BSc (Hons) in Information Technology
Year 2

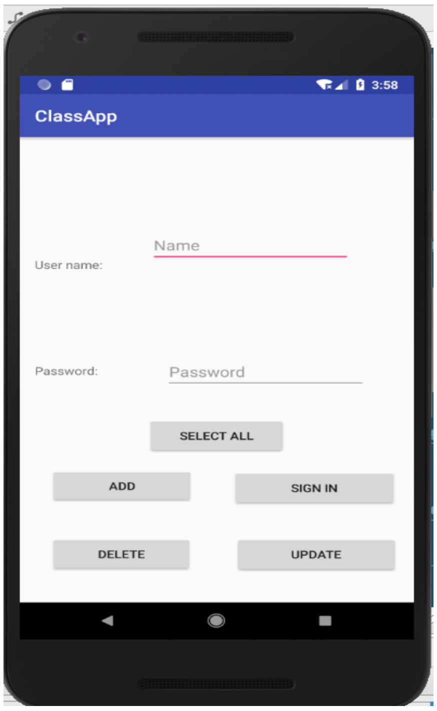
Lab Exercise 6

IT2010 – Mobile Application Design and Development

Semester 2, 2018

---

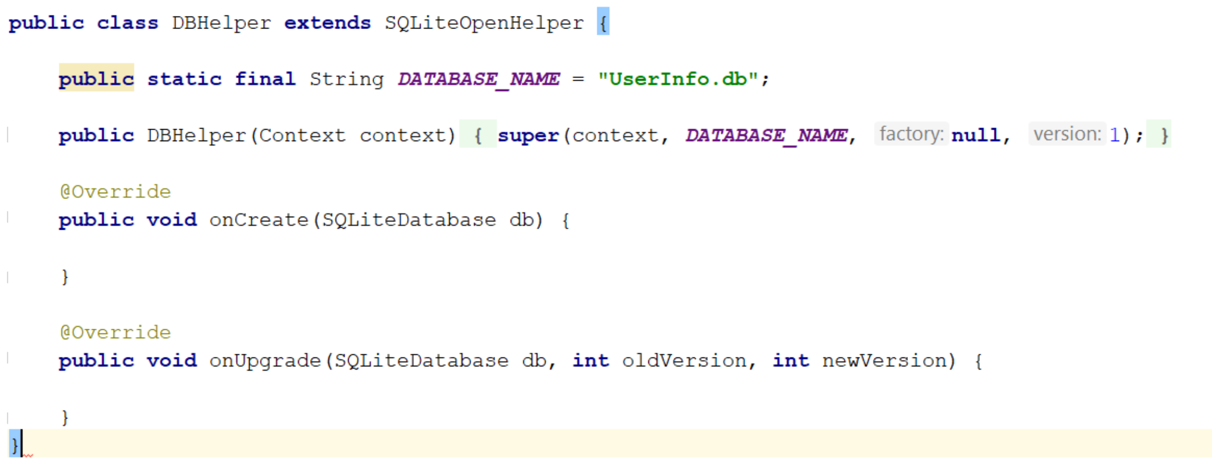
1. Create a login screen as follow containing
   a. UserName (Text)
   b. Password (Text Password)
   c. 5 Buttons
      i. Select all
      ii. Add
      iii. Sign In
      iv. Delete
      v. Update



2. Create a new folder as Database and inside that, create a Final class called 'UsersMaster'. Make the default constructor private. Create an inner class called 'Users' by implementing 'BaseColumn' interface. Inside the inner class, define the columns you need along with the table.

