ANDROID

SMB116

Android :

Android est un système d'exploitation open source développé par l'Open Handset Alliance sous l'autorité de Google, et crée au départ pour les téléphones mobiles.

Android est basé sur un kernel Linux et possède une licence Apache License 2.0. Android n'est pas seulement un système d'exploitation, c'est aussi un framework. Les développeurs ont accès au SDK (Software Development Kit) et à tout le code source de la plateforme. Avec ces outils il est possible de créer des versions personnalisées de l'OS Android et de développer des applications. C'est un des atouts qui a fait le succès d'**Android**.

La genèse d'Android :

Android est aujourd'hui associé à Google mais son histoire commence bien plus tôt. Il a été développé à partir de 2003 par la société américaine éponyme. Google a fait l'acquisition de cette société en 2005. On suppose qu'un « Google Phone » va voir le jour mais rien ne se passera jusqu'à la sortie du **G1 Phone** en 2008.

A l'origine, chaque constructeur de téléphone et d'équipement mobile créait son propre système d'exploitation embarqué et il était de ce fait impossible de concevoir une application compatible avec tous les appareils. En 2007 **Apple** sort l'**Iphone**, le système **IOS** est très moderne, très ergonomique et très en avance sur tous les concurrents. A la fin de cette même année 35 entreprises dont Google se regroupent et constituent le « **Open Handset Aliance »** pour créer un système d'exploitation **open source** destiné à révolutionner le marché et contrer les concurrents comme **Windows mobile** mais surtout **IOS**.

En 2008 la version 1.0 du SDK d'Android est publiée et le **G1 Phone** arrive sur le marché

2009 voit l'arrivée de nombreux équipements basés sur **Android** (20 nouveaux appareils). Et de nouvelles versions du système sont publiées.

2010 les appareils utilisant **Android** sont les plus vendus juste derrière les **Black Berry**. (Plus de 60 appareils utilisent **Android 2.2**)

2011 **Android** est le numéro 1 mondial en nombre de nouvelles machines vendues et prend de plus en plus de parts du marché des tablettes.

Cette année (2017) Android est utilisé sur 80% des smartphones dans le monde.
http://www.journaldugeek.com/2017/02/17/android-ios-dominent-marche-smartphones/

Les versions d'Android :

Android 1.0 (Octobre 2008) utilisée par G1 Phone (Fabriqué par HTC):

- Innovation : La barre de notification
- Pas de clavier ni gestion du multitouch
- Apparition des Widgets
- Android Market (ancêtre du Play Store)

Android 1.1 « Banana Bread » (Février 2009)

- Correction des nombreux bugs
- Mise à jour via OTA (par téléchargement en Wifi ou 3G)

Android 1.5 « CupCake» (Avril 2009)

- Amélioration interface graphique
- Arrivée du clavier tactile et du presse papier
- Capture et publication de vidéo

Android 1.6 « Donut» (Septembre 2009)

- Prise en charge réseau CDMA
- Ecran portrait / paysage, nouvelles définitions jusqu'à la HD (1280x720)
- Amélioration interface APN

Android 2.0 « Eclair» (Octobre 2009) et Maj 2.1

- Amélioration clavier tactile
- Fonds d'écran animés
- Bluetooth 2.1
- Google Map
- Nouvelles fonctionnalités pour APN
- Speech To Text
- Synthèse vocale

Android 2.2 « Froyo» (Mai 2010) et 2.3 « Gingerbread »

- Nombreuses améliorations pour le « Nexus One » (premier smartphone de la marque Google créé par HTC)
- Flash Player
- Prise en charge densité écran à 320 ppi
- Ecran de verrouillage à mot de passe ou code PIN
- VoIP
- NFC
- Système de fichier ext4
- Outils pour le développement d'applications en natif

Android 3.x « Honeycomp» réservée aux tablettes (Février 2011)

- 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**

Android 4.0 « Ice Cream Sandwich» (Octobre 2011)

- 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

Android 4.1, 4.2, 4.3 « Jelly Bean» (Juin 2012) 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

Android 5.0, 5.0.1, 5.0.2 et 5.1 « Lollipop» (Octobre 2014)

