AMIO : Applications mobiles et Internet des Objets
Troisième année

**T.P. 1 : Programmation Android 101**

**Ressources à consulter sans modération durant les séances de TP :**
Documentation API Android:
http://developer.android.com/guide/index.html
Cours avec exemples de code:
http://www.univ-orleans.fr/lifo/Members/Jean-Francois.Lalande/teaching.html

## Exercice 1 : Prise en main de l'IDE

L'objectif de cet exercice est de se familiariser avec l'environnement de développement utilisé dans ce TP et les suivants, à savoir l'IDE Eclipse et le Android SDK.

- Question 1 - Créer une nouvelle application Android.
  - dans Android Studio, créer un nouveau projet “ProjetAMIO” pour téléphone et tablette
    - choisir le template Empty View Activity
    - choisir l'API 26: Android 8 (Oreo) comme SDK Minimum
    - choisir JAVA comme langage de programmation
  - une fois l'application créée, laisser le code par défaut
  - retrouver les éléments vus en cours dans l'arborescence du projet (Manifest, code source, ressources)
  - ajouter juste la ligne suivante dans la méthode onCreate() de l'activité principale:

```
Log.d("MainActivity", "Création de l'activité");
```

    - permet d'afficher une ligne de log de niveau “debug” (équivalent à println)

- Question 2 - Créer un émulateur Android.
  - utiliser AVD (Android Virtual Device) Manager
  - choisir un virtual device avec une version d'API >= 26 (minimum pour pouvoir exécuter votre application, mais vous pouvez aussi prendre l’API la plus récente fournie par Android Studio car la rétrocompatibilité est assurée). Le type de smartphone émulé importe peu (un pixel 3 sera très bien).
  - note: l'accélération matérielle de la virtualisation rend l'émulation beaucoup plus fluide (paquet qemu-kvm sous Debian* ; penser à ajouter l'utilisateur au groupe kvm)

- Question 3 - Exécuter l'application.
  - appuyer sur Run et choisir le téléphone virtuel instancié. L'activité « Main » de l'application doit apparaître au premier plan.
  - visualiser les logs générés par l'application : ouvrir logcat et trouver la ligne de log de la Question 1. Ajouter un filtre pour ne voir que les messages de l'activité principale.

## Exercice 2 : Service et tâche périodique

Cet exercice reprend et étend l'embryon d'application créé lors de l'exercice précédent. L'objectif est de comprendre le fonctionnement de certains composants que l'on peut rencontrer dans une application Android.

- Question 1 - Ajouter un service à l'application.

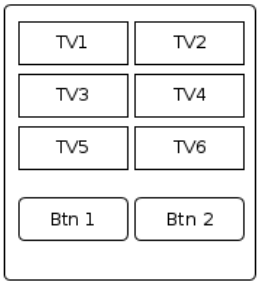
- créer un nouveau service « MainService » dans le répertoire des sources
- le service doit être lancé et stoppé depuis l'activité
  - création et démarrage dans la méthode onCreate() de l'activité principale
  - démarrage du service en mode « sticky » (redémarrage automatique en cas d'arrêt)
  - arrêt dans la méthode onDestroy()
- afficher des lignes de logs pour vérifier le bon fonctionnement du service
- note: par défaut, le service partage le même thread que les activités de l'application et ne doit pas faire de traitement lourd sous peine de figer l'UI. Solution : demander explicitement dans le Manifest son exécution dans un processus indépendant ou créer des threads dans le service

- Question 2 - Implémenter un timer dans le service.
  - API: TimerTask, tâche pouvant être exécutée par un timer périodique (par exemple toutes les 30 secondes)
  - afficher une ligne de log à chaque fois que la tâche périodique est exécutée (méthode run() )
  - note: l'objet Timer est exécuté dans un thread différent, il faut utiliser la fonction cancel() quand le service est détruit (onDestroy) afin de l'arrêter. Alternative : utiliser dans l'API ScheduledThreadPoolExecutor

## Exercice 3 : Conception d'interface utilisateur

Cet exercice introduit quelques éléments d'interface graphique et d'interaction avec les utilisateurs.

- Question 1: Ajouter à l'interface graphique de l'activité principale les éléments suivants :
  - 6 TextViews (3x2)
    - TV1: “Service status: “ & TV2: “???”
    - TV3: “Last result:” & TV4: “???”
    - TV5: “Last alert:” & TV6: “???”
  - 2 boutons
    - 2 boutons switch (ou toggle) “On/Off”
      - sur Off par défaut
  - Changer éventuellement le Layout pour avoir une meilleure disposition




- Question 2: Réagir aux appuis sur le bouton toggle.
  - dans le code de l'activité principale
  - méthode setOnCheckedChangeListener()
  - prérequis: retrouver dans le code les composants graphiques (Textview TV2 + bouton toggle)
    - méthode findViewById()
  - bouton toggle pressé une première fois => On
    - démarrer le service
    - modier TV2: “en cours”
  - bouton toggle pressé une seconde fois => Off
    - stopper le service
    - modifier TV2: “arrêté”
  - note: une fois que le listener est correctement implémenté, enlever les lignes de code de l'exercice 2 question 1 permettant le démarrage et l'arrêt du service.

## Exercice 4 : Persistance de données et gestion des Broadcasts

Cet exercice porte d'une part sur l'implémentation des préférences et réglages que peut offrir une application Android, et d'autre part sur les mécanismes de type IPC (à l'aide d'Intent-Filter et de Broadcast Receiver) qui peuvent être utilisés dans Android pour faire communiquer les applications entre elles.

- Question 1: Ajouter une checkbox dans l'interface utilisateur de l'activité principale.
  - texte de la checkbox: “Start at boot”
  - ajouter un écouteur sur la checkbox
    - méthode setOnCheckedChangeListener()
    - afficher une ligne de log quand l'état de la checkbox change

- Question 2: Rendre l'application paramétrable/customisable.
  - API: SharedPreferences
  - Créer 1 entrée de type booléen dans les préférences de l'application
    - cette entrée va sauver l'état de la checkbox
    - true si la checkbox a été explicitement cochée
    - false par défaut ou si la checkbox a été explicitement décochée

- Question 3: Démarrer automatiquement le service après le boot du smartphone.
  - Si le paramètre a été positionné - i.e. checkbox cochée - au préalable
  - note: l'application doit être installée dans la mémoire interne
  - déclarer la permission et le broadcast receiver nécessaires dans le AndroidManifest.xml
    - dans l'exemple suivant, le nom de la classe implémentant le receiver sera MyBootBroadcastReceiver, et sera contenu le même package (ici « com.amio »)

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

<receiver
    android:name="eu.telecomnancy.cholez.projetamio.MyBootBroadcastReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

  - Implanter le broadcast receiver MyBootBroadcastReceiver.java
    - afficher une ligne de log pour vérifier que le receiver fonctionne correctement
    - démarrer le service dans la méthode onReceive() après avoir vérifié dans les SharedPreferences que l'utilisateur a bien coché l'option correspondante
    - utiliser adb shell pour simuler l'envoi d'un Broadcast et ainsi tester votre classe

```
./adb shell am broadcast -a android.intent.action.BOOT_COMPLETED
```