Développer une application ANDROID

PLAN:

I) Langage JAVA

① Description du langage

Syntaxe

Elle est assez proche du C.
Toutefois elle apporte des nouveautés comme un opérateur ternaire:

(expression logique) **?** valeurSiVrai **:** ValeurSiFaux

=> selon le résultat de l'opération logique, le retour sera soit ValeurSiVrai, soit ValeurSiFaux.

Machine virtuelle Java

- On **développe** un programme avec la syntaxe java (éditeur texte en ASCII): nom.java
- On **compile** le programme .java, on obtient du Bytecode => nom.class
- Un **interpréteur** java dédié à la machine cible permet d'exécuter sur le programme .class sur cette machine = **Machine virtuelle**.

Il faut distinguer 2 programmes Java:

- JRE (Java Runtime Environment) = interpréter (machine virtuelle seule = JVM Java Virtual Machine).
- JDK (Java Development Kit = SDK Java = Software Development Kit Java) puis J2SE puis JSE.
  = JRE Machine virtuelle ET compilateur (compiler et interpréter).

Java est un **langage objet** = Création de manière simple de **nouveaux types de données**, le type est une **classe.**
L'objet est un petit programme qui a ses propres variables et instructions.
=> Au lieu de faire un gros programme, on développe des petits qui communiquent entre eux.

Class = modèle ou prototype à suivre pour fabriquer objet = **instanciation.**
La classe décrit un ensemble d'objets partageant des caractéristiques communes (données) et des comportements (fonctionnalités). Une classe est un type de données.

Les classes sont composées + d'**attributs** (variables de description = données)
+ de **méthodes** (actions associés à l'objet, appel d'autres classes méthodes).
Attributs et Méthodes sont les membres de la classe.

Package = Projet Java qui regroupe plusieurs classes.
Une des méthodes du projet doit être "main".

Remarque : Le premier langage objet était "Simula 67" créé pour les banques en 1967 = Simulation de comptes.

Objet = un objet est une **instance de classe** (déclinaison selon le modèle qu'est la classe).
Instancier une classe = Créer un objet dynamiquement avec l'**opérateur new.**
Il faut d'abord déclarer une **référence à un objet** du type voulu puis créer (construire) l'objet grâce à l'opérateur new.
On peut regrouper les deux étapes en une ligne:
nom_classe nom = new nom_classe(); où nom = référence d'un objet

3 types de variables en Java :

- les variables primitives ou simples: 8 **types primitifs**: char, short, int, byte, float, long, double, boolean.
- les variables objets = variables dont le **type est une classe**.
- les références d'objets (équivalents de pointeurs pour désigner un objet en mémoire et qui permet de le manipuler.)

La manipulation de référence est plus simple que celle de pointeur puisqu'après création de l'objet (instanciation = créer une instance de classe), une référence peut quasiment être confondue avec l'objet qu'elle désigne.

- Pas d'accès à l'adresse mémorisée dans une référence (pas d'opérateur &, pas d'arithmétique de pointeurs)
- Pas de syntaxe particulière pour accéder à l'objet référencé (pas de déréférencement avec opérateur * ou ->).

L'accès à l'objet est direct par sa référence => On confond souvent objet et référence. C'est le compilateur Java s'il s'agit une variable objet ou d'une référence (pas d'opérateur spécial pour signifier que la variable déclarée est une référence).

C'est le compilateur qui reconnaît s'il s'agit d'une variable simple ou d'une référence :

- si le type est primitif => La variable déclarée est simple et contenue dans la pile
- si le type est une classe => La variable déclarée est une référence sur un objet de la classe, on devra construire avec new un objet désigné par cette référence.

Remarque: Si on fait une égalité de 2 références, on ne copie pas un objet dans un second mais on fait pointer une autre référence sur le premier => Un seul objet en terme de contenus.

Méthode:

Déclarer une méthode:

Qui a accès à la méthode ?	Comment utiliser la méthode ?	type de donnée renvoyée par la méthode	nom de la méthode	(arguments envoyés à la méthode)
public	static	void	main	(String[] args)

Appeler une méthode = syntaxe nom_classe . nom_méthode (paramètres = arguments)

Private, Protected, Public = Accès aux membres d'une classe

Un membre de classe est soit Private, Protected ou Public = niveau d'accès (Monde, Package, Classe, sous-classe)

Public = Accès Monde, Package, Classe, Sousclasse = Les sous-classes ont accès aux membres public de la classe mère.
Remarque: Une seule classe public par fichier Java
Private = Accès Class seulement = Les sous-classes n'ont pas accès aux membres private de la classe mère
(ils sont créés à l'instanciation mais accessible seulement par la classe mère)
Protected = Accès Package seulement = permet de rendre les membres accessibles pour les classes filles.
Aucun des trois =: Accès Package et Class.

Constructeurs = méthode particulière appelée automatiquement à la création d'un objet pour l'initialiser.
En java, l'instanciation par new appelle la méthode constructeur.

- Le constructeur * porte le même nom que la classe
  * n'a pas de type de retour (même pas void).
- On peut définir plusieurs constructeurs pour une même classe s'ils ont une signature différente (nb de paramètres). Selon les paramètres lors de l'appel, Java choisira la version utilisée.

Si aucun constructeur n'est défini, le compilateur en crée un par défaut => Pas de paramètres et initialisation de tous les attributs à leur valeur nulle (variables numériques = 0, références = NULL, chaînes de caractère = "", …).

=> Instanciation d'un objet = déclaration de la référence + opérateur new (allocation en mémoire + construction par appel d'un constructeur)

En résumé: Classe = Modèle d'objet (prototype) constituée de
- déclaration de champs
- déclaration de méthodes
- déclaration de constructeurs
- déclaration d'initialisations static

② Application "Bonjour cela marche" sous Java

- Installer le Development Kit java (nom qui a évolué: JDK puis J2SE puis JSE).
  Ne pas confondre : JRE : Java Runtime Environnement := Exécuter les applications Java sur un type de machine.
  J2SE : Java 2 Platform Standard Edition : Développer et exécuter des applications Java
- Editer un fichier texte nomfichier.java

```
public class nomfichier
    {
    public static void main(String[ ] args)
        {
        System.out.println("Bonjour Cela marche");
        }
    }
```

- Compiler le fichier: javac nomfichier.java
  Cela crée un fichier .class
- Exécuter le fichier: java nomfichier

③ Manipuler les objets

Héritage (Inherance): mot clé **Extends**

On a 2 classes: Mère et Fille => class Fille extends Mère

classe mère = **surclasse**

classe fille = **sous classe**

La classe fille dispose des membres de la classe mère (sauf les private) et en plus des siens => Extension.
En java, toutes les classes sont dérivée de la classe Object (sommet de la hiérarchie: classe de base de toutes les autres => Elle n'a pas de classe mère.

=> Structure hiérarchique, Tous les objets en Java sont du type Object => Tous les objets possèdent déjà à leur naissance un certain nombre d'attributs et de méthodes dérivées d'Object. Une classe déclarée sans extends a Object pour surclasse immédiatement supérieure.

Static = défini une seule fois en mémoire même s'il y a plusieurs objets instanciant la classe.
=> La variable ou méthode n'appartient pas à un objet, on en a une pour une classe.

- Donnée static modifiée par un objet, le sera pour tout les autres.
- Méthode static = n'a accès qu'aux données static
  = on y accède avec le nom de la classe avant (classe.méthode)

Final = ne peut être modifié, il faut l'initialiser dès sa déclaration.
=> Classe final = Pas d'héritage = Classe ne pouvant être étendue = ne peut pas avoir de filles.
=> Méthode et Attribut final = Pas modifiable

- Méthode que l'on ne peut redéfinir (super) dans les sous classes
- Attribut déclaré final est constant (exemple PI = 3.1416).

Abstract

- Une classe abstraite ne peut être instanciée directement => Au contraire de final, il faut un héritage pour l'utiliser, elle doit être dérivée pour générer indirectement des objets. Seules ses sous-classes sont instanciables.
- Méthode abstraites = Pas d'implémentation possible avec cette classe, doit être redéfinie dans une classe fille.

L'intérêt est de regrouper des attributs et méthodes communs à un groupe de classes.
Une classe abstraite possède au moins une méthode abstraite.
Une classe avec une méthode abstraite est forcément abstraite.

Surcharge = Overloading

On implémente des méthodes dans une même classe avec le même nom mais des arguments (en type et/ou en nombre) et contenus différents.
Le type de retour doit être le même.
On parle **de polymorphisme** = plusieurs formes.

On n'a pas de mot clé, c'est implicite. L'appel selon le nombre d'arguments enverra vers une des implémentations.

@Override = Redéfinition de méthodes héritées = Overriding

On redéfinit une méthode de la classe mère **dans une classe fille** en gardant le même nom et arguments avec un contenu différent.
On utilise le mot clé @Override.
C'est utilisé dans Android avec le mot clé super (voir ensuite).

Super:

Quand on a redéfini une méthode, il y a une version surclasse et une version sous-classe.
La redéfinition cache la méthode d'origine de la classe mère mais on peut quand même appeler la version surclasse de la classe mère, on utilise le mot clé super. super.méthode();

C'est surtout utilisé pour ne pas tout réécrire: Modification = Ajout

=> On place super.méthode(); en début de méthode ce qui incorpore la méthode de classe mère puis on place le code que l'on ajoute.

Intérêts de @Override et super:

- Etre sûr de ne pas faire d'erreur de syntaxe dans les noms de méthodes (Android veut absolument un des siens onCreate, onStart ou onResume... En cas de redéfinition avec erreur dans le nom, sans @Override le compilateur ne verra pas d'erreur.
- Ajouter des éléments en incluant la méthode de surclasse.

Implements: Spécification d'interface

Java interdit l'héritage multiple pour éviter des mélanges dans l'utilisation. Mais on a parfois besoin d'utiliser des éléments de plusieurs origines (gestion clic bouton: OnClickListener, pour démarrer un autre Thread Runnable,...)

Java propose les interfaces pour cela. L'interface précise quelles opérations on peut effectuer sur un objet particulier. (ce qui sort et rentre). Une classe peut implémenter plusieurs interfaces (alors qu'elle n'hérite que d'une seule surclasse =extends).

implements permet de spécifier quelle interface on utilise pour une classe. On peut implémenter plusieurs interfaces dans la même classe. => Oblige une classe à définir certaines méthodes.
Une interface est à la base une classe abstraite avec toutes ses méthodes abstraites.

On l'utilise avec les écouteurs (Listener) par exemple où il faut juste redéfinir le contenu de la méthode, avec le multithread pour démarrer un nouveau Thread.

Remarque: On utilisera des interfaces. Si on veut en créer une, on utilise sous Android AIDL (Android Interface Definition Langage: http://developer.android.com/guide/developing/tools/aidl.html)

Résumé: On veut faire notre programme en réutilisant du code déjà écrit = héritage: extends.
On ajoute à sa classe les éléments existants d'une seule classe = étendre la classe mère
(class ClasseFille extends ClasseMere) si ClasseMère n'est pas final.

Si on doit adapter des méthodes de la classe mère pour notre utilisation, on les redéfinit en réécrivant (pas de mot clé).
- soit utiliser la version originale (rien à redéfinir, on appellera la méthode ClasseMère.Méthode)
- soit compléter la version originale pour l'adapter aux apports de la classe fille: super.méthode(); puis code ajouté
- soit réécrire la méthode: Redéfinition

En cas de réécriture, le mot clé super fera appel à la méthode de surclasse.

=> 4 types de classes:
Classe simple
Classe dérivant d'une super-classe; Extends
Classe implémentant une interface: Implements
Classe dérivant et implémentant

View

Classe qui permet de gérer l'écran en fournissant des méthodes de gestion d'interaction écran (clic, zoom, ...)

Bundle

Lot d'éléments regroupés.
Utilisé dans Android pour mémoriser des lots de valeurs non sauvegardées entre changements d'états de l'application.

Exception

Il y a 2 types d'erreurs en Java: Surveillées / non surveillées (trop graves).
Dans le cas des erreurs surveillées, Java génère une exception et demande de prévoir un traitement.
=> On essaie un traitement sensible (try), on "attrape" l'exception avec catch (e est un objet de la classe Exception).
try { le_traitement } catch(Exception e) { A faire en cas d'erreur }

Message

Demande d'exécution d'une méthode à un objet. Voir chapitre sur les Intent.

④ Classes prédéfinies: Utilisation de import

Chaque version de Java apporte des classes prédéfinies. Pour pouvoir utiliser un élément d'une classe qui n'est pas System, il faut utiliser **import** = Importer une classe pour l'utiliser dans la classe que l'on définit. (Les classes sont regroupées dans des fichiers JAR: fichier compressé java).

On place en début de programme import nom_paquet

⑤ Sources

Java pour les enfants: http://ftp-developpez.com/java/livres/javaEnfants/JavaEnfants.pdf
Du C au Java: http://www.eteks.com/coursjava/tdm.html
Penser en Java: http://uv.utbm.free.fr/tutoriels/Android/docs_Android_Java_Linux/Java/PenserEnJava.pdf
http://download.oracle.com/javase/tutorial/java/IandI/index.html
http://www.particle.kth.se/~lindsey/JavaCourse/Book/Part1/Supplements/Chapter04/overrideVsOverload.html
Learn Java for Android http://mobile.tutsplus.com/tutorials/android/java-tutorial/
http://java.sun.com/developer/onlineTraining/Programming/BasicJava2/oo.html

II) Plateforme de développement Android SDK

① Installation

Source: http://developer.android.com/sdk/installing.html

Installation de l'environnement de **développement Java JDE**
Installation du **SDK Android**
Installation de l'**IDE Java eclipse**
Installation dans eclipse du **plugin ADT** (Android Development Tool)

Voir annexes

Remarque: NDK (Native Development Kit)

Le NDK est utilisé avec le SDK. Il permet de développer des applications en code natif. On peut réutiliser les applications développées en C/C++. L'application développée avec le NDK est encore mise dans un fichier paquet .apk et démarrera dans une machine virtuelle dans l'appareil. => On garde le modèle Android.

