MIT Global Startup Labs México 2013

http://gsl.mit.edu
Coming Soon!

Lección 04 + 05 – Databases

Agenda

- Databases in general
- Database in Android (SQLite)
- Quick review of SQL
- Exercise

- SQLite Architecture
- Lab - Contacts

Databases in general

- Database = data storage mechanism
- Useful for making data *persist* (keep track of data even when application is closed and reopened).
- Many different ways of implementing a database.
- One common approach: Relational Databases using **SQL** (a language used to insert, delete, and update data in a database)

Transactional DB

- Changes and queries are (by definition):
  - Atomic, Consistent, Isolated, Durable (ACID)
- All changes within a single transaction either occur completely or not at all, even if :
  - Program crashes
  - Operating system crashes
  - Power failure

SQL in app Benefits

- Cache
  - Contacts, systems settings, bookmarks

Databases on Android (SQLite)

- The Android OS provides a built-in database management system called **SQLite** (a DB system specialized for embedded devices)

- Each Android application can have its own SQLite database, but may not access the database of any other application (for security)

Advantages of SQLite

- Uses standard SQL syntax
- Open-source, zero-configuration (no effort required by developer to set up the DB before using it)
- SQLite system is not a client-server system (there's no SQLite server process that is always running).
- Each SQLite database exists in its own, single file (very secure)

Other Options

- http://www.sqlite.org/

- Can use other db system:
  - JavaDB, MongoDB
    - Will have to bundle required libraries
    - Can't rely on Android's built-in db support
  - SQLite not alternative to full SQL server, alternative to local file with arbitrary format

Quick review of SQL:

Table 1: "notes"

_id	title	body
0	myFirstNote	Hi, abc…
1	anotherNote	blaablaablaa

SQLite Database with two tables

SQL statement for creating table "notes":

```
CREATE TABLE notes (_id integer primary key
autoincrement, title text not null, body text not
null);
```

Table 2: "employees"

_id	emp_name	emp_salary
0	Sally	$123,456
1	Bobby	$65,432

Quick review of SQL:

Table 1: “notes”

_id	title	body
0	myFirstNote	Hi, abc…
1	anotherNote	blaablaablaa

SQLite Database with two tables

<u>SQL statement for inserting into tables:</u>

```
INSERT INTO notes VALUES(‘myFirstNote’, ‘Hi,abc…’);
INSERT INTO employees VALUES (‘Sally’, ‘123456’);
```

Table 2: “employees”

_id	emp_name	emp_salary
0	Sally	$123,456
1	Bobby	$65,432

Quick review of SQL:

Table 1: "notes"

_id	title	body
0	myFirstNote	Hi, abc…
1	anotherNote	blaablaablaa

SQLite Database with two tables

SQL statement for selecting/deleting specific rows in the tables:

```
SELECT * FROM notes
   WHERE title = 'anotherNote'
   AND body = 'blaablaablaa';

DELETE FROM employees WHERE emp_salary < 100000;
```

Table 2: "employees"

_id	emp_name	emp_salary
0	Sally	$123,456
1	Bobby	$65,432

SQLite3

- For debugging
- Command line utility to execute SQL commands against SQLite database

http://www.sqlite.org/sqlite.html

http://www.w3schools.com/sql/

Exercise

- Make database ‘Contactos’
- 2 tables
  - Email_Priority: Stores emailID, priority (1-10)
  - Email_Info: Stores emailID, FirstName, Lastname, PhoneNumber
- Add 10 contacts
- Select statement shows LastName + Phonenumber of contact who has priority>8
- Output to important_phone.txt

SQLite Architecture

- android.database contains all general classes for working with databases.
- android.database.sqlite contains the SQLite specific classes.
- Need “connection” to database
  - SQLiteOpenHelper Class
    - Returns instance of SQLiteDatabase

Best practice Exceptions

- Outside of direct control
- Database might be running out of space or be corrupted
- Good practice:
  - Surround database calls with try/catch blocks

DBHelper

- CRUD operations
  - Create, read (query), update, delete
- DBHelper:
  - insert() Inserts one or more rows into the database
  - query() Requests rows matching the criteria you specify
  - update() Replaces ones or more rows that match the criteria you specify
  - delete() Deletes rows matching the criteria you specify

Cursor

- Query returns set of rows along with pointer called *cursor*
  - Return results one at a time, causing cursor to advance each time to next row
  - Empty cursor means you have retrieved all rows

