**Université Paris Diderot** **Programmation des composants mobiles**

**M1** **Année 2018-2019**

# TP n° 1

# Prise en main, Première application

## IMPORTANT

Inscrivez-vous sur le Moodle `Programmation des composants mobiles (Android) 2018-2019`.

Ce TP s'appuie principalement sur le cours 1. Pour le premier TP, nous vous guiderons beaucoup. Ce sera moins le cas par la suite.

## Ressources:

`https://developer.android.com/studio/` : vous y trouverez `Android studio` et le `sdk` si vous voulez l'utiliser chez vous. Vous y trouverez aussi des explications et des références dans la section `https://developer.android.com/docs/` .

# 1 Démarrage

**Important: Les sections 1.1 à 1.4 ne vous concernent pas si vous utilisez votre propre ordinateur.**

## 1.1 Avertissement

Ne pas utiliser la machine de nom "mekanik" (première rangée vers le milieu).

## 1.2 Pour ceux qui ont déjà une version d'android

Si vous aviez déjà installé android l'an dernier et que ça fonctionnait correctement, vous pouvez essayer ceci: Faites le point 1.4. Puis lancez `android-studio`, en choisissant de garder vos paramétrages au premier écran.

Si ça ne marche pas: supprimez les dossiers `.android`, `.AndroidStudioXX`, et `.gradle`. Gardez `AndroidStudioProjects` qui est le répertoire où se trouvent vos projets de l'an dernier. Puis, suivez ce qui est expliqué ci-dessous.

## 1.3 Pour ceux qui ne l'ont pas déjà installé

Android Studio n'est pas vraiment conçu pour fonctionner avec des montages de type NFS, ce qui pose un certain nombre de problèmes. Pour les résoudre, nous vous demandons de suivre **attentivement** les explications suivantes.

Tout d'abord, vous allez être obligés de travailler en partie "en local", c'est-à-dire en utilisant la mémoire disque propre au PC sur lequel vous êtes pour stocker votre émulateur.

A priori, cette mémoire ne sera pas systématiquement nettoyée, donc il peut être plus confortable mais pas indispensable de travailler toujours sur le même PC.

## 1.4 Parametrages 1– A faire à chaque fois que vous changez de PC

**et à refaire si le répertoire `android-`*votrelogin* n'est plus là (nettoyage)**

Ceci consiste à faire stocker votre émulateur sur le disque local, sans qu'Android ne le sache (lien symbolique).

```
mkdir -p /local2/tmp/android-$USER
mkdir -p ~/.android
rm -rf ~/.android/avd
cd ~/.android && ln -s /local2/tmp/android-$USER avd
cd
```

## 1.5 Paramétrages 2 – À faire une seule fois:

Depuis la ligne de commande, lancez la commande `android-studio`. Suivez **à la lettre** les indications ci-dessous:

- Au premier écran, choisissez "Do not import settings".
- On tombe ensuite sur un message "unable to access Android SDK add-on list" cliquez sur "Setup Proxy", puis "Manual proxy configuration", mettez "cache" comme hostname et 3128 comme port, puis OK.
- Vous arrivez alors à un écran de "Setup Wizard". **Important: *annuler* ce wizard en cliquant sur Cancel,** sinon il télécharge et met dans le compte utilisateur toute une installation d'Android :-(. Confirmez cette annulation dans l'écran suivant en choisissant **"Do not re-run the setup wizard"** puis "OK".
- L'écran suivant a pour titre "Welcome to Android Studio". Tout en bas, choisissez "Configure" puis "Project Default" puis "Project Structure". Tapez `/opt/android` dans le champ Android SDK Location, puis "OK". "Default Settings", puis "Build, Execution, Deployement", puis "Gradle" et cochez "Work off line".
- Vous pouvez maintenant choisir de créer un nouveau projet comme au premier paragraphe du 1.6 **Important:** Vous aurez durant la création un écran sur le réglage du proxy, mettez de nouveau "cache" et "3128", et choisissez aussi "Enable HTTPS Proxy" (et "Do not show this dialog in the future"). Puis "Ok".
- En cas de souci avec cette configuration, effacer le dossier /.AndroidStudio2.1 et relancer android-studio.