② Application basique "Bonjour cela marche" sous Android

a) Créer un appareil virtuel = AVD (Android Virtual Device)

On crée un téléphone virtuel afin de tester le programme = AVD (Android Virtual Device).

Sous Eclipse: Accès à SDK Android et AVD Manager par l'icône  ligne AVD, new,...

(ne pas mettre trop de mémoire sinon cela plante => 64 Mib, pas 6000 Mib).

On pourrait aussi le faire plus loin au lancement de l'application ou en ligne de commande sous tools en root:
android create avd --target 2 --name rtphone (target = version d'android, rtphone = nom du téléphone virtuel)

b) Créer le projet sous eclipse

- lancer eclipse

- file / new / other / android / android project, next
  - * mettre un **nom de projet : intro**
  - * mettre la **cible (target)** qui correspond à l'étape précédente AVD = cocher devant Android 1.6
    (à la création de l'AVD, des messages indiquent la version Android correspondante)
  - * mettre un **nom d'application**: Essai
  - * mettre un **nom de paquet Java**: Package Name com.exemple.UV_UTBM (pas de -)
  - * cocher "**create activity**" si ce n'est fait,
    mettre le **nom de projet**
  - * **Min SDK Version***:* Mettre la valeur de la colonne "API Level" de la cible choisie (voir tableau Build Target)

  Finish

Remarque: Pour modifier à postériori le nom de l'application "application name":
choisir strings.xml dans l'arborescence du projet,
puis dans la fenêtre du milieu, prendre l'onglet en bas "strings.xml" (pas ressources).

- sur le bureau d'eclipse, cliquer sur "goto workbench"
  => On voit le projet dans la fenêtre Package. Cliquer sur le projet
  Aller dans src > com.exemple.UV_UTBM, ouvrir le fichier nom_projet`.java`
  On a le code de base du projet.

c) Construire l'UI avec fichier XML = User Interface

On s'intéresse ici à la construction de l'interface graphique par fichier xml associé, pas à la méthode par programmation Java. (dans le programme Java, on instancie des objets graphique comme on va le faire en XML, puis on affiche chacun d'eux avec setContentView contrairement à la méthode par XML où on appelle l'affichage une fois pour le contenu du fichier XML).

- Modifier le **code JAVA** en ajoutant les éléments en gras

```
package com.exemple.UV_UTBM;

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

public class intro extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
   }
} /* fin intro extends Activity */
```

- Activity est la classe
- Nécessaire pour utiliser widget TextView
- extends de la classe Activity
- Au démarrage de l'application: OnCreate
- Affiche le **layout** défini dans **main.xml**

- Modifier le **fichier main.xml**

```
ajouter les lignes suivantes entre les balises de layout
(  Entre <...Layout et ses lignes > et </...Layout>,
   "..." précédent étant Absolute ou Relative ou ... ):

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Bonjour, cela marche"
/>

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Je confirme"
/>
```

- fichier xml
- Si LinearLayout = ligne après ligne
- Première ligne de texte "Bonjour, cela marche"
- Seconde ligne de texte "Je confirme"

Remarque: Fin de balise: On peut utiliser /> ou ></nom> Ici: TextView se termine par />
EditText se termine par > </EditText>

d) Exécuter l'application

Enregistrer avant de compiler et démarrer !!!

Si on fait compiler et démarrer sans enregistrer, cela lance la dernière version enregistrée.

D'autre part Run fait la compilation avant démarrage si on a coché Project/Build Automatically (par défaut).

- Dans eclipse: Run / Run ce qui compile et démarre l'application.
Choisir "Android Application".

ETRE PATIENT !! (écran noir puis petit android puis un gros avec changement de brillance de lettre puis téléphone).
(regarder l'onglet Console, on doit voir après installation de nom.apk, puis "Starting Activity")

- Lancer l'application: Touche Menu pour déverrouiller le téléphone et voir l'application.
On voit une fenêtre avec le nom de l'application et dedans le message.

En cas d'incohérence de nom entre projet, activité, on aura une erreur avant compilation ou après avec message de fermeture
(Exemple: package = calculatrice,
appli = Calculatrice)

Source: http://developer.android.com/guide/tutorials/hello-world.html
(qui n'utilise que le fichier Java sans main.xml dans un premier temps).

e) Modification de la manière de définir une valeur

Au lieu de définir le texte lors de la définition du widget dans main.xml, on va faire référence à une chaine de caractère de nom hello définie **dans res/values/strings.xml** (syntaxe @type valeur/valeur).

- Modifier la ligne android:text="Je confirme" qui est une définition directe du texte
  en android:text="**@string**/hello" qui est une définition dans res/values/strings.xml car de type string et utilise la balise de nom hello dans ce fichier
- Aller dans le fichier res/values/strings.xml, modifier la ligne de la balise hello
  <string name="hello">autre affichage</string>
- Enregistrer et démarrer pour vérifier que l'affichage change.

③ Outils de Débogage

- Ajouter des messages de Log pour indiquer l'évolution du programme et renvoyer des valeurs de variables.
  On a le choix entre différents contenus de message.

Log.v	Log.d	Log.i	Log.w	Log.e
Verbose	Debug	Info	Warn	Error

Verbose est le plus détaillé mais ne doit pas être compilé dans une application sauf en phase de développement, Error est le moins détaillé.

Attention: Ces messages ralentissent l'application et consomment des ressources mémoires (Log.v surtout).

- Pour utiliser les messages de Log:
  * import android.util.Log;
  * Définir le nom de TAG des messages: private static final String TAG = "Nom de mon Activité";
  * Appeler les messages. Exemple affichage de valeur de variable i: Log.e(TAG, "index=" + i);
  * Ouvrir la fenêtre LogCat (onglet DDMS ou Debug ou Window/Perspective/...), ne pas filtrer (icônes V, D,...).

Sources: http://developer.android.com/reference/android/util/Log.html
http://android.cyrilmottier.com/?p=57

- Pour avoir des informations sur une erreur: Cliquer sur Debug, pour revenir on cliquera sur Java.
  Ensuite Window/ Other Window / **LogCat**.

- Pour utiliser l'outil DDMS (Dalvik Debug Monitor Server):
  *L'installer quand on ajoute dans eclipse Android - https://dll-ssl.google.com/android/eclispse/
  => On a "developer tools" qui s'affiche, DDMS est dans les sous-menus.
  Il est placé dans le SDK/tools.
  * Afficher son icône dans eclipse = Menu Window / Open Perspective / Other.../DDMS

Remarques: - Si on ne voit pas de message: - Aller dans la perspective DDMS au lancement de l'application,
- Cliquer juste sur le nom d'un processus actif (Onglet Devices après qu'il passe Online).
- Si on veut filtrer les messages, cliquer sur v, d, ,i, w ou e. (on verra tous les messages de hiérarchie inférieure)

④ Installer un projet existant

Lors de l'installation des outils de développement, si on l'a accepté, des exemples de projet développés par Google (samples) ont été installés. Ils sont dans le SDK puis Samples puis le version Android choisie.

Pour installer le projet et en même temps une copie du projet dans son répertoire de travail pour le modifier en gardant la source d'origine. (on aurait pu faire un nouveau projet et cocher "Create Project from existing source" mais on travaille sur la source sans copie)

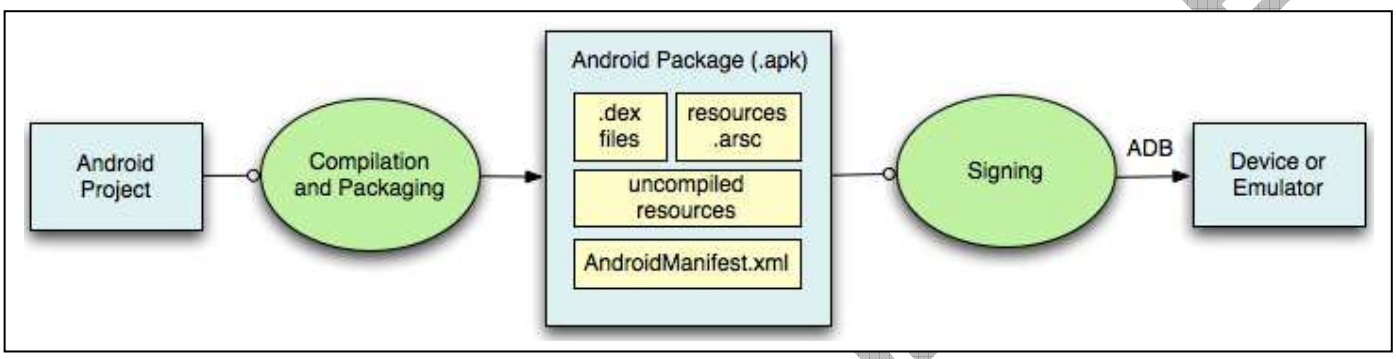
Dans Eclipse:
* File / Import
* Choisir General puis "Existing Projects into Workspace"
* "Select root Directory", Browse
  -> <répertoire SDK>/samples/android-7/BluetoothChat
* Cocher "Copy Projects into workspace"
* Finish

III) Android

① Etapes du développement d'application

Android est un OS qui utilise Linux pour gérer ses composants logiciels et matériels => 1 matériel = 1 cas particulier pour Linux. Pour qu"Android soit utilisable sur des machines différentes, il utilise une machine virtuelle: DALVIK.

- Le développeur fait son application en **JAVA** avec le SDK puis compile.
- La compilation va engendrer un fichier exécutable Android **.dex** (anDroid Executable)
- Puis il y a création d'un paquet Android .**apk** (APK = Android PacKage) qui regrouper exécutable et ressources externes (autorisations, images, sons, ...)
- Le système démarre le composant virtuel ou vérifie que le composant réel est connecté
- Il envoie le paquet apk sur la machine cible.
- L'appareil démarre l'application en utilisant le paquet apk



Source: http://developer.android.com/guide/developing/building/index.html

② Contenu d'un projet Android

Android permet d'utiliser et d'intégrer l'utilisation de différents éléments (Multimédia, navigation GPS, affichage de cartes, jeux, téléphones sans fil voix et données, messages, photo et vidéo, wifi, bluetooth, compas, accéléromètres,...)

Dans le répertoire res, on placera les ressources nécessaires pour le fonctionnement de l'application (images, définitions d'affichages, valeurs de constantes,...) définies dans des fichiers xml.

Pour gérer les fichier xml par eclipse, utiliser le bouton "New Android XML files"

Le projet Android contient un dossier **res** = ensemble des ressources liées projet.
Il est utilisé par ADT pour créer R.java.
Il contient des sous dossiers regroupant les ressources selon leur type.

Par défaut à la création du projet, les sous-dossiers sont:
drawable : images (png, jpg, gif), Drawables.
layout : Android crée l'interface graphique avec des fichiers XML qui sont placés dans ce dossier.
values : Contient des fichiers décrivant les valeurs utilisées par l'application. (strings.xml = chaines de caractères, arrays.xml = tableaux, couleurs utilisées, dimensions, …)

L'arborescence de l'application est:

```
AndroidManifest.xml        ➔  Autorisations

src ──┬ nom.java             ➔  Fichiers sources du projet
      └ nom.java

res ──┬ drawable             ➔  Répertoire des icône, images, fond écran
      ├ layout               ➔  Affichages: main.xml = organisation de l'interface utilisateur
      │                                     autre xml si on a plusieurs fenêtres
      └ values               ➔  Définition de constantes: strings.xml = chaines de caractères
                                                          integer.xml, ...
```

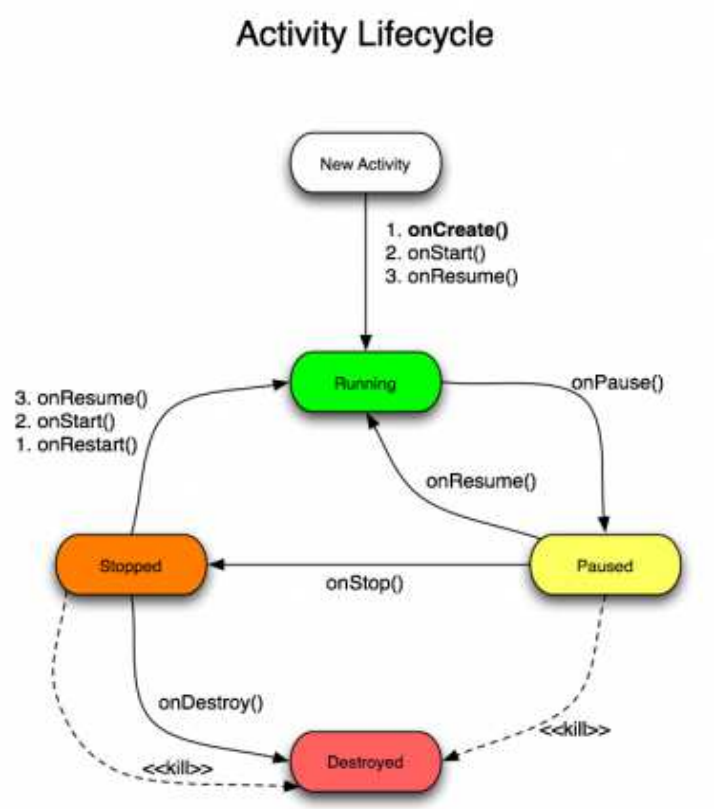
Sources architecture: http://www.ibm.com/developerworks/opensource/library/os-android-devel/

③ Cycle de vie (Lifecycle) d'une application Android

L'activité passe entre différents états:

- **Running:** Application active = visible en premier plan (Foreground).
- **Paused:** Application visible en arrière plan, toujours prête à fonctionner (ressources système préservées).
- **Stopped:** Invisible car une autre activité est passée en premier plan (ressources système préservées sauf si le système en a besoin).
- **Destroyed:** Désactivée, ressources système libérées.