Example: Android Notes App

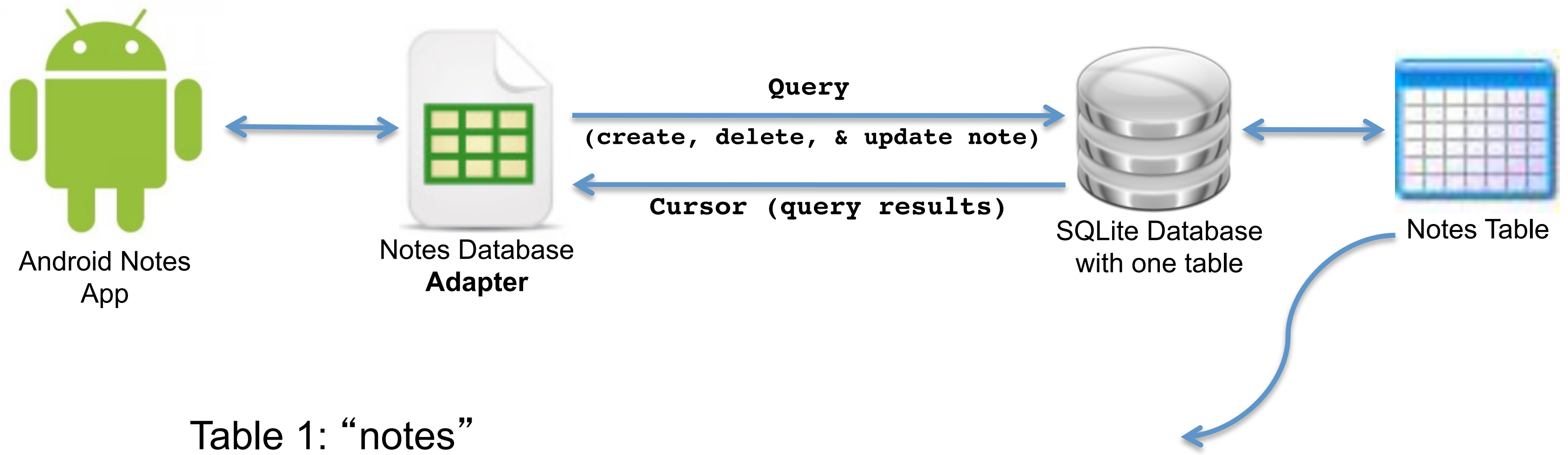
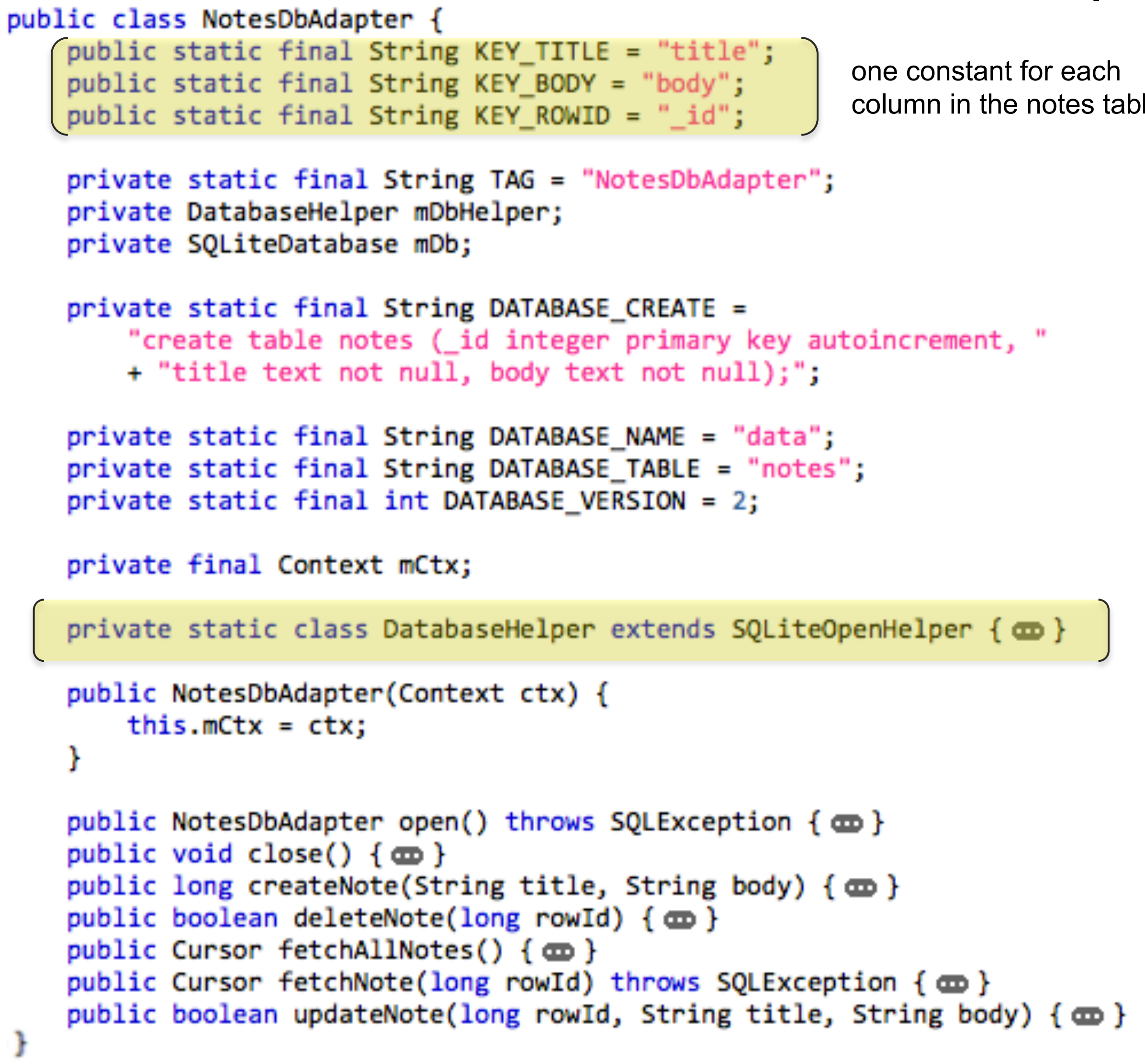


Table 1: “notes”

