SQLite pour Android

On veut, dans ce TP, compléter l'application Android par une base de données SQLite contenant la liste des stations vélibs. Plus précisément, lorsque les stations vélibs sont obtenues, on veut les stocker en interne sur le téléphone dans une BD SQLite.

Construction de la base de données

1°) Définir un fichier de constantes (= une interface) contenant :

```
public static final int BASE_VERSION = 1;
public static final String BASE_NOM = "velibs.db";

public static final String TABLE_DES_VELIBS = "table_des_velibs";

public static final String COLONNE_ID = "id";
public static final String COLONNE_NUMERO = "numero";
public static final String COLONNE_NOM = "nom";
public static final String COLONNE_LATITUDE = "latitude";
public static final String COLONNE_LONGITUDE = "longitude";
```

2°) Construire un "Database Open Helper" permettant de gérer l'ouverture de la base de données. On utilisera les constantes de la question précédente. On pourra (ce n'est pas obligatoire) utiliser les `import static`.
Le début de cette classe est :

```
public class MaBaseOpenHelper extends SQLiteOpenHelper {

    private static final String REQUETE_CREATION_TABLE = "create table "
            + TABLE_DES_VELIBS
            + " (" + COLONNE_ID + " integer primary key autoincrement, "
            + COLONNE_NOM + " text not null, "
            + COLONNE_NUMERO + " text not null, "
            + COLONNE_LATITUDE + " text not null, "
            + COLONNE_LONGITUDE + " text not null);";
```

Compléter cette classe de sorte à créer la table `TABLE_DES_VELIBS` de la base de données avec les 3 méthodes fondamentales d'un `SQLiteOpenHelper`.

On a déjà la classe `StationVelib` qui modélise une … station vélib.

3°) Créer une classe DAO, appelée `VelibDB_DAO`, avec les méthodes appropriées de sorte à :
a) Construire et repérer le `SQLiteOpenHelper` lorsque le DAO est créé.
b) ouvrir la base de données en lecture-écriture
c) refermer cette base de données
d) pouvoir insérer une `StationVelib` dans cette base
e) pouvoir, ayant une `ArrayList<StationVelib>`, mettre (insérer !) toutes les stations vélibs de cette `ArrayList` dans cette base.
f) récupérer toutes les stations vélibs de la base à l'aide de la méthode :

```
public ArrayList<StationVelib> getToutesLesStationsVelib(){ … }
```

g) vider la table des stations vélibs de cette base de données à l'aide d'une méthode `videLaTableDesStationsVelibsBase()`.

Cette classe `VelibDB_DAO` doit aussi avoir une donnée

```
private int nbStationsVelib;
```
indiquant le nombre de stations vélibs connu par ce DAO ainsi que sa méthode `get` associée.

Utilisation de cette base de données SQLite

Dans toute l'app, on veut utiliser si possible le même DAO. Il est alors bon de construire une classe qui va repérer, à l'aide de ses données membres références statiques ce DAO.

4°) Construire donc la classe :

```
public class LaClasseDeDonneesStatiques {
    private static VelibDB_DAO leDao;

    public static VelibDB_DAO getLeDao() {
        return leDao;
    }

    public static void setLeDao(VelibDB_DAO leDao) {
        LaClasseDeDonneesStatiques.leDao = leDao;
    }
}
```

(cf. `http://stackoverflow.com/questions/4878159/whats-the-best-way-to-share-data-between-activities`)

On va construire cette base de données lorsqu'on a obtenu toutes les stations vélibs. On veut faire cette construction ailleurs que dans la UI Thread.

5°) Indiquer dans quelle partie de l'application Android vous allez faire cette construction.

6°) Plus précisément le code suit l'algorithme suivant :
on récupère le DAO repéré à l'aide de `LaClasseDeDonneesStatiques`,

```
si ce DAO n'existe pas ou n'a pas de données
      le construire
      l'ouvrir
      insérer toutes les stations vélibs qu'on a déjà récupérées
      positionner la référence statique de la classe
            LaClasseDeDonneesStatiques qui repère ce DAO
fin si
```

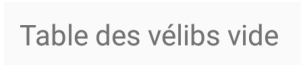
Ainsi, à l'issue de cette partie, vous avez créé une base de données SQLite contenant les stations vélibs.
On veut maintenant la manipuler c'est-à-dire afficher son contenu, la vider, la reconstruire, etc.

Manipulation de cette base de données SQLite

7°) Dans la `ListView` de la première activité (`MainActivity`), ajouter deux items "Montrer le contenu de la table des vélibs" et "Vider la table des vélibs" :

8°) Lorsque l'utilisateur choisi l'item "Montrer …", une activité est lancée et se charge d'afficher le contenu de la table des stations vélibs dans une TableLayout si celle-ci a été déjà créée. On peut avoir :

ou bien



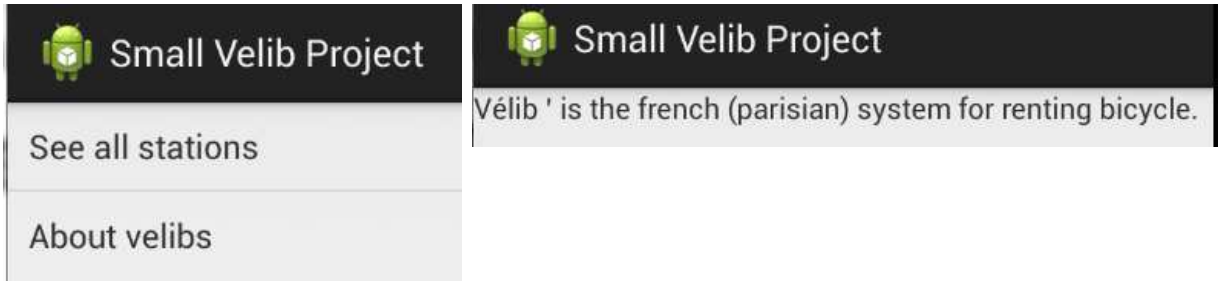
si cette table est vide.

9°) Lorsque l'utilisateur choisi l'item "Vider …", la table des stations vélibs est vidée (DROP …), et un `Toast` est affiché pour confirmation.

Des Bonus

Internationalisation

1°) Configurer l'application de sorte à avoir une version de votre projet en anglais, les noms et adresses des stations restent en français :



2°) Configurer votre émulateur pour revenir à l'IHM en français.

Rangement dans les paquetages

Notre application devient riche en classes et elles sont toutes dans le même paquetage : ce n'est pas raisonnable !
Créer des paquetages qui regroupe les activity (`xxx.activity`), un paquetage pour les classes métiers (`xxx.business`), pour des classes d'aide (`xxx.helpers`).
Remarquer que le fichier `AndroidManifest.xml` a été mis à jour même après changement de paquetage.

ORM (Object-Relational Mapping)

On a manipulé les bases de données SQLite sur Android comme indiqué par `https://developer.android.com/training/basics/data-storage/databases.html`. C'est très bien !
On peut aussi utiliser une bibliothèque de "mapping objet relationnel". Faire une nouvelle version de traitement de base de données SQLite en Android avec un ORM. On pourra utiliser OrmLite, SugarORM, etc.