- Material Design
- Amélioration sécurité du noyau
- Economie d'énergie (amélioration de la veille)
- Abandon de la machine Dalvik au profit d'ART (Android Runtime)

Android 6.0 « Marshmallow» (Octobre 2015)

- Support de **Google Now** améliorée
- Amélioration autonomie
- Nouveau système de permissions
- Nombreuses autres évolutions

Android 7.0 « Nougat» (Août 2015)

- Multi-fenêtrage
- Modification écran paramètres
- Amélioration du multitâche
- Blocage denuméros
- Outils pour la réalité virtuelle

L'environnement de développement

Java

Les codes sont écrits en java et sont dans une première phase compilés en bytecode, d'où la nécessité d'avoir installé un JDK (Java Development Kit).

Android Studio

Android Studio est un ide basé sur IntelliJ qui a été adapté pour développer des applications Android. Android Studio permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android. Il propose aussi des outils pour gérer le développement d'applications, multilingues et permet de concevoir et de réaliser les interfaces graphiques sur des écrans de résolutions variées simultanément. **Android Studio** est venu en remplacement d'**Eclipse** qui était utilisé jusqu'en 2015 (avec un plugin spécifique) pour le développement d'applications Android.

SDK Android

Le SDK Android fournit tous les outils nécessaires pour créer une application Android. Il est disponible pour Windows, Mac OS et Linux Chaque version du SDK contient :

L'**AAPT** - Android Asset Packaging Tool : cet outil sert à créer et à analyser les fichiers *.apk (Android Application Package). Un fichier APK est une archive compressée qui contient une application.

L'**ADB** - Android Debug Bridge : le but de cet outil est d'établir des connexions avec un téléphone Android ou un émulateur. Ceci afin d'accéder à son système de fichier, d'exécuter des commandes mais surtout de transférer une application en développement ou finalisée et packagée sur un appareil Android ou un émulateur.

Le **DDMS** - Dalvik Debug Monitor Service: cet outil est utilisé principalement pour le débogage et le monitoring des applications :

- Observer les threads en cours d'exécution.
- Consulter le Logcat.
- Connaître la liste des processus en cours d'exécution sur l'appareil.
- Simuler l'envoi de messages et d'appels.
- Simuler une localisation
- Transmettre des simulations de mesures comme si elles étaient fournies par les différents capteurs.

Le SDK contient aussi la documentation pour chaque version d'Android et des exemples d'utilisation des différentes API.

Emulateur

L'émulateur est un outil qui permet de simuler un téléphone ou une tablette qui fonctionne sous Android. Un émulateur est donc utile pour tester et mettre au point une application durant son développement. Avec l'émulateur, il est aussi possible de créer une large gamme d'émulateurs pour simuler tous les types d'appareils ainsi que toutes les résolutions et toutes les tailles d'écran.

ART

ART (Android RunTime) est une nouvelle machine virtuelle sur laquelle s'appuie Android, depuis la version **5.0 Lollipop**. Elle est venue remplacer la machine virtuelle précédente du nom de **Dalvik**, qui est toujours disponible dans les dernières versions du système d'exploitation Android).

ART possède plusieurs nouveautés ou spécificités dont la principale est l'**AOT Compilation** (Ahead of Time, ce qui veut dire en français compilation anticipée). L'AOT Compilation permet de compiler toutes sortes d'applications lors de l'installation et non pas à l'exécution, comme le faisait la version Dalvik.

Dalvik se base sur la compilation à l'exécution, qu'on appelle **Just In Time**. Dans les faits, l'application est compilée à chaque exécution. La machine virtuelle ART permet maintenant d'augmenter les performances et l'espace de stockage pour sauvegarder une application (stockage de la version compilée de l'application).

Cette machine virtuelle ART possède plusieurs autres nouveautés telles que l'amélioration du **Garbage Collector** ou encore l'amélioration du développement et du débogage et du profilage d'applications.

NDK

Le **NDK**, **N**ative **D**evelopment **K**it permet le développement natif sous Android. Ce développement natif se fait en langage C ou en langage C++.

Les bibliothèques et composants natifs sont utilisables à partir des parties développées en Java en utilisant **JNI** (**J**ava **N**ative **I**nterface).

