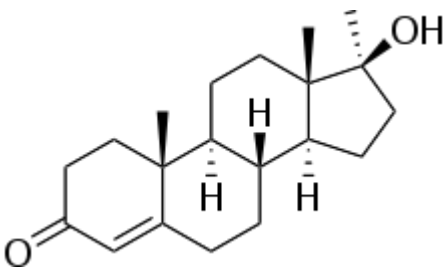
Introduction à Android

Master 2 Informatique 2012-2013

Android ?

- Robot anthropomorphe
- Nom commercial de la méthyltestostérone
- Jeu de société d'aventure futuriste
- Marque de vêtements
- Marque de dentisterie
- Mais aussi... système d'exploitation Linux pour appareils nomades avec bibliothèques intégrées (SQLite, WebKit, OpenGL...), machine virtuelle et plate-forme de développement basée sur Java



Une petite histoire d'Android

- Octobre 2003 : conception d'un OS mobile par Android Inc. (co-fondé par Andy Rubin) à Palo Alto
- Août 2005 : rachat d'Android Inc par Google
- Novembre 2007 : présentation du consortium Open Handset Alliance (Google + industriels) et de l'Android Open Source Project (AOSP), version beta sous licence Open Source Apache
- Septembre 2008 : 1ère version finale avec le téléphone HTC Dream
- Octobre 2011 : sortie de la dernière version majeure (4.0 Ice Cream Sandwich)

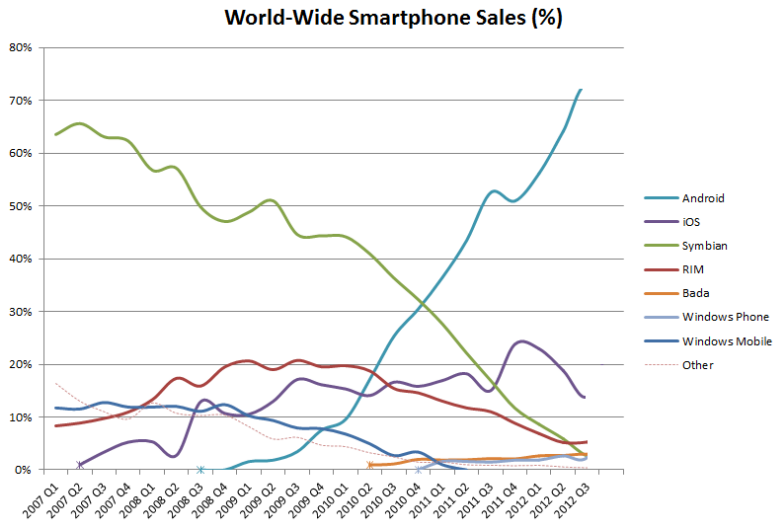
Composantes d'Android

- Noyau Linux standard avec modifications :
  - Gestion d'énergie avec WakeLocks (à terme, intégration dans branche principale)
  - Mécanisme de communication inter-processus avec Binder
- Machine virtuelle Dalvik :
  - Conversion du bytecode : .class → .dex
  - VM à registres avec un jeu important d'instructions
  - JIT introduit par Froyo

- Bibliothèques implantant l'API (compatible avec JDK6) :
  - Sources natives en C et en Java
  - Briques pour la communication entre applications, l'accès aux senseurs (GPS, {accéléro,magnéto,baro,...}-mètre) et à la communication réseau (cellulaire, WiFi, Bluetooth, NFC...)

Principaux OS orientés mobiles

Parts de marché sur 4T 2012 (source IDC)



- Android [75,0%]
- iOS [14,9%]
- BlackBerry OS [4,3%]
- Symbian S60 [4,1%]
- Windows Phone / Mobile / RT [3,6%]
- Autres basés sur Linux [2,8 %] (MeeGo, Maemo, LiMo, Openmoko, webOS...)
- Feature phones (supportant J2ME) : Bada, S40...
- À venir : Firefox OS, Tizen, Ubuntu OS...