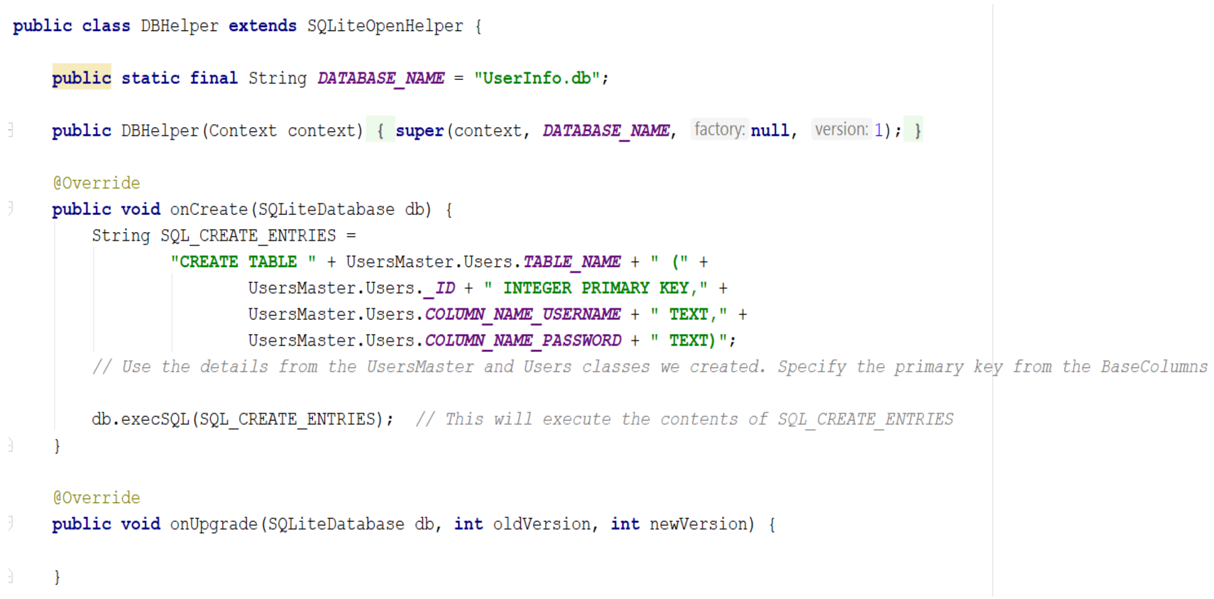
```java
import android.provider.BaseColumns;

public final class UsersMaster {
    private UsersMaster() {}

    /* Inner class that defines the table contents */
    public static class Users implements BaseColumns {
        public static final String TABLE_NAME = "users";
        public static final String COLUMN_NAME_USERNAME = "username";
        public static final String COLUMN_NAME_PASSWORD = "password";
    }
}
```

3. Create another class called DBHelper inside the database folder by extending the class SQLiteOpenHelper as its superclass. Implement the relevant methods and constructors.



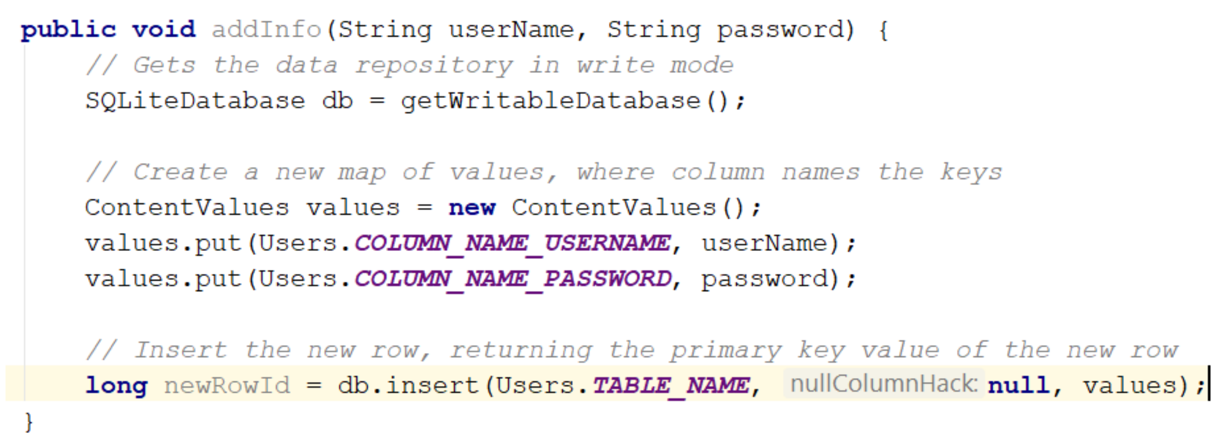
```java
public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "UserInfo.db";

    public DBHelper(Context context) { super(context, DATABASE_NAME, factory: null, version: 1); }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}
```