Le NDK met à disposition plusieurs outils qui permettent de compiler des fichiers source en langage C ou en langage C++.

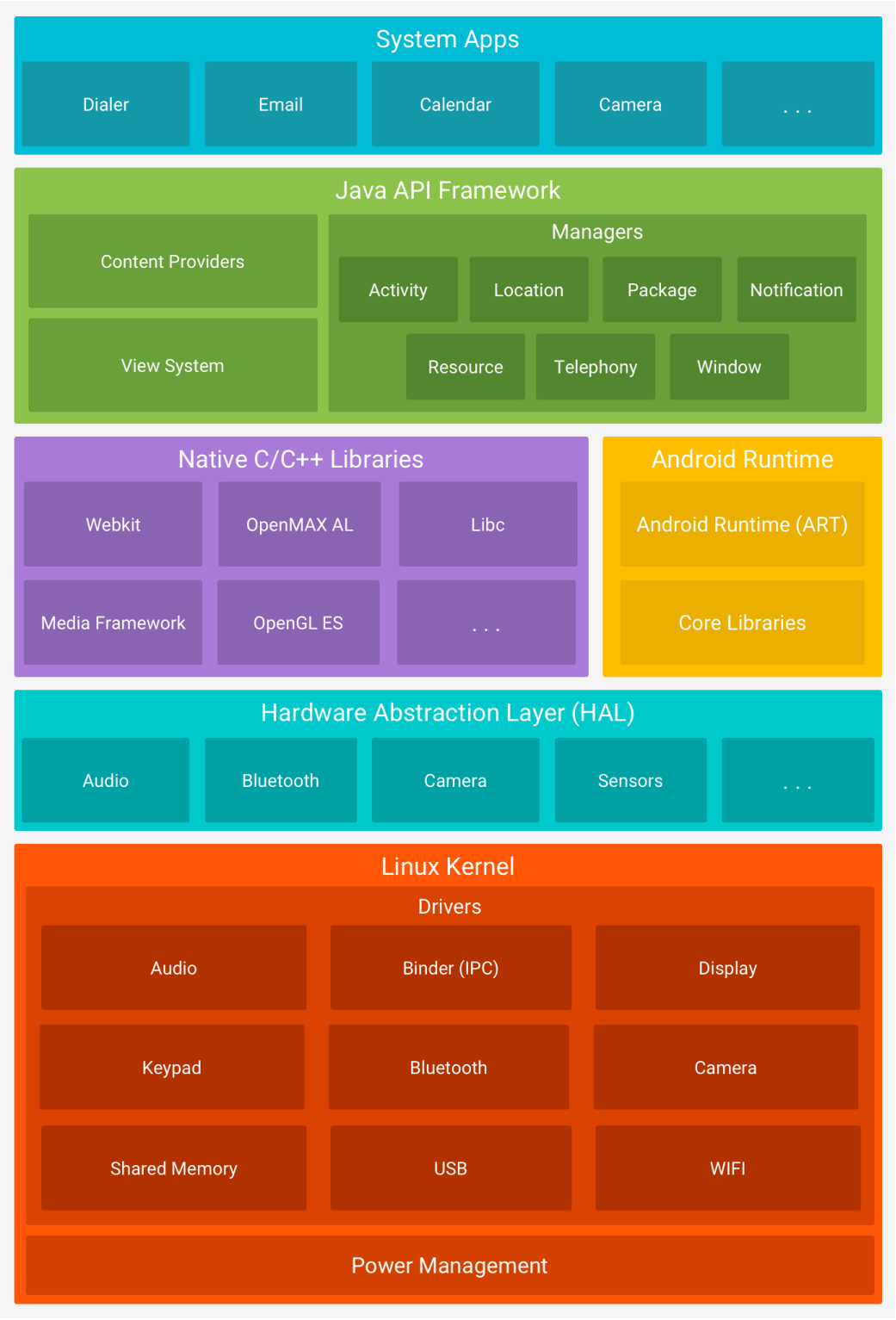
http://lmi92.cnam.fr/SMB116/tp2_enonce/

Architecture de la plateforme Android

https://developer.android.com/guide/platform/index.html

Android est un système d'exploitation basé sur Linux et créé pour être utilisable sur un grand nombre d'appareils et de définitions d'écrans.