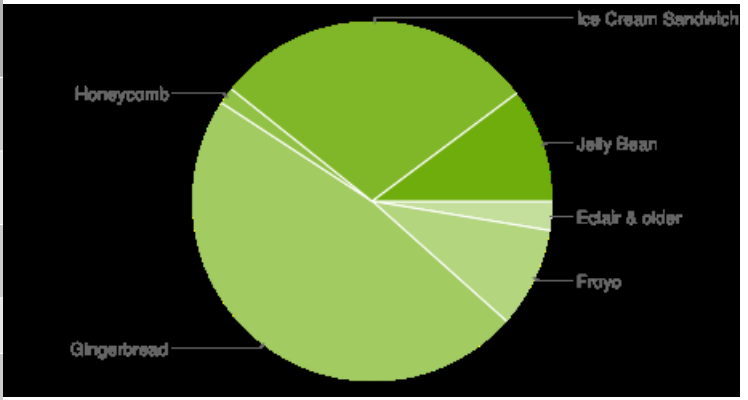
Pré-requis pour un OS mobile

- Gestion intelligente de l'énergie : optimisation de l'autonomie
- Adaptabilité aux spécificités matérielles :
  - Petit écran, résolution plus ou moins élevée
  - Interface tactile
  - Senseurs : accéléromètre, boussole, GPS, luxmètre, thermomètre, baromètre...
  - Communication réseau : Bluetooth, Wi-Fi, cellulaire

Versions d'Android

Source (données du 3/01/2013) : http://developer.android.com/about/dashboards/index.html

Sortie	Version	Friandise	# API	Part de marché
09/2008	1.0		1	
02/2009	1.1	Petit four	2	
04/2009	1.5	Cupcake	3	
09/2009	1.6	Donut	4	0,2 %
01/2010	2.1	Eclair	7	2,4 %
05/2010	2.2	Froyo	8	9,0 %
12/2010	2.3 – 2.3.2	Gingerbread 1	9	0,2 %
02/2011	2.3.3 – 2.3.7	Gingerbread 2	10	47,4 %
05/2011	3.1	Honeycomb 1	12	0,4 %
07/2011	3.2	Honeycomb 2	13	1,1 %
12/2011	4.0	Ice Cream Sandwich	15	29,1 %
07/2012	4.1	Jelly Bean 1	16	9,0 %
11/2012	4.2	Jelly Bean 2	17	1,2 %



Écosystème Android

- Implantation de référence (AOSP) maintenue par Google sous licence Apache 2.0 (sauf patches Linux sous GPLv2)
- Applications de la distribution standard : client HTTP WebKit, calculatrice, calendrier, caméra, horloge, gallerie, SMS/MMS, musique, carnet d'adresses, téléphone, recherche, enregistreur de voix...
- Distributions Android dérivées par les fabricants (de téléphones cellulaires, tablettes, boîtiers multimédia, appareils photos, autoradios...) :
  - Noyau repatché
  - Modules pour support de matériel (quelquefois closed source)
  - Nouvelles applications plus avancées
  - Limitations courantes : accès root bloqué (déblocage par faille d'escalade de privilège), bootloader bloqué (MAJ du firmware difficile)
- Quelques magasins d'applications :
  - Google Play (ex-Android Market) : ~675K apps et 25G téléchargements en 09/2012
  - Amazon Appstore
  - Opera Mobile Store
  - GetJar : créé en 2004, propose également des applications J2ME
  - F-Droid : magasion d'applications libres