4. Add the code for creating the table of 'Users' inside the onCreate method. This will be executed each time when an instance of this class is created. If the table exists, nothing will happen, else table will be created.



```java
public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "UserInfo.db";

    public DBHelper(Context context) { super(context, DATABASE_NAME, factory: null, version: 1); }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String SQL_CREATE_ENTRIES =
                "CREATE TABLE " + UsersMaster.Users.TABLE_NAME + " (" +
                        UsersMaster.Users._ID + " INTEGER PRIMARY KEY," +
                        UsersMaster.Users.COLUMN_NAME_USERNAME + " TEXT," +
                        UsersMaster.Users.COLUMN_NAME_PASSWORD + " TEXT)";
        // Use the details from the UsersMaster and Users classes we created. Specify the primary key from the BaseColumns

        db.execSQL(SQL_CREATE_ENTRIES);  // This will execute the contents of SQL_CREATE_ENTRIES
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
```

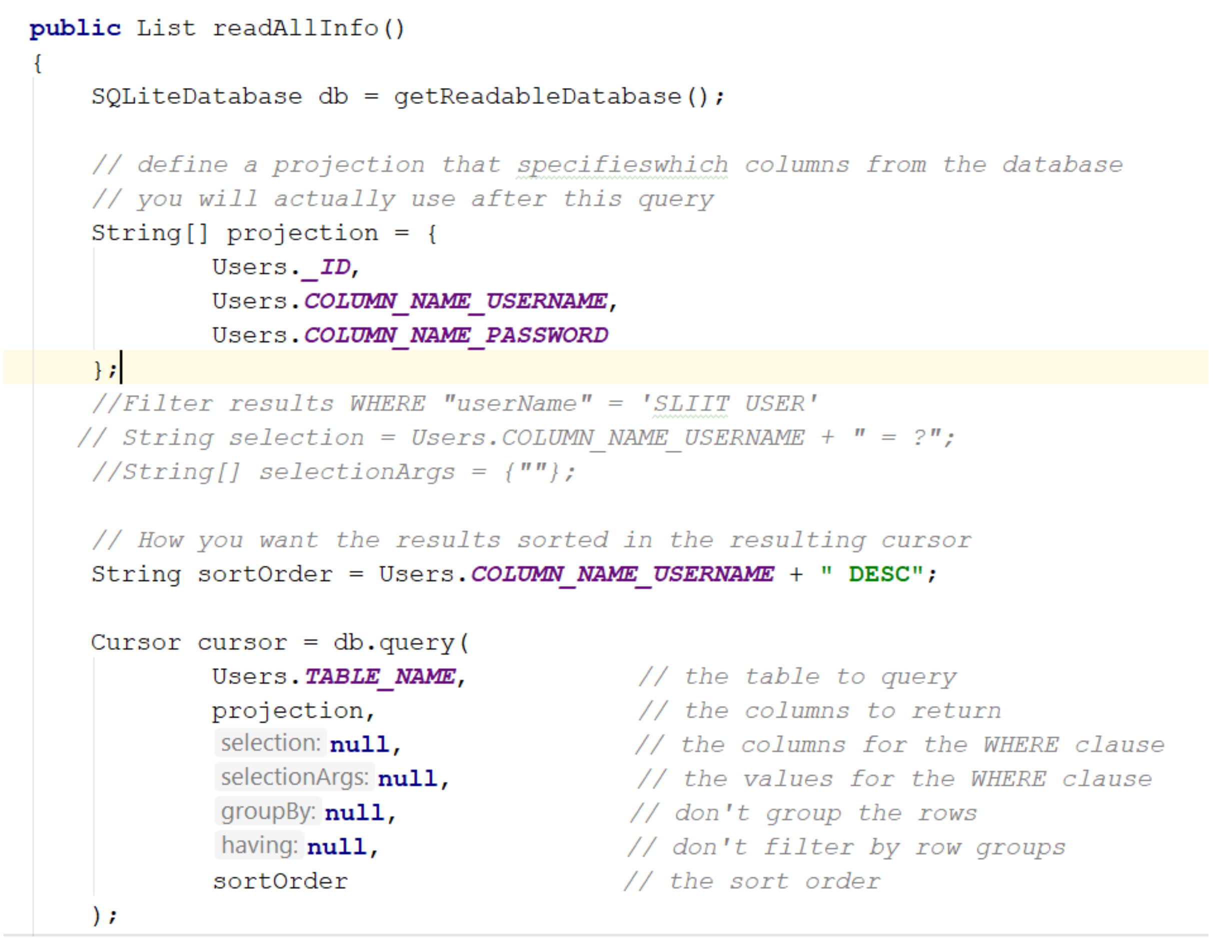
5. Let's add some data. Create a new method 'addInfo()' in DbHelper.java file to add the User Name and Password. (accepts UserName and passwords as parameters). Implement the button click event for Add button and call to addInfo().



