Mobile Computing

Lab 4

1 Using an embedded database

1.1 Android Studio

1. In Android Studio, create a new Android Application Module called DatabaseExample. Be sure to change the domain to `example.com` or the package to `com.example.databaseexample`, and include an Empty Activity.

2. Next, download the zip file for this weeks lab from moodle (Lab 4 - Android) and extract it somewhere.

3. Copy the java files from the zip file into the `com.example.databaseexample` package, so that the `MainActivity` from the zip file overwrites the one that is already there.

4. Next, copy `activity_main.xml` into the `res/layout` folder.

2 Example Code

The app that is provided today is intended to teach you how to use local databases based on the SQLite database system. In your project, you will see a `DatabaseHelper` class, which provides an interface to android's database functions, and `MainActivity`, which is the activity demonstrating its use. Some key points of these classes are highlighted below

2.1 DatabaseHelper

- Look through the DatabaseHelper class, and search for the `onCreate` method. This method specifies what to do on a device which doesn't have the database yet (Perhaps because it is the first time the app is being run). Notice that in this method, we create the database and add some initial data using SQL.

- This class also provides `doQuery` and `doUpdate` methods, which are used to interact with the database. Note that these functions are provided purely to give you an example of how to use functions that query and update the database.

2.2 MyActivity

- `MainActivity` uses the `activity_main.xml` layout, which contains three buttons, each of which demonstrates something you need to know. In the Android alternative, there is no XML file, but the layout is set up the same way using the `createLayout` function.

- The List button, bound to the `doList` method, adds a `TextView` to the screen. It does this by creating a `TextView` and adding it to a layout that is already on the screen as it was added in `activity_main.xml` (or the `createLayout` function). Note that in order to do this, it must obtain a reference to the layout that is already on the screen. In Android,

it does this using `findViewById`, which references the id that is given to the layout in `activity_main.xml`. In the android alternative, it makes use of a global variable called output, which provides a global reference to the LinearLayout created in the `createLayout` function. This part of the code is extremely important. If you do not understand it, please pause here and ask questions so we can explain it better.

- The Query button, bound to the `doQuery` method, runs a query that selects all student names and ages from the database and prints them out, which means that they appear in the logcat view in Android Studio, or in the window running `run.bat` in the android alternative. It does this by running an SQL query which returns a Cursor object. The Cursor object contains the results of the query, and has a pointer indicating the current row of the Cursor. By calling `moveToNext()`, we move to the next row in the Cursor so that we can get the next set of data. The `getString` and `getLong` methods return the values of the indicated fields in the current row.

- The Insert button, bound to the `doInsert` method, adds a row to the table in the database. Note how the parameters are bound to the query. We put ? characters in the query itself and use the params array to bind values to the parameters.

3 Display query results

Change the `doList` method so that it displays all the students names and ages on the device's screen. You can get the names and ages in the same way that the `doQuery` method does, but instead of printing them out using `System.out.println`, create `TextView` objects with the appropriate text that you can add to the screen.

4 Insert items

Add an interface to the current screen allowing users to insert students into the system. You should add two EditTexts, with id's `nameInput` and `ageInput` respectively, as well as a Button with id `addButton` and text "Add". The user can then enter the student's name and age into `EditText` objects, and on clicking the Add button, the specified student should be inserted into the database.

5 Persistence

Note that the items remain in the database even after you quit the app and restart it. This is the main reason to use a database instead of variables.

Submit your code to the marking system by zipping up the contents of the src folder of your project. Note that this week's lab is not automatically marked at the moment as I am still figuring out the technicalities behind automatically marking Android code so you may get an error when submitting, but please do submit anyway.