_id	title	body
0	myFirstNote	Hi, abc…
1	anotherNote	blaablaablaa

Example: Android Notes App

A closer look at the Notes Database Adapter

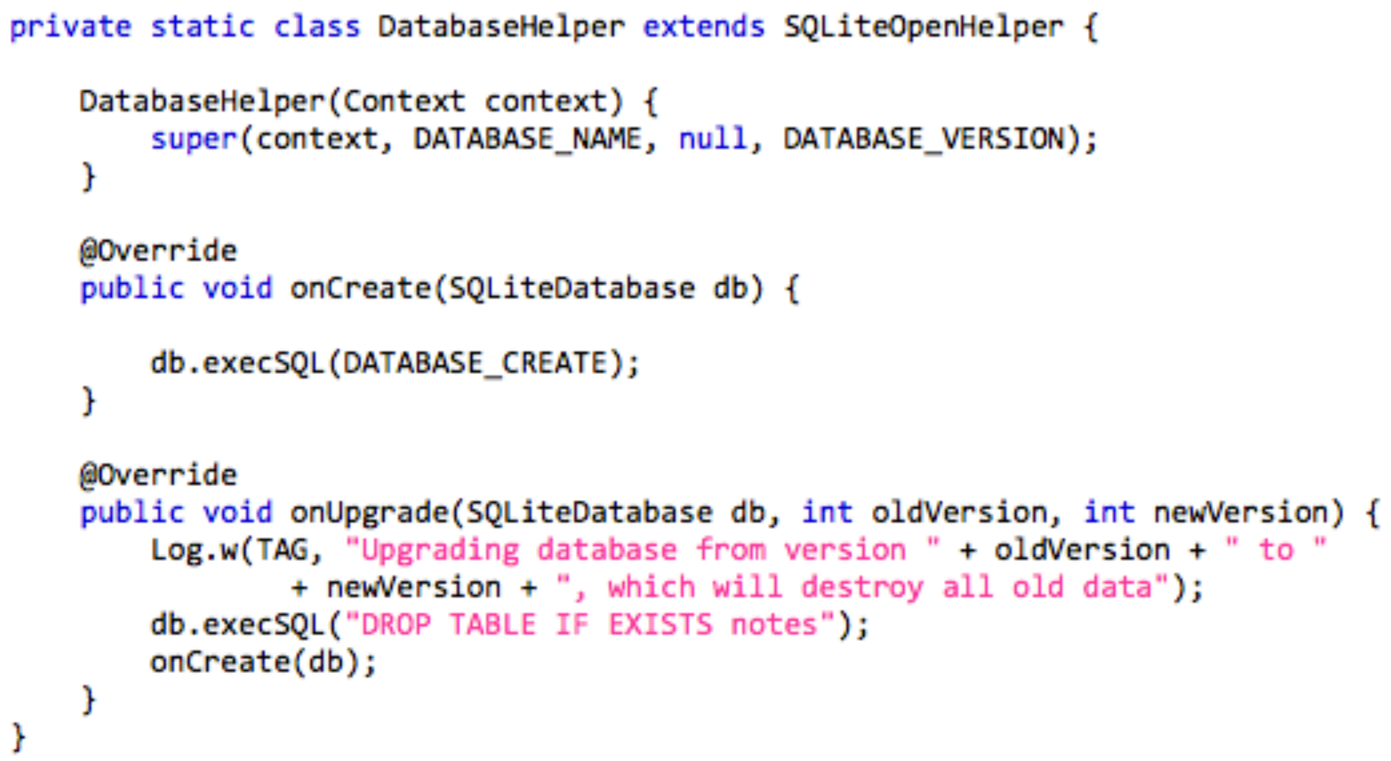


```
public class NotesDbAdapter {
    public static final String KEY_TITLE = "title";
    public static final String KEY_BODY = "body";
    public static final String KEY_ROWID = "_id";

    private static final String TAG = "NotesDbAdapter";
    private DatabaseHelper mDbHelper;
    private SQLiteDatabase mDb;

    private static final String DATABASE_CREATE =
        "create table notes (_id integer primary key autoincrement, "
        + "title text not null, body text not null);";

    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "notes";
    private static final int DATABASE_VERSION = 2;

    private final Context mCtx;

    private static class DatabaseHelper extends SQLiteOpenHelper {⋯}

    public NotesDbAdapter(Context ctx) {
        this.mCtx = ctx;
    }

    public NotesDbAdapter open() throws SQLException {⋯}
    public void close() {⋯}
    public long createNote(String title, String body) {⋯}
    public boolean deleteNote(long rowId) {⋯}
    public Cursor fetchAllNotes() {⋯}
    public Cursor fetchNote(long rowId) throws SQLException {⋯}
    public boolean updateNote(long rowId, String title, String body) {⋯}
}
```

one constant for each column in the notes table

Example: Android Notes App

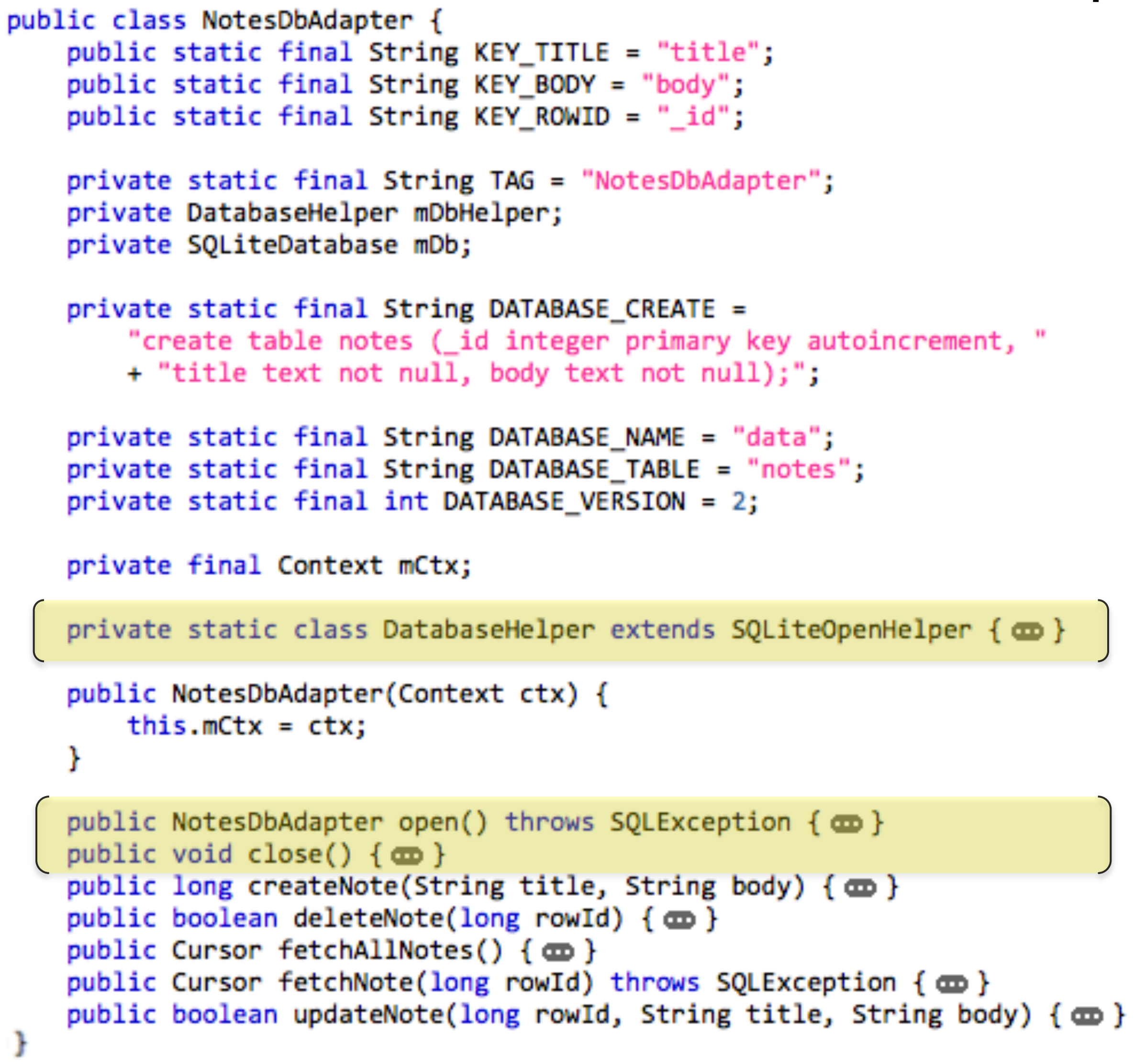
A closer look at the Notes Database Adapter



```java
private static class DatabaseHelper extends SQLiteOpenHelper {

    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

        db.execSQL(DATABASE_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
                + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS notes");
        onCreate(db);
    }
}
```