Les principales couches de la plateforme Android sont présentées dans le diagramme suivant :



Le noyau Linux :

La base de la plateforme Android est le noyau Linux, durant l'exécution de tout processus l'ART utilise les primitives du système Linux pour la gestion de la mémoire, l'accès au système de fichier ou la gestion des processus en exécution.

L'utilisation de Linux comme base a plusieurs avantages :

- Les fonctionnalités sécurité de Linux sont réutilisées
- les fabricants de périphériques pourront plus facilement créer les pilotes pour Linux

HAL : Hardware Abstraction Layer

HAL est une couche intermédiaire qui met à disposition les fonctions pour l'utilisation des périphériques hardware (son, Bluetooth, caméra, divers capteurs, etc.) par les API du framework Java pour Android. Le HAL est composé de modules de bibliothèques comportant un ensemble de méthodes pour manipuler une catégorie de périphérique particulier. Ces modules sont chargés par le noyau lorsqu'une API du framework a besoin d'accéder à un périphérique.

Android Runtime (ART)

Depuis la version 5.0 de Android chaque application s'exécute dans son propre processus et son propre Android Runtime. ART est conçu pour permettre l'exécution de plusieurs machines virtuelles sur des appareils disposant de peu de mémoire. ART exécute des fichiers de bytecode DEX créés à partir de source Java compilés puis convertis.

Les apports principaux de ART :

- Compilation AOT (Ahead-of-time) et JIT (just-in-time).
- Ramasse miette optimisé (Garbage Collector)
- Amélioration des possibilités de débogage et de profilage.
- Traces et messages d'erreur plus clair

ART comporte aussi un ensemble de bibliothèques mettant à disposition des fonctionnalités utilisées par le framework Java.

Bibliothèques native C/C++

Un ensemble de bibliothèques natives écrites en C/C++ sont disponibles dans le framework Android et sont exploitées par ART mais sont aussi utilisables à partir d'API du framework Java.

API Java

Toutes les fonctionnalités du framework Android sont accessibles à travers les API Java. Toutes les applications Android créées par des développeurs ou fournies de base avec le système utilisent ces APIs.

- **View System APIs :** pour créer des interfaces graphiques à partir de composants graphiques comme des zones de texte (textbox), des boutons, des Grille de données et des Layouts.
- **Resource Manager APIs :** fonctionnalités pour accéder aux ressources autres que le code comme les fichiers I18n (internationalisation), les images, les fichiers XML de layout.
- **Notification Manager :** fonctionnalités pour permettre aux applications de créer des messages d'alerte dans la barre de 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. Par exemple : Accès à la liste des contacts.

Applications systèmes

Android est livré avec un ensemble d'applications : gestion des contacts, navigateur Web, communication par SMS etc. Il n'y a aucune différence entre ces applications systèmes et les applications créées par des développeurs.

Ces applications systèmes mettent à disposition un ensemble de fonctionnalités. Ces fonctionnalités sont exploitables directement par l'utilisation de ces applications ou bien à travers d'autres applications qui utilisent ces applications systèmes pour obtenir des services. Par exemple : envoi d'un SMS.

Les composants d'une application Android

Activity (Activité):

Les **activités** sont une des briques fondamentales des applications Android. Elles sont le point d'entrée pour les interactions entre un utilisateur et une application. Leur enchainement constitue la navigation de l'utilisateur dans l'application courante ou entre différentes applications. Pour simplifier une activité peut être vue comme un élément associé à une IHM. Les activités ont un cycle de vie composé d'états et de callbacks géré par l'Activity Manager d'Android.

BroadcastReceiver (Récepteur de messages de notification) :

Les applications Android peuvent envoyer ou recevoir des messages en « Broadcoast » en utilisant Android. Les **BroadcoastReceiver** sont des composants qui s'abonnent auprès d'Android à un type de message. Si un émetteur crée et transmet par Android un message de ce type, le **BroadcoastReceiver** recevra ce message.