Les transitions se font selon le scénario d'utilisation (lancement d'autres applis, arrivée de sms, ...)
A chaque transition, le système exécute des méthodes: onCreate, onStart, onResume, onPause, onStop, onDestroy



L'évolution de l'activité dans le cycle de vie vient:

- ou des actions de l'utilisateur (appuyer sur une icône).
  Démarrage:
  Passage à l'état Running = onCreate, onStart, onResume.
  Arrêt:
  Passage à Stopped puis Destroyed = onStop, onDestroy

- ou d'une tâche prioritaire que le téléphone doit gérer (réception appel téléphonique):
  Passage de Running à Paused = onPause,
  Retour à Running = onResume.

- ou le programme utilise plusieurs activités,
  Passage de Running à Paused = onPause,
  Retour à Running = onResume.

Exemple avec un scénario d'utilisation	Succession d'opérations correspondantes
1) Premier démarrage de l'activité	onCreate / onStart / onResume
2) Saisie d'une valeur puis clic sur le bouton Home	onPause, onStop
3) Redémarrage de l'activité	on Restart, onStart, onResume
4) Fermeture de l'activité	onFinish, onStop, onDestroy
5) Redémarrage	onCreate / onStart / onResume
6) Retour au menu Home	onPause (pas finish ! contrairement au 4), onStop
7) Démarrage du navigateur, puis Home	Rien
8) Redémarrage de l'activité	on Restart, onStart, onResume
9) Mise en veille du téléphone, l'activité étant 1er plan	onPause (pas finish)
10) Sortie de veille, l'activité revient au 1er plan	onResume

Sources Cycle de vie: http://androtruc.wordpress.com/2010/07/19/le-cycle-de-vie-dune-application-android/
http://developer.android.com/reference/android/app/Activity.html

Attention: Le basculement d'affichage entre paysage et portrait correspond au redémarrage de l'activité.
(CTRL F11/ F12 avec l'émulateur)
=> On passe par: onSaveInstanceState() - onPause() - onStop() - onDestroy() - onStart() - onRestoreInstanceState()

Source émulateur: http://developer.android.com/guide/developing/tools/emulator.html#controlling

Pour éviter cela, on peut verrouiller l'activité dans un mode d'affichage.

Dans le code JAVA	setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
	ou
	setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

Dans le xml	android:screenOrientation="landscape" ou "portrait"

④ Gestion des permissions: AndroidManifest.xml

Par défaut une application n'a pas le droit d'utiliser les composants du téléphone. Cela signifie que pour les utiliser on doit autoriser leur utilisation.

Pour cela on déclare les permissions dans AndroidManifest.xml avec la balise <uses-permission.

Exemples: Pour que le programme gère la lumière flash du téléphone:

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

Pour que le programme ait accès au niveau de batterie:

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

On verra plus loin les exemples du bluetooth, vibreur, puce GPS, wifi.

⑤ Type d'organisation d'écran

L'affichage est défini par des Layout dans des fichiers xml, main.xml étant le premier. On appelle l'affichage défini dans un fichier xml nom.xml par setContentView(R.layout.nom);

Il existe différents layout:

AsoluteLayout	Tous les éléments sont placés en position absolue (on spécifie x et y par android:layout_x et y, le haut à gauche étant 0,0). Solution dépréciée car pas évolutive (inadapté à modif écran ou ajout d'élément).
FrameLayout	Affiche un seul élément à la fois. Si on a plusieurs éléments, chacun est traité séparément = position basée sur le coin gauche en haut d'écran. Des éléments se chevauchant sont affichés ainsi. => Utile pour une image en fond. Utile aussi si le programme gère et modifie la visibilité d'éléments (setVisibility en java équivaut à android:visibility dans xml,). La visibilité est soit visible, invisible (pas affiché mais présent) ou gone (pas affiché et supprimé du layout).
LinearLayout	Affiche sur une ligne les éléments. android:orientation indique si c'est vertical ou horizontal. N'a d'intérêt que si on imbrique plusieurs layout (nested). Exemple: un LinearLayout vertical qui comprend un bouton puis un LinearLayout horizontal comprenant Textview et EditView puis un second LinearLayout horizontal comprenant Textview et EditView
RelativeLayout	Les positions des éléments sont définies par rapport au conteneur ou par rapport à d'autres éléments.
TableLayout	Définition en ligne et colonne d'un tableau. On définit dans le xml les lignes par des éléments Tablerow. Android déduira des déclarations d'un Tablerow les colonnes à utiliser (colonne 0 pour le premier élément du Tablerow, colonne 1 pour le 2ème,... On peut spécifier la colonne de début par android:layout_column="1" pour la 2ème

Remarque: On peut définir des interfaces selon l'orientation de l'affichage qui peut changer durant l'utilisation.

On utilisera des fichiers xml dans les répertoires res/layout-land – landscape UI paysage
res/layout-port – portrait UI
res/layout-square – square UI

On peut utiliser différentes instructions dans un xml définissant un layout:

- Définir le widget à afficher:
  - Button pour un bouton,
  - Imageview pour une image,
  - ScrollView pour faire tourner le Layout
  - EditText pour une case de saisie ou affichage de variable texte
  - Textview pour un texte fixe
  - ...
- Définir ses caractéristiques
  - id identifiant de l'élément à utiliser dans le programme
  - layout_width pour la largeur layout_height pour la hauteur
  - layout_marginTop, layout_marginBottom, layout_marginLeft, layout_marginRight pour les marges
  - layout_gravity pour la position relative (top, bottom, left, right, center_vertical, fill_vertical,...)
  - layout_weight Specifies how much of the extra space in the layout to be allocated to the View
  - layout_x, layout_y Coordonnées absolues
  - scaleType
  - onclick (pour un bouton) Méthode associée en cas de clic
  - editable (pour un EditText) Pas éditable = pas de saisie, seulement utilisé pour affichage
  - src="@drawable/nom" indique qu'il faut utiliser l'élément nom dans le répertoire drawable
  - background
- Dans le cas de l'affichage relatif:

Position relative au conteneur	layout_alignParentTop (true/false) haut élément aligné sur haut du conteneur layout_alignParentBottom, layout_alignParentLeft, layout_alignParentRight layout_centerHorizontal = élément centré dans le conteneur layout_centerVertical, layout_centerInParent pour horizontal et vertical
Position relative à d'autres éléments	layout_above="@id/nom" placé au dessus de nom, layout_below placé en dessous, layout_toLeftOf, layout_toRightOf, layout_alignTop le haut de notre élément est aligné avec celui de nom layout_alignBottom, layout_alignLeft, layout_alignRight, layout_alignBaseLine

Remarques: * **wrap_content**: limité au contenu = minimum.
match_parent (fill_parent avant API 8): Le plus grand possible selon espace disponible.
* Unités des valeurs: On peut utiliser:

Unités absolues:	px: Pixels de l'écran. in: Inches mm: Millimeters. pt: Points - 1/72 de inch.
Unités relatives:	dp: Density-independent Pixels - Unité abstraite basée sur la densité de l'écran. 1 dp = 1 pixel pour un écran 160 dpi. sp: Scale-independent Pixels - Comme le dp mais tenant compte aussi de la taille de police. A utiliser quand on spécifie la taille de police..

* Imbrication de Layout: A l'intérieur du xml, on peut placer plusieurs layout
Exemple: pour faire des onglets, on fait un Absolute Layout en y incorporant un LinearLayout (voir source)
Exemple: Dans un LinearLayout vertical, on place des LinearLayout horizontaux.

Sources: Définition des layout: http://developer.android.com/guide/topics/ui/layout-objects.html
Objets d'un layout: http://developer.android.com/guide/topics/ui/layout-objects.html
Onglets: http://androtruc.wordpress.com/2010/07/13/onglets/
Généralités: http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layouts
http://android.developpez.com/cours/

⑥ Multitâches: Notions de Threads = Tâches ou fil, Handler, Messages

On a vu en tp un système à µP qui n'exécute qu'un seul programme = séquentiel.
Le µP exécute le programme, si un événement externe survient, il reçoit une interruption et exécute une routine d'interruption, le programme principal n'est plus exécuté. Puis il reprend où il en était dans le programme principal.

Android est un système multitâche qui exécute plusieurs programmes à la fois. Le système gère l'utilisation du µP qui fait fonctionner la machine Android.

Android utilise un seul processus mais plusieurs tâches (Threads). Il y a un Thread principal qui gère l'affichage: "UI Thread".

ON NE PEUT BLOQUER L'UI THREAD PLUS DE 5s
=> Pas de boucle d'attente de saisie de valeurs ou choix

Exemple: Si on fait une autre opération trop longue dans l'UI Thread (télécharger, scruter des données reçues), l'affichage est bloqué, le téléphone ne répond plus. Ensuite il ferme l'application.

ON NE PEUT STOPPER L'EVOLUTION DE onCreate à onResume pour attendre l'initialisation d'une variable.

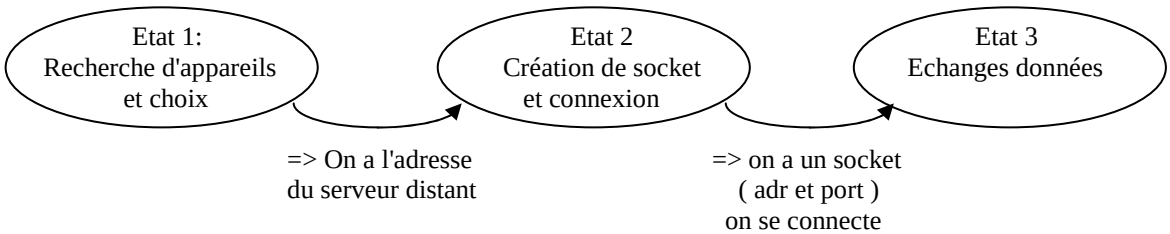
Exemple: Si on ne fait pas attention à l'évolution du programme, on demande un choix de valeur, on l'utilise ensuite. Mais Android n'attend pas que le choix soit fait, donc si on ne met pas des conditions, la suite utilisera une valeur vide (pointeur null => erreur => fermeture).

SOLUTION:

On programme par état: Définir des états du système
Faire le traitement correspondant à un état
Si on remplit les conditions passer à l'état suivant.

=> Pas de variables utilisées alors qu'elles ne sont pas définies.
On programme avec plusieurs tâches parallèles que le système gère
=> Pas de blocage, c'est le système qui gère (inconvénient: moins réactif).

Exemple: Bluetooth



Dans l'état 3, avec le socket, on a des flux entrant/Sortant que l'on gère.

Pour la lecture, n'ayant pas d'interruption matérielle, on doit scruter le file d'entrée => boucle infinie
=> Nouvelle tâche en parallèle pour ne pas bloquer.

Il faut créer d'autres Thread. Pour créer des Thread en tâches de fond (background):

- Utiliser la classe AsyncTask permet de définir un opération réalisée en tâche de fond.
  Elle fournit une structure fixe prédéfinie de "event handlers" = moyen d'échanger avec l'UI principale.

- Créer ses threads, créer et utiliser ses propres Handlers pour communiquer avec l'UI principale.

a) AsyncTask

➔ Création dans l'UI Thread de Thread avec AsyncTask: On étend la classe AsyncTask en indiquant les types de variables d'entrée, les messages d'état du thread progress reporting values) et les valeurs résultats renvoyées possibles (void s'il 'y en a pas).

➔ Dans la nouvelle tâche AsyncTask, on a

- Une méthode **doInBackground** pour définir les traitements faits en tâche de fond.
  Cette méthode ne dialogue pas avec l'UI Thread.
  En utilisant **PublishProgress** elle envoie des valeurs à la méthode **onProgressUpdate.**
- Une méthode **onProgressUpdate** qui peut communiquer en cours de traitement avec l'UI Thread
  On peut envoyer des valeurs intermédiaires. (ex: Progression d'un téléchargement).
- Une méthode **onPostExecute** pour envoyer les résultats en fin de traitements à l'UI Thread.

Source: http://developer.android.com/reference/android/os/AsyncTask.html

Remarque: On ne peut exécuter une instance d'AsyncTask qu'une fois.

b) Création de Thread

AsyncTask est une organisation stricte. Si on veut plus de flexibilité, on crée son Thread et on dialogue entre les Threads par messages de Handler.

Pour créer un Thread:

- Soit on hérite (extends) de la classe java.lang.Thread en redéfinissant (override) sa méthode run.
- Soit on implémente (implements) l'interface Runnable en utilisant sa méthode run.