Example: Android Notes App

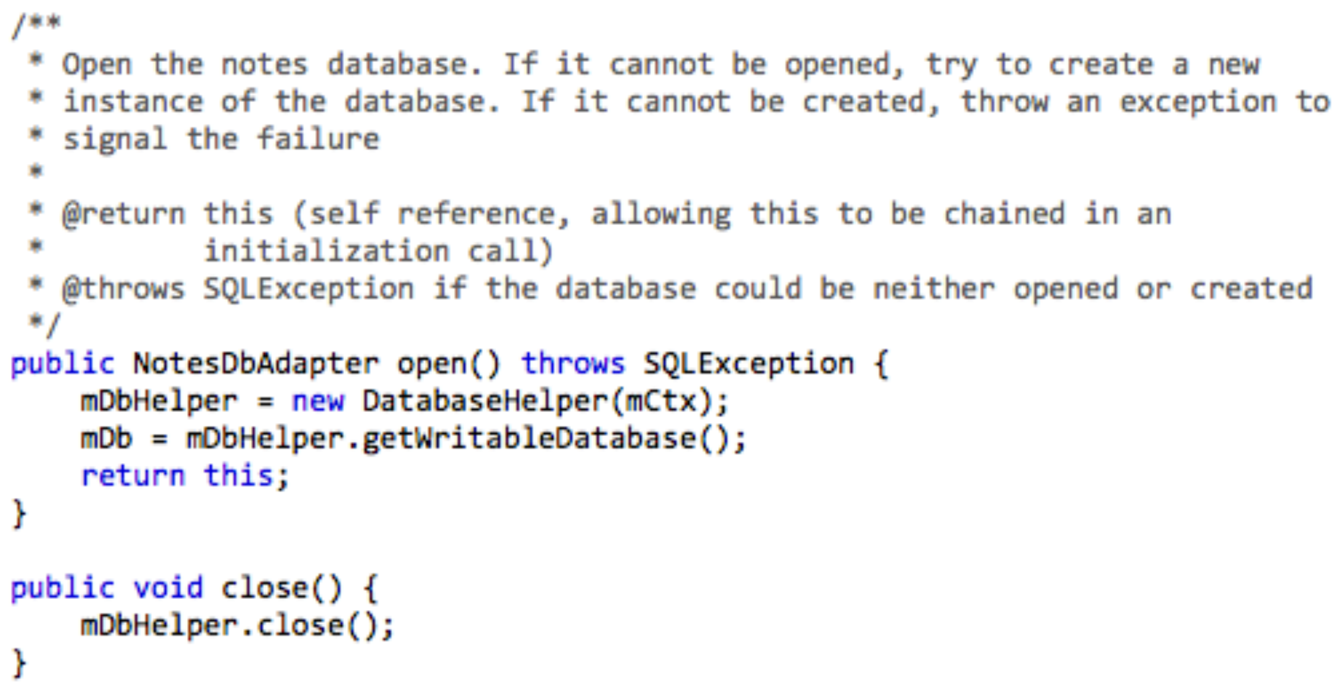
A closer look at the Notes Database Adapter



```java
public class NotesDbAdapter {
    public static final String KEY_TITLE = "title";
    public static final String KEY_BODY = "body";
    public static final String KEY_ROWID = "_id";

    private static final String TAG = "NotesDbAdapter";
    private DatabaseHelper mDbHelper;
    private SQLiteDatabase mDb;

    private static final String DATABASE_CREATE =
        "create table notes (_id integer primary key autoincrement, "
        + "title text not null, body text not null);";

    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "notes";
    private static final int DATABASE_VERSION = 2;

    private final Context mCtx;

    private static class DatabaseHelper extends SQLiteOpenHelper { }

    public NotesDbAdapter(Context ctx) {
        this.mCtx = ctx;
    }

    public NotesDbAdapter open() throws SQLException { }
    public void close() { }
    public long createNote(String title, String body) { }
    public boolean deleteNote(long rowId) { }
    public Cursor fetchAllNotes() { }
    public Cursor fetchNote(long rowId) throws SQLException { }
    public boolean updateNote(long rowId, String title, String body) { }
}
```

