Lab 4: SQLite

Submission Due: 03/02/2021

Objective

1. Learn to create a Student database
2. Learn how to perform CRUD operation

Before Lab

1. Download the lab4 starter code from Camino (Files -> Lab -> Lab4_StarterCode).
2. Unzip and open the Lab4_StarterCode project in Android Studio.

Lab4_StarterCode

Lab4_StarterCode contains MainActivity and several fragments. The focus of this lab session is on SQLite. Hence Fragment will not be discussed.

MainActivity contains code that initializes layout to display HomeFragment on activity creation. It also implements OnFragmentInteractionListener – an interface provided by HomeFragment to replace Fragments based on the user interaction (Button clicks).

HomeFragment contains buttons to perform CRUD operations on the Student database and implements Button Click event listeners for these buttons -
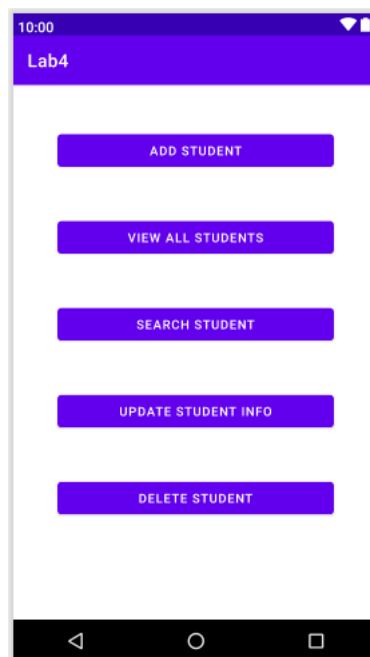
For fragment overview, refer to https://developer.android.com/guide/fragments and class notes.

App Description & Tasks

In this lab, you are going to define database schema by creating a contract class, implement methods that create and maintain the Student database and tables. You will also add code

- to insert data into the database
- to read from the database
- to delete rows from the database
- to modify data in the database
- to search from the database

You need to complete the following tasks:

Task 1: Define Schema by creating Contract Class

1. Create Contract class called '**StudentInfoContract'**
   a. Project -> New -> Java Class -> Class
   b. Set Class name – StudentInfoContract
   c. Make the class- final class by adding the *final* keyword.
2. To prevent someone from accidentally instantiating the contract class, implement an empty constructor and make the constructor private.
3. For each table in the database, an inner class must be added within Contract Class.
   Let Student Database contain a single table called **StudentsInfo** with the following schema-
   <p align="center">StudentsInfo(_ID, student_id, name, email)</p>

   Corresponding code:

```java
public static class Students implements BaseColumns{
    public static final String TABLE_NAME = "StudentsInfo";
    public static final String STUDENT_ID = "student_id";
    public static final String STUDENT_NAME = "name";
    public static final String STUDENT_EMAIL = "email";
}
```

   Note: By implementing the BaseColumns interface, an inner class can inherit a primary key field called _ID that some Android classes such as CursorAdapter expect it to have.

Task 2: Create a database using an SQL helper

1. Create DB Helper class called **StudentDbHelper** a subclass of SQLiteOpenHelper
   a. Project -> New -> JAVA class -> class
   b. Set Class Name – StudentDbHelper
   c. Make it a subclass of SQLiteOpenHelper
      (Add extends SQLiteOpenHelper)

2. Implement constructor as shown

```
public StudentDbHelper(Context context){
    super(context,DATABASE_NAME,null,DATABASE_VERSION);
}
```

Note: DATABASE_NAME, DATABASE_VERSION are private static final instance variables with values "students.db" (string) and 1 (int) respectively.

3. Add the below statements that create and delete the StudentsInfo table
   a. Create StudentsInfo Table-

```
CREATE TABLE StudentsInfo(_id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT NOT NULL, student_id TEXT NOT NULL,
email TEXT);
```

Corresponding code:

```
private static final String CREATE_TABLE = "CREATE TABLE " +
StudentInfoContract.Students.TABLE_NAME + "(" +
StudentInfoContract.Students._ID  + " INTEGER PRIMARY KEY
AUTOINCREMENT," +
StudentInfoContract.Students.STUDENT_NAME + " TEXT NOT NULL, " +
StudentInfoContract.Students.STUDENT_ID + " TEXT NOT NULL, " +
StudentInfoContract.Students.STUDENT_EMAIL + " TEXT" + ")";
```

   b. Delete StudentsInfo Table-

```
DROP TABLE IF EXISTS StudentsInfo;
```

Corresponding code:

```
private static final String DROP_TABLE = "DROP TABLE IF EXISTS "+
StudentInfoContract.Students.TABLE_NAME;
```

