Lab (part 2) - Database App SQL & Log

Revision 01

# Create Project

Create a new Android project with an Empty Activity template, with the project name dblog and an *activity* called *MainActivity*.

Note that this app will have no UI. It will run in the logcat window using log statements only.

## Database and Data Model

### Create the record class

Firstly create a class in the same package as MainActivity called Contact. The Code is given below. It contains a row with a primary key and two text fields, name and phone number.  
  
*/\*  
 \* This class contains the getters and setters for the row being used in the table  
 \*/***public class** Contact {  
 **int \_id**;  
 String **\_name**;  
 String **\_phone\_number**;  
  
 **public** Contact(){ }  
 **public** Contact(**int** id, String name, String \_phone\_number){  
 **this**.**\_id** = id;  
 **this**.**\_name** = name;  
 **this**.**\_phone\_number** = \_phone\_number;  
 }  
  
 **public** Contact(String name, String \_phone\_number){  
 **this**.**\_name** = name;  
 **this**.**\_phone\_number** = \_phone\_number;  
 }  
 **public int** getID(){  
 **return this**.**\_id**;  
 }  
  
 **public void** setID(**int** id){  
 **this**.**\_id** = id;  
 }  
  
 **public** String getName(){  
 **return this**.**\_name**;  
 }  
  
 **public void** setName(String name){  
 **this**.**\_name** = name;  
 }  
  
 **public** String getPhoneNumber(){  
 **return this**.**\_phone\_number**;  
 }  
  
 **public void** setPhoneNumber(String phone\_number){  
 **this**.**\_phone\_number** = phone\_number;  
 }  
}

### Create the database helper class

Second, create the helper class which is called DatabaseHandler.java. This is created in the same level as the other two classes in the project. In this example, the code in this class performs both the Table and record CRUD operations. The code is given below:  
  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
  
**public class** DatabaseHandler **extends** SQLiteOpenHelper {  
 **private static final int *DATABASE\_VERSION*** = 1;  
 **private static final** String ***DATABASE\_NAME*** = **"contactsManager"**;  
 **private static final** String ***TABLE\_CONTACTS*** = **"contacts"**;  
 **private static final** String ***KEY\_ID*** = **"id"**;  
 **private static final** String ***KEY\_NAME*** = **"name"**;  
 **private static final** String ***KEY\_PH\_NO*** = **"phone\_number"**;  
  
 **public** DatabaseHandler(Context context) {  
 **super**(context, ***DATABASE\_NAME***, **null**, ***DATABASE\_VERSION***);  
 *//3rd argument to be passed is CursorFactory instance* }  
  
 *// Creating Tables* @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 String CREATE\_CONTACTS\_TABLE = **"CREATE TABLE "** + ***TABLE\_CONTACTS*** + **"("** + ***KEY\_ID*** + **" INTEGER PRIMARY KEY,"** + ***KEY\_NAME*** + **" TEXT,"** + ***KEY\_PH\_NO*** + **" TEXT"** + **")"**;  
 db.execSQL(CREATE\_CONTACTS\_TABLE);  
 }  
  
 *// Upgrading database* @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
 *// Drop older table if existed* db.execSQL(**"DROP TABLE IF EXISTS "** + ***TABLE\_CONTACTS***);  
  
 *// Create tables again* onCreate(db);  
 }  
  
 **public void** emptyContacts() {  
 *// Drop older table if existed* SQLiteDatabase db = **this**.getWritableDatabase();  
 db.execSQL(**"DROP TABLE IF EXISTS "** + ***TABLE\_CONTACTS***);  
  
 *// Create tables again* onCreate(db);  
 }  
 *// code to add the new contact* **void** addContact(Contact contact) {  
 SQLiteDatabase db = **this**.getWritableDatabase();  
  
 ContentValues values = **new** ContentValues();  
 values.put(***KEY\_NAME***, contact.getName()); *// Contact Name* values.put(***KEY\_PH\_NO***, contact.getPhoneNumber()); *// Contact Phone  
  
 // Inserting Row* db.insert(***TABLE\_CONTACTS***, **null**, values);  
 *//2nd argument is String containing nullColumnHack* db.close(); *// Closing database connection* }  
  
 *// code to get the single contact* Contact getContact(**int** id) {  
 SQLiteDatabase db = **this**.getReadableDatabase();  
  
 Cursor cursor = db.query(***TABLE\_CONTACTS***, **new** String[] { ***KEY\_ID***,  
 ***KEY\_NAME***, ***KEY\_PH\_NO*** }, ***KEY\_ID*** + **"=?"**,  
 **new** String[] { String.*valueOf*(id) }, **null**, **null**, **null**, **null**);  
 **if** (cursor != **null**)  
 cursor.moveToFirst();  
  