Example: Android Notes App

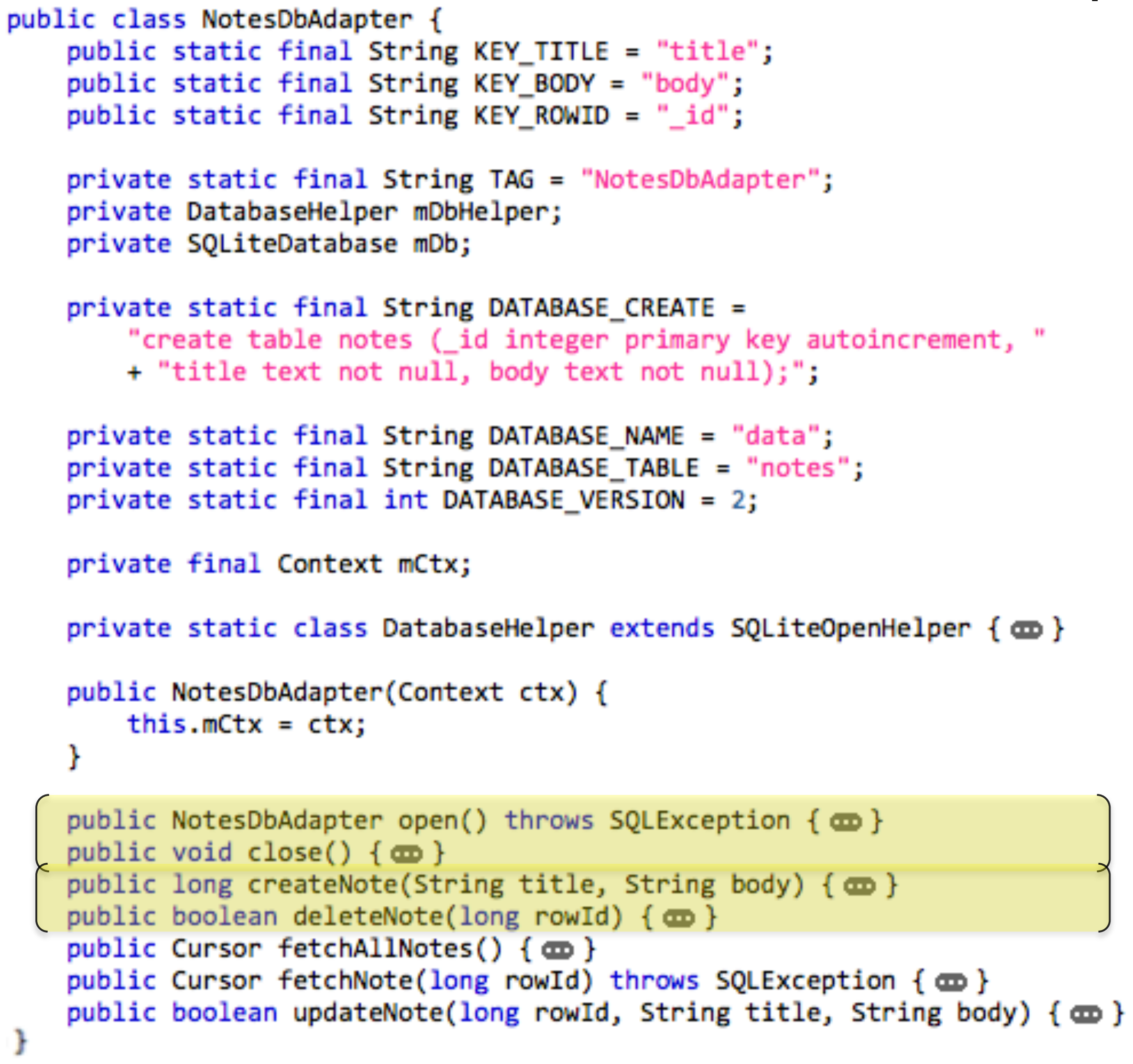
A closer look at the Notes Database Adapter



```java
/**
 * Open the notes database. If it cannot be opened, try to create a new
 * instance of the database. If it cannot be created, throw an exception to
 * signal the failure
 *
 * @return this (self reference, allowing this to be chained in an
 *         initialization call)
 * @throws SQLException if the database could be neither opened or created
 */
public NotesDbAdapter open() throws SQLException {
    mDbHelper = new DatabaseHelper(mCtx);
    mDb = mDbHelper.getWritableDatabase();
    return this;
}

public void close() {
    mDbHelper.close();
}
```

Example: Android Notes App

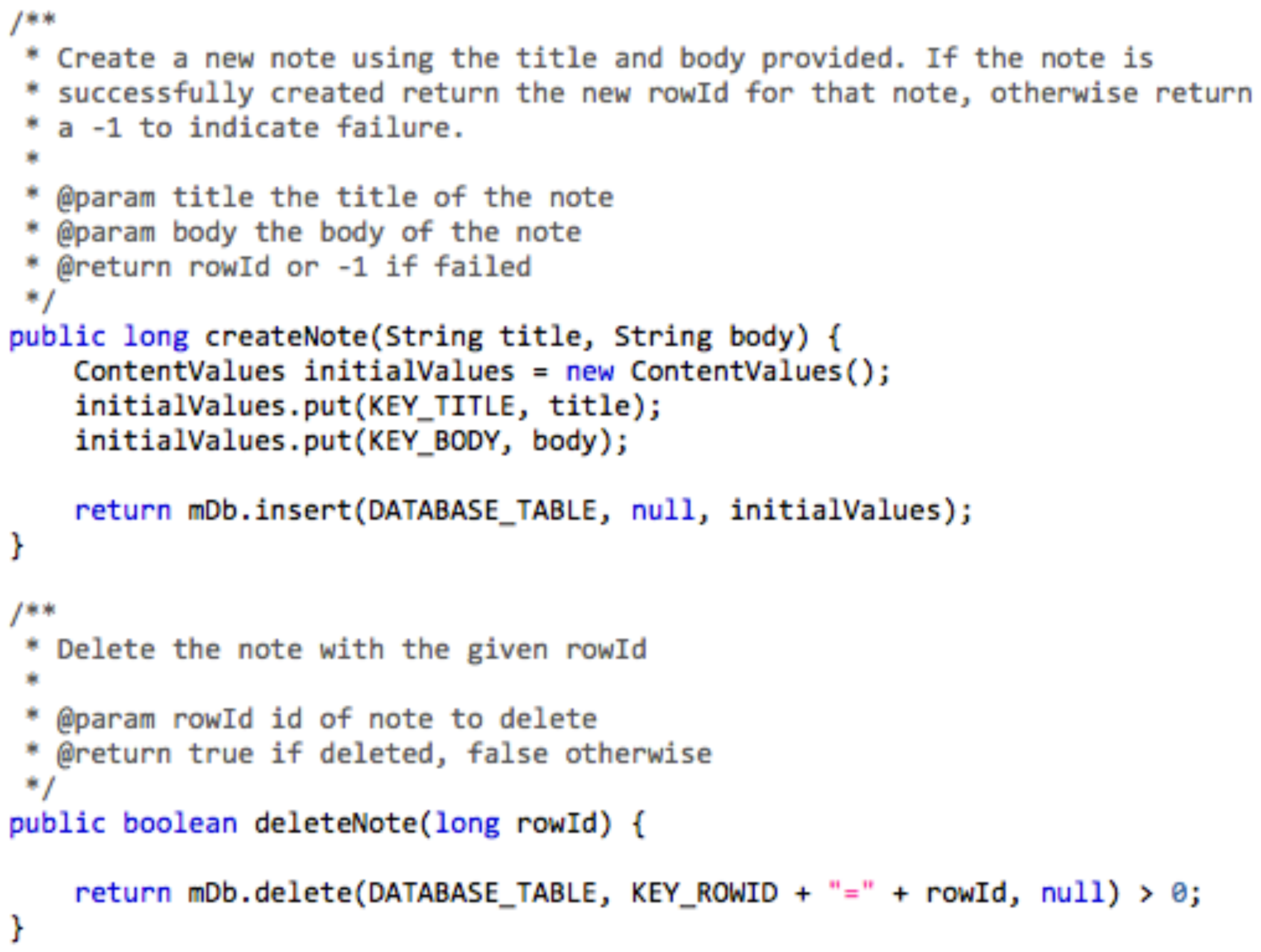
A closer look at the Notes Database Adapter



```
public class NotesDbAdapter {
    public static final String KEY_TITLE = "title";
    public static final String KEY_BODY = "body";
    public static final String KEY_ROWID = "_id";

    private static final String TAG = "NotesDbAdapter";
    private DatabaseHelper mDbHelper;
    private SQLiteDatabase mDb;

    private static final String DATABASE_CREATE =
        "create table notes (_id integer primary key autoincrement, "
        + "title text not null, body text not null);";

    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "notes";
    private static final int DATABASE_VERSION = 2;

    private final Context mCtx;

    private static class DatabaseHelper extends SQLiteOpenHelper { ⋯ }

    public NotesDbAdapter(Context ctx) {
        this.mCtx = ctx;
    }

    public NotesDbAdapter open() throws SQLException { ⋯ }
    public void close() { ⋯ }
    public long createNote(String title, String body) { ⋯ }
    public boolean deleteNote(long rowId) { ⋯ }
    public Cursor fetchAllNotes() { ⋯ }
    public Cursor fetchNote(long rowId) throws SQLException { ⋯ }
    public boolean updateNote(long rowId, String title, String body) { ⋯ }
}
```