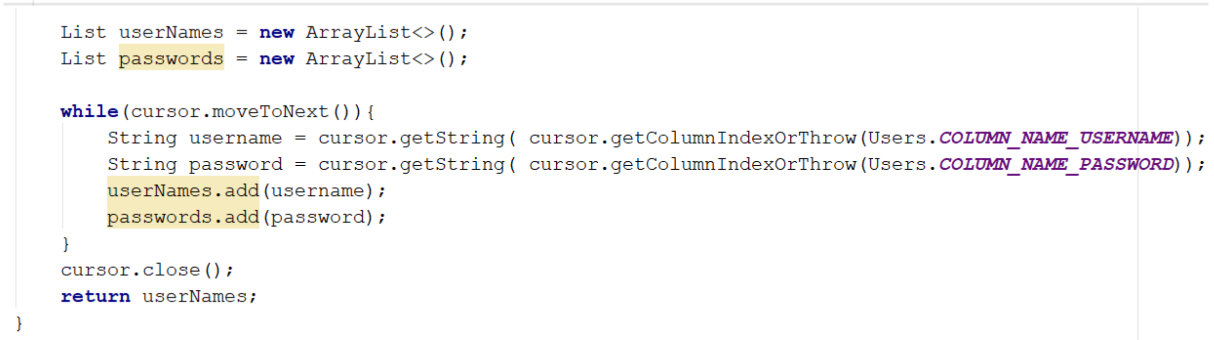
```
public void addInfo(String userName, String password) {
    // Gets the data repository in write mode
    SQLiteDatabase db = getWritableDatabase();

    // Create a new map of values, where column names the keys
    ContentValues values = new ContentValues();
    values.put(Users.COLUMN_NAME_USERNAME, userName);
    values.put(Users.COLUMN_NAME_PASSWORD, password);

    // Insert the new row, returning the primary key value of the new row
    long newRowId = db.insert(Users.TABLE_NAME,  nullColumnHack: null, values);
}
```

6. Implement the button click event for SelectAll button.
    a. Can specify a condition/ where statement
    b. Grouping of data
    c. Sort / order the selection data

- Create a new method ReadAllInfo() in DbHelper.java file to read the user info.
- Call to the above method form SelectAll button click event and display all user names in Log window.



```
public List readAllInfo()
{
    SQLiteDatabase db = getReadableDatabase();

    // define a projection that specifieswhich columns from the database
    // you will actually use after this query
    String[] projection = {
            Users._ID,
            Users.COLUMN_NAME_USERNAME,
            Users.COLUMN_NAME_PASSWORD
    };
    //Filter results WHERE "userName" = 'SLIIT USER'
  // String selection = Users.COLUMN_NAME_USERNAME + " = ?";
    //String[] selectionArgs = {""};

    // How you want the results sorted in the resulting cursor
    String sortOrder = Users.COLUMN_NAME_USERNAME + " DESC";

    Cursor cursor = db.query(
            Users.TABLE_NAME,              // the table to query
            projection,                    // the columns to return
            selection: null,               // the columns for the WHERE clause
            selectionArgs: null,           // the values for the WHERE clause
            groupBy: null,                 // don't group the rows
            having: null,                  // don't filter by row groups
            sortOrder                      // the sort order
    );
```



```
    List userNames = new ArrayList<>();
    List passwords = new ArrayList<>();

    while(cursor.moveToNext()){
        String username = cursor.getString( cursor.getColumnIndexOrThrow(Users.COLUMN_NAME_USERNAME));
        String password = cursor.getString( cursor.getColumnIndexOrThrow(Users.COLUMN_NAME_PASSWORD));
        userNames.add(username);
        passwords.add(password);
    }
    cursor.close();
    return userNames;
}
```

- ** Lists userNames and Passwords will contain all the info (since no *where* statement is specifies) from the table Users. So, a logic can be written accordingly.