Quelques applications Android utiles

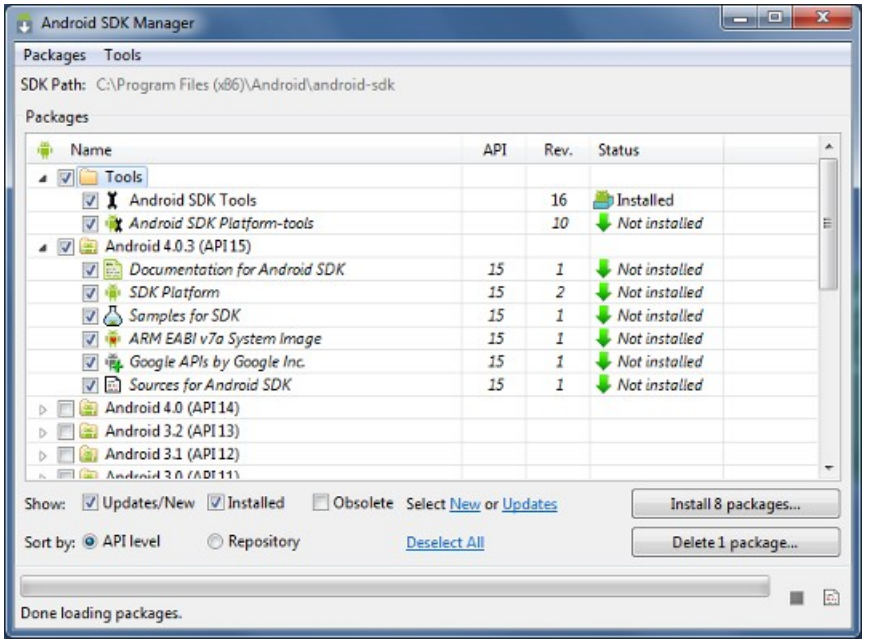
- Réseau
  - ConnectBot : client SSH
  - K-9 Mail : client IMAP/POP assez complet
  - SipDroid, CsipSimple : clients de téléphonie SIP
- Jeux
  - FreeCiv : clone de Civilization
  - FrozenBubble : jeu d'arcade
- Sécurité
  - Android Privacy Guard : implantation openPGP
  - Keepassdroid : gestionnaire de mots de passe
  - Google Authenticator : générateur de mots de passe jetables
- Multimédia
  - VLC : lecteur multimédia
  - XBMC : lecteur multimédia initialement pour TV
- Divers
  - Barcode Scanner : lecteur de code-barres 2D
  - FBReader : lecteur de livres électroniques (supportant entre autres les formats Epub et mobi)
  - OSMAnd : logiciel de navigation GPS offline utilisant les cartes d'OpenStreetMap
  - MyTracks : gestionnaire de traces GPS

SDK Android

- SDK manager : gestion de paquetages du SDK (updates)
- emulator : émulateur ARM basé sur QEMU
- Android Debug Bridge (ADB) : communication avec un appareil Android
- Android Developer Tools : plugin Eclipse
- android : création et gestion de machines virtuelles (Android Virtual Devices)
- Monitor : GUI pour déverminage (DDMS, traceview...)
- Outils de gestion d'images (9patch, ETC1...)
- Lint : analyse statique du code
- proguard : compresseur, optimiseur et obfuscateur de fichiers apk
- Outils de tests automatisés (monkey, monkeyrunner et uiautomator)
- Code-source d'Android
- Documentation de l'API, exemple de code
- Android Native Development Kit (NDK) : intégration de code natif en C/C++

SDK Manager

- Téléchargement (versions Windows, Linux) : http://developer.android.com/sdk/
- Lancement : `android sdk`



Création d'une machine virtuelle

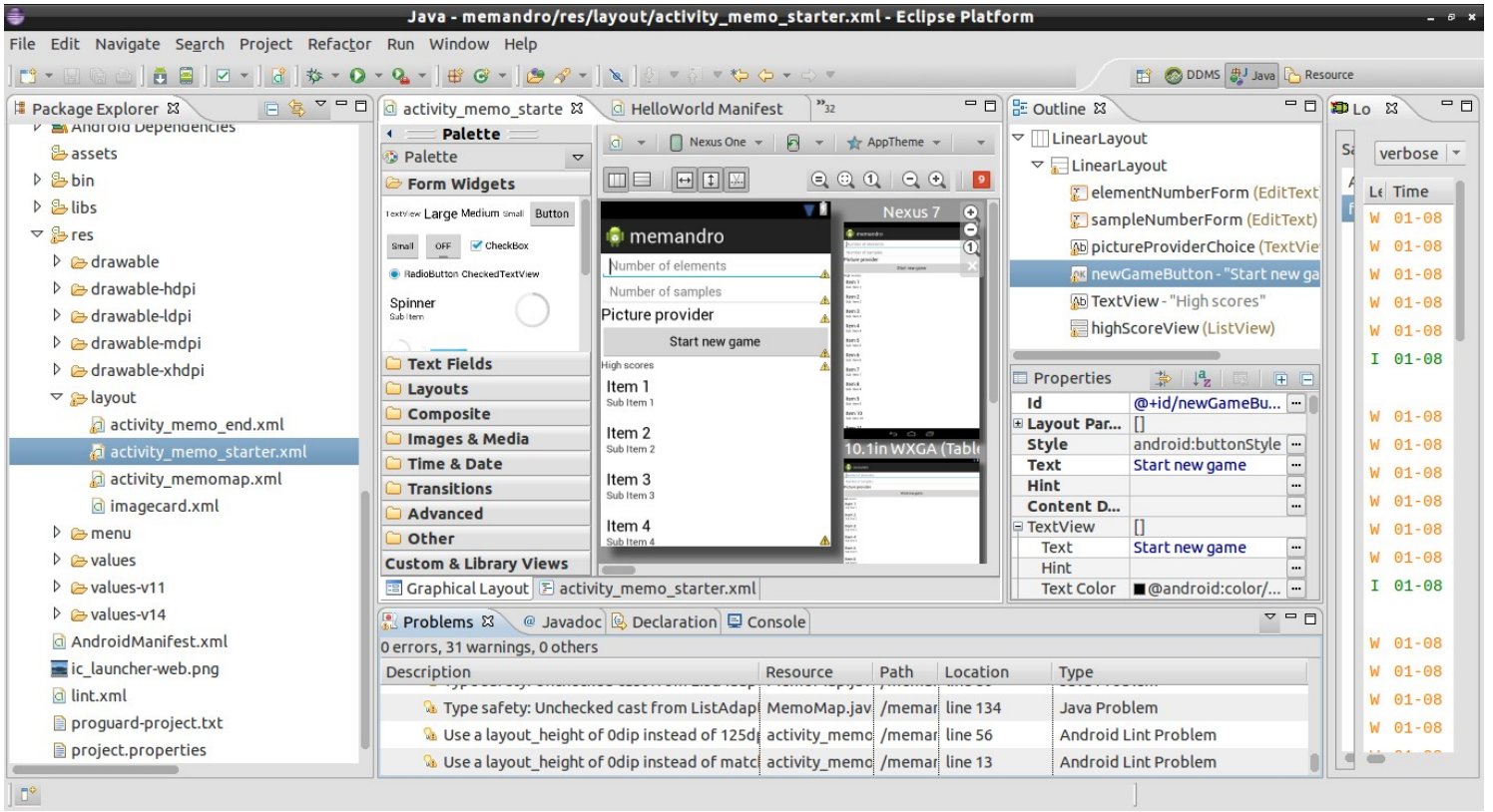
[https://developer.android.com/tools/devices/managing-avds-cmdline.html]

