Projet Long M1 **Année 2009-2010**

TP1

1 Informations diverses

Dans cet énoncé, les noms de classe seront systématiquement préfixés par le nom du paquetage. La documentation sur l'API est sur

`http://developer.android.com/reference/packages.html`.

`eclipse` permet d'importer les paquets manquants grâce à la combinaison de touches `Ctl-Maj-o`.

Un site Didel nommé PROJLONG a été créé pour le cours de Projet Long. Le site est accessible à partir de votre ENT. Vous y trouverez des documents.

Pour ce TP, on vous fournira des fichiers, ils seront sur le site Didel et aussi dans le répertoire `/ens/fagnot/Public/ANDROID/` sur `lucien`.

2 But du TP

le but de ce TP est de créer une application qui implémente un jeu d'anagramme : on donne au joueur un mot dont les lettres ont été mélangées (par exemple, "isonam"), il doit retrouver le mot initial (dans l'exemple, "maison"). La liste des mots proposés sera dans une base de données.

3 Créer l'application

Si vous êtes sur les PC de l'UFR, utilisez la commande `eclipse-galileo` pour lancer Eclipse.
Créez un nouveau projet Android. Remplir les champs comme suit :

- **Project name :** Anagramme
- Choisir **create new project** et **use default location**
- **Build Target :** Android 1.5
- **Aplication name :** jeu d'anagramme
- **Package name :** essai.jeu
- **create Activity :** Main

4 Première activité

4.1 Installation du layout

Récupérez le layout (*présentation de la Vue*) de la première activité (`Main`) `main.xml` (sur Didel ou sur `lucien`).

Remplacez le fichier `res/layout/main.xml` de votre application par celui que vous avez récupéré.

Ouvrez le fichier `res/values/strings.xml` et ajoutez-y les définitions que vous récupérerez dans `string` que l'on vous a fourni.

Vous pouvez lancer l'application, vous verrez apparaître un texte, avec deux boutons qui pour l'instant ne réagissent pas au clic.

4.2 Accéder aux boutons

Ouvrez `main.xml` : vous verrez qu'on y déclare deux boutons, on attribue à chacun un identifiant par : `<Button android :id="@+id/accepter" ...>`.

Ouvrez maintenant `src/essai/jeu/Main.java`.

Déclarez deux Boutons `boutonOui` et `boutonNon` (`android.widget.Button`) comme attributs de la classe `Main`.

Pour avoir accès à l'objet bouton correspondant à celui du layout, on écrira

```
boutonOui = (Button) findViewById(R.id.accepter) ;
```

(On rappelle que la classe `R` est engendrée automatiquement à partir des ressources déclarées dans le répertoire `res`. La variable `R.id.accepter` est donc engendrée à partir de layout `main.xml`.)

4.3 Définition d'un contrôleur

Dans Android, comme pour les interfaces graphiques en Java, c'est un objet d'une classe spéciale qui gère ce qui se passe quand on clique sur un bouton. On parle de *contrôleur*.

Créez une telle classe en implémentant l'interface `android.view.View.OnClickListener`, sous forme de classe interne. Comme spécifié dans la documentation, il suffit d'implémenter la méthode `void onClick(View v)`, où `v` est la vue (ici le bouton) cliquée. Le corps de la méthode consistant en la description de ce que vous voulez faire dans chaque cas. Pour l'instant, dans le cas où le bouton "NON" est choisi, on finit l'activité (`finish()`), dans l'autre cas, on ne fait rien.

Il reste à ajouter dans la méthode `OnCreate()` un tel contrôleur aux deux boutons (méthode `setOnClickListener(View.OnClickListener l)`). On ajoute le **même** contrôleur pour les deux boutons.

Testez votre programme.

5 Créer une deuxième activité

Vous allez créer une deuxième activité, dans laquelle le jeu proprement dit se déroulera. Pour cela, créez une classe `Jeu` et un nouveau layout `jeu.xml` qui correspondra à l'activité.

Pour l'instant, le `OnCreate()` de `Jeu` se contentera d'afficher le layout `jeu.xml`.

5.1 Layout

Faites un layout qui contiendra au moins
- Une zone de texte (`TextView`) qui contiendra l'anagramme (cette zone doit avoir un identifiant).
- Une zone de saisie (`EditText`) pour la réponse du joueur.
- Un bouton "Confirmer" que le jouer cliquera pour envoyer sa réponse.

Vous pouvez ajouter des zones de texte supplémentaires pour indiquer au joueur ce qu'il doit faire.

5.2 Déclaration dans le manifeste

Déclarez l'activité dans le fichier `AndroidManifest.xml` (inutile de déclarer un Intent-Filter).

5.3 Affichage d'un mot

Dans `OnCreate()`, faites afficher un mot dans la zone de texte (méthode `setText(CharSequence text)`). Pour l'instant, le mot qu'on affiche est une constante, par exemple "isonam".

5.4 Passage d'une activité à une autre

Modifiez la première activité pour qu'elle appelle la deuxième activité quand on clique sur le bouton "OUI" (méthode `startActivity()`).

Pour cela, tapez le code suivant où `contexte` est l'activité depuis laquelle on lance l'appel :

```
Intent i = new Intent(contexte, Jeu.class);
startActivity(i);
```

Testez de nouveau votre application.

5.5 Finir l'activité Jeu

A partir de maintenant, vous devrez aller chercher une partie des informations dans l'API. Pour vous aider, la classe où se trouve la méthode est précisée.

On va maintenant tester dans l'activité `Main` si le joueur a gagné ou perdu, pour l'instant, il a gagné s'il tape "maison". On va ensuite modifier les activités `Main` et `Jeu` pour que le résultat ("GAGNÉ" OU "PERDU") s'affiche dans `Main` dans le premier `TextView` :

- Pour appeler une activité avec résulat, on utilisera `startActivityForResult()` (classe `Activity`).
- Pour donner le résultat (sous forme d'Intent), on utilisera `setResult(int resultCode, Intent data)` (classe `Activity`).
- L'intent sera rempli avec les données résultats grâce à la méthode `putExtra()` de `Intent`.
- Pour récupérer le résultat de l'appel, utilisez `getExtras()` sur l'intent passé en paramètre de `OnActivityResult()` et redéfinissez la méthode `OnActivityResult()` de `Activity`.

Récupérez le fichier `Algo.java` qui calcule à partir d'un mot un de ses anagrammes.

6 Base de données

Complétez le fichier `Base.java`.

Vous pouvez ensuite finir l'application de telle sorte que l'anagramme proposé à l'utilisateur varie.