Password Keeper Application

Prerequisites
It is assumed that you already know how to program in Java, have a basic understanding of XML, and have a basic understanding how to create Android Projects using Android Studio. Please refer to my Introductory Lab Project if you need help setting up and implementing a simple Activity in Android Studio.

Project Setup
Create a new project following the steps outlined in the Introductory Lab Tutorial. Give this project a name such as PwKeeper.

Main Activity / GUI
In the activity_main.xml file create a GUI with the following elements (See Figure 1):

- A TextView control with the label “ID”.
- A readonly EditText control to display the record ID.
- A TextView control with the label “Description”.
- An editable EditText control for the password description.
- A TextView control with the label “Password”.
- An editable EditText control for the actual password.
- Three buttons labeled “Add”, “Find” and “Delete”.

You can also use a TextView field to display the record ID since it is a readonly element of the GUI. Make sure you give your EditText controls meaningful IDs so that it will be easy to create references to them in your Java code. For example, name your EditText controls txtId, txtDescription and txtPassword. Name your buttons btnAdd, btnFind and btnDelete. Add “onclick” attributes to your buttons that name the methods that will handle the click events for your buttons:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add"
    android:id="@+id/btnAdd"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="35dp"
    android:clickable="true"
    android:onClick="addPassword" />
```
“addPassword”, “findPassword” and “deletePassword”. See Figure 2 for an example XML specification for the Add button.

Wallet Class
Under the “java” folder in your project window on the right side of the screen your will find the package name for your java code. Right click the package name and select New/Java Class. Name your class Wallet. Your class should include three private properties: id (integer), description (string) and password (string). Create a no-arg constructor, a constructor that has the two string formal parameters and getter and setters for your properties. Objects of this class will be used to store password records that are saved and retrieved from the SQLite database.

```java
public class Wallet {
    private int id;
    private String description;
    private String password;

    public Wallet() {};

    public Wallet( String description, String password ) {
        this.setDescription(description);
        this.setPassword(password);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

DbHandler Class
Create another Java class and name it DbHandler. This will be the class that will handle the interactions between your main Activity and the SQLite database.

Here are the imports:
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.ContentValues;
import android.database.Cursor;

The class itself will be a subclass of SQLiteOpenHelper and will use the following constants:

```java
public class DbHandler extends SQLiteOpenHelper{
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "passwordDB.db";
    private static final String TABLE_CONTENT = "pswdWallet";

    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_DESCRIPTION = "description";
    public static final String COLUMN_PSWD = "password";
}
```

Here are the constructor, onCreate and onUpgrade methods as discussed in the tutorial covered in class:

```java
public DbHandler(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
    super(context, DATABASE_NAME, factory, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_PRODUCTS_TABLE = "CREATE TABLE " + TABLE_CONTENT + "(
        _id INTEGER PRIMARY KEY," + COLUMN_DESCRIPTION + " TEXT," + COLUMN_PSWD + " TEXT" + ")"
    db.execSQL(CREATE_PRODUCTS_TABLE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTENT);
    onCreate(db);
}
```

The next three methods are called to access the SQLite database. They implement the button functions of your main Activity.

```java
public void addPassword(Wallet passwordRec) {
    ContentValues values = new ContentValues();
    values.put(COLUMN_DESCRIPTION, passwordRec.getDescription());
    values.put(COLUMN_PSWD, passwordRec.getPassword());

    SQLiteDatabase db = this.getWritableDatabase();

    db.insert(TABLE_CONTENT, null, values);
    db.close();
}
public Wallet findPassword(String sPassword) {
    String query = "Select * FROM " + TABLE_CONTENT + " WHERE " + COLUMN_DESCRIPTION + " = \"" + sPassword + \"\";

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(query, null);
    Wallet wallet = new Wallet();
    if (cursor.moveToFirst()) {
        cursor.moveToFirst();
        wallet.setId(Integer.parseInt(cursor.getString(0)));
        wallet.setDescription(cursor.getString(1));
        wallet.setPassword(cursor.getString(2));
        cursor.close();
    } else {
        wallet = null;
    }
    db.close();
    return wallet;
}

public boolean deletePassword(String sDescription) {
    boolean result = false;
    String query = "Select * FROM " + TABLE_CONTENT + " WHERE " + COLUMN_DESCRIPTION + " = \"" + sDescription + \"\";

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(query, null);
    Wallet wallet = new Wallet();
    if (cursor.moveToFirst()) {
        wallet.setId(Integer.parseInt(cursor.getString(0)));
        db.delete(TABLE_CONTENT, COLUMN_ID + " = ?", new String[] { String.valueOf(wallet.getId()) });
        cursor.close();
        result = true;
    }
    db.close();
    return result;
}

MainActivity Class
Now we need to modify the MainActivity class so that we can glue the pieces of your app together. Open up your MainActivity class and make the following additions.

Add the following to your onCreate class. The control ids used in the parameter to the findViewById method should match the ids you gave your controls in the activity_main.xml file of your project.

    idView = (EditText) findViewById(R.id.txtId);
    descriptionBox = (EditText) findViewById(R.id.txtDescription);
    passwordBox = (EditText) findViewById(R.id.txtPassword);
Now we simply have to add the onclick event handlers for the command buttons on your GUI. The names of the methods should match the method names given in the XML onclick attribute of each button when it was defined in activity_main.xml.

```java
public void addPassword (View view) {
    DbHandler dbHandler = new DbHandler(this, null, null, 1);

    Wallet wallet =
        new Wallet(descriptionBox.getText().toString(),
                   passwordBox.getText().toString());
    dbHandler.addPassword(wallet);
    descriptionBox.setText("");
    passwordBox.setText("");
}

public void findPassword (View view) {
    DbHandler dbHandler = new DbHandler(this, null, null, 1);

    Wallet wallet =
        dbHandler.findPassword(descriptionBox.getText().toString());
    if (wallet != null) {
        idView.setText(String.valueOf(wallet.getId()));
        passwordBox.setText(String.valueOf(wallet.getPassword()));
    } else {
        idView.setText("No Match Found");
    }
}

public void deletePassword(View view) {
    DbHandler dbHandler = new DbHandler(this, null, null, 1);

    boolean result = dbHandler.deletePassword(
        descriptionBox.getText().toString());
    if (result) {
        idView.setText("Record Deleted");
        descriptionBox.setText("");
        passwordBox.setText("");
    } else {
        idView.setText("No Match Found");
    }
}
```

**Testing**

You should now be able to start your Android Virtual Device and run your app. Test the Add, Find and Delete buttons to make sure your app is functioning properly.