Exemple: Implémentation de l'interface Runnable

Dans la classe

```
Dans onCreate, ou onStart ou onResume, on appelle la méthode Traitement();

private void Traitement() {
      // Déplacement des traitements dans un Thread enfant.
      Thread monThread = new Thread(null, TraitementEnFond,"Background");
       monThread.start();
}

// Runnable qui exécute la méthode des traitements de fond.
 private Runnable TraitementEnFond = new Runnable() {
      public void run() {
            monThreadTraitements();   // appel de la méthode des traitements
      }
};

private void monThreadTraitements() {   // Méthode qui fait les traitements en tâche de fond.
          // Traitements
}
```

Mais il faut ajouter des moyens d'échanges d'informations d'état et de résultats pour que le Thread des traitements échange avec le Thread principal = Synchronisation.

Voir exemple Bluetooth en fin de cours pour la réception.

c) Synchronisation = Echanges entre Threads

Pour faire des échanges entre Thread, on déclare un **gestionnaire = Handler**. Il permettra d'échanger des messages entre les Threads.

-> Dans l'UI Thread

* Créer une instance de la classe Handler dans l'UI Thread
  et redéfinir sa méthode de gestion de message: handleMessage()

```
final monhandler = new Handler() {
    @override
    public void handleMessage(Message Msg) {
      Traitements du message en utilisant ses champs   Msg.xx
    }
}
```

* Créer un Thread depuis l'UI Thread.

-> Dans le Thread créé:

* Créer un message du handler créé: Message Msg =monHandler.obtainMessage();
* Affecter les données à échanger à un champ du message.
* Envoyer le message du Thread créé vers l'UI Thread: monhandler.sendMessage(Msg);

Remarque: On peut aussi envoyer des messages retardés ou à une date précise sendMessageDelayed() et sendMessageAtTime().

Message est un objet prédéfini dans android.os.message avec les champs:

arg1 = un int.
arg2 = un int.
obj = objet à envoyer dans le message.
replyTo = objet Messenger qui identifie le destinataire.
what = code défini par l'usager pour identifier le message (par exemple type de réponse)

Source: http://developer.android.com/reference/android/os/Message.html
voir l'exemple Bluetooth plus loin pour envoyer les données reçues à l'UI Thread

d) Thread et Handler

Si on conjugue Thread du b et échanges (handler, message) du c.

- Ajouter les imports:

```
import android.os.Handler;
import android.os.Message;
```

- Définir une nouvelle instance de gestionnaire en début de classe en redéfinissant sa méthode handleMessage

```
Handler handler=new Handler() {
@Override
public void handleMessage(Message msg) {
        if ( message != null )
          {
        }
}
};
```

- Dans onStart, créer le nouveau Thread pour le traitement

```
Thread background=new Thread(new Runnable() {
        public void run() {
                try {
                        handler.sendMessage(handler.obtainMessage());
                        }
                } catch (Throwable t) {
                // Termine le thread en arrière–plan
                }
        } // fin Run
});
```

- Lancer le Thread

```
background.start();
```

Remarque: Pour faire des messages textes transitoires simples qui s'affichent: Toast

On les mettra dans le run du runnable doUpDateGUI

```
Toast monToast = Toast.makeText(getApplicationContext(), " mon message ", Toast.LENGTH_SHORT ).show();
```

IV) Charger l'application sur le téléphone réel

L'émulateur ne permet pas de tester des applications plus évoluées, en particulier utilisant des modules de communication du téléphone. On doit donc les tester sur matériel réel. Pour cela on utilise l'ADB (Android Debug Bridge).

① Installation et préparation de ADB

On développe et débugge avec un appareil réel Android comme avec un émulateur. Mais il faut préparer l'implémentation :

a) Dans le programme, fichier Manifeste AndroidManifest.xml:

- Déclarer l'application comme "debuggable" dans le manifeste Android:
  Dans AndroidManifest.xml, dans la partie <application>, ajouter avant le > de fermeture

```
android:debuggable="true"
```

b) Sur l'appareil Android recevant le programme:

- Activer « USB debugging » sur l'appareil qui reçoit l'application:
  Menu / Applications / Development / USB Debugging
- Redémarrer sinon ensuite on verra le téléphone toujours offline

c) Sur le PC de développement:

- Configurer le PC sur lequel on développe l'application avec éclipse :

Windows	Installer les drivers ADB = Android Debug Bridge (charger les drivers, connecter l'appareil, indiquer les drivers)
Linux	+ Passer en **root** + Créer un fichier **/etc/udev/rules.d/51-android.rules** + Ajouter dans ce fichier la ligne pour les versions Ubuntu après la 6 (Dapper) **SUBSYSTEM=="usb", SYSFS{idVendor}=="0bb4", MODE="0666"** La valeur de SYSFS{idVendor} dépend du constructeur de l'appareil : Acer `0502` Dell `413c` Foxconn `0489` Garmin-Asus `091E` HTC = `0bb4` Huawei `12d1` Kyocera `0482` LG `1004` Motorola `22b8` Nvidia `0955` Pantech `10A9` Samsung `04e8` Sharp `04dd` Sony Ericsson `0fce` ZTE `19D2` `Google Nexus One 18d1` `voir http://www.linux-usb.org/usb.ids` Vérifier le code en faisant **lsusb** dans un terminal, code = 1ère partie après ID + Changer les permissions : **chmod a+rx /etc/udev/rules.d/51-android.rules** + Redémarrer (au moins ./adb kill-server puis ./adb start-server qui démarre un pid)

② Utilisation de ADB

- Vérifier le bonne connexion : Dans le répertoire Platform Tools du SDK, taper "adb devices ".
  (./adb devices sous linux)

* Si on a connexion, on verra apparaître une liste qui inclut l'appareil.
* S'il y a un problème de déclaration (valeur SYSFS{idVendor} par exemple),
  on a « ?????????? no permissions »
* S'il est marqué offline, alors qu'allumé, vérifier que " USB Debugging" a été activé
  et le téléphone redémarré.

- Lancer le transfert et l'exécution : Sous éclipse, run ou debug comme d'habitude.
  Choisissez l'appareil qui ne doit pas être offline.

- Pour installer l'application sur l'appareil, on utilise "adb install <chemin vers apk>".

③ Supprimer une application

Sur le téléphone: - Paramètres / Applications / Gérer les applications /
- Sélectionnez l'application à désinstaller :
- Désinstaller / OK

④ Sources

http://developer.android.com/guide/developing/device.html
http://wiki.frandroid.com/wiki/Cours_sur_l%27utilitaire_Android_ADB
http://developer.android.com/guide/developing/tools/adb.html
http://androidcommunity.com/forums/f4/how-to-install-apps-using-adb-4482/

V) Réalisation de l'interface graphique: UI User Interface

① Notions sur le XML

xml est fait de balises (groupe d'éléments définissant un objet). On peut créer ses balises (contrairement à http).

```
<nom
        def simple
        def simple
        >
    <def complexe
            def simple
            def simple
    />                                  <!—Une balise ouverte doit être fermée -->
    <def complexe
            def simple
            def simple
    >
    </def complexe>
</nom>
```

Remarques:	
	Une balise doit commencer par sa racine <nom_de_balise>. Elle doit être fermée</nom_de_balise>.
	Mise en remarque de lignes de code En java, mettre entre /* et */ ou // comme en C. En xml, mettre entre <!-- et --> comme en html.
	Les attributs sont mis entre guillemets " ". Pas de ' (=> dans les textes en français, pas d'apostrophe)
	Pas de majuscule dans les noms de balises. Pas d'espace dans les noms de balises.
	On peut terminer une balise par /> ou ></nom_de_balise>

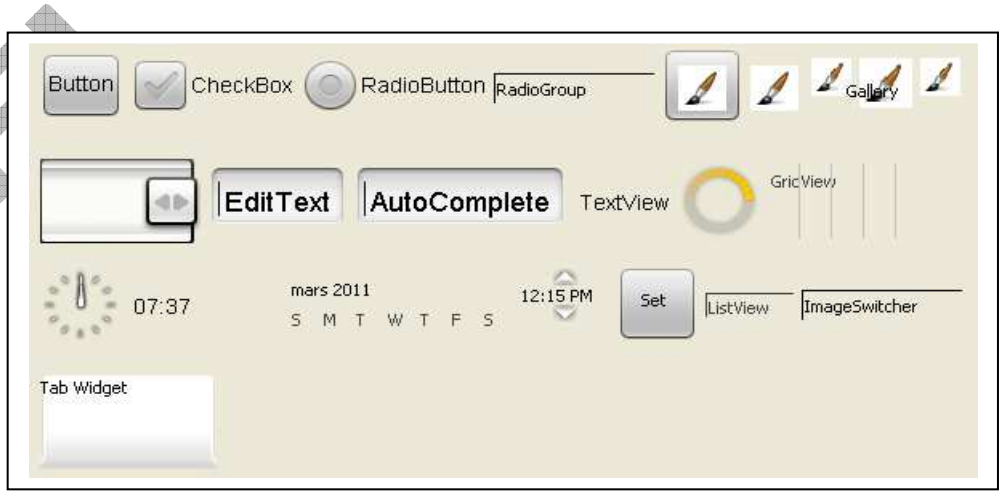
ATTENTION: XML est très strict. Android bloquera si un fichier est mal défini (affichages, chaine de caractère ou autre valeur) ou si la définition n'est pas au bon endroit.

Android accède à <u>un affichage</u> par R . répertoire . nom_affichage
R.layout.main => Affichage des balises du fichier main.xml

Android accède à <u>une valeur</u> par R . nom_fichier.valeur où nom_fichier.xml est défini dans le répertoire Values
R.strings.message1 => Affichage de la balise message1 du fichier strings.xml du répertoire values

② Eléments graphiques

On peut utiliser plusieurs éléments dans l'interface graphique. Dans le fichier xml de déclaration, un identifiant est associé à l'élément pour l'utiliser dans le programme.



- Aller dans res / layout, cliquer sur main.xml.
  En bas on a soit la vision graphique, soit le code xml.

Source: fin de http://developer.android.com/guide/tutorials/hello-world.html

③ Utilisation de Droiddraw

DroidDraw est une application gratuite qui permet de concevoir le layout de l'application en wysiwyg:
Téléchargeable à http://www.droiddraw.org/

- Lancer l'application DroidDraw.
- Passer en AbsoluteLayout.
- Placer un EditText et en-dessous un bouton.
- Cliquer sur Generate.

On retrouve 2 balises: EditText et Button. Chaque a son identifiant (ligne android:id="@id/nom")

- Modifier la balise EditText en ajoutant la ligne android:editable="false"
  => La case de texte ne sera utilisable qu'en sortie (affichage), pas en entrée (entrée de texte = getText)
- Modifier la balise EditText en ajoutant la ligne android:text="texte de départ"

- Copier/Coller le code XML.
- Lancer l'application et vérifier que l'on retrouve l'organisation définie dans DroidDraw.

Remarque: Google a défini une charte graphique :
http://developer.android.com/guide/practices/ui_guidelines/widget_design.html

④ Ajout d'icône

Version 1.5 et précédente: 1 écran cible

Faire un fichier icon.png de 48 pixels x 48 pixels contenant l'image à associer comme icône.
Modifier la ligne dans **AndroidManifest.xml** si nécessaire: <application android:label="@string/app_name"> en <application android:icon="@drawable/icon" android:label="@string/app_name">
Créer un répertoire "drawable" sous le répertoire \res si nécessaire.
Mettre dedans le fichier image représentant l'icône: icon.png

Version 1.6 et suivante: Multiscreeen

On reprend le principe précédent mais avec plusieurs cibles: ldpi: Low icône 36 x 36 pixels mdpi: Medium icône 48 x 48 pixels hdpi: High icône 72 x 72 xhdpi: Extra High après version 2.2
Définir quel type d'écran peut accueillir l'application avec les variables dans la partie supports-screens du manifeste: android:smallScreens, android:normalScreens, android:largeScreens, android:anyDensity
Mettre l'image de l'icône dans les répertoires correspondants de res/drawable

⑤ Ajout de fond d'écran

a) Changer la couleur du fond

Ajouter dans main.xml à la définition du layout android:background="#000044" (000044 = code couleur)

b) Image en fond

- Placer l'image background.9.png dans les sous-répertoires (en multiscreen) de res/drawable.
- Ajouter dans main.xml à la définition du layout android:background="@drawable/background"

Pour construire son image, soit on met une image 480x640, soit on utilise l'utilitaire draw9patch (dans tools du répertoire SDK). Cela supprime 1 pixel au bord et permet de définir les zones d'image qui vont s'étirer si on change d'écran (strech)..

- Placer l'image fondecran.png qui fait 480x640 dans les répertoires drawable, modifier la ligne du main.xml
  android :background="@drawable/fondecran"

Sources: Type d'écran: http://developer.android.com/guide/practices/screens_support.html
9-patch: http://android.cyrilmottier.com/?p=127

VI) Applications sans variable ni communication (pas de variable, pas de communication)

① Vibreur après appui de bouton

On va maintenant associer des traitements aux éléments graphiques.
On veut que l'appui du bouton fasse vibrer le téléphone et ouvre une boite de message avec un texte (Toast).

a) Associer une action au bouton

- Dans **main.xml**, définir comme id du bouton "monBouton".

- Ajouter **en début de programme JAVA**: import android.widget.Button;
  import.android.widget.Toast;
  import android.view.View;
  import android.view.View.OnClickListener;

- Ajouter à la **fin du programme JAVA** après setContentView(R.layout.main);
  avant le crochet de fermeture de onCreate.

```
final Button Instbouton = (Button) findViewById(R.id.monBouton);

Instbouton.setOnClickListener(new OnClickListener() {
   public void onClick(View v) {
        Toast.makeText(Nomprojet.this, "Vibration", Toast.LENGTH_SHORT).show();  // action à réaliser au clic
                /* Nomprojet est défini au début = nom de l'activité ( voir public class Nomprojet extends Activity  ) */
   }
});     /* attention: crochet puis parenthèse */
```

=> On a instancié un bouton = InstBouton associé au widget monBouton
=> On a défini un écouteur d'événements sur notre élément de Layout monBouton.
Instbouton.setOnClickListener
=> En cas de clic (Onclick), on appelle le traitement à réaliser, ici afficher un message de durée Toast.LENGTH.SHORT.

