SQLite pour Android

On veut, dans ce TP, construire une application Android qui permet de gérer des contacts (essentiellement un nom associé à un numéro de téléphone). Ces contacts seront mis dans une base de données Android gérée par SQLite.

Construction de la base de données

1°) Définir une classe `Contact` contenant les informations :

```
public class Contact {
    private int _id;
    private String nom;
    private String numTelephone;
       ...
}
```

Compléter cette classe avec des accesseurs et des constructeurs appropriés (à la souris sous Eclipse !)

2°) Construire un "Database Helper" permettant de gérer une base de données. Le début de cette classe est :

```
public class LeDatabaseHandler extends SQLiteOpenHelper {
      // All Static variables
      // Database Version
      private static final int DATABASE_VERSION = 1;

      // Database Name
      private static final String DATABASE_NAME = "contactsManager";

      // Contacts table name
      private static final String TABLE_CONTACTS = "contacts";

      // Contacts Table Columns names
      private static final String KEY_ID = "id";
      private static final String KEY_NAME = "name";
      private static final String KEY_PH_NO = "phone_number";
      // A compléter ...
}
```

3°) Compléter cette classe de sorte à :
a) créer la base de données
b) pouvoir insérer des `Contact`s dans cette base
c) récupérer tous les contacts de la base à l'aide de la méthode :

```
      public List<Contact> getAllContacts() { ... }
```

4°) Manipuler cette base à l'aide d'une activité principale possédant la méthode :

```
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      LeDatabaseHandler db = new LeDatabaseHandler(this);

      /*
```

```
         * Operation CRUD
         */
        Log.d("JMF", "Insertion de Contact");
        db.addContact(new Contact("Jo", "9100000000"));
        db.addContact(new Contact("Jack", "9199999999"));
        db.addContact(new Contact("William", "9522222222"));
        db.addContact(new Contact("Averel", "9533333333"));

        // Reading all contacts
        Log.d("JMF", "Lecture des Contacts");
        List<Contact> contacts = db.getAllContacts();

        for (Contact cn : contacts) {
              String log = "Id: "+cn.getID()+" ,Name: " + cn.getName() + " ,Phone: "
+ cn.getPhoneNumber();
              // Writing Contacts to log
              Log.d("JMF", log);
        }
}
```

Cette partie est grandement inspiré de l'article de Ravi Tamada à `http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/`

Voici une partie additionnelle permettant de mieux manipuler la base de données.

Une IHM pour manipuler une base de données

5°) Définir une activity qui fait afficher :

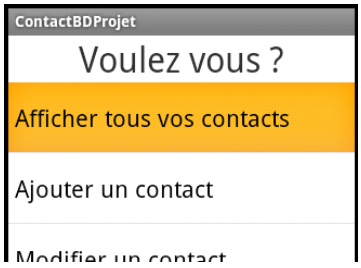
il s'agit d'une `TextView` ("Voulez vous ?") suivi d'une `ListView` à 4 items.

6°) Ajouter le code de sorte que lorsque l'utilisateur clique sur l'item "Initialisation de la base !", la base (table) est recréée avec 4 `Contacts`. On rappelle que le code de gestion d'une `ListView` est :

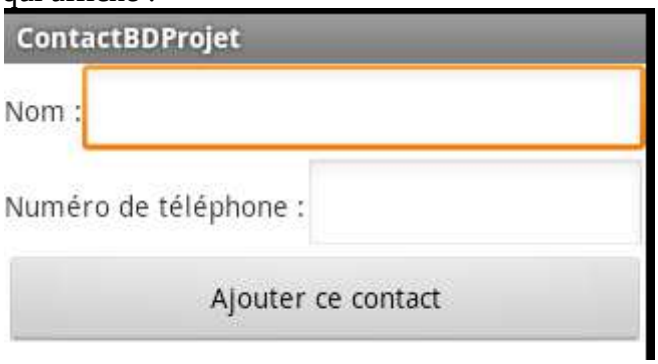
```
lv.setOnItemClickListener(new OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent, View view,
                  int position, long id) {
            if (position == 0) {
            // traitement si l'utilisateur a choisi le premier item de la ListView
            } else { ...
            }
      }
});
```

où `lv` est la `ListView`.

7°) De même, écrivez le code qui affiche tous les contacts lorsque l'utilisateur a choisi le premier item. On pourra utiliser un `TableLayout`.

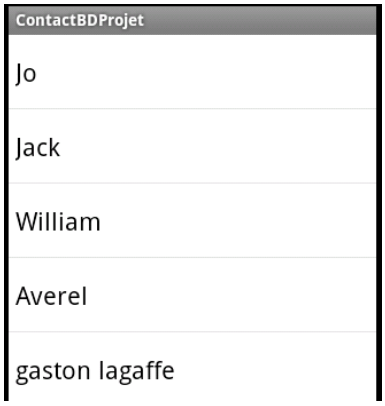


8°) Ecrire une activity qui affiche :

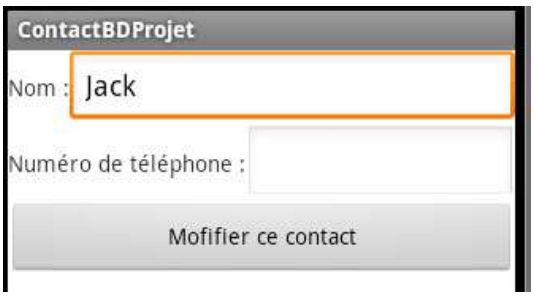


Cette activité est lancée lorsque l'utilisateur sélectionne l'item "Ajouter un contact".
Ecrire le code permettant d'ajouter un `Contact` dans la base de données grâce à cette interface.

9°) Ecrire l'activité qui présente tous les contacts dans une `ListView` pour pouvoir en modifier. L'enchaînement des activités doit être :



Par la suite, lorsque l'utilisateur choisit un des contacts, une nouvelle activité est affichée, initialisée par le nom du `Contact` à modifier. L'enchaînement des activités peut être :



Il faudra donc passer le `Contact` d'une activité à une autre. Pour cela on utilisera le code (peut être à adapter) :

```
Contact ctAModifier = arContacts.get(position);
Intent i = new Intent(getApplicationContext(), ContactAModifierActivity.class);
Bundle b = new Bundle();
b.putSerializable(Constants.CONTACT, ctAModifier);
i.putExtras(b);
startActivity(i);
```

Pour récupérer ce Contact dans l'activité destinataire on peut écrire le code :

```
Bundle b = this.getIntent().getExtras();
if (b != null) {
    ct = (Contact)(b.getSerializable(Constants.CONTACT));
}
```

10°) Enrichir l'activité qui présente tous les contacts dans une `ListView` pour pouvoir en détruire le contact choisi par l'utilisateur. On pourra construire de nouvelles activités ou enrichir celles déjà développées.

Conclusion

On a bien mis en place toutes les opérations CRUD, Create (question 8 et 6), Read (question 7), Update (question 9), Delete (question 10)