Android TP 1
Présentation et mise en place

1 Internationalisation

On souhaite adapter le texte affiché par le programme à la langue du *device* ou dans notre cas à celle de l'émulateur.

1. Dans un nouveau dossier, créer un projet par défaut.
2. Dans le dossier `res/values` de ce projet, modifier le fichier `strings.xml` et ajouter une nouvelle chaîne de caractères de nom `hello` dont le contenu est `Hello!`.
3. Modifier le fichier `res/layout/main.xml` et ajouter un champs de texte de type `TextView` dont le contenu sera celui de la chaîne `hello` en modifiant son attribut `android:text` : `android:text="@string/hello"`.
4. Compiler le projet et installer le de manière à vérifier que l'affichage est conforme.
5. Dans le dossier `res`, créer un dossier `values-fr` et copier le fichier `res/values/strings.xml` dedans.
6. Modifier cette copie et remplacer le contenu de la chaîne `hello` par `Bonjour!`.
7. Compiler à nouveau le projet et installer le. Changer ensuite la langue par défaut de l'émulateur pour passer de l'anglais au français et réciproquement. Relancer votre programme et vérifier que le champs de texte contient bien la chaîne `Bonjour` lorsque la langue par défaut est le français et `Hello` si une autre langue est choisie.
8. Ajouter le support d'une autre langue, par exemple l'espagnol.

Les curieux peuvent consulter les sites suivants pour plus d'informations sur l'internationalisation :
— `http://developer.android.com/training/basics/supporting-devices/languages.html`
— `http://developer.android.com/guide/topics/resources/localization.html`

2 Interface

Pour réaliser les exercices de cette section vous devrez lire les 2 documentations suivantes :
— `http://developer.android.com/guide/topics/ui/declaring-layout.html`,
— `http://developer.android.com/guide/topics/ui/controls.html`,

1. Ajouter un bouton à l'interface au-dessous des champs de texte, occupant toute la largeur de l'écran et dont la hauteur s'adapte au contenu.
2. Imbriquer un *layout* au dessous du bouton et y ajouter un champs de texte et un bouton côte à côte.
3. Tester les autres éléments disponibles : `TextField`, `ToggleButton`, ...

3 Évènements

1. En suivant les guides :
   — `http://developer.android.com/guide/topics/ui/controls/button.html#HandlingEvents`,
   — `http://developer.android.com/guide/topics/ui/notifiers/toasts.html`,
   ajouter un évènement sur les boutons déclenchant l'affichage de `Toasts`. Utiliser la méthode xml et la méthode "à la Swing".
2. Modifier également le texte et/ou la couleur du bouton à chaque appui.

4 Log

Lors d'une erreur dans l'exécution de votre programme, le seul message affiché à l'utilisateur est "L'application s'est arrêtée" ou un message similaire, ce qui n'est pas très utile pour débugger. La classe `Log`, dont l'utilisation est décrite ici : `http://developer.android.com/tools/debugging/debugging-log.html`, sert à produire des messages d'information, de warning ou d'erreur, consultables à l'aide de la commande `adb logcat`.

1. Lancer la commande `adb logcat` dans un terminal pendant que vous installer et lancer votre application. Par défaut tous les messages du système et des applications sont affichés mais il est possible de filtrer cette sortie : `http://developer.android.com/tools/debugging/debugging-log.html#filteringOutput`
2. Ajouter dans la méthode `onCreate` un message d'information et vérifier que celui-ci s'affiche dans la console à l'exécution de votre programme en filtrant uniquement les messages de votre application.