Remarque: Si on gère plusieurs clics de boutons, on peut utiliser la structure:

```
public void onClick(View v) {
        switch (v.getId()) {
        case R.id.identité_1:
            Action_1();
            break;
        case R.id. identité_2:
            Action_2();
            break;
        default:
            break;
        }
}
```

Source: Gestion clic bouton: http://developer.android.com/guide/tutorials/views/hello-formstuff.html
Message Toast: http://developer.android.com/guide/topics/ui/notifiers/toasts.html

b) Appeler une action

En plus d'ouvrir un message, on veut faire vibrer le téléphone.

- Autoriser la mise en vibration: Dans **AndroidManifest.xml,** ajouter la ligne

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

Tout écrire même s'il y a d'autres lignes <uses-permission ...>

- En début de programme JAVA: import android.os.Vibrator

- A l'intérieur de la classe principale (en parallèle de onCreate), définir une méthode dans le fichier JAVA qui va afficher le message, faire vibrer et modifier le texte de la boite texte du dessus.

```
public void Action(View v) {
      Toast.makeText(intro.this, "Vibration", Toast.LENGTH_SHORT).show();

      // Instanciation de Vibrator pour le context courant = intro = nom de la classe du "extends Activity"
      Vibrator vibreur = (Vibrator) getSystemService(intro.VIBRATOR_SERVICE);
      vibreur.vibrate(2500);      // Vibration de 2500 milliseconds
   } /* fin Action */
```

- Appeler cette méthode à la place du message dans le onCreate: Action(v);

NB: L'émulateur ne simule pas de vibration, il faut un téléphone réel

Récapitulatif:

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget33"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
android:background="#000044"    >

<TextView
android:id="@+id/widget2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/hello"
android:textSize="18sp"
android:layout_x="50px"
android:layout_y="22px"     />

<EditText
android:id="@+id/widget27"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="mon petit Texte"
android:editable="false"
android:textSize="18sp"
android:layout_x="50px"
android:layout_y="92px"        ></EditText>

<Button
android:id="@+id/monBouton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Vibreur"
android:layout_x="110px"
android:layout_y="182px"     ></Button>

</AbsoluteLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.exemple.UV_UTBM"
    android:versionCode="1"
    android:versionName="1.0">
  <uses-sdk android:minSdkVersion="4" />

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

  <application android:icon="@drawable/icon"
           android:label="@string/app_name"
           android:debuggable="true" >
    <activity android:name=".intro"
           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>
```

```
package com.exemple.UV_UTBM;

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

public class intro extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final Button InstBouton = (Button) findViewById(R.id.monBouton);
        /*avec override InstBouton.setOnClickListener(new View.OnClickListener() { */
        InstBouton.setOnClickListener(new OnClickListener() {
            public void onClick(View v){
                Action(v);
            }
        });
    }  /* fin de onCreate */

    public void Action(View v) {
        Toast.makeText(intro.this, "Vibration", Toast.LENGTH_SHORT).show();
          // Get instance of Vibrator from current Context
     Vibrator vibreur = (Vibrator) getSystemService(intro.VIBRATOR_SERVICE);
     // Vibrate for 300 milliseconds
     vibreur.vibrate(2500);
    }  /* Fin Action */

}  /* fin de intro extends Activity */
```

Si on veut afficher un message dans la case texte:

- Ajouter dans la méthode Action:

```
Au début, instancier l'objet:   EditText macasetexte = (EditText)findViewById(R.id.widget27);
                                                où widget27 est le nom du EditText dans main.xml
A la fin utiliser l'instance:   macasetexte.setText("c'est fini");
```

② Lecture de fichier son

Après les premières actions, on veut lire un fichier son.

- Créer un sous-répertoire raw dans res. Y placer le fichier son : fichier_son.mp3

- Ajouter dans le fichier JAVA au début :

```
import android.media.MediaPlayer ;
import android.os.handler ;
```

- Ajouter dans public void Action(View v) du fichier JAVA avant l'accolade de fin:

```
final MediaPlayer lecteur = MediaPlayer.create(intro.this, R.raw.fichier_son);
lecteur.start();
Handler h = new Handler(){
      @Override

final MediaPlayer lecteur = MediaPlayer.create(intro.this,R.raw.fichier_son);
lecteur.start();

Handler h = new Handler(){
    @Override
     public void handleMessage(Message msg){
           if (msg.what==0){
               lecteur.stop();
            }
           super.handleMessage(msg);
     }
};

Message m=new Message();
m.what=0;
h.sendMessageDelayed(m, 20000); /* envoi message sur handler h après 20000ms => lecteur.stop*/
```

On fait une instance de l'objet Mediaplayer dans le contexte (intro.this) et pour le fichier son.
Puis on lance la lecture avec la méthode start();
On ajoute un handler pour gérer la fon de lecture après 20 ms, sinon cela s'arrête à la fin.
NB : Avec ce code, si on appuie encore sur le bouton, cela relance Action qui crée une nouvelle lecture sur celle existante.

Source : http://bunjix.fr/tutoriel-faire-vibrer-le-telephone-et-lire-mp3-au-moment-du-reveil/301

VII) Applications sans communication

① Application avec gestion de variable: Calculatrice

On va saisir un nombre par le widget EditText, cliquer sur un bouton d'opération, saisir un second nombre, cliquer sur le bouton =, ce qui affichera le résultat du calcul de l'opération sur les 2 nombres saisis dans un autre EditText non éditable (pas de saisie possible, affichage seulement)

=> Définition des EditText en affectant un écouteur

- Définir la case de saisie final EditText Saisie = (EditText)findViewById(R.id.Saisie);
- Appeler la méthode setOnKeyListener pour l'objet Saisie

```
Saisie.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View view, int keyCode, KeyEvent keyEvent) {
      if ((keyEvent.getAction() == KeyEvent.ACTION_DOWN) && (keyCode == KeyEvent.KEYCODE_ENTER)) {
            ---
```

=> Association d'écouteur à un bouton et traitements

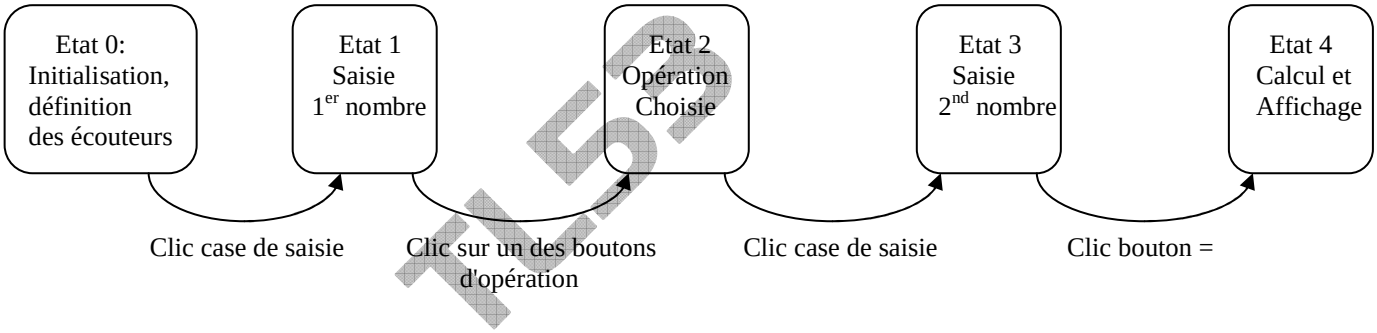
- Implémenter la méthode **OnClickListener** à la définition de l'activité
- Définition de l'écouteur dans onResume pour l'objet addi qui a été défini comme Button dans main.xml:

```
findViewById(R.id.addi).setOnClickListener(this);
```

- Traitement en cas de clic: Définir la méthode onClick(View vue) pour une vue.

```
switch(v.getId()){
    case R.id.addi:
      Traitement();
    break;
```

=> Gestion des états:




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

<RelativeLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_height="wrap_content">

<TextView
android:id="@+id/textesaisie"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Saisie valeurs"
></TextView>

<EditText
android:id="@+id/Saisie"
android:layout_width="200sp"
android:layout_height="wrap_content"
android:textSize="30sp"
android:layout_below="@id/textesaisie"
></EditText>
```

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

  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".calculatrice"
          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>
```

****

```
<Button
android:id="@+id/egal"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="="
android:layout_below="@id/Saisie"
android:layout_marginLeft="10sp"
></Button>

<Button
android:id="@+id/addi"
android:layout_width="50sp"
android:layout_height="50sp"
android:text="+"
android:layout_alignParentRight="true"
android:layout_marginRight="10sp"
></Button>

<Button
android:id="@+id/soust"
android:layout_width="50sp"
android:layout_height="50sp"
android:text="-"
android:layout_below="@id/addi"
android:layout_alignParentRight="true"
android:layout_marginRight="10sp"
></Button>

<Button
android:id="@+id/multi"
android:layout_width="50sp"
android:layout_height="50sp"
android:text="x"
android:layout_below="@id/soust"
android:layout_alignParentRight="true"
android:layout_marginRight="10sp"
></Button>

<Button
android:id="@+id/divi"
android:layout_width="50sp"
android:layout_height="50sp"
android:text="/"
android:layout_below="@id/multi"
android:layout_alignParentRight="true"
android:layout_marginRight="10sp"
></Button>

<TextView
android:id="@+id/texteresultat"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Résultat"
android:layout_below="@id/egal"
></TextView>

<EditText
android:id="@+id/resultat"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="30sp"
android:layout_below="@id/texteresultat"
android:layout_marginTop="20sp"
android:editable="false"
android:background="#FFFF0000"
android:layout_marginLeft="50sp"
></EditText>

</RelativeLayout>
```

```
package fr.UV_UTBM.calculatrice;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnKeyListener;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.util.Log;

public class calculatrice extends Activity implements OnClickListener {
  public int calcul=0;
  public String message;
  public String var1,var2,ChaineRes;
  private static final String TAG = "Calculatrice";
  public int Etat=0;   // 0 init
                       // 1 saisie var1
                       // 2 clic + - x ou /
                       // 3 saisie var 2
                       // 4 calcul

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  } // fin onCreate

  @Override
  public void onStop() {
    super.onStop();
  } // Fin onStop

  @Override
  public void onDestroy() {
    super.onDestroy();
  } // Fin onDestroy

  @Override
  public void onResume() {
    super.onResume();

    // listener +, -, x, /   => vide saisie, affecte variable calcul
    // +, -, x, / => calcul = 1, 2, 3 ou 4

    findViewById(R.id.addi).setOnClickListener(this);
    findViewById(R.id.soust).setOnClickListener(this);
    findViewById(R.id.multi).setOnClickListener(this);
    findViewById(R.id.divi).setOnClickListener(this);

    // Affection d'écouteur d'évènement au bouton
    findViewById(R.id.egal).setOnClickListener(this);

    if ( Etat == 0 ) {
      Log.e(TAG, "Etat = 0, lecture chaine var1");
      LectureChaine();
    }
  } // fin onResume

  //Méthode utilisée au clic sur un bouton
  public void onClick(View v) {
      if ( Etat == 1) {    // var1 saisie
       switch(v.getId()){

       case R.id.addi:
         calcul=1;
         break;
       case R.id.soust:
         calcul=2;
         break;
       case R.id.multi:
         calcul=3;
         break;
       case R.id.divi:
         calcul=4;
         break;
       }
```

```
        Etat=2;
        Log.e(TAG, "Etat = 2, lecture chaine var2");
        LectureChaine(); // lire var2
      } // fin if

        if (( Etat == 3) && (v.getId()==R.id.egal) ) {
           Log.e(TAG, "Etat = 3, lancement calcul "+calcul);
           ChaineRes=Operation(var1,var2,calcul);
           Etat=4;
           Log.e(TAG, "Etat = 4, après calcul affichage");
           final EditText CaseResult = (EditText)findViewById(R.id.resultat);
           CaseResult.setEnabled(true);
           CaseResult.setText(ChaineRes);
           Etat=0;
           }
  } // fin onClick

  public void LectureChaine() {
     // definition case saisie
      final EditText Saisie = (EditText)findViewById(R.id.Saisie);
      Saisie.setEnabled(true);

      // listener saisie: si premier => affecte à var chiffre1,  si pas premier => affecte à var chiffre 2
     // ecouteur case saisie et clavier
     Saisie.setOnKeyListener(new OnKeyListener() {
           public boolean onKey(View view, int keyCode, KeyEvent keyEvent) {
              if ((keyEvent.getAction() == KeyEvent.ACTION_DOWN) && (keyCode == KeyEvent.KEYCODE_ENTER)) {   // appui enter clavier
                  String message = Saisie.getText().toString();
                  if ( Etat == 0 ) {
                     var1=message;
                     message="";
                     Saisie.setText("");
                     Etat=1;
                     Log.e(TAG, "Etat = 1, sortie lecture chaine var1= "+var1);
                     return(true);
                  }
                  if ( Etat == 2 ) {
                     var2=message;
                     message="";
                     Saisie.setText("");
                     Etat=3;
                     Log.e(TAG, "Etat = 3, sortie lecture chaine var2="+var2);
                     }
                  return(true);
               }
               return(false);
        }
     });
  }// fin LectureChaine

  public String Operation(String var1, String var2, int calcul) {
     double var=0,val1=0,val2=0;

     val1=Double.valueOf(var1).doubleValue();
     val2=Double.valueOf(var2).doubleValue();
     if ( calcul == 1)
        var=val1+val2;
     if ( calcul == 2)
        var=val1-val2;
     if ( calcul == 3)
        var=val1*val2;
     if ( calcul == 4)
        var=val1/val2;
     // conversion var en string
     String res=Double.toString(var);
     return(res);