4. Override the onCreate() and onUpgrade() callback methods of SQLiteOpenHelper as shown-

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    //create all tables
    sqLiteDatabase.execSQL(CREATE_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int
i1) {
    //drop existing tables
    sqLiteDatabase.execSQL(DROP_TABLE);
    //start over
    onCreate(sqLiteDatabase);
}
```

Task 3: Insert data into the database

When a user clicks **ADD STUDENT** Button in the HomeFragment, AddFragment will be visible to the user where the user can enter the id, name, and email of new student and click on Button **SAVE.** You need to complete the implementation of this button click event listener method-saveStudentInfoIntoDatabase to insert data into the Student database.
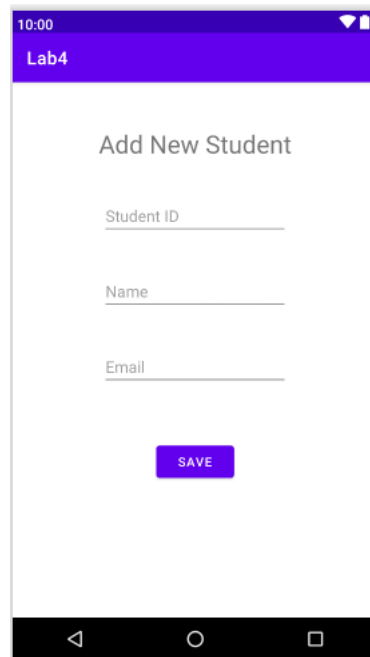
1. To access database, instantiate StudentDbHelper
   ```
   StudentDbHelper dbHelper = new StudentDbHelper(getActivity());
   ```
2. Get the data repository in write mode
   ```
   SQLiteDatabase db = dbHelper.getWritableDatabase();
   ```
3. Create ContentValues object
   ```
   ContentValues contentValues = new ContentValues();
   ```
4. Use put() method to insert data where column names of the table are keys and user entered data are values.
   ```
   contentValues.put(StudentInfoContract.Students.STUDENT_ID, id);
   ```
   (Similarly, add name and email)
5. Insert the new row by calling db.insert()and assign the return value (primary key value of the new row) to **recordId**
   ```
   long recordId =
   db.insert(StudentInfoContract.Students.TABLE_NAME,null,contentVal
   ues);
   ```
6. Close the database connection using **db.close()**
7. Check whether insertion is successful or not
   a. If recordId is -1, add a Toast message to show "Insertion failed"
   b. Else, add a Toast message to show "Successfully added student <name> in the db"
8. Clear the edit text fields using setText()
   ```
   et_std_id.setText("");
   ```
   (Similarly, clear name and email)

Task 4: Retrieve information from the database

When a user clicks **VIEW ALL STUDENTS** Button in the HomeFragment, HomeFragment will be replaced by ViewAllFragment where all the students' info must be displayed.

1. Instantiate StudentDbHelper (refer task 3)
2. Get the data repository in read mode (refer task 3 but call getReadableDatabase())
3. Use the query() method, passing it selection criteria and desired columns (projection) for the SQL command –
   `SELECT * FROM StudentsInfo ORDER BY student_id;`

   Corresponding code:

```java
Cursor cursor = db.query(StudentInfoContract.Students.TABLE_NAME, columns: null,
        selection: null, selectionArgs: null, groupBy: null, having: null,
        StudentInfoContract.Students.STUDENT_ID);
```

4. Results of the query are returned as a Cursor object which can be used to retrieve the data from the query result.

```java
String result = "";
while(cursor.moveToNext()){
    String id =
cursor.getString(cursor.getColumnIndex(StudentInfoContract.Studen
ts.STUDENT_ID));
    String name =
cursor.getString(cursor.getColumnIndex(StudentInfoContract.Studen
ts.STUDENT_NAME));
    String email =
cursor.getString(cursor.getColumnIndex(StudentInfoContract.Studen
ts.STUDENT_EMAIL));

    //append this record to result
    result = result + "\n\nID: "+id+"\nNAME: "+name+"\nEMAIL:
"+email;
}
```
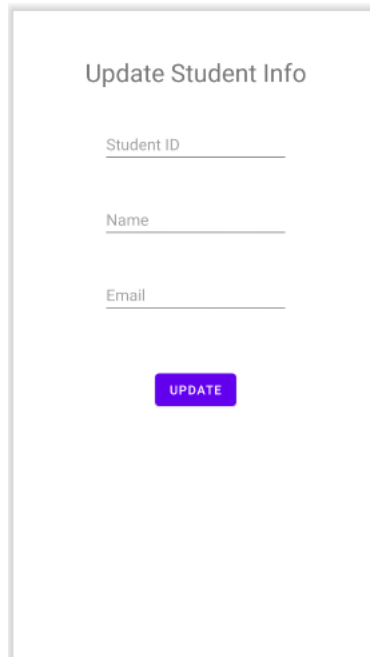
5. Close the database connection using **db.close()**
6. Check whether the result is empty. If empty, set result= "No records Found"
7. Set TextView's text=result.