- GUI de gestion d'AVD : `android avd`
- Nouvelle carte SD vfat :
  `mksdcard -l carteSD 1024M carteSD.img`
- Nouvelle machine virtuelle :
  `android create avd -c carteSD.img -n myAVD --snapshot --target android-17`
- Lancement de la VM :
  `emulator @myAVD`
  Options utiles : -sdcard <file>, -memory <sizeInMBs>, -shell, -logcat <tags>, -tcpdump <file>

- Communication avec la VM avec adb :
  - `adb push <source> <dest>` : copie un fichier
  - `adb pull <source><dest>` : récupère un fichier
  - `adb logcat <tags>` : affiche les logs courants
  - `adb shell` : ouvre un shell sur la machine
  - `adb {install <file>, uninstall <package>}` : installe ou désinstalle une application
  - `adb {backup -f <file>, restore <file>}` : sauvegarde ou restaure les données utilisateurs
  - `adb forward <local> <remote>` : redirige une socket locale vers une socket de la machine

Android Developer Tools

- Plugin Eclipse passerelle vers tous les outils de développement et déverminage
- Edition WYSIWYG de layouts graphiques



API Android

- Réimplantation de l'API du JDK 1.6 : java.lang, java.util, java.{io,nio}, java.math, java.net, java.text
- Bibliothèques XML et JSON : javax.xml, org.w3c.dom, org.xml.sax, org.xmlpull et org.json
- Intégration de JUnit
- Paquetages android.*

Paquetages android.*

- util : classes utilitaires pour analyse de texte, logging, tableaux creux, caches...
- os : accès aux primitives IPC, au gestionnaire d'énergie, d'horloge, aux variables d'environnement, au vibreur...
- graphics : bibliothèque graphique bitmap permettant la manipulation d'images
- text : outils pour l'affichage de texte
- database : gestion de BDDs privées (implantation SQLite3 fournie)
- content, provider : gestion de contenus
- view : définiition des interfaces pour les vues graphiques
- widget : implantation de vues utiles
- app : définition des interfaces de base des applications
- provider : accès aux ContentProvider de base (CallLog, Contacts, MediaStore...)
- telephony, bluetooth, net : accès bas-niveau aux interfaces de communication
- webkit : affichage de contenu HTML
- location : API de géolocalisation (par GPS, bornes WiFi et triangulation cellulaire)
- media : lecture et enregistrement de données audiovisuelles
- opengl : API pour le rendu graphique 3D OpenGL
- ...