Service

Un **Service** est une tâche qui s'exécute en arrière-plan sans IHM. Un service continue à fonctionner même si l'utilisateur bascule entre plusieurs applications (exemple : écouter de la musique tout en consultant ses mails).

ContentProvider (Fournisseur de contenu)

Un **ContentProvider** assure la persistance de données. Il permet à une application de sauver ses données mais aussi de les mettre à disposition pour d'autres applications. Un **ContentProvider** comporte un mécanisme pour définir des droits d'accès aux données qu'il peut fournir.

Intent (Intention) et IntentFilter

Les **Intent** sont des messages qui sont utilisés pour demander une action à une activité ou une autre application. Les 3 cas principaux d'utilisation des **Intent** sont :

- Démarrer une activité
- Démarrer un service
- Créer et propager un message en Broadcast

Il existe 2 types d'**Intent** :

- Les **Intent** explicites : ils sont définis avec le nom (nom complet de la classe) du composant destinataire.
- Les **Intent** implicites : Le destinataire n'est pas nommé mais l'**Intent** est défini avec le nom d'une action à réaliser. Le composant qui s'est enregistré auprès d'Android pour réaliser l'action (recevoir les actions de ce nom) portant ce nom reçoit l'**Intent** et effectue l'action. L'enregistrement auprès d'Android s'effectue en utilisant un **IntentFilter**.

> *Dans le cas d'utilisation des Intent implicites, nous avons un composant qui s'abonne à un type d'action et un autre composant qui lance une demande pour un type d'action.*
> ***Cela ne fait-il pas penser à un design pattern ?***

Fragment

Un Fragment représente le comportement d'une portion de l'IHM d'une activité (détaillé plus tard)

Loader

Un loader met à disposition des méthodes pour lire les données d'un **ContentProvider** ou d'autres sources de données.

Intérêt des **Loader** :

- Chargements en asynchrone sans bloquer l'IHM
- Optimisation par l'utilisation de caches
- Possibilité de lui confier l'observation de la ressource lue pour détecter les changements

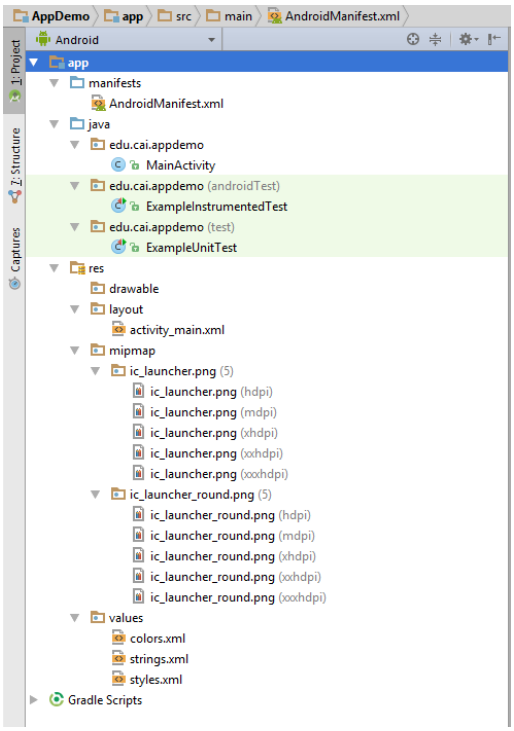
AppWidget

Les AppWidget sont des applications miniatures avec leur IHM qui peuvent être embarquées dans d'autres applications (Exemple : le widget météo d'un écran d'accueil).

Processes and Threads

Chaque application s'exécute dans son propre processus, par défaut tous ses composants s'exécutent dans le même processus et Thread. Il est toutefois possible pour une application de lancer l'exécution de certains de ses composants dans d'autres Processus. L'application peut aussi créer des threads supplémentaires dans n'importe lequel de ses processus.

Structure d'un projet Android :



App Manifest :

Toutes les applications doivent avoir un fichier nommé **AndroidManifest.xml** dans le répertoire racine. Ce fichier fournit toutes les informations liées à l'application parmi lesquelles :

- Le nom du package de l'application
- La description des activités
- La liste des composants
- La liste des permissions que doit obtenir l'application
- Le numéro minimum de version de la version d'Android que nécessite l'application
- La liste des bibliothèques utilisées
- Icone et libellé de l'application

Structure du fichier manifest :

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

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

(https://developer.android.com/guide/topics/manifest/manifest-intro.html)

Exemple de fichier manifest :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="edu.cai.appdemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Les fichiers sources Java :

Le code d'une application Android est écrit dans des classes Java. Ces classes implémentent les différents composants d'une application Android : Activity, View, Intent, ContentProvider etc.