 Contact contact = **new** Contact(Integer.*parseInt*(cursor.getString(0)),  
 cursor.getString(1), cursor.getString(2));  
 *// return contact* **return** contact;  
 }  
  
 *// code to get all contacts in a list view* **public** List<Contact> getAllContacts() {  
 List<Contact> contactList = **new** ArrayList<Contact>();  
 *// Select All Query* String selectQuery = **"SELECT \* FROM "** + ***TABLE\_CONTACTS***;  
  
 SQLiteDatabase db = **this**.getWritableDatabase();  
 Cursor cursor = db.rawQuery(selectQuery, **null**);  
  
 *// looping through all rows and adding to list* **if** (cursor.moveToFirst()) {  
 **do** {  
 Contact contact = **new** Contact();  
 contact.setID(Integer.*parseInt*(cursor.getString(0)));  
 contact.setName(cursor.getString(1));  
 contact.setPhoneNumber(cursor.getString(2));  
 *// Adding contact to list* contactList.add(contact);  
 } **while** (cursor.moveToNext());  
 }  
  
 *// return contact list* **return** contactList;  
 }  
  
 *// code to update the single contact* **public int** updateContact(Contact contact) {  
 SQLiteDatabase db = **this**.getWritableDatabase();  
  
 ContentValues values = **new** ContentValues();  
 values.put(***KEY\_NAME***, contact.getName());  
 values.put(***KEY\_PH\_NO***, contact.getPhoneNumber());  
  
 *// updating row* **return** db.update(***TABLE\_CONTACTS***, values, ***KEY\_ID*** + **" = ?"**,  
 **new** String[] { String.*valueOf*(contact.getID()) });  
 }  
  
 *// Deleting single contact* **public void** deleteContact(Contact contact) {  
 SQLiteDatabase db = **this**.getWritableDatabase();  
 db.delete(***TABLE\_CONTACTS***, ***KEY\_ID*** + **" = ?"**,  
 **new** String[] { String.*valueOf*(contact.getID()) });  
 db.close();  
 }  
  
 *// Getting contacts Count* **public int** getContactsCount() {  
 **int** count = 0;  
 String countQuery = **"SELECT \* FROM "** + ***TABLE\_CONTACTS***;  
 SQLiteDatabase db = **this**.getReadableDatabase();  
 Cursor cursor = db.rawQuery(countQuery, **null**);  
 count = cursor.getCount();  
 cursor.close();  
  
 *// return count* **return** count;  
 }  
  
}

### Add the MainActivity working code

Finally, we add the code for the main activity. Note that the output for this program will be in the logcat area of Android Studio. Set the log level to ‘info’ to see the messages clearer.

MainActivity.java  
  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.os.Bundle;  
**import** android.util.Log;  
  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 DatabaseHandler db = **new** DatabaseHandler(**this**);  
  
 *//db.emptyContacts(); // empty table if required  
  
 // Inserting Contacts* Log.*i*(**"Insert: "**, **"Inserting .."**);  
 db.addContact(**new** Contact(**"Joe"**, **"0873456789"**));  
 db.addContact(**new** Contact(**"Mary"**, **"0863111122"**));  
 db.addContact(**new** Contact(**"Jack"**, **"0859999888"**));  
 db.addContact(**new** Contact(**"Andrew"**, **"083334444"**));  
 db.addContact(**new** Contact(**"Harold"**, **"0831112222"**));  
 db.addContact(**new** Contact(**"Joe"**, **"0835554444"**));  
  
 *// Reading all contacts* Log.*i*(**"Reading: "**, **"Reading all 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.*i*(**"Name: "**, log);  
 }  
  
 Log.*i*(**"divider"**, **"===================="**);  
  
 Contact singleUser = db.getContact(5);  
 Log.*i*(**"contact 5 is "**, singleUser.getName());  
  
 Log.*i*(**"divider"**, **"===================="**);  
  
 *// Calling SQL statement* **int** userCount = db.getContactsCount();  
 Log.*i*(**"User count: "**, String.*valueOf*(userCount));  
  
 }  
}

## Finished Project View

Graphical user interface, text, application, chat or text message

Description automatically generatedYou should have three java files all of the same package, which will be different to below.

Figure - Project View

## Running the App

Build and run the app on the emulator. You should see the following output in the logcat section:

Graphical user interface, text

Description automatically generated

Figure Example of run of program in logcat

If you examine the code in the MainActivity, you will see the calls made to insert the contacts, list all the contacts, and list contact number 5 and finally get a count of all the records (using SQL).

Note the db.rawquery() command is used in DatabaseHelper class to implement the SQL query.

## Final part of lab

In the final part of this lab, you are asked to add the SQL to print out all the Contacts where the name is like “Joe”.

Add the code to the MainActivity and the DatabaseHelper class to achieve this.

For your lab report, simply add the code changes for the above two classes, not the whole program.