Point commun entre implantations Oracle et Google

```
private static void rangeCheck(int arrayLen, int fromIndex, int toIndex) {
	if (fromIndex > toIndex)
		throw new IllegalArgumentException("fromIndex(" + fromIndex + ") > toIndex(" + toIndex+")");

	If (fromIndex < 0)
		throw new ArrayIndexOutOfBoundsException(fromIndex);

	if (toIndex > arrayLen)
		throw new ArrayIndexOutOfBoundsException(toIndex);
}
```

- JDK : présent dans java.util.Arrays
- Android : présent dans java.util.Timsort (algorithme de tri hybride, tri par insertion et par fusion)
- rangeCheck supprimé à partir d'Android 4.0

Composantes d'applications Android

- *Activité* : brique élémentaire pour interface graphique d'interaction avec l'utilisateur
  - *Fragment* : élément de GUI réutilisable
- *Service* : démon réalisant une tâche en arrière plan
- Fournisseur de contenu (*ContentProvider*) : pour gérer des données accessibles à d'autres applications (ex : carnet d'adresse, agenda...)
- Récepteur d'évènements diffusés (*BroadcastReceiver*) : pour réagir à des évènements systèmes

Application, processus et threads

- Processus
  - 1 application (plusieurs composantes) = 1 processus (par défaut)
  - Mapping composant → processus modifiable dans le manifeste
  - Processus le moins important (avec composantes) tuable par le système si pénurie
    Hiérarchie : 1er plan > visible > service > arrière plan > vide
- Threads
  - Thread principale automatiquement crée pour l'affichage graphique
    - Ne pas la bloquer avec de longs calculs ou communications réseau (sinon « Application Not Responding »)
  - Création par l'utilisateurs de nouvelles threads :
    new Thread(new Runnable() { … }).start()

Création d'un projet

- Création possible depuis ADT sous Eclipse
- Création en ligne de commande :

```
android create project
    --target <target_ID> \
    --name <your_project_name> \
    --path path/to/your/project \
    --activity <your_activity_name> \
    --package <your_package_namespace>
```

- Mise à jour depuis un précédent SDK :

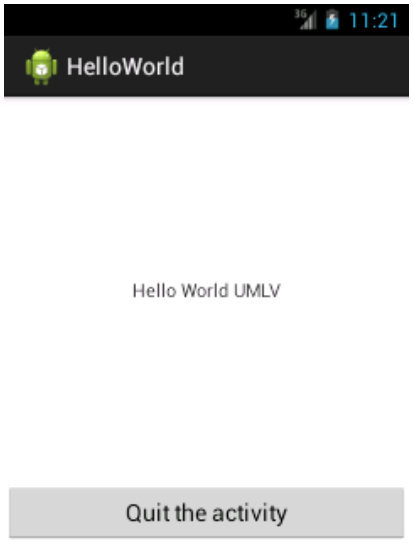
```
android update project \
    --name <project_name> \
    --target <target_ID> \
    --path <path_to_your_project>
```

Structure d'un projet

- *src/* : sources Java
- *lib/* : jars de bibliothèque
- *res/* : ressources, données statiques utilisées par l'application (chaînes i18n, description des layouts, menus, images, sons...)
- *gen/* : code Java autogénéré (fichier R.java référençant par constantes les ressources)
- *AndroidManifest.xml* : déclaration des métadonnées du projet (permissions, activité principale, description des composantes...)
- *bin/* : classes compilées, fichier apk

Une première activité : HelloWorld

```
public class HelloWorld extends Activity
{
	@Override
	protected void onCreate(Bundle savedInstanceState)
	{
		// We must never forget to call the super method
		super.onCreate(savedInstanceState);
		// We create ourselves the layout rather than loading it from a XML description
		LinearLayout layout = new LinearLayout(this); // New layout: container for the graphical elements
		layout.setOrientation(LinearLayout.VERTICAL);
		TextView tv = new TextView(this);
		tv.setText("Hello World UMLV"); // The string should be externalized as a resource
		tv.setGravity(Gravity.CENTER);
		LinearLayout.LayoutParams tvParams =
			new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT /*width*/, LayoutParams.WRAP_CONTENT /*height*/, 1/*weight*/);
		layout.addView(tv, tvParams);
		Button b = new Button(this);
		b.setText("Quit the activity");
		// We choose a weight of 1 for the TextView and 0 for the button (only the TextView will be resized)
		LinearLayout.LayoutParams buttonParams =
			new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT, 0);
		layout.addView(b, buttonParams);
		// We add a listener on the click event of the button
		b.setOnClickListener(new OnClickListener()
		{
			@Override
			public void onClick(View v)
			{
				HelloWorld.this.finish(); // Finish the activity
			}
		});
		setContentView(layout);
	}
}
```



