Université Paris Diderot - Paris 7
Master 1 Informatique, Programmation de composants mobiles
20 juin 2018, durée : 2h15

Documents et matériel autorisés :

- tous les documents papier sauf livres,
- documentation android `https://developer.android.com/index.html`,
- documentation java `https://docs.oracle.com/javase/7/docs/api/index.html`
- tous les documents sur moodle du cours,
- vos propres programmes de TP et votre projet,
- vos appareils android pour tester vos programmes.

Il est strictement **interdit** d'utiliser d'autres adresses WWW que mentionnées dessus, en particulier :

- il est interdit d'utiliser google ou n'importe quel autre moteur de recherche (vous devez aller directement sur la page de la documentation d'android et faire les recherches dans la documentation depuis le site d'android , sans passer par google),
- d'utiliser les forums de développeurs,
- utiliser l'ordinateur, tablettes, smartphones à d'autres fins que le développement et le test de vos programmes.

Le sujet comporte 4 pages.

Consignes

Pour chacune de deux applications vous devez compresser le répertoire **VotreAppli/app/src/-main** qui contient les sources de votre programme et les fichiers layout :

```
tar cvf VotreNom_Application_i.tar  VotreAppli/app/src/main
gzip VotreNom_Application_i.tar
```

où i sera remplacé par le numéro de l'application et `VotreNom` par votre nom.
Déposez le fichier `tar.gz` qui contient la solution sur moodle (cours Android) dans le dépôt dans la section Examen Juin 2018. Vous pouvez ajouter un fichier txt pour expliquer quelles questions sont implémentées.
J'accepte aussi les archives zip. **Par d'archives rar.** Il est prudent de sauvegarder au moins la dernière version qui marche avant de passer à la question suivante, sinon vous risquez de vous retrouver avec un programme où rien ne marche plus.

Avant de commencer les exercices récupérez sur moodle (section Examen Juin 2018) les fichiers layout pour toutes les activités et le(s) fichier(s) *.java.
En ce qui concerne les fichiers layout récupérés sur moodle il vous faut adapter au moins la ligne `tools:context` pour qu'elle soit conforme à votre projet. Vous pouvez modifier à votre guise tous les fichiers qu'on vous fournit (ou même fabriquer vos propres layout).

Dans les fichiers `java` fournis pour la dernière application (la bases de données et Content-Provider) vous pouvez faire vos modifications tant que cela ne change pas la structure de la table. (Mais à priori je ne vois aucune raison de modifier les fichiers `*.java` fournis sur moodle.)

Vos programmes seront testés avec un logiciel de détection de plagiat.
Si vous travaillez sur les machines de l'ufr c'est à vous de protéger votre répertoire de travail contre les visites non autorisées.

Application 1

Le but de l'application est de développer un programme qui aidera à apprendre des langues étrangères, ce sera juste un prototype très incomplet.

Exercice 1

L'application est composée de deux activités.
L'activité principale est l'activité `Choisir` dont le fichier layout est `activity_choisir.xml`.
L'activité `Choisir` est composée de : un `Spinner`, deux `RadioButtons fr` et `eng` groupés dans un `RadioGroup`, un `Button button`.
Le contenu de `Spinner` est alimenté à l'aide de deux tableaux dont le contenu se trouve dans le fichier ressources `array.xml` Les deux tableaux sont de la même longueur, le premier tableau contient les mots en anglais, le deuxième les mots en français de telle sorte que les mots à la même position dans les deux tableaux désignent la même couleur.
Au lancement de l'activité c'est le `RadioButton fr` qui est automatiquement sélectionné. Cela implique qu'initialement le `Spinner` est alimenté avec les mots français.
A chaque moment quand l'utilisateur change la sélection de `RadioButton` le `Spinner` doit être alimenté avec le tableau qui correspond à la langue sélectionnée. Implémentez cette fonctionnalité.

Indication 1. RadioGroup.OnCheckedChangeListener sur `RadioGroup` (méthode `setOnCheckedChangeListener`).
Android active ce listener chaque fois quand l'utilisateur sélectionne un nouveau `RadioButton` appartenant à ce groupe. L'unique méthode du listener possède deux paramètres, le deuxième paramètre fournit le **id** du `RadioButton` sélectionné. (Comme toujours les ids de deux `RadioButtons` sont donnés dans le fichier `activity_choisir.xml` donc il suffit de comparer le deuxième paramètre de la méthode avec `R.id.nom_id` approprié).

Indication 2. Pour alimenter un `Spinner` on procède comme pour alimenter une `ListView`, c'est-à-dire en utilisant un `ArrayAdapter`. Le fichier `ligne.xml` est un fichier layout pour une ligne du `Spinner`.