## 1.6 Un premier projet

**Commencez un projet:** Choisissez "start a new Android Studio project". À l'écran suivant, vous pouvez choisir, par exemple, comme nom d'application "TP1", appuyez sur "Next", choisissez comme "Minimum SDK" la version 15. Appuyez encore sur "Next", acceptez le choix "Empty Activity", puis sur "Finish" sans rien changer à l'écran suivant.

Le démarrage du projet est un peu lent la première fois, mais si vous avez l'impression qu'il bloque vraiment, appelez votre chargé de TD. L'écran de travail d'Android Studio doit s'ouvrir durant ce processus.

**Configurez un simulateur – A faire une seule fois** En haut de la fenêtre, vers la droite, cliquez sur le bouton sur lequel on voit un téléphone violet (AVD Manager). Dans la fenêtre qui s'ouvre cliquez sur "create virtual device", Sélectionnez Nexus 6. Cliquez sur "Clone device..".

Dans la fenêtre suivante, cliquez sur "Marshmallow x86", puis "Next" et enfin "Finish". Il y a un peu d'attente avant d'avoir une fenêtre "Your Virtual devices", fermez cette fenêtre.

Vous n'aurez besoin de refaire cette opération que si vous utilisez les PC de la salle machine et que vous changez de PC.

**Lancez le simulateur (et l'application):** Cliquez sur la flêche verte. **Le simulateur met un certain temps à charger, ne l'éteignez surtout pas une fois qu'il est lancé.** Au bout d'un temps un peu long, le simulateur apparaît, puis votre application (Hello World) se lance, testez-la.

Par la suite, vous pourrez lancer une application sur le même simulateur avec cette même flèche verte, dans ce cas-là: cliquez sur ok pour lancer l'application sur l'émulateur déjà lancé.

# 2 La première "vraie" application

Nous allons maintenant transformer cette première application pour en faire une application qui permet d'écrire un nombre en binaire à l'aide de boutons "0" et "1" et de convertir ce nombre simultanément en décimal.

## 2.1 Mise en page (layout)

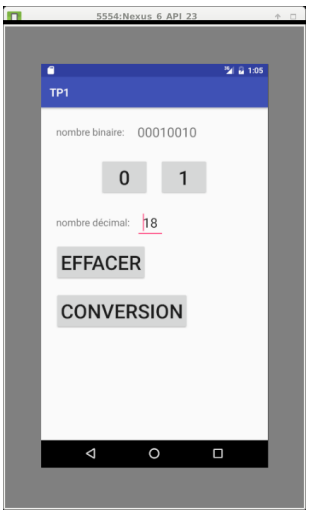


Figure 1: Capture d'écran

Ouvrez le fichier contenu dans `res/layout`, cliquez sur le bouton "Text" en bas à gauche pour voir le fichier xml. Effacez-le et copiez-collez à la place le fichier `activity_main.xml` que vous récupérerez sur Moodle, (ouvrez le fichier de Moodle sous emacs ou gedit, pas avec un navigateur web). Pour que le fichier s'indente correctement, utilisez "Code/Reformat code" (utile aussi avec le code java).

La prévisualisation ne fonctionne pas toujours, ne vous inquiétez pas.

Lancez l'application (flêche verte), si vous cliquez sur un des boutons "0" ou "1", il y aura un bug c'est normal.

## 2.2 Code java

### 2.2.1 Comment va fonctionner le programme?

L'idée de base est de garder l'état de l'application sous la forme d'une chaîne de caractères `cumulBinaire` qui contiendra la suite des "0" et des "1" tapés depuis le dernier effacement. Dans l'exemple de la capture d'écran, on a donc `"00010010"` dans `cumulBinaire`.

Nous aurons une méthode `void miseAJour()` qui sera chargée de modifier les informations sur l'écran en fonction de `cumulBinaire`.

### 2.2.2 Étape 1

Dans un premier temps, on ne va s'occuper que des deux boutons "0" ou "1" et du `TextView` d'identifiant `nombre_binaire`.

Déclarez dans le code java des attributs correspondant à ces 3 éléments, ainsi qu'un attribut `cumulBinaire` de type `String` initialisé à la chaîne vide, qui va contenir le nombre binaire déjà tapé:

```
private Button mBoutonZero;
.........
```

Dans le `onCreate(...)`, intialisez ces attributs, **après** le `setContentView(..)`:

```
mBoutonZero = (Button)findViewById(R.id.bouton_zero);
.........
```

Les identifiants `R.id....` se retrouvent en regardant le layout.