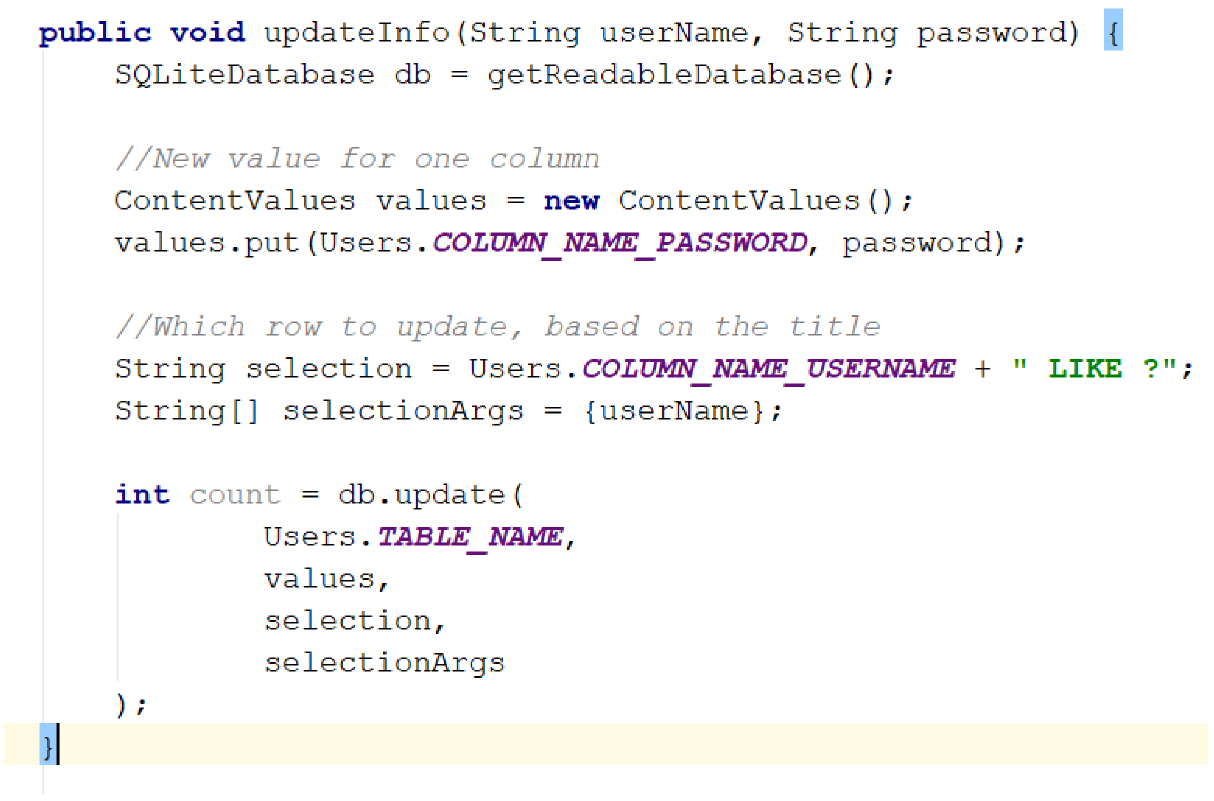
7. Modify button click event of ‘SignIn’ to check the given username and password are existing in the database and give a Toast message if the user is not existing. Modify the readAllInfo() as suitable in a separate method called readInfo().

8. Implement the button click event for Delete button. And write a method called deleteInfo() in DbHandler.java.

   Modify above methods to display a Toast message after completing the task.

```
//This will delete a particular user from the table
public void deleteInfo(String userName){
    SQLiteDatabase db = getReadableDatabase();
    //Define 'where' part of query
    String selection = Users.COLUMN_NAME_USERNAME + " LIKE ?";
    //Specify arguments n placeholder order
    String[] selectionArgs = { userName };
    //Issue SQL statement
    db.delete(Users.TABLE_NAME, selection, selectionArgs);

}
```

9. Implement the Update button to update user details for the given user name. Implement a method called ‘updateUser()’ in DbHelper.java file.



```
public void updateInfo(String userName, String password) {
    SQLiteDatabase db = getReadableDatabase();

    //New value for one column
    ContentValues values = new ContentValues();
    values.put(Users.COLUMN_NAME_PASSWORD, password);

    //Which row to update, based on the title
    String selection = Users.COLUMN_NAME_USERNAME + " LIKE ?";
    String[] selectionArgs = {userName};

    int count = db.update(
            Users.TABLE_NAME,
            values,
            selection,
            selectionArgs
    );
}
```