  } // fin Operation  si on met return dans if, on a une erreur, il faut un return bidon à la fin hors if

} // fin Activité Calculatrice
```

source: http://www.tutomobile.fr/faire-une-application-calculatrice-tuto-android-n%C2%B06/30/06/2010/
Conversion de température http://www.mathcs.org/java/android/temp_converter.html

② Localisation GPS

a) Classes utilisées en localisation

Le GPS utilise les classes

LocationManager	Accès aux services de localisation
LocationListener	Réception des notifications venant de LocationManager quand la position change
Location	Position géographique
Address	Adresse = ensemble de chaines décrivant la position

A cela on peut ajouter la classe "Geocoder" qui donne accès au service de géolocalisation de Google.

b) Programme

- Créer un nouveau projet: file / new / other / android / android project, next
  Mettre le nom monGPS si on veut utiliser le code JAVA suivant ou modifier le code pour mettre votre nom.

- Modifier **AndroidManifest.xml** pour avoir les permissions d'utilisation du GPS par puce GPS.

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

Pour le GPS par réseau téléphonique, il faut aussi ajouter:

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

- Créer l'interface utilisateur dans **main.xml** (voir tableau suivant).

- Créer le programme JAVA (voir tableau suivant).

c) Test du programme avec émulateur

Pour modifier la position GPS de l'émulateur, on peut utiliser 2 méthodes::

Par telnet	Par DDMS
* Se connecter en telnet à l'émulateur lancé: Ouvrir un terminal et taper Telnet localhost 5554 * help ou help gps * Pour définir une position gps geo fix 47.494813 6.803015 (syntaxe: gps fix longitude latitude altitude, l'altitude est optionnelle)	- Ouvrir la perspective DDMS dans eclipse (bouton en haut à droite ou window/perspective/Other.../DDMS) - Partie "Emulator Control" sur la gauche: Descendre à "Location Control" - Entrer Longitude et Latitude 47.494813 6.803015 - Cliquer sur Send.

Remarque: En telnet, on a aussi la simulation de messages NMEA: Pour définir une trame NMEA
* geo nmea
$GPGGA,001431.092,0118.2653,N,10351.1359,E,0,00,,-19.6,M,4.1,M,,0000*5B

ATTENTION: Cela marche mal sur un PC français car l'altitude n'est pas traitée et le point est traduit en virgule.
=> Incompréhension des valeurs et affichage "bancal".
=> Utiliser l'outil suivant DDMS.

Remarque DDMS: DDMS (Dalvik Debug Monitor Server) est dans le répertoire Tools du SDK.
On a ajouté à eclipse un onglet via: window/perspective/Other.../DDMS
http://developer.android.com/guide/developing/debugging/ddms.html

source: http://www.tutomobile.fr/geolocalisation-grace-au-gps-ou-au-reseau-mobile-tutoriel-android-n%C2%B015/13/08/2010/
http://www.developpez.net/forums/d844470/java/general-java/java-mobiles/android/android-simuler-gps/
http://blog.developpez.com/android23/p8494/android/simuler-le-gps-avec-le-sdk/#more8494
http://www.insideandroid.fr/post/2009/03/28/Simulation-du-GPS-avec-l-emulateur-android
http://www.devx.com/wireless/Article/39239/1954

Organisation du programme

onCreate
- affichage du main.xml
- Création d'une instance de LocationManager
- Création de l'écouteur de clic de bouton

onClick
- Mettre à jour la position GPS, la stocker

onLocationChanged
- Appeler la méthode "AfficherLocalisation"

AfficherLocalisation
- afficher la latitude, longitude, altitude

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/RelativeLayout01"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"

xmlns:android="http://schemas.android.com/apk/res/android">

   <TextView android:text="Latitude"
      android:id="@+id/TextView01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:layout_marginLeft="10dip" />
    <EditText android:text="0.0"
      android:id="@+id/latitude"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:editable="false"
      android:focusable="false"
      android:gravity="center"
      android:layout_marginRight="10dip"
      android:layout_width="150dip" />
    <TextView android:text="Longitude"
      android:id="@+id/TextView02"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/longitude"
      android:layout_alignTop="@+id/longitude"
      android:gravity="center"
      android:layout_marginLeft="10dip" />
    <EditText android:text="0.0"
      android:id="@+id/longitude"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:editable="false"
      android:focusable="false"
      android:gravity="center"
      android:layout_marginRight="10dip"
      android:layout_width="150dip"
      android:layout_below="@+id/latitude" />
    <TextView android:text="Altitude"
      android:id="@+id/TextView03"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBottom="@+id/altitude"
      android:layout_alignTop="@+id/altitude"
      android:gravity="center"
      android:layout_marginLeft="10dip" />

   <EditText android:text="altitude"
      android:id="@+id/altitude"
      android:layout_height="wrap_content"
      android:layout_alignParentRight="true"
      android:editable="false"
      android:focusable="false"
      android:gravity="center"
      android:layout_marginRight="10dip"
      android:layout_below="@+id/TextView02"
      android:layout_width="150dip" />
    <LinearLayout android:id="@+id/LinearLayout01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/TextView03"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="10dip">
       <Button android:layout_height="wrap_content"
         android:id="@+id/lecture_position"
         android:text="Interrogation GPS"
         android:layout_width="150dip" />
    </LinearLayout>
    <TextView android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/adresse"
      android:layout_below="@+id/LinearLayout01"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="10dip" />
 </RelativeLayout>
```

programme JAVA

```
package fr.monGPS.UV_UTBM;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.Toast;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;

public class monGPS extends Activity implements OnClickListener , LocationListener {
   private LocationManager lmg;
   private Location location;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      //Utilisation du service de localisation
      lmg = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

      //Affection d'écouteur d'évènement au bouton
      findViewById(R.id.lecture_position).setOnClickListener(this);
   }  // Fin onCreate


   //Méthode utilisée au clic sur un bouton
   public void onClick(View v) {
      // le service de localisation notifie tout changement de position toutes les 30s = 30000 ms.
      // paramètre this = notre classe implémente LocationListener et recevra les notifications.
      lmg.requestLocationUpdates(LocationManager.GPS_PROVIDER, 30000, 0, this);
      }   // Fin onClick


   public void onLocationChanged(Location location) {
      this.location = location;     // on sauvegarde la position
      AffichePosition ();          // on l'affiche
      lmg.removeUpdates(this); // indique au service que l'on ne veut plus de mise à jour
                              // seul le clic permet de relancer la lecture
   }


   public void onProviderDisabled(String provider) { //source est désactivée
      Toast.makeText(monGPS.this,"source désactivée",Toast.LENGTH_SHORT).show();
       // => affiche un Toast
      lmg.removeUpdates(this); // on spécifie au service que l'on ne veut plus de mise à jour
   }

   public void onProviderEnabled(String provider) {
      //  La source a été activée, vide par simplification
   }
   public void onStatusChanged(String provider, int status, Bundle extras) {
      //  Le statut de la source a changé, vide par simplification
   }

   private void AffichePosition() {
      // affichage des informations de position à l'écran
      ((TextView)findViewById(R.id.latitude)).setText(String.valueOf(location.getLatitude()));
      ((TextView)findViewById(R.id.longitude)).setText(String.valueOf(location.getLongitude()));
      ((TextView)findViewById(R.id.altitude)).setText(String.valueOf(location.getAltitude()));
   }  // Affichage après lecture position

}
```

=> onCreate fait l'affichage de main.xml, crée un service LocationManager, crée un écouteur de bouton.

=> onClick active la lecture du GPS pour notre bouton

=> Si il y a un changement de position = onLocationChanged, on lance l'affichage = AffichePosition

On a aussi d'autres fonctions liées au GPS: onProviderDisabled, onProviderEnabled, onStatusChanged.

VIII) Applications avec communication

① Bluetooth: Compléments de cours

a) Eléments de bluetooth utilisés pour la configuration

Le composant Bluetooth est défini par son adresse = 48 bits = 12 codes hexa, unique et statique
un nom (display name): configurable
les protocoles implémentées qu'ils sait gérer = Profils.

Description d'une connexion:

→ Le client est l'appareil à l'initiative de la connexion.
Il envoie une requête de découverte en **broadcast = Inquiry scan**
Cela peut être long (15s) car il doit changer de fréquences d'émission souvent.

→ Les appareils qui écoutent répondent s'ils y sont autorisés.
La réponse est l'adresse et en entier identifiant le type d'appareil (classe: écouteur, téléphone, modem,...)

On distingue "Inquiry scan" = appareil visible = découvrable
"Page scan" = appareil qui peut répondre

Inquiry scan = visible = détectable	Page scan = serveur	
on	on	détectable et accepte les connexions entrantes
off	on	pas détectable mais accepte les connexions entrantes (d'appareils qui connaissent son adresse)
on	off	détectable mais refuse les connexions entrantes => Peu utile
off	off	pas détectable et refuse les connexions entrante => seulement connexion sortante

→ Si on utilise plusieurs applications simultanément sur une même machine, il faut les distinguer = numéro de port.
En Bluetooth, le port est dynamique (contrairement à TCP/IP où un serveur utilise toujours le même numéro de port pour un service 80 = HTTP, 21 = FTP,...).
Pour indiquer à quel service on veut se connecter, il faut avoir son numéro de port => Protocole SDP Service Discovery Protocol. => Le client doit le demander ce numéro à un serveur SDP.

=> Le client choisit sa cible (Target) et lui envoie une requête SDP (à lui seul) = Demande de connexion en indiquant le service demandé. Le serveur SDP lui indique le port à utiliser ensuite.

Un serveur Bluetooth qui démarre s'enregistre auprès de son serveur SDP en **décrivant son service = Service Record**.
Le serveur SDP possède ainsi une liste des services disponibles sur son appareil.
Pour RFCOMM, on parle de canal plutôt que de port.

Sous linux, on peut utiliser **sdptool** : "sdptool scan" ou "sdptool search SP" pour le seul profil série.
Un téléphone Android renvoie

```
~$ sdptool search SP
Inquiring ...
Searching for SP on 8C:71:F8:E5:XX:XX . . .
Service Name: 1808130054
Service RecHandle: 0x10003
Service Class ID List:
UUID 128: 00001101-0000-1000-8000-00805f9b34fb
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 13
```

Le "Service Record" qui décrit le service est fait de
- n° de port
- Service ID
- ServiceClassIDList (même application codée différemment pileBT windows, toshiba,...)
- Service Name (nom ou n° défini par constructeur)
- ...

Le service ID est décrit par un numéro standardisé par l'IETF: UUID = Universally Unique Identifier

Il est fait de 128 bits = 32 codes hexa
8 codes – 4 codes – 4 codes – 4 codes – 12 codes

Il existe des UUID réservés = UUID courts sur 16 ou 32 bits. On remplace les 32 bits à gauche, pour le reste on met toujours:
00000000-0000-1000-8000-00805F9B34FB

Exemple d'UUID réservés: Protocole RFCOMM = 0x0003
Protocole SDP = 0001
Profil Port série SPP = 0x1101 => 0000**1101**-0000-1000-8000-00805F9B34FB

→ Ayant adresse et numéro de port, le client peut envoyer une requête de connexion au serveur
Adresse + n° de port d'une com = numéro de **SOCKET** (j'envoie un message à "ADR" pour lui parler de "APPLI=n° de port")

Création du socket des 2 côtés (commande create) = **Binding** = associer le socket à son matériel et son n° de port puis le serveur reçoit la commande listen (mise en écoute de requêtes pour son socket = adr et port)

Le client demande la connexion (commande connect), le serveur accepte (commande accept)

→ Connecté (commandes send, receive) (je dis que "message")

→ Echange de données

Remarque: Sécurité.
BT peut utiliser authentification (code PIN) avant communication, cryptage pendant communication.
La première fois qu'il y a authentification = Procédure de Pairing ou Bonding.
Pour les appareils sans interface d'entrée, il y a un code PIN défini par défaut.
En BT2.1, le code PIN est généré automatiquement, l'utilisateur doit juste accepter ou refuser la connexion.

b) Bluetooth et Android

- Bluetooth n'est utilisable avec Android que depuis la version Android 2.1

- Les classes à utiliser en Bluetooth sont :

BluetoothAdapter	Représente l'adaptateur local utilisé.
BluetoothDevice	Un appareil distant qui sait faire des échanges Bluetooth
BluetoothSocket	Cela permet un appel à createRfcommSocketToServiceRecord Qui va créer un socket BT sur l'appareil distant permettant de lui faire une requête de connexion et d'initialiser la communication.
BluetoothServerSocket	Création d'un socket BT serveur local = en écoute de requêtes. On utilisera la méthode listenUsingRfcommWithServiceRecord pour se mettre en écoute de connexion entrante.