AndroidManifest.xml : exemple

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.upemlv.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppBaseTheme" >
        <activity
            android:name="fr.upemlv.helloworld.HelloWorld"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Gestion des ressources

- Ressources dans *res/*
- Ressources contexte-dépendantes :
  - Différentes langues : i18n
    res/values-LG/strings.xml centralise les chaînes pour une langue (*res/values/strings.xml* pour chaînes par défaut)
  - Différentes tailles et densité d'écran
    les répertoires *res/drawable-X/* contiennent les images pour chaque type d'écran (X={l,m,h,xh}dpi)
- Description de GUI :
  - *res/layout* : descriptions XML de layout (édition graphique sous Eclipse)
  - *res/menu* : descriptions XML de menu

Internationaliser HelloWorld

- Fichier res/values/string.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello_world">Hello World UMLV!</string>
</resources>
```

- Fichier res/values-fr/string.xml :

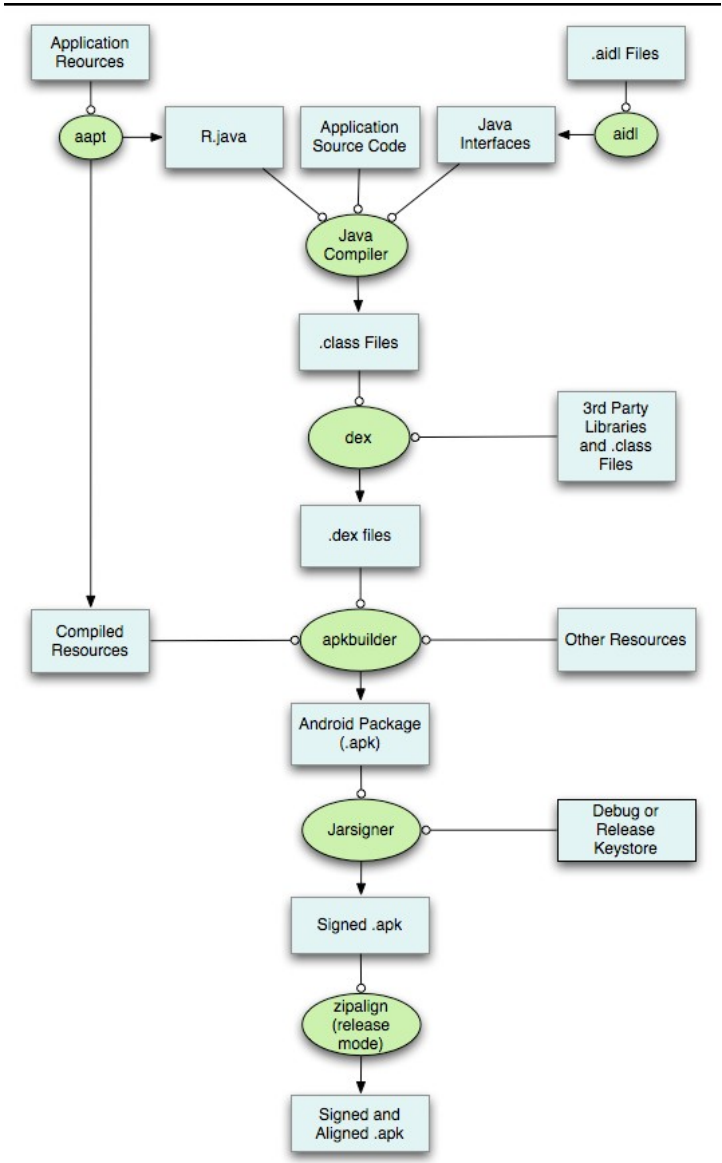
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello_world">Bonjour le monde UMLV !</string>
</resources>
```

- Remplacement par la référence de la ressource dans l'activité :

```
tv.setText(R.string.hello_world);
```