Exercice 2 Dans l'activité `Choisir` ajouter le code à exécuter quand l'utilisateur clique sur le bouton `ENVOYER`.

En appuyant sur le bouton l'utilisateur lance une nouvelle activité `Comparer` dont le layout est `activity_comparer.xml`.
Dans `Intent` qui lance la nouvelle activité il faut faire passer deux extras :

- le premier extra, dont la clé est `mot`, est un entier – l'indice de l'élément actuellement choisi dans le `Spinner`,
- le deuxième extra, dont la clé est `lang`, est un `String` qui indique la langue du mot sélectionné dans le `Spinner`.

Le code de la nouvelle activité est à écrire dans l'exercice suivant, ici vous pouvez utiliser l'activité `Comparer` vide.

Exercice 3 Il reste à écrire le code de l'activité `Comparer`.
L'activité `Comparer` est composé d'un `Spinner`, d'un `TextField` initialement vide, et d'un bouton `VERIFIER` Dans le `Spinner` on affiche les mots dans la langue différente de celle indiquée dans l'extra `lang` de l'intent qui a lancé cette activité.
L'utilisateur choisit un mot dans le Spinner et appuie sur le bouton.
Si ce mot est une traduction correcte du mot choisi auparavant dans l'activité `Choisir`, on affiche BRAVO dans le TextField. Par contre si le mot sélectionné dans l'activité `Comparer` ne correspond pas au mot sélectionné dans l'activité `Choisir` on affichera RATE dans le `TextField`.

Application 2

Les exercice suivants développent une nouvelle application dont le but est similaire que dans l'application précédente mais cette fois-ci nous utiliseront une base de donnée.
Cette BD contient une seule table qui possède deux colonnes, fr et eng. Chaque ligne de la table donne un couple de mots, un mots anglais et un mot français.
La base de données est implémentée par la classe `DicoDB` dont le code vous pouvez récupérer dans le fichier `DicoDB.java`. Cette classe est dérivée de `SQLiteOpenHelper`. Pour simplifier les choses on ne demande pas d'utiliser `ContentProvider`, vous pouvez faire directement sur la BD. Pour cela il suffit de créer l'objet `DicoBD` et récupérer `SQLiteOpenHelper` (dicoBD.getReadable/WritableDatabase()). **Exercice 4**

L'activité principale de l'application contient trois boutons et un `EditText`. Le fichier layout : `activity_aprentissage.xml`. Les trois boutons de l'activité principale servent à lancer trois nouvelles activités, respectivement : `Ajouter`, `Afficher` et `Verifier`.

Dans cet exercice on demande d'implémenter l'activité `Ajouter` lié au bouton `Ajouter` de l'activité principale. Le but de l'activité `Ajouter` est d'alimenter la base de données.
L'activité `Ajouter` est composée de deux `EditText` et d'un bouton, le fichier layout `activity_ajouter.xml`.
L'utilisateur met un couple de mots, un mot français l'autre anglais, dans les deux `EditText` et appuie sur le bouton. Les deux mots seront ajoutés dans la table de la BD. Après avoir ajouté le couple de mots dans la BD le contenu de deux `EditText` sera effacé.

Exercice 5 L'activité `Afficher` affiche le contenu de la BD dans une liste `ListView` de couples de mots, le fichier layout `content_afficher.xml`
Le layout `bd.xml` c'est le layout d'une ligne d'affichage avec Adapter.

Rappel. Quand vous faites la requête SELECT sur un table il ne faut pas oublier la colonne `_id` qui est exigée par le curseur, donc il y aura trois colonnes : deux colonnes pour les mots anglais et français et la troisième `rowid as _id`
indication : `SimpleCursorAdapter`.

Exercice 6 Le troisième bouton de l'activité principale est accompagné d'un `EditText`
L'utilisateur entre un mot en français dans `EditText`, appuie sur le bouton et lance la dernière activité `Verifier`. Le mot fournit par l'utilisateur est envoyé comme un extra dans l'intent qui lance l'activité `Verifier` dont le layout est `activity_verifier.xml`
L'activité `Verifier` est composée de : un `TextView`, un `EditText` et un bouton.
Dans `TextView` on affichera le mot français envoyé dans l'intent.
Ensuite l'utilisateur entre un mot anglais dans `EditText`, appuie sur le bouton et l'activité interroge la BD pour vérifier si le mot anglais correspond bien au mot français.
Si c'est le cas on affiche un toast `BRAVO` sinon un toast `RATE`.
Après l'affichage de Toast on vide `EditText` pour que l'utilisateur puisse donner une autre réponse.