Exemple: Pour définir son adaptateur Bluetooth local: monAdaptateur = BluetoothAdapter.getDefaultAdapter();
Pour définir l'adaptateur distant: BluetoothDevice device = monAdaptateur .getRemoteDevice("ab:cd:ef:gh:ij:kl");
Pour créer un socket en client tant que client vers le serveur "device" avec le service demandé mon_UUID:
btSocket = device.createRfcommSocketToServiceRecord(mon_UUID);
Pour créer un socket en serveur avec le nom name:
btSocket = monAdaptateur.listenUsingRfcommWithServiceRecord(name,mon_UUID)
Attention: Depuis la version 2.2 il y a une version secure, une autre insecure.

Source: http://www.brighthub.com/mobile/google-android/articles/103281.aspx
http://developer.android.com/guide/topics/wireless/bluetooth.html

② Emission SPP vers un serveur BT (adresse non modifiable, appareil appairé avant par une autre application)

- Créer un projet sous Android clientBTsimplex

- Ajouter au fichier xml manifeste la permission:

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

- Faire le code JAVA de l'application (**en adaptant l'adresse à joindre** dans le programme) :

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget0"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<EditText
android:id="@+id/monTexte"
android:layout_width="300px"
android:layout_height="wrap_content"
android:text=""
android:textSize="18sp"
android:layout_x="100px"
android:layout_y="42px"
>
</EditText>
</AbsoluteLayout>
```

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

  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN">
                                                                  </uses-permission>
  <uses-permission android:name="android.permission.BLUETOOTH">
                                                                  </uses-permission>

  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".clientBTsimplex"
              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>
```

```
package fr.UV_UTBM.clientBTsimplex;

import java.io.IOException;
import java.io.OutputStream;
import java.util.UUID;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnKeyListener;
import android.widget.EditText;
import android.widget.Toast;


public class clientBTsimplex extends Activity {

  private BluetoothAdapter mBluetoothAdapter = null;
  private BluetoothSocket btSocket = null;
  private OutputStream outStream = null; ////

  // UUID de SPP, probablement canal 1 RFCOMM channel 1 s'il est dispo
  private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
  private static String adresse = "00:22:58:C7:1E:50"; //<== adresse serveur à joindre

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

     mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();       // recherche de l'adaptateur local

    if (mBluetoothAdapter == null) {
      Toast.makeText(this, "Pas de Bluetooth sur cet appareil", Toast.LENGTH_LONG).show();
      finish();
      return;
    }

    if (!mBluetoothAdapter.isEnabled()) {
        Toast.makeText(this, "Activer le BT du phone et recommencer", Toast.LENGTH_LONG).show();
        finish();
        return;
    }
```

```
} // Fin onCreate


@Override
public void onStart() {
   super.onStart();
} // Fin onStart

@Override
public void onResume() {
   super.onResume();

   // on définit l'appareil bluetooth distant grâce à son adresse ( pas de recherche ).
   BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(adresse);

   // Création de socket, le protocole est chosii avec la valeur d'UUID.
   // En général RFCOMM 1, mais pas forcément 1.
   try {
      btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
   } catch (IOException e) {
      // message étape RESUME: Echec création socket.
   }

   //Discovery peut être en cours sur l'android. Pour être sur que non on l'annule
   mBluetoothAdapter.cancelDiscovery();

   try {
      btSocket.connect();
      //message étape RESUME: Connexion établie, liaison ouverte
      Toast.makeText(this, "Connexion établie", Toast.LENGTH_LONG).show();

   } catch (IOException e) {
      try {
         btSocket.close();
      } catch (IOException e2) {
         // message étape RESUME: Impossible de fermer le socket après échec de conenxion
      }
   }

   // definition case saisie
   final EditText saisieTexte = (EditText)findViewById(R.id.monTexte);
   saisieTexte.setEnabled(true);

   // ecouteur case saisie et clavier
   saisieTexte.setOnKeyListener(new OnKeyListener() {
      public boolean onKey(View view, int keyCode, KeyEvent keyEvent) {
         if ((keyEvent.getAction() == KeyEvent.ACTION_DOWN) &&
               (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)) {    // appui bouton du PAD au centre
          String message = saisieTexte.getText().toString();
          EnvoiMessage(btSocket, message);
          saisieTexte.setText("");
        return(true);
      }
      return(false);
    }
   });
} // fin onResume


public void EnvoiMessage(BluetoothSocket btSocket, String message) {
   // vérification saisie
   //Toast.makeText(this,message, Toast.LENGTH_LONG).show();

   // Définition de la trame à envoyer pour notre socket
   try {
      outStream = btSocket.getOutputStream();
   } catch (IOException e) {
      // message étape RESUME: Echec de création de la trame à envoyer
   }
   ////String message = "Message du client au serveur";
   byte[] msgBuffer = message.getBytes();
   try {
```

```
            outStream.write(msgBuffer);
            //Toast.makeText(this, "message envoyé", Toast.LENGTH_LONG).show();

        } catch (IOException e) {
            // message étape RESUME: Exception pendant écriture
        }
    } // fin EnvoiMessage


    @Override
    public void onPause() {
        super.onPause();
        if (outStream != null) {
            try {
                outStream.flush();
            } catch (IOException e) {
                // message PAUSE : Impossible de vider le flux de sortie
            }
        }
        try {
            btSocket.close();
        } catch (IOException e2) {
            // message étape PAUSE: Impossible de fermer le socket
        }
    } // Fin onPause


    @Override
    public void onStop() {
        super.onStop();
    } // Fin onStop


    @Override
    public void onDestroy() {
        super.onDestroy();
    } // Fin onDestroy

} // Fin activité
```

Source: http://www.anddev.org/viewtopic.php?p=35487#35487
ou http://www.mail-archive.com/android-beginners@googlegroups.com/msg20753.html

Remarque: Utilisation de la saisie au clavier avec validation du bouton PAD au centre
On utilise un EditText. On lui associe un écouteur clavier onKeyListener
A chaque événement (keyEvent.getAction) on analyse l'événement.
Quand c'est l'appui du bouton PAD au centre, on lit le texte avec getText.
Source: http://developer.android.com/reference/android/view/KeyEvent.html

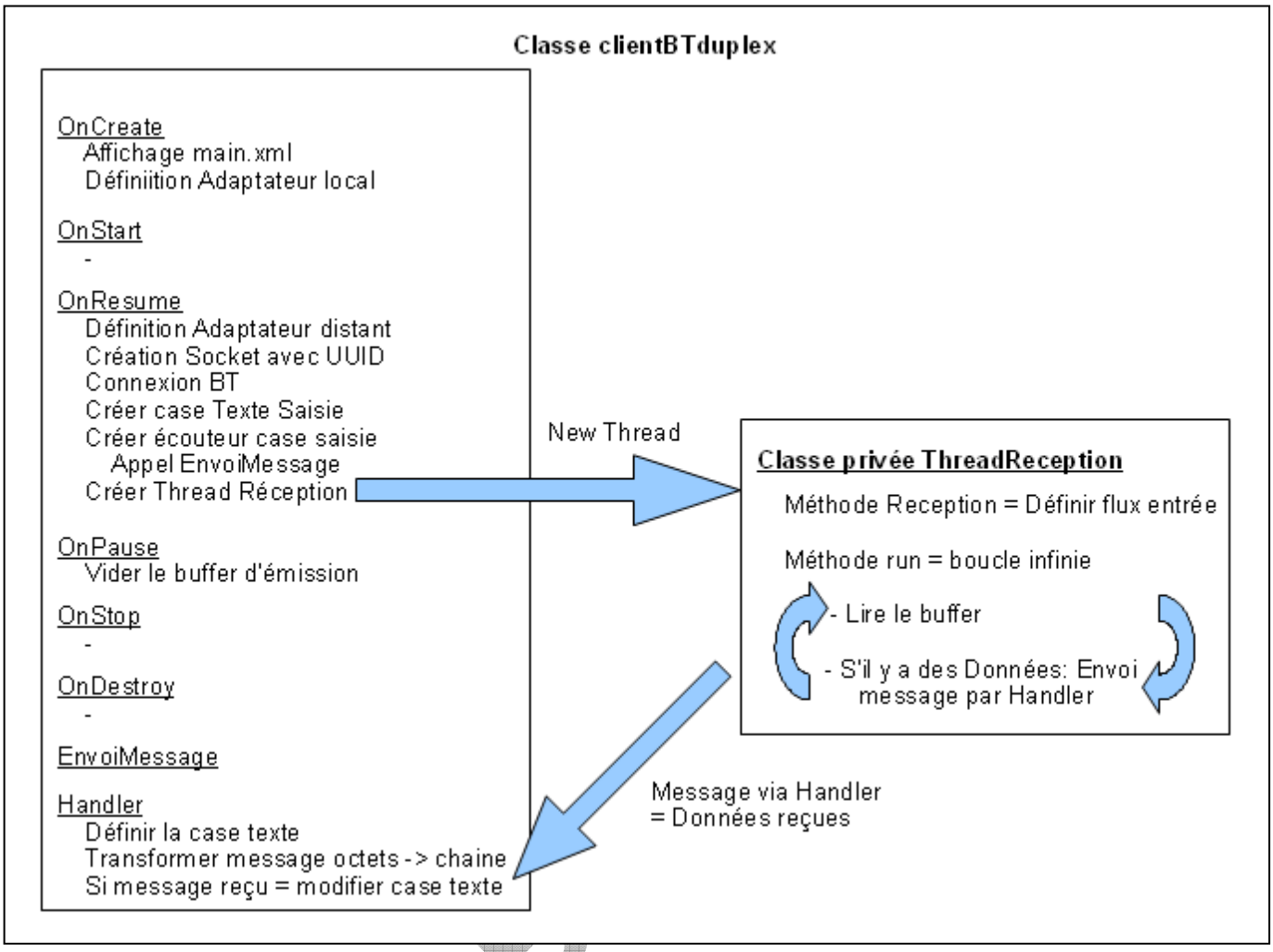
Test: - Relever l'adresse de l'adaptateur Bluetooth sur le PC, la mettre dans le programme.
- Ouvrir Hyperterminal en mettant le port com Bluetooth SERVEUR ! (Win7 = Panneau Config / Matériel / Port Com Local Bluetooth)
- Lancer la connexion Hyperterminal si ce n'est fait (icône téléphone raccroché = en com).
- Envoyer le message depuis l'appareil Android (on aura connexion BT sur le PC et réception du message)

③ Emission/réception SPP vers un serveur BT (adresse non modifiable, appareil appairé avant)

Ne disposant pas d'interruption matérielle, on doit utiliser la lecture par scrutation = boucle d'attente. Mais on ne peut pas bloquer l'UI Thread plus de 5 s.
=> On va créer un autre Thread et laisser le système Android partager les accès. L'UI Thread relancera l'exécution quand il le pourra.

Pour envoyer des données reçues d'une Thread à l'autre, on utilise un Handler qui permet l'envoi de messages..

a) Schéma de principe:



b) Ajouts dans le code de clientBTsimplex

Dans main.xml

Ajouter un EditText (case de texte) de nom AfficheTexte
qui n'est qu'en lecture (mettre la ligne android:editable="false")

Dans le code JAVA

Dans les "import":

```
import android.os.Handler;
import android.os.Message;
import java.io.InputStream;
```

Dans la définition de la classe:

```
private ThreadReception monThreadReception;
boolean flag;
```

Dans le onResume:

```
// definition case saisie et écouteur comme en clientBTsimplex

        // creation Thread reception
        monThreadReception = new ThreadReception(btSocket);
    monThreadReception.start();
```

Après les onCreate, onStart,... onDestroy, dans la classe on définit le handler:

```
private Handler handler = new Handler() {
        public void handleMessage(Message msg) {
      if ( msg.what == 0) {
          byte[] readBuf = (byte[]) msg.obj;
```

```
            // faire un string depuis les octets du buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            // définir la variable de case de saisie id=AfficheTexte
            EditText AfficheTexte = (EditText)findViewById(R.id.AfficheTexte);

              AfficheTexte.setText(readMessage+AfficheTexte.getText());
           }
       };
   }; // fin Handler
```