Activité principale de l'application:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Les ressources :

Les ressources sont des fichiers qui contiennent des éléments utilisés par l'application et qui ne sont pas du code. Il s'agit d'éléments graphiques, de fontes, de descriptions d'écran, de labels, de couleurs, de valeurs diverses.

Selon leur type, ces fichiers de ressources sont répartis dans des dossiers différents et leur contenu est codé différemment.

Dossier	Type de ressource	Type de contenu
`res/anim`	Animations	XML
`res/drawable`	Eléments graphiques	Bmp, Png, Gif, XML
`res/color`	Couleurs selon état (ColorStateList)	XML
`res/layout`	Disposition des éléments dans les vues	XML
`res/menu`	Menus	XML
`res/values`	Couleurs, Labels, Styles	XML
`res/font`	Fontes	XML
`res/raw`	Tous les autres types de ressources	indéfini

Les ressources organisées dans ces dossiers sont les ressources que l'application doit utiliser par défaut. Il est possible de spécifier les ressources à utiliser en fonction de divers critères comme Langage et région, Taille de l'écran, Orientation de l'écran, Résolution de l'écran, Version d'Android. Pour cela il suffit de créer des dossiers avec le nom d'origine suffixé par des *Qualifieurs.*

Exemples :

`res/drawable-small` pour avoir des images spécifiquement pour les petits écrans.

`res/layout-fr` pour les layouts destinées aux utilisateurs qui ont un système en français.

`res/layout-land` pour les layouts destinées à être utilisés lorsque l'écran est orienté en mode paysage.

Pour plus d'informations : https://developer.android.com/guide/topics/resources/providing-resources.html#table2

Accès aux ressources dans le code Java :

Toutes les ressources d'un projet Android sont identifiées par un identifiant unique. Ces identifiants sont référencés dans une classe Java nommée **R** maintenue automatiquement par Android Studio lorsque des ressources sont créées ou supprimées.

Exemple d'accès à un Layout dans une Activity : setContentView(**R.layout.*activity_main***);

Installation de l'environnement de développement :

Installation d'un JDK :

Pour tester qu'un JDK est installé on utilise les commandes : *Java –version* , et *Javac*

Téléchargement du JDK à partir de http://www.oracle.com/technetwork/java/javase/downloads/index.html

Installation Android studio :

1ère Méthode :

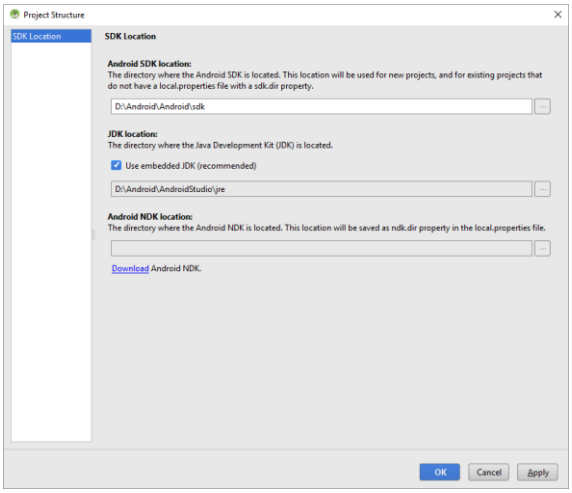
Télécharger Android studio https://developer.android.com/studio/index.html

Lancer le .exe téléchargé et suivre les instructions.