Si vous regardez le layout, vous trouverez qu'on a déclaré une méthode callback de nom `nouveauChiffre` sur les deux boutons (attribut `onClick` dans le XML). Déclarez cette méthode (`void nouveauChiffre(View bouton)`) dans le code java et faites en sorte qu'elle mette à jour l'attribut `cumulBinaire` (par une simple concaténation), puis appelle une méthode de mise à jour de l'écran (voir plus bas). On remarquera que l'argument de type `View` correspond au bouton appuyé (ici un objet de type `Button` en java, `Button` étant une sous-classe de `View`), pour savoir quel bouton a été appuyé, il suffit donc de le comparer avec l'attribut `mBoutonZero` avec `==`.

Écrivez la méthode de mise à jour qui pour l'instant consiste juste à écrire dans le `TextView` la chaîne `cumulBinaire` (méthode `setText`).

Vérifiez que cela fonctionne.

Pour voir les erreurs de compilation, cliquez sur "Build" en bas d'android-studio, pour les erreurs d'exécution, cliquez sur "Logcat": en général, il ne faut tenir compte que des erreurs en rouge.

Corrigez les éventuels bugs.

### 2.2.3 Étape 2

Maintenant, occupons-nous du nombre décimal (i.e. nombre en base 10): Il sera écrit dans un `EditText`, (classe `EditText`), pour cette étape-ci, on va l'utiliser comme un `TextView`, mais on le déclare comme un attribut `EditText`.

Il suffit de convertir la chaîne représentant le nombre en binaire en utilisant `Integer.parseInt( , )` avec les bons arguments (le deuxième argument est la base de numération) et d'afficher ce nombre en le convertissant en chaîne de caractères avec `Integer.toString( )`.

### 2.3 Un bouton en plus

On va maintenant gérer le bouton "Effacer" qui effacera le nombre binaire et le nombre décimal. Il n'y a pas d'attribut `onClick` dans le XML, on va utiliser un écouteur (listener) pour programmer son fonctionnement. (Voir `setOnClickListener` dans le cours à la page 11.) L'effacement se fera en affectant la chaîne vide à `cumulBinaire` et en utilisant la méthode de mise à jour qu'il faudra modifier pour qu'elle traite le cas où cette chaîne est vide.

## 3 S'il vous reste du temps

### 3.1 Mode paysage

On veut que, lorsque le téléphone est tourné, l'application utilise un autre layout.

Pour créer ce layout, dans l'onglet "Design" du fichier `XML` du layout, cliquez avec le bouton droit sur l'icône représentant un téléphone en train de tourner, et choisissez "create Landscape Variation". Il doit apparaître dans la fenêtre "Project", dans le répertoire `layout` avec le nom puis "(land)" entre parenthèses. (Si vous êtes curieux, vérifiez à partir du terminal qu'en fait le fichier est dans le répertoire `layout-land`!)

Ouvrez ce nouveau fichier, puis changez la couleur de fond avec par exemple `android:background="#AA0088"` (à glisser après `android:orientation="vertical"` vers la ligne 6). Cela permettra de différencier les deux layout.

Maintenant, testez votre application: écrivez quelques chiffres, puis tournez l'appareil. Que se passe-t'il?

Pour remédier à ce problème, il vous faudra sauvegarder la valeur de `CumulBinaire` en redéfinissant `onSaveInstanceState` (voir cours p.18 et suivantes). Dans `onCreate` il faudra contrôler la valeur de `savedInstanceState` et agir en fonction. Pensez à appeler la mise-à-jour. On utilisera les méthodes `putString` et `getString` au lieu de `putInt` et `getInt` dans le cours.

Vérifiez en faisant tourner l'appareil après avoir écrit des chiffres et aussi juste après un effacement.

## 4 Pour ceux qui ont fini

Vous pouvez programmer le comportement du dernier bouton: celui qui va convertir un nombre décimal en un nombre binaire. La partie où est écrit le nombre décimal est un `EditText` qui permet une saisie par l'utilisateur. Pour récupérer la valeur saisie, on utilise la méthode `getText()` qui retourne un objet de type `CharSequence`, qu'il faut donc convertir en `String` avec `toString()`.

Ensuite, il faudra changer la valeur de `CumulBinaire` et faire une mise à jour de l'affichage.