Utilisation des listes

Exercice 1 – Afficher des listes

On souhaite développer une application pour présenter la liste des ressources enseignées au semestre 4 du BUT Informatique. L'application sera dotée de 4 boutons, pour au choix de l'utilisateur, visualiser la liste de toutes les ressources du semestre 4, la liste des ressources du tronc commun, la liste des ressources du parcours A, la liste des ressources du parcours D.

A son lancement, l'application affichera la totalité des ressources, dans une liste dotée d'une barre de défilement. Dans cette première version, à tout moment, si l'utilisateur clique sur l'une des ressources, un message de type *Toast* apparaîtra et lui rappellera le nom de la ressource qu'il a sélectionnée.

Application à son lancement

L'utilisateur a cliqué sur le bouton « Ressources parcours D »

Indications

- ✓ Les noms des ressources seront définis dans le fichier *strings.xml*, en tant que *string-array*.
- ✓ Dans le code Java, les listes de ressources seront représentées par des *ArrayList*. En effet, avec un tableau, un problème se produirait lorsque la liste serait vidée.
- ✓ Un seul adaptateur suffit. Il faudra bien sûr adapter le contenu de la liste qu'il gère en fonction du clic de l'utilisateur. A chaque clic sur un bouton, le plus simple est de vider la liste associée à l'adaptateur, en agissant directement sur l'adaptateur, et ensuite d'ajouter à celui-ci (en fait à la liste qui lui est associée) la liste des ressources que l'on souhaite afficher.

- ✓ Les couleurs suivantes seront définies :
  - couleur du titre : `#0E8891`
  - couleur des libellés des boutons : `#082D73`
  - couleur de fond des boutons : `#A5DDE1`

- ✓ Il existe dans la classe *Arrays*, classe prédéfinie du langage Java standard, une méthode permettant de transformer un tableau en liste.

A faire

Développer l'application ainsi décrite.

1) Récupérer sur Moodle les fichiers suivants :
   - *strings.xml*
   - *dimens.xml*
   - *a ajouter au fichier des styles.xml* : ce fichier contient la définition d'un style pour les boutons
   - *a inserer dans main_activity.xml* : ce fichier contient une partie de la déclaration des widgets

2) Intégrer à votre projet les fichiers *strings.xml* et *dimens.xml*

3) Dans le fichier des couleurs qui a été généré automatiquement par Android Studio, définir les couleurs qui auront pour identifiant : **couleurTitre**, **couleurFondBouton**, **couleurLibelle** (consulter les fichiers *xml* fournis pour voir l'usage des ces identifiants)

4) Ajouter au fichier des styles qui a été généré automatiquement, le style décrivant les boutons (voir fichier fourni).

5) Dans le fichier *xml* décrivant la vue de l'activité :
   - remplacer le *ConstraintLayout* par le gestionnaire de mise en forme qui vous semble adapté
   - insérer le code fourni et décrivant en partie les *widgets*
   - appliquer le style adéquat sur les boutons
   - déclarer le *widget* permettant d'afficher la liste des ressources
   - effectuer d'autres adaptations si nécessaire

6) Coder la classe activité. Tester et mettre au point l'application.

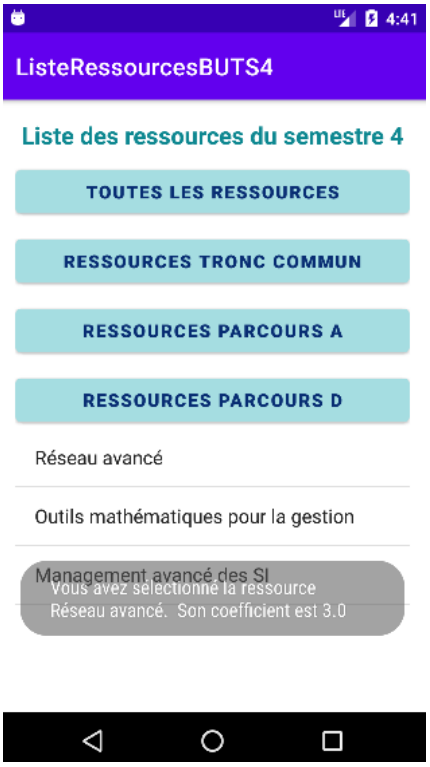
Exercice 2 – Logo et affichage des coefficients

L'objectif est de compléter l'application précédente pour lui ajouter les 2 éléments suivants.

1) On souhaite que l'icône de lancement de l'application ne soit pas celle par défaut, mais le logo du BUT Informatique. Cette image est disponible sur Moodle. Modifier l'application et notamment le fichier *AndroidManifest.xml* pour doter l'application de cette icône de lancement.

2) Faire en sorte que lorsque l'utilisateur clique sur un module, le coefficient de celui-ci s'affiche dans le message de type *Toast*. Ces coefficients seront fictifs. Les coefficients pourront être définis dans un tableau, en tant que ressources, dans le fichier *strings.xml*. La balise à utiliser est *integer-array* (au lieu de *string-array).*

```
<integer-array name="coef_parcours_D">
    <item name="R4_D_08">30</item>
    <item name="R4_D_09">31</item>
    <item name="R4_D_10">32</item>
</integer-array>
```



Il faut noter qu'il n'est pas possible de définir des tableaux contenant des nombres à virgules (dans le fichier xml des *strings*). Or certains des coefficients à stocker pourraient comporter des décimales : 1,5 par exemple. Une solution possible consistera à définir dans le fichier *xml*, le coefficient multiplié par 10 (donc 15 au lieu de 1,5). Bien sûr, il sera nécessaire dans le code Java de rétablir le véritable coefficient.

Exercise 3 (Bonus) – *International Application*

Android runs on many devices in many regions. To reach the most users, your app should handle text, audio files, numbers, currency, and graphics in ways appropriate to the locales where your app is used.

Resources are text strings, layouts, sounds, graphics, and any other static data that your Android app needs. An app can include multiple sets of resources, each customized for a different device configuration. When a user runs the app, **Android automatically selects and loads the resources that best match the device.**

When you write your app, you create **default and alternative resources** for your app to use. When users run your app, the Android system selects which resources to load, based upon the device's locale. To create resources, you place files within specially named subdirectories of the project's res/ directory.

Whenever the app runs in a locale for which you have not provided locale-specific text, Android loads the default strings from res/values/strings.xml. If this default file is absent, or if it's missing a string that your app needs, then your app doesn't run and shows an error.

Suppose that your app's default language is French. Suppose also that you want to localize all the text in your app to English. In this case, you could create two alternative strings.xml files, each stored in a locale-specific resource directory:

res/values/strings.xml	contains French text for all strings that the app uses
res/values-en/strings.xml	contains English text for all strings that the app uses

1) Duplicate strings.xml file and translate all texts in English (resource names, button labels, app title…)

2) Test - Testing on an emulator

a) Launch an emulator
b) The default locale is French, so the app launches with French text
c) From emulator settings (do it like on a smartphone), pick the locale English
d) Restart the app : the app launches with English text
e) From emulator settings (do it like on a smartphone), pick the locale Japanese
f) Restart the app : which locale is used