Compiler un projet

- Auto-compilation avec Eclipse
- Script Ant autogénéré : ant debug, ant release
- Pour les courageux : compilation à la main
  - Génération de code
    - Gestion d'IPC grâce à Android Interface Definition Language
    - Recensement des ressources dans R.java avec aapt
    - Génération de code depuis RenderScript (depuis HoneyComb) pour le calcul et le rendu graphique
    - Génération du BuildConfig (informations de déverminage pour la compilation)
  - Traditionnel javac
  - Utilisation optionnelle de ProGuard pour optimiser et obfusquer le bytecode
  - Conversion du bytecode Java en bytecode Dalvik avec dex
  - Empaquetage et compression des ressources avec aapt
  - Empaquetage final du bytecode et des ressources avec apkbuilder
  - Signature optionnelle de l'apk avec jarsigner
  - Alignement des ressources non compressées avec zipalign (pour faciliter l'accès avec mmap)



Tester un projet

- Avant de compiler, analyse statique avec `lint <projectdir>`
- Compiler le projet : `ant {debug, release}`
- Lancer un appareil de test sous Android :
  - `emulator @myAVD`
  - Ou connecter une vraie machine en USB (vérifier la connexion avec `adb devices`)
- Installer l'application (préalablement empaquetée dans un apk) :
  `adb install HelloWorld.apk`
- Lancer l'activité principale :
  `adb shell am start -n fr.upemlv.helloworld/.HelloWorld`
- Analyse de l'exécution
  - Ssurveiller les logs avec logcat (avec tag ActivityThread et de niveau INFO minimum) :
    `adb logcat ActivityThread:I`
  - Utiliser la GUI Dalvik Debug Monitor (DDMS)

Tâches

- Tâche = regroupement logique d'activités en pile ; permet la navigation temporelle
- Création d'une nouvelle tâche : typiquement depuis le launcher
- Coexistence possible de plusieurs tâches, mais une unique tâche affichée
- Manipulation de la pile d'une tâche :
  - Activité A lance activité B : B est empilé sur A
  - Touche "retour" : B est détruite et dépilée
  - Touche "home" : la tâche est mise en arrière-plan ; l'activité de bas de pile est préservée, les autres peuvent être détruites après un laps de temps

Comportements spécifiques pour les instances d'activité

- launchMode :
  - standard : une activité peut être instantiée plusieurs fois (et donc être multi-présente dans la pile)
  - singleTop : plusieurs instances de la même activité ne peuvent être présentes consécutivement dans la pile
  - singleTask : l'activité ne peut être présente que dans une seule tâche à sa racine
  - singleInstance : l'activité est l'unique élément dans sa tâche (nouvelle activité lancée --> nouvelle tâche)
- alwaysRetainTaskState : si vrai pour l'activité racine ne détruit pas les activités de sommet de pile d'une tâche restée en arrière-plan
- clearTaskOnLaunch : si vrai pour l'activité racine détruit systématiquement les activités de sommet de pile lors du passage en arrière-plan
- finishTaskOnLaunch : si vrai, détruit l'activité en cas de passage en arrière-plan
- Ajout d'un intent-filter pour permettre le lancement depuis le launcher :