2nde Méthode :

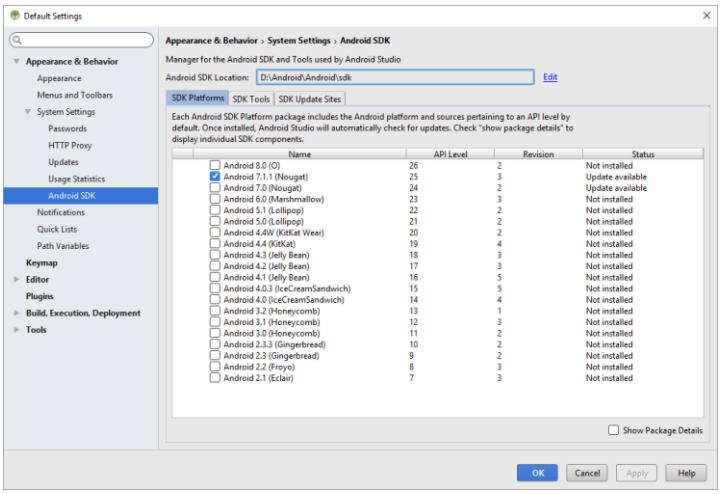
Copier une installation existante puis personnaliser la configuration :

- Définition localisation du SDK Android et du JDK :
  File -> Other settings -> Default Project Structure

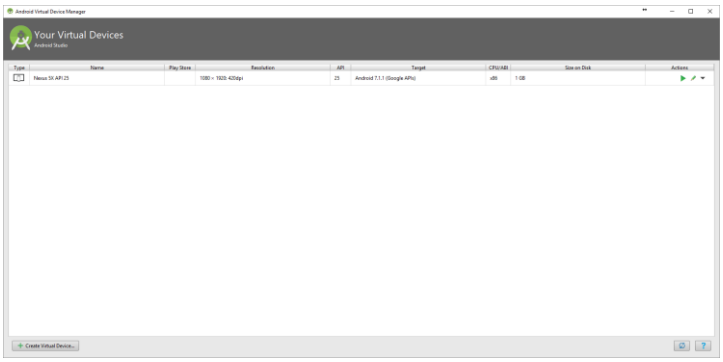


Configurer l'environnement de développement Android studio :

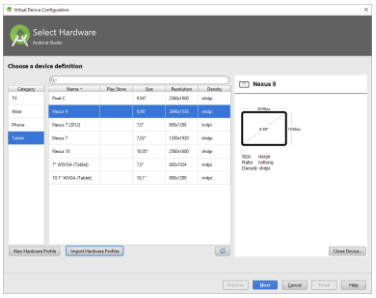
Définir la version de SDK à utiliser : *Tools -> Android -> SDK Manager :*

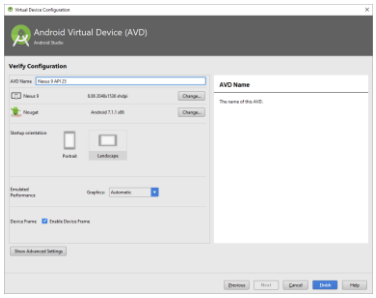


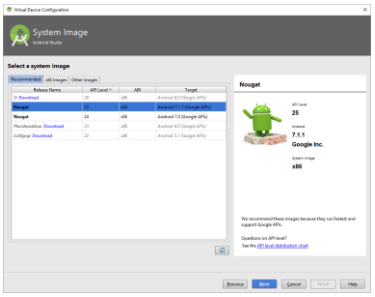
Définir les « Virtual Device » : *Tools -> Android -> AVD Manager :*



Création d'une nouvelle VD :







Notre premier TP : « La calculatrice »

- Quand elle démarre elle affiche 0
- L'appuie sur RAZ remet la calculatrice dans le même état que quand elle démarre
- Exemple d'une opération :
    - On appuie sur 1 la calculatrice affiche 1
    - On appuie sur 2 la calculatrice affiche 12
    - On appuie sur + la calculatrice affiche 12
    - On appuie sur 3 la calculatrice affiche 3
    - On appuie sur = la calculatrice affiche 15
    - On appuie sur * la calculatrice affiche 15
    - On appuie sur 2 la calculatrice affiche 2
    - On appuie sur + la calculatrice affiche 30
    - On appuie sur 1 la calculatrice affiche 1
    - On appuie sur 0 la calculatrice affiche 10
    - On appuie sur = la calculatrice affiche 40
    - On appuie sur RAZ la calculatrice affiche 0