Example: Android Notes App

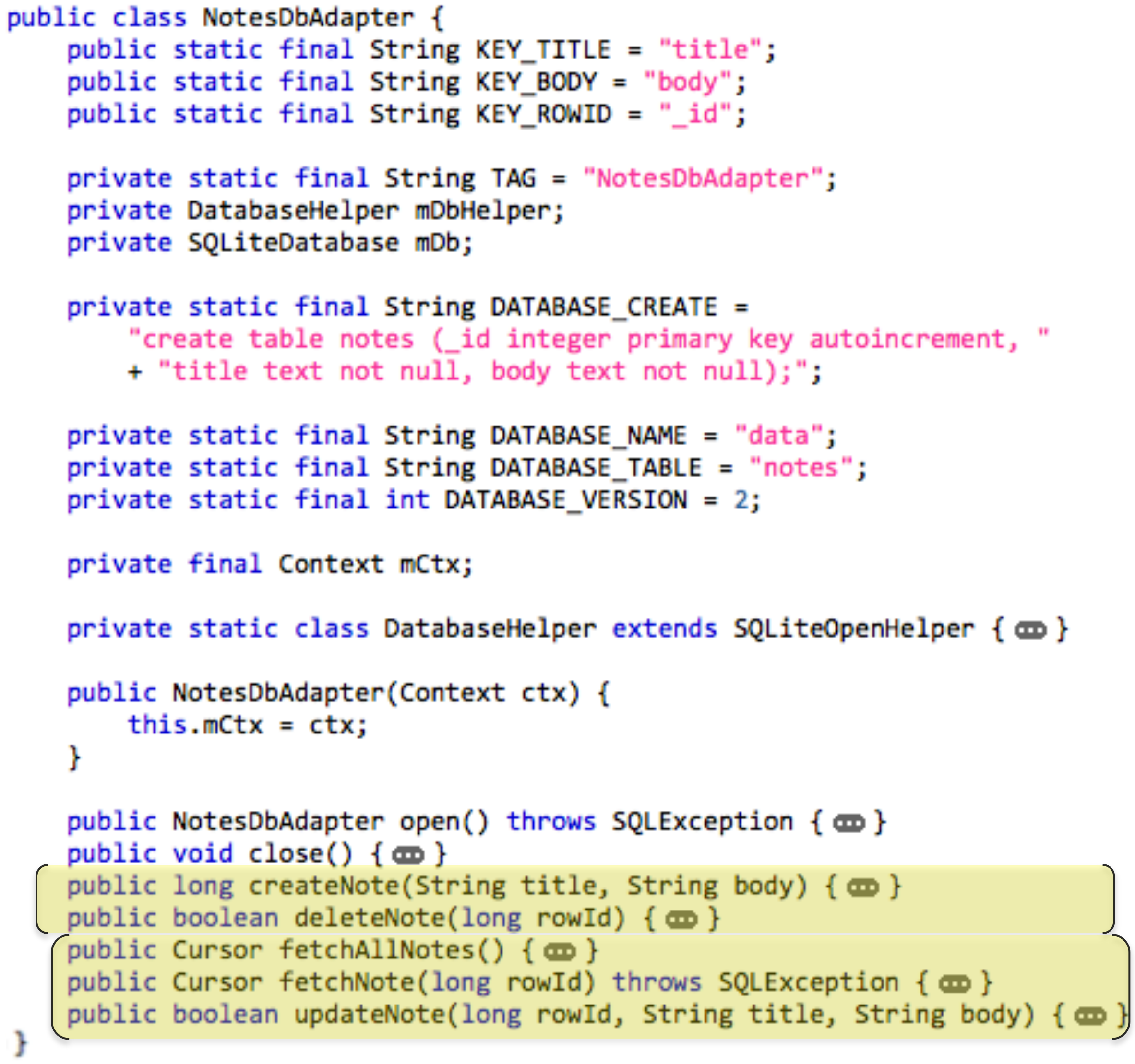
A closer look at the Notes Database Adapter



```
/**
 * Create a new note using the title and body provided. If the note is
 * successfully created return the new rowId for that note, otherwise return
 * a -1 to indicate failure.
 *
 * @param title the title of the note
 * @param body the body of the note
 * @return rowId or -1 if failed
 */
public long createNote(String title, String body) {
    ContentValues initialValues = new ContentValues();
    initialValues.put(KEY_TITLE, title);
    initialValues.put(KEY_BODY, body);

    return mDb.insert(DATABASE_TABLE, null, initialValues);
}

/**
 * Delete the note with the given rowId
 *
 * @param rowId id of note to delete
 * @return true if deleted, false otherwise
 */
public boolean deleteNote(long rowId) {

    return mDb.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
}
```

