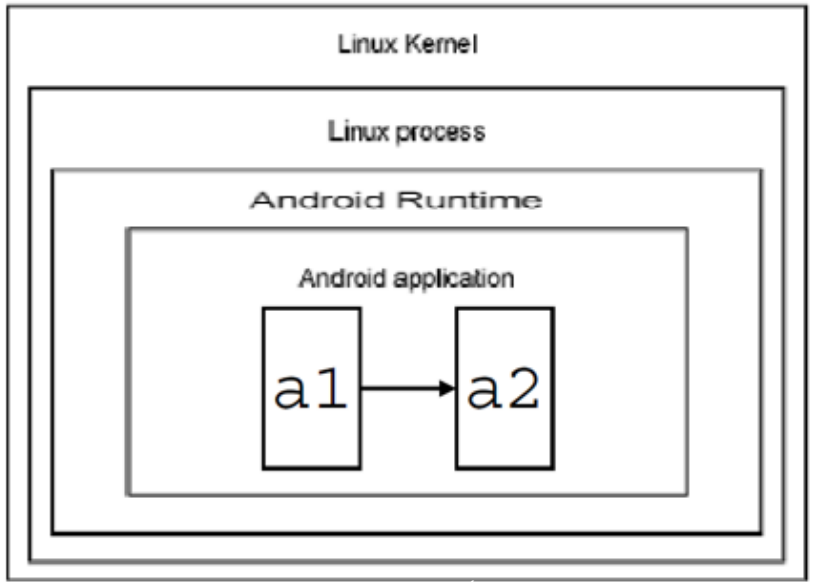
a1 est une activité



Les composants d'une application Android

Processes et Threads

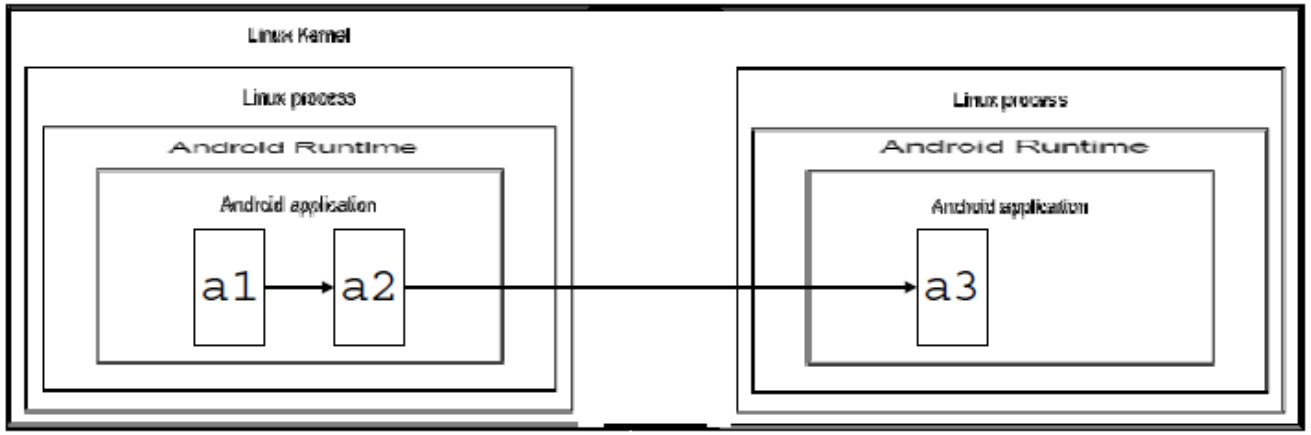
- Un Processus, un ART, une Application, deux Activités
- Une activité peut en lancer une autre:
  - a1 lance a2
  - a1 et a2 s'exécutent dans le même processus, la même application, le UIThread
  - a1 est en pause, a2 est actif



Les composants d'une application Android

Processes et Threads

- Deux Processus, Deux ART, Deux Applications, Trois Activités
- Une activité peut en lancer une autre:
  - a1 a lancé a2
  - a1 et a2 s'exécutent dans le même processus, la même application, le UIThread de la première application
  - a2 a lancé a3 (d'une autre application) qui s'exécute dans un second processus, un second ART, le UIThread de la seconde application



Conclusions

- Android est un système d'exploitation pour les « Equipements Mobiles »
- La SDK permet de développer en Java ou Kotlin
- C'est un Système bâti en couche au-dessus d'un Linux
- Architecture d'une application Android :
  - à base de composants
  - communication par messages.
- De nombreux design pattern sont utilisés dans le Framework
- Comme dans la majorité des framework avec des UI : 1 seul Thread Graphique