```java
tv_all_student.setText(result);
```

Task 5: Update records in the database

When a user clicks **UPDATE STUDENT INFO** Button in the HomeFragment, UpdateFragment will be visible to the user where the user can enter id, name, and email of the student and click on Button **UPDATE.** You need to complete the implementation of this button click event listener method- **updateStudentInfo()** which updates the student info of the students whose names **match** the user entered name.

```
Update Student Info

    Student ID _____

    Name _____

    Email _____


            UPDATE
```

1. Instantiate StudentDbHelper (refer task 3)
2. Get the data repository in write mode (refer task 3)
3. Create ContentValues object and add user-entered values using put() (refer task 3)
4. Using update() method to update the StudentsInfo table where column name = user entered name.
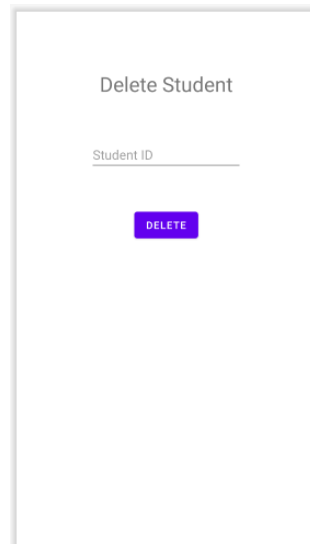
```java
String whereClause = StudentInfoContract.Students.STUDENT_NAME+"=?";
String[] whereArgs = {name};

int count =
db.update(StudentInfoContract.Students.TABLE_NAME,contentValues,wher
eClause,whereArgs);
```

5. Close the database connection using **db.close()**
6. Check whether any records were updated or not
     a. If count < 1, add Toast message to show "No records updated"
     b. Else, add toast message to show "Updated <count> records"
7. Clear all edit texts (refer task 3)

Task 6: Delete records in the database

When a user clicks on the **DELETE STUDENT** button in HomeFragment, HomeFragment will be replaced by DeleteFragment. Users can enter the student id of the student whose information should be deleted from the database. You need to implement the **deleteStudent**() which gets invoked when the user clicks on the **DELETE** button.

1. Instantiate StudentDbHelper (refer task 3)
2. Get the data repository in write mode (refer task 3)
3. Use delete() to delete the records where id = user-entered id
   (similar to update() in task 5 but where clause & args contain id instead of name)
4. Close database connection
5. Check count value which is the return value from delete() and toast appropriate message (refer task 5)
6. Clear edit text field.

Task 7: Search database

When a user clicks on the **SEARCH STUDENT** button in HomeFragment, SearchFragment will be visible to the user. Users can enter the name of the student and click on the **SEARCH** button. You need to implement searchStudent() method of the SearchFragment which displays all the records in the database that ***contains*** the user entered name.

1. Instantiate StudentDbHelper (refer task 3)
2. Get the data repository in read mode (refer task 3 but call getReadableDatabase())
3. Use the query() method, passing it selection criteria and desired columns (projection) for the SQL command –
   ```
   SELECT name,student_id FROM StudentsInfo WHERE name LIKE
   "%<entered_name>%" ORDER BY name;
   ```

   Corresponding code:

   ```
   String[] columns = {StudentInfoContract.Students.STUDENT_NAME,
           StudentInfoContract.Students.STUDENT_ID};
   String selection = StudentInfoContract.Students.STUDENT_NAME+ "
   LIKE? ";
   String[] selectionArgs = {"%"+name+"%"};

   Cursor cursor =
   db.query(StudentInfoContract.Students.TABLE_NAME,columns,selectio
   n,selectionArgs,null, null,
   StudentInfoContract.Students.STUDENT_NAME);
   ```

4. Iterate through records in the result using cursor object and append results to String **result** (Similar to task 4, but fetch only name and student_id from each record)
5. Close the database connection using **db.close()**
6. Check whether the result is empty, If empty, set result= "No records Found"
7. Set TextView's text=result.

Task 8: Execute & Observe

1. Run the app on an emulator or an Android device.
2. Test whether the application is working as expected (perform CRUD operations).

Deliverables

1. Final code (Format: zip)