Example: Android Notes App

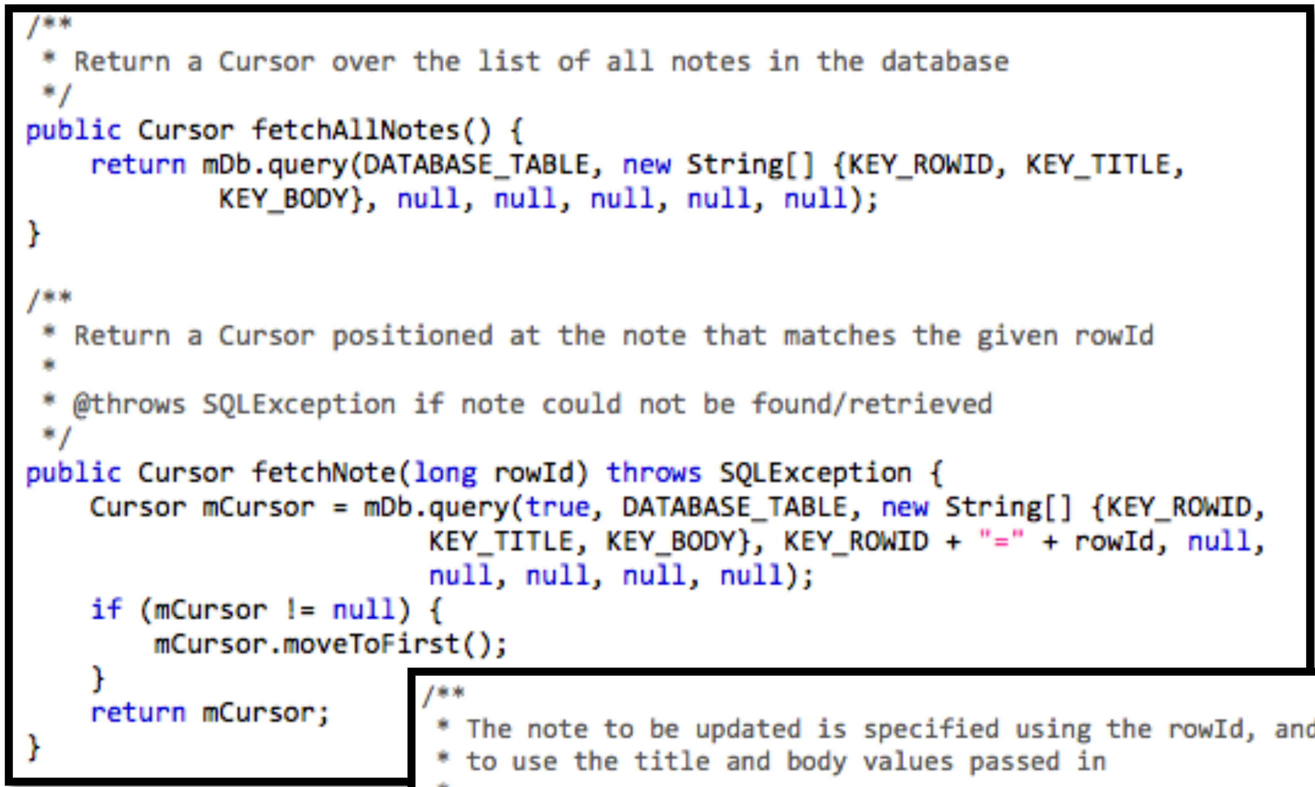
A closer look at the Notes Database Adapter



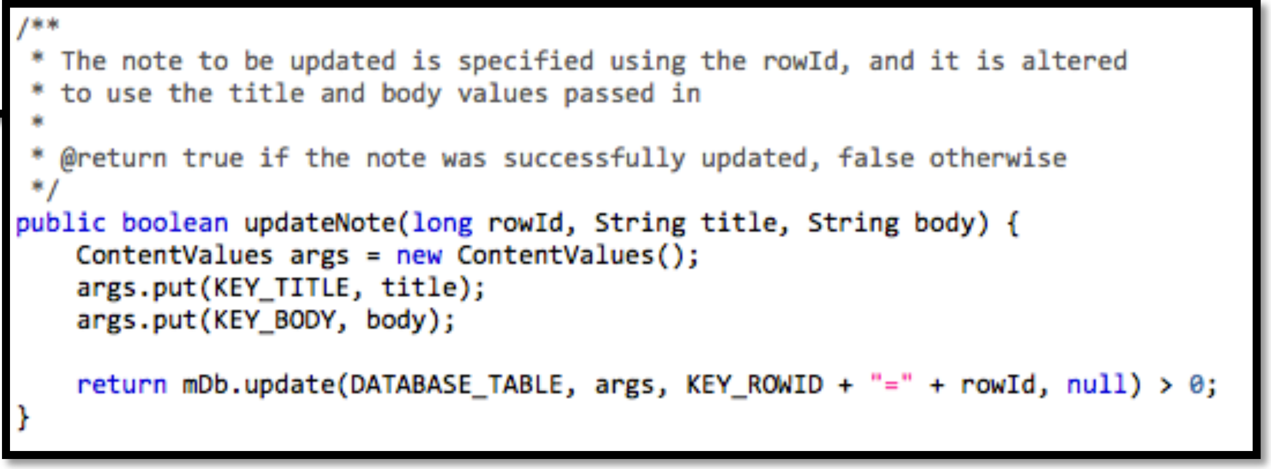
```
public class NotesDbAdapter {
    public static final String KEY_TITLE = "title";
    public static final String KEY_BODY = "body";
    public static final String KEY_ROWID = "_id";

    private static final String TAG = "NotesDbAdapter";
    private DatabaseHelper mDbHelper;
    private SQLiteDatabase mDb;

    private static final String DATABASE_CREATE =
        "create table notes (_id integer primary key autoincrement, "
        + "title text not null, body text not null);";

    private static final String DATABASE_NAME = "data";
    private static final String DATABASE_TABLE = "notes";
    private static final int DATABASE_VERSION = 2;

    private final Context mCtx;

    private static class DatabaseHelper extends SQLiteOpenHelper { }

    public NotesDbAdapter(Context ctx) {
        this.mCtx = ctx;
    }

    public NotesDbAdapter open() throws SQLException { }
    public void close() { }
    public long createNote(String title, String body) { }
    public boolean deleteNote(long rowId) { }
    public Cursor fetchAllNotes() { }
    public Cursor fetchNote(long rowId) throws SQLException { }
    public boolean updateNote(long rowId, String title, String body) { }
}
```

