TP02 - Cycle de vie d'une Activité

Objectif:

- Comprendre le cycle de vie de chaque activité avec l'utilisation d'un Tracker;
- Implementer different layout;
- Implementer quelques valeurs défaut dans le dossier */values*

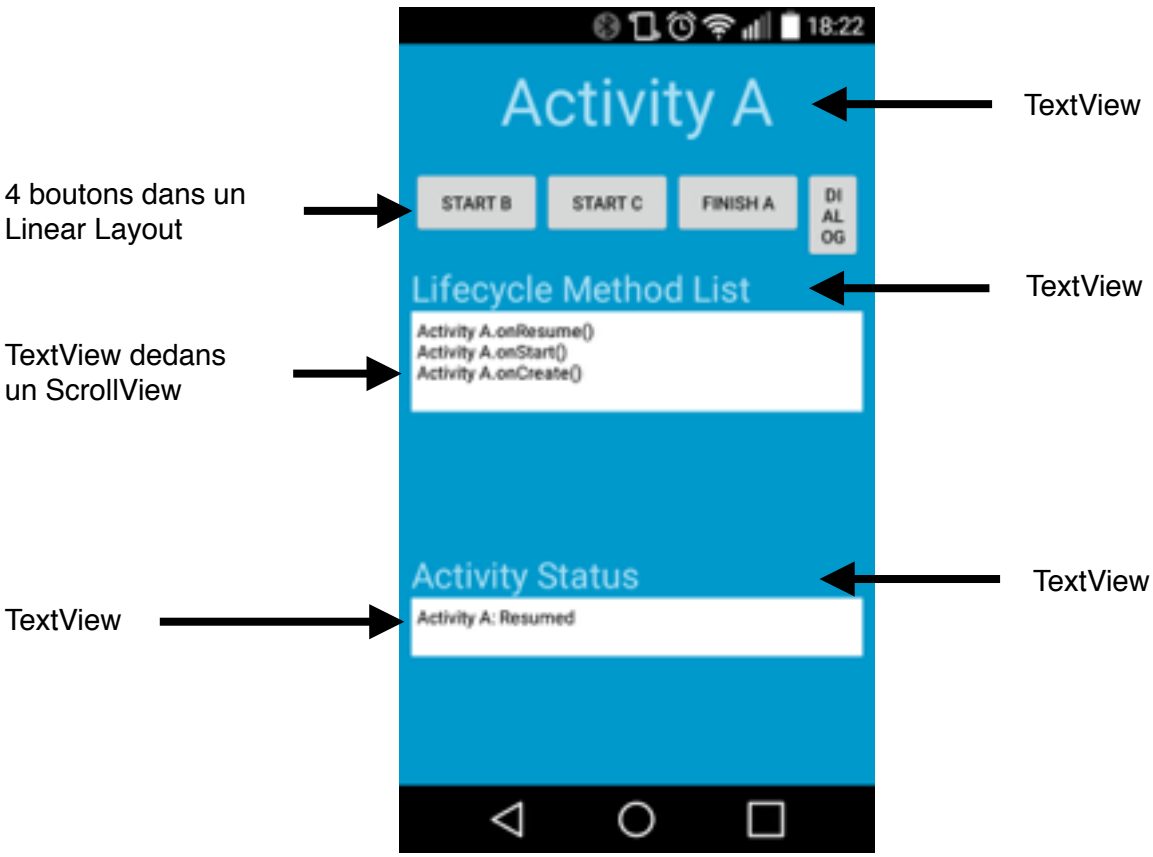
Dépendances

- Télécharger les 2 fichiers *StatusTracker* et *Utils* que se situe dans:

http://mkpereira.tk/realisations_app/tp02/

Exercices

1 - Créer une activité appeler *ActivityA* (un **extends** de la classe *Activity*) avec un layout *activity_a*. Le layout doit avoir la disposition suivante (pas nécessaire d'être exactement pareil):



2 - Utiliser des textes défini dans le fichiers strings.xml dans le dossier */values* (ne pas les écrire en dur).

3 - Définir des ids pour chaque Widget et faire qu'on que les boutons appellent des fonctions qu'on va implementer en Java (on peut les appeler startActivityB(), startActivityC(), startDialog(), finishActivityA())

4 - Dans notre classe ActivityA:

- Déclarer un objet StatusTracker comme global de la façon suivante :

  private StatusTracker mStatusTracker = StatusTracker.getInstance();

- Déclarer 2 objets TextView et les associé au 2 TextView sur fond blanc déclarer dans notre layout;
- Déclarer un String avec le nom de l'activité.
- Implementer les différentes méthodes du cycle de vie d'une activité:
  - onCreate(Bundle savedInstance),
  - onStart(),
  - onRestart(),
  - onPause(),
  - onPause(),
  - onDestroy()
- Sur chacune des méthodes, envoyer le status de l'activité en utilisant l'objet mStatusTracker avec la méthode *setStatus* qui prend 2 *String* comme arguments, le nom de l'activité, et son status (en consequence de la méthodes du cycle de vie appeler). Après le status défini, utiliser la méthode statique *printStatus* de la classe *Utils* (pas nécessaire de déclarer un objet) qui prend comme arguments 2 TextView (nos TextView déclarer auparavant, ceux sur fond blanc).

5 - Créer une ActivityB et une ActivityC avec leur layout respectifs (ne pas oublier de les déclarer dans le *AndroidManifest.xml*). On va répéter le layout de l'ActivityA mais avec des fonds de couleurs différents. Implementer les méthodes du cycle de vie et les différents objets, comme pour l'ActivityA.

6 - Créer une 4ème activité appeler DialogActivity. Cette activité auras un style spécial, une boite de dialogue, pour ça on la déclare dans le *AndroidManifest.xml* avec l'attribut suivant:

android:theme="@android:style/Theme.Dialog"

Aussi, dans la méthode *onCreate(Bundle savedInstance)* on rajoute l'appel suivant:

requestWindowFeature(Window.FEATURE_NO_TITLE);

7- Son layout sera très simple: un *LinearLayout* en orientation vertical avec un *TextView* et un *Button* qui appellera la méthode *finishDialog().* Dans cette méthode, on forcera l'arrêt de l'activité en appellant la méthode *finish()* de la façon suivante:

DialogActivity.this.finish();

8 - Déclarer les différentes méthodes appeler par les boutons, pour quelles puissent, chacune d'entre elles démarrer les autres activités. Pour démarrer une activité on va utiliser un *Intent.* Exemple:

```
Intent intent = new Intent(getApplicationContext(), ActivityB.class);
startActivity(intent);
```

Chaque méthode appeler par chaque boutons dans les différentes activité devra démarrer une autre activité. Dans ActivityA, il devra être possible de démarrer ActivityB, ActivityC et DialogActivity. Dans ActivityB, démarrer ActivityA, ActivityC et DialogActivity. Dans ActivityC, démarrer ActivityA, ActivityB et DialogActivity.

On a aussi les boutons *finishA* (dans Activity A)*, finishB* (dans ActivityB) et *finishC* (dans ActivityC). Ces derniers forceront la destruction de l'activité en question, tout comme le bouton sur la DialogActivity:

ActivityA.this.finish();

Tout ça implanter, on pourra visualiser le passage sur chaque méthode grace a notre mStatusTracker et la classe Utils qui va imprimer les different status dans les TextViews donner en arguments.