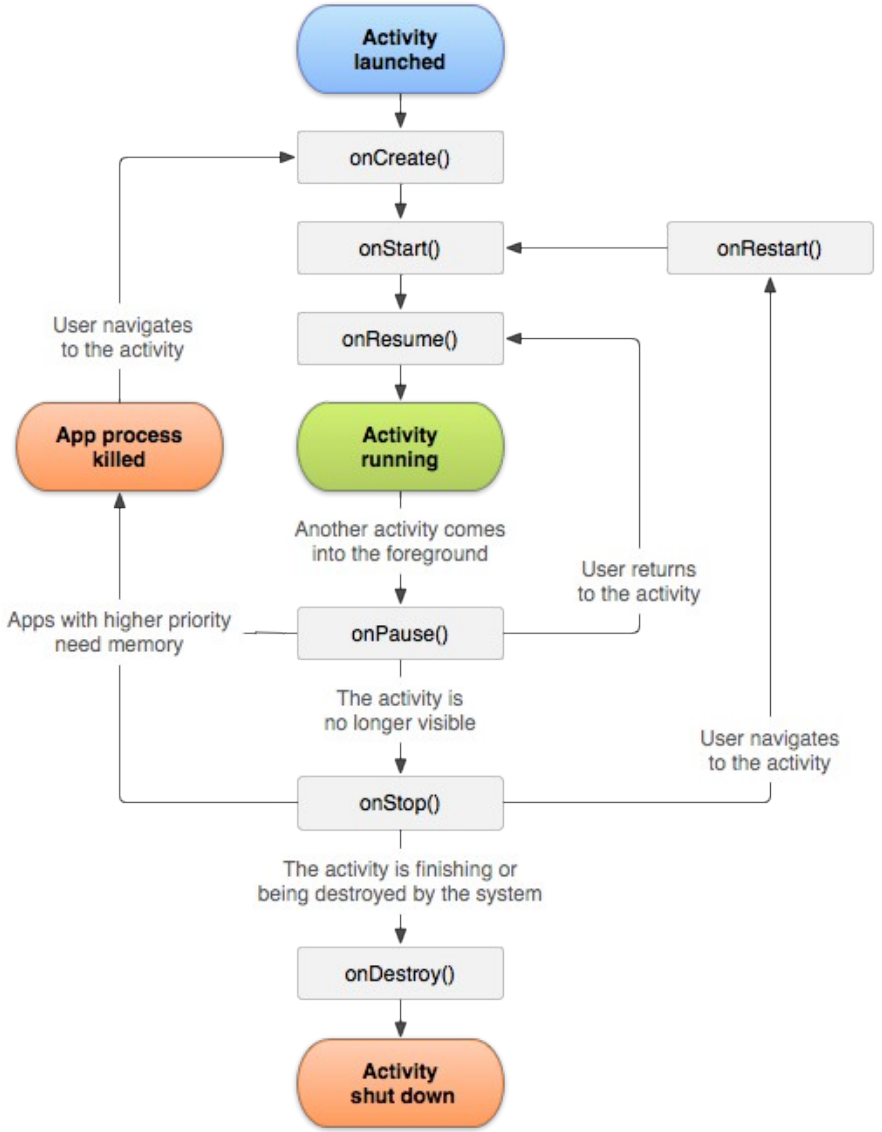
```
<activity ... >
    <intent-filter ... >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    ...
</activity>
```

Cycle de vie d'une activité

- Une seule activité visible à l'écran en avant plan (sommet de pile de la tâche courante)
- Activités invisibles (autres tâches, fond de pile) :
  - RAM : présentes (ou pas si pénurie de mémoire)
  - CPU : ne doivent pas exécuter des threads de calcul (tâche dévolue aux services)
- Passage de l'état visible à invisible :
  - En lançant une nouvelle activité masquant l'activité courante
  - En détruisant l'activité courante (finish(), bouton "retour")
- Envoi d'évènements à chaque changement de statut : gestion par listener *onX()* dans *Activity*

Diagramme de cycle de vie

http://developer.android.com/training/basics/activity-lifecycle/index.html



Listeners de changement d'état

- Les méthodes appelées lors des changements d'état :
  - onCreate(Bundle savedInstanceState) : initialisation des structures, (re)chargement des données dans le Bundle
  - onStart() : affichage de l'activité à l'écran ; s'il s'agit d'un redémarrage, onRestart() est appelé avant
  - onResume() : passage au 1er plan
  - onPause() : remplacement de l'activité courante par une autre (sauvegarde de champs en cours d'édition, arrêt de threads d'animation)
  - onStop() : activité rendue invisible, peut être ensuite tuée si pénurie de mémoire
  - onDestroy() : destruction imminente de l'activité, libération de ressources
- Redéfinir toues les méthodes n'est pas obligatoire ; si redéfinition ne pas oublier d'appeler d'abord *super.onX()*

Quelques sites utiles

- Site officiel Android (téléchargement du SDK, référence et tutoriels pour l'API) : http://developer.android.com/develop/
- Exemples d'applications dans le répertoire samples/ du SDK
- Code source Android (repository Git) : http://source.android.com/
- Blog sur nouveautés et actualités Android : http://android-developers.blogspot.fr/
- Micro-blogging Android : https://twitter.com/Android
- Google Groups Android : https://groups.google.com/forum/?fromgroups#!forum/android-developers
- Forum XDA : http://forum.xda-developers.com/
- ...

Quelques livres

- Professional Android Application Development (Reto Meier)
  Wrox (existe aussi en version FR)
- Hello, Android: Introducing Google's Mobile Development Platform (Ed Burnette)
  Pragmatic Bookshelf
- Android (Florent Garin) 3ème édition
  Dunod