Example: Android Notes App

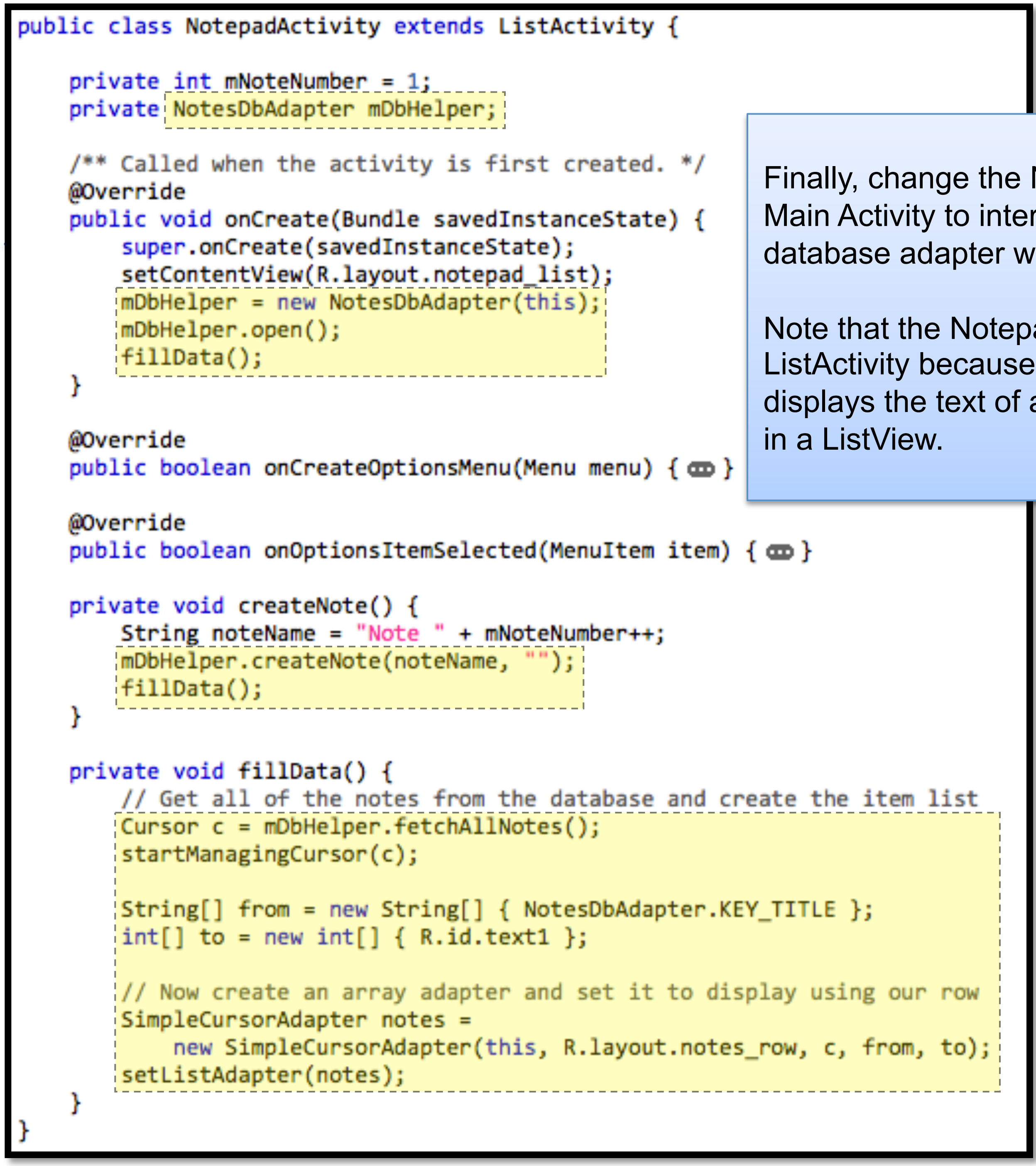
A closer look at the Notes Database Adapter



```java
/**
 * Return a Cursor over the list of all notes in the database
 */
public Cursor fetchAllNotes() {
    return mDb.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_TITLE,
            KEY_BODY}, null, null, null, null, null);
}

/**
 * Return a Cursor positioned at the note that matches the given rowId
 *
 * @throws SQLException if note could not be found/retrieved
 */
public Cursor fetchNote(long rowId) throws SQLException {
    Cursor mCursor = mDb.query(true, DATABASE_TABLE, new String[] {KEY_ROWID,
                    KEY_TITLE, KEY_BODY}, KEY_ROWID + "=" + rowId, null,
                    null, null, null, null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}
```



```java
/**
 * The note to be updated is specified using the rowId, and it is altered
 * to use the title and body values passed in
 *
 * @return true if the note was successfully updated, false otherwise
 */
public boolean updateNote(long rowId, String title, String body) {
    ContentValues args = new ContentValues();
    args.put(KEY_TITLE, title);
    args.put(KEY_BODY, body);

    return mDb.update(DATABASE_TABLE, args, KEY_ROWID + "=" + rowId, null) > 0;
}
```



```java
public class NotepadActivity extends ListActivity {

    private int mNoteNumber = 1;
    private NotesDbAdapter mDbHelper;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.notepad_list);
        mDbHelper = new NotesDbAdapter(this);
        mDbHelper.open();
        fillData();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {}

    private void createNote() {
        String noteName = "Note " + mNoteNumber++;
        mDbHelper.createNote(noteName, "");
        fillData();
    }

    private void fillData() {
        // Get all of the notes from the database and create the item list
        Cursor c = mDbHelper.fetchAllNotes();
        startManagingCursor(c);

        String[] from = new String[] { NotesDbAdapter.KEY_TITLE };
        int[] to = new int[] { R.id.text1 };

        // Now create an array adapter and set it to display using our row
        SimpleCursorAdapter notes =
            new SimpleCursorAdapter(this, R.layout.notes_row, c, from, to);
        setListAdapter(notes);
    }
}
```

Finally, change the Notepad App's Main Activity to interact with the database adapter we just created.

Note that the NotepadActivity is a ListActivity because the app displays the text of all saved notes in a ListView.

Lab

- Follow tutorial to implement Contacts
  - http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/

- Make user interface:
  - Add contact
  - Delete contact
  - Update contact
  - Display all contacts