Puis définition du Thread de réception

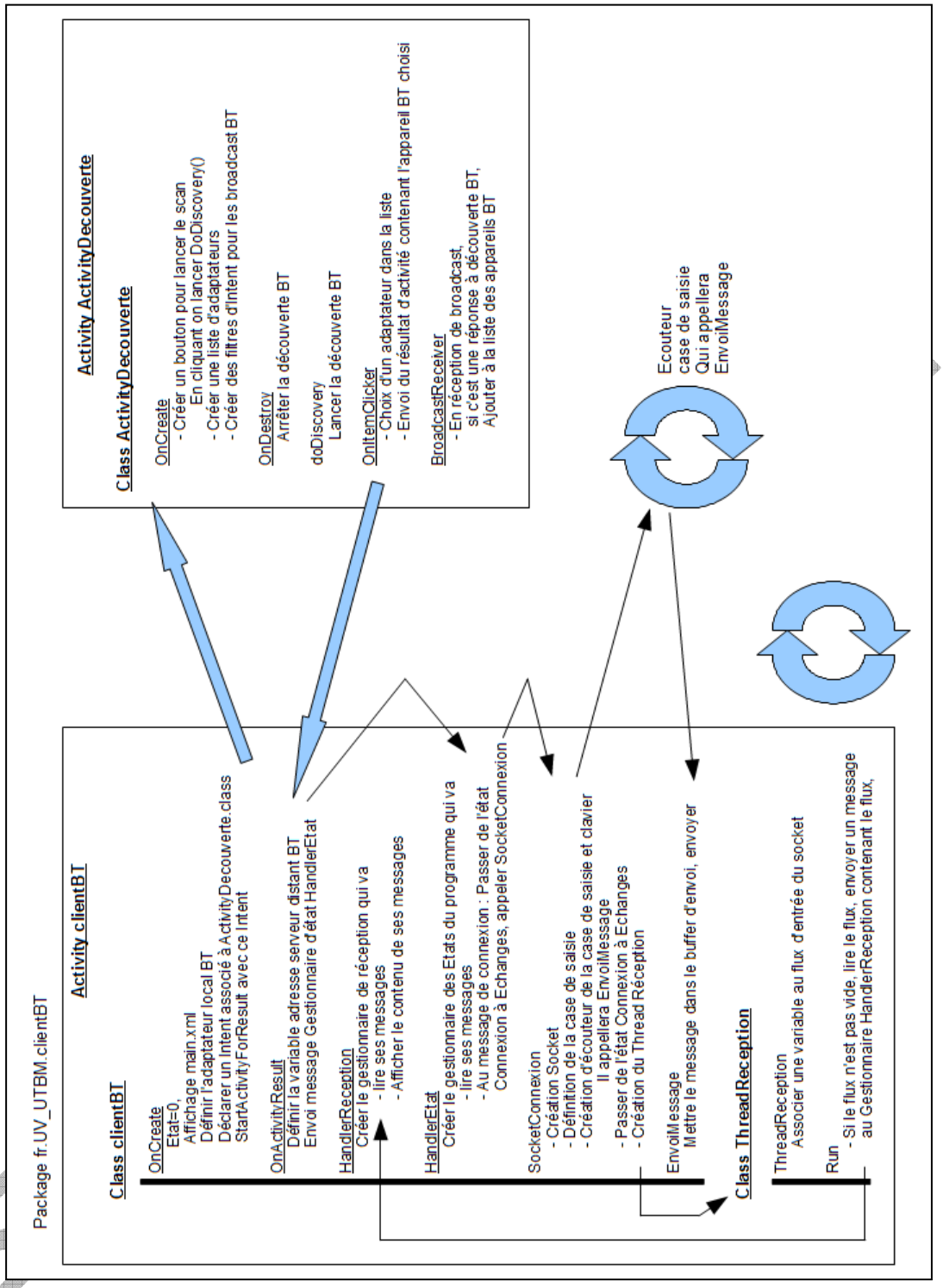
```
   private class ThreadReception extends Thread {
      private final BluetoothSocket SocketLocal;
      private final InputStream mon_inpstream;

      public ThreadReception(BluetoothSocket socket) {
         SocketLocal = socket;
         InputStream tmpIn = null;

         // Lecture flux d'entrée
         try {
            tmpIn = socket.getInputStream();
         } catch (IOException e) {
            //Log.e(TAG, "socket non créé", e);
         }
         mon_inpstream = tmpIn;
      }

      public void run() {
         byte[] buffer = new byte[1024];
         int bytes;

         // Ecoute en permanence
         while (true) {
            try {
               // Lecture des octets
               bytes = mon_inpstream.read(buffer);

               // Envoi des octets au Thread de l'UI
               handler.obtainMessage(0, bytes, -1, buffer).sendToTarget();
            } catch (IOException e) {
               // perte de connexionconnectionLost();
               break;
            }
         }
      }
   }
} // Fin class activité
```

④ Evolutions

a) Ajout de découverte, appairage et choix du serveur

On utilise une partie des sources de Google (sample de Bluetooth Chat) => On utilise la classe ActivityDecouverte

Schéma de principe



b) Traitement des données échangées

On est capable d'échanger des 0 et 1 via Bluetooth. Dans une application, il y aura un protocole qui donnera une signification aux échanges. Il faudra ajouter:

- Stockage dans un buffer circulaire des données reçues au lieu de les afficher (avec les index du buffer)
- Création de trame lors de l'envoi de données (byte stuffing, contrôle d'erreur même si Bluetooth fait le sien)
- Remplacement de la fonction d'affichage des données reçues par une fonction de traitements (nouveau Thread pour traitement asynchrone).

Autres sources:
- Adaptation de BluetoothChat pour oscilloscope sur téléphone
  http://projectproto.blogspot.com/2010/09/android-bluetooth-oscilloscope.html
- Adaptation de BluetoothChat pour dialoguer avec un robot
  http://www.pocketmagic.net/wp-content/uploads/2010/11/android-control-robot-via-bluetooth1.zip
- Code en norvégien pour échange par port série (découverte)
  http://stackoverflow.com/questions/3547658/connecting-to-paired-bluetooth-device-bluetoothsocket-in-android-2-1

② Wifi

a) Gestion du WIFI

On utilise une instance de WifiManager.

Permissions	android:name="android.permission.ACCESS_WIFI_STATE"
Instancier WifiManager	WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
Etat du wifi	wifiManager.getWifiState()
Connaitre son réseau	WifiInfo wifiInfo = wifiManager.getConnectionInfo(); wifiInfo.getSSID() pour avoir le SSID wifiInfo.getLinkSpeed() pour avoir la vitesse Log.d("wifiInfo",wifiInfo.toString());
Connaitre le niveau de réception	wifiManager.calculateSignalLevel

Découvrir les réseaux	1) Déclarer un filtre d'Intent qui sera associé à un BroadcastReceiver
	IntentFilter intentfil = new IntentFilter();
	intentfil.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
	
	2) Enregistrement d'un BroadcastReceiver qui préviendra de l'arrivée des informations.
	registerReceiver(
	new BroadcastReceiver() {
	@Override
	public void onReceive(Context context, Intent intent) {
	WifiManager wifimanager = (WifiManager)
	getSystemService(Context.WIFI_SERVICE);
	List<ScanResult> hotspot = wifiManager.getScanResults();
	System.out.println("Nombre de points d'accès: " + hotspot.size());
	}
	}
	, intentfil);

b) Etablissement de socket

Soit on utilise du http avec des navigateurs web, soit on fabrique soit même son protocole (synchro, byte stuffing, correction d'erreur,...).

Pour établir la connexion logicielle, il faut établir un socket. Comme en Bluetooth, on a des sockets serveur et client.

Pour se connecter à un serveur, on fera un socket client:

Permission	android.permission.INTERNET

Déclaration	Private String Requete(String Adr_serveur, int Port, String Msg) throws IOException {
	SocketClient = new Socket(Adr_serveur, Port); // requête création socket avec Adr_serveur
	BufferedWriter MemEmission = new BufferedWriter(new
	OutputStreamWriter(SocketClient.getOutputStream()));
	BufferedReader MemReception = new BufferedReader(new
	InputStreamWriter(SocketClient.getInputStream()));
	MemEmission.write(Msg+"\n");
	MemEmission.flush();
	return MemReception.readLine();
	}

Appel par	String Reponse = Requete("192.168.1.1",5060,chaine);

Source: http://developer.android.com/reference/android/net/wifi/package-summary.html

ANNEXES

Installer les logiciels sous windows

JAVA

- installer java JDK = JSE Dk (JRE insuffisant)

SDK

- charger sdk android http://developer.android.com/sdk/index.html
- installer dans \android\android_sdk

- ajouter le chemin du SDK Android à la variable Path du système.

Ceci permettra d'utiliser les outils du SDK sans devoir spécifier le chemin complet à chaque fois.

•Windows 7 et Windows Vista :

Clic droit sur Ordinateur > Propriétés > Protection du système (à gauche),
ensuite cliquez sur l'onglet Paramètres système avancés puis sur le bouton Variable d'environnement.
Sélectionnez dans les Variables système, la variable Path puis cliquez sur Modifier.
Il suffit de rajouter à la fin de la ligne un point virgule (si ce n'est pas déjà fait) et d'ajouter le chemin du SDK décompressé
(par défaut: C:\Program Files\Android\android-sdk-windows).

•Windows XP :

Double clic sur Poste de travail > Afficher informations système (à gauche),
ensuite cliquez sur l'onglet Avancé puis sur le bouton Variable d'environnement .
Sélectionnez dans les Variables système, la variable Path puis cliquez sur Modifier.
Il suffit de rajouter à la fin de la ligne un point virgule (si ce n'est pas déjà fait) et d'ajouter le chemin du SDK décompressé
(pour moi il s'agit de C:\tuto_mobile\android-sdk-windows).

Eclipse

- charger eclipse = environnement de développement java = IDE Java open-source
  Eclipse IDE for Java Developers. à http://www.eclipse.org/downloads/
- Aucune installation n'est à réaliser: * décompresser le fichier téléchargé
  * faire un double clic sur le fichier eclipse.exe pour lancer le logiciel.

Plugin Android pour Eclipse

- Installation du plugin Android pour Eclipse
  * Lancez Eclipse,
  * menu Help > Install New Software,
  * cliquez sur le bouton Add et remplissez le formulaire.
    Dans le champs Name écrivez par exemple Plugin Android
    Dans le champ Location indiquez l'adresse suivante : https://dl-ssl.google.com/android/eclipse/ puis cliquez sur OK.

Il se peut que cette adresse ne fonctionne pas parfois => pour résoudre ce problème il m'a suffit d'enlever le « s » du « https ».

  * Pending devient Developper tools
  * Cochez Developer Tools (!!! Installation de DDMS, ADB,...) puis cliquez sur Next et encore une fois sur Next.
  * Cochez « I accept the terms of the license agreements » puis cliquez sur Finish.
  * Un message d'avertissement devrait apparaître vous signalant que le logiciel que vous essayez d'installer n'est pas signé, cliquez sur OK puis cochez le certificat Eclipse.org Foundation puis cliquez sur OK.
  * Une fois l'installation finie redémarrez Eclipse: OK à "restart now".
  * Une fois Eclipse relancé, allez dans le menu Window > Préférences > Android puis SDK Location
    Browse pour aller chercher le dossier dans lequel se trouve le SDK que vous avez décompressé tout à l'heure
    (pour moi il s'agit de C:\Program Files\Android\android-sdk-windows) puis cliquez sur Apply, OK.

- redémarrer le pc.
- relancer eclipse puis Configuration du plugin Android dans eclipse
  Cliquez sur la petite icône android (robot vert). refaire si besoin menu Window > Préférences > Android puis SDK Location
- Available packages, Cochez Accept All puis cliquez sur Install.
  Cliquez sur Installed Packages puis sélectionnez le package Android SDK Tools et cliquez sur Update All, accept all, install.

Source: http://developer.android.com/sdk/installing.html

Installer les logiciels sous Linux

Installer Ant, Java, eclipse

- sudo apt-get install ant

- pour java6 sous ubuntu 10 Maverick
  (version précédente d'Ubuntu: apt-get install sun-java6-bin)

  sudo add-apt-repository ppa:sun-java-community-team/sun-java6
  sudo apt-get update
  sudo apt-get install sun-java6-bin
  (utiliser tab et entrée pour les validations de license)

- sudo apt-get install eclipse

Remarque: Pour Ubuntu 64 bits, il faut aussi installer apt://ia32-libs

Installer le sdk android

- Charger la source à http://developer.android.com/sdk/index.html

Copier le fichier de /Home/rt/Téléchargements dans /Home/RT

Ouvrir un terminal en root: sudo su
cd /Home/RT
tar -zxvf android-sdk_r*-linux_x86.tgz
supprimer le tgz

A ce stade, on n'a pas de developpement-tools. Pour installer Platform-tools, Android platforms et les autres add-ons,
il FAUT UNE CONNEXION INTERNET:
cd android-sdk-linux_x86/tools/ pour aller dans tools,
Exécuter le programme android (en su !!): android update sdk

Toujours dans tools:
echo "export PATH=$(pwd):\${PATH}" >> ~/.bashrc

cd ../platform-tools/ (après install via tools/android)
echo "export PATH=$(pwd):\${PATH}" >> ~/.bashrc
. ~/.bashrc

Il vous faudra alors mettre à jour la liste des paquets et choisir ceux que vous voulez installer.
Les paquets appelés "Target" correspondent à la version d'Android cible (1.5, 1.6, 2.0…) sous laquelle vous souhaitez développer.
./android list targets donnera la liste des cibles installées.

Installer plugin ADT dans eclipse (Android Developpement Tools)

- Lancer eclipse: menu Ubuntu Applications / Programmation / Eclipse

J'ai laissé le répertoire de travail /home/uv/workspace

- Help → Install New Software, Cliquer sur « Add » et ajouter les « Software sites » suivants selon le schéma
[Name] – [Location]

android - https://dl-ssl.google.com/android/eclipse/

Il se peut que cette adresse ne fonctionne pas parfois => pour résoudre ce problème il m'a suffit d'enlever le « s » du « https ».

recommencer add
GEF - http://download.eclipse.org/tools/gef/updates/releases/
cocher la dernière version sinon conflit

EMF - http://download.eclipse.org/modeling/emf/updates/releases/
GMF - http://download.eclipse.org/modeling/gmf/updates/releases
Webtools - http://download.eclipse.org/webtools/updates/
Google eclipse Plugin - http://dl.google.com/eclipse/plugin/3.5

- installer la totalité des champs qui se trouvent dans le nom de schéma "android" (ligne android ci-dessus)

- **cocher devant "Developper Tools"** (installation de DDMS, adb,...), next, cocher "I accept", finish

Une fois Eclipse relancé, allez dans le menu Window > Préférences > Android puis SDK Location
Si c'est vide, cliquer Browse pour aller chercher le dossier dans lequel se trouve le SDK que vous avez décompressé tout à l'heure (pour moi il s'agit de /Home/ RT/android-sdk-linux_x86 puis cliquez sur Apply, OK.
On y voit aussi les versions android cibles (targets).

Source: http://developer.android.com/sdk/installing.html