

# 中山大学移动信息工程学院本科生实验报告

## (2017 年秋季学期)

课程名称：移动应用开发

任课教师： 郑贵锋

年级	2015	专业（方向）	移动互联网
学号	15352344	姓名	吴文标
电话	13112312320	Email	wwb.bill@qq.com
开始日期	2017. 12. 8	完成日期	2017. 12. 17

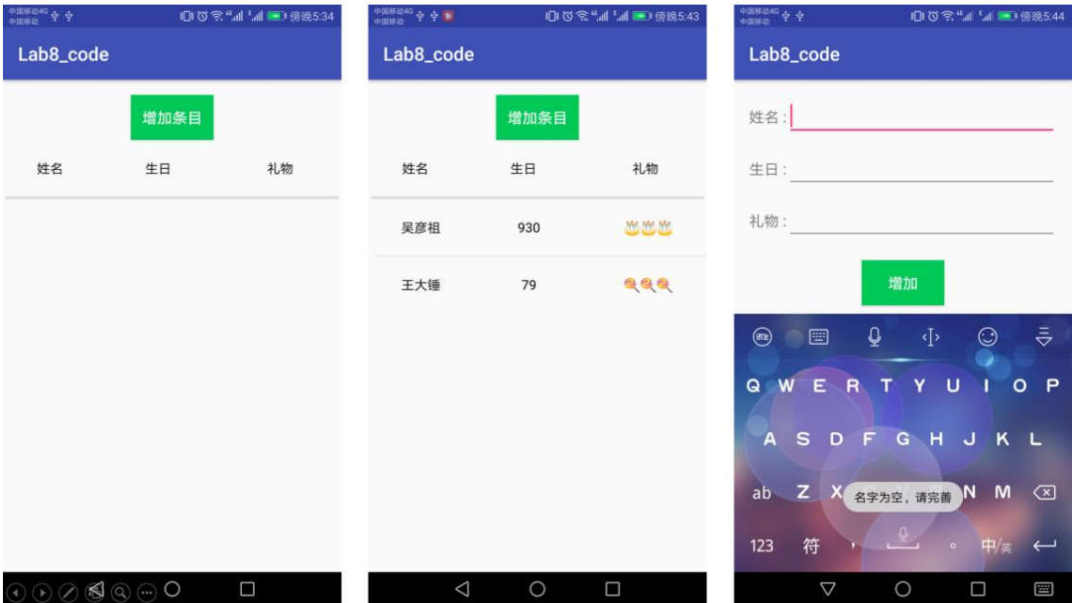
### 一、 实验题目

实验八 数据存储（二）

### 二、 实验目的

- a、 学习 SQL 数据库的使用；
- b、 学习 ContentProvider 的使用；
- c、 复习 Android 界面编程。

### 三、 实现内容



从左至右， 依次为： 初始界面， 添加一部分条目， 名字不能为空。



从左至右， 依次为： 名字不能重复， 点击条目显示信息（可修改） ， 长按删除条目。

实现一个生日备忘录， 要求实现：

使用 SQLite 数据库保存生日的相关信息， 并使得每一次运行程序都可以显示出已经存储在数据库里的内容；

使用 ContentProvider 来获取手机通讯录中的电话号码。

功能要求：

- A. 主界面包含增加生日条目按钮和生日信息列表；
- B. 点击“增加条目” 按钮， 跳转到下一个 Activity 界面， 界面中包含三个信息输入框（姓名、生日、礼物） 和一个“增加” 按钮， 姓名字段不能为空且不能重复；
- C. 在跳转到的界面中， 输入生日的相关信息后， 点击“增加” 按钮返回到主界面， 此

时，主界面中应更新列表，增加相应的生日信息；

D. 主界面列表点击事件：

点击条目：

弹出对话框，对话框中显示该条目的信息，并允许修改；

对话框下方显示该寿星电话号码(如果手机通讯录中有的话,如果没有就显示“无” )

点击“保存修改” 按钮， 更新主界面生日信息列表。

长按条目：

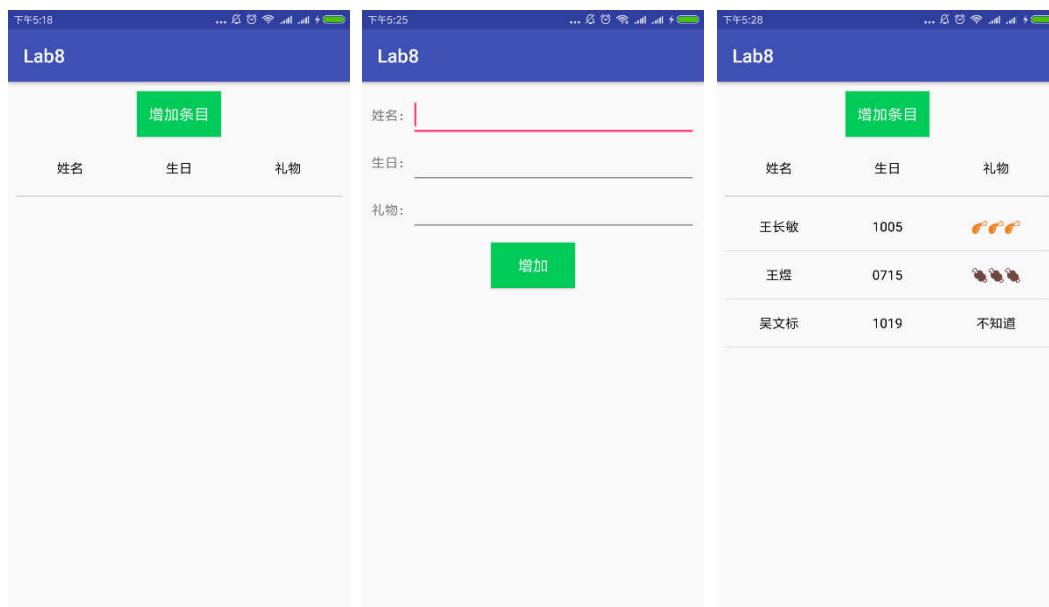
弹出对话框显示是否删除条目；

点击“是” 按钮，删除该条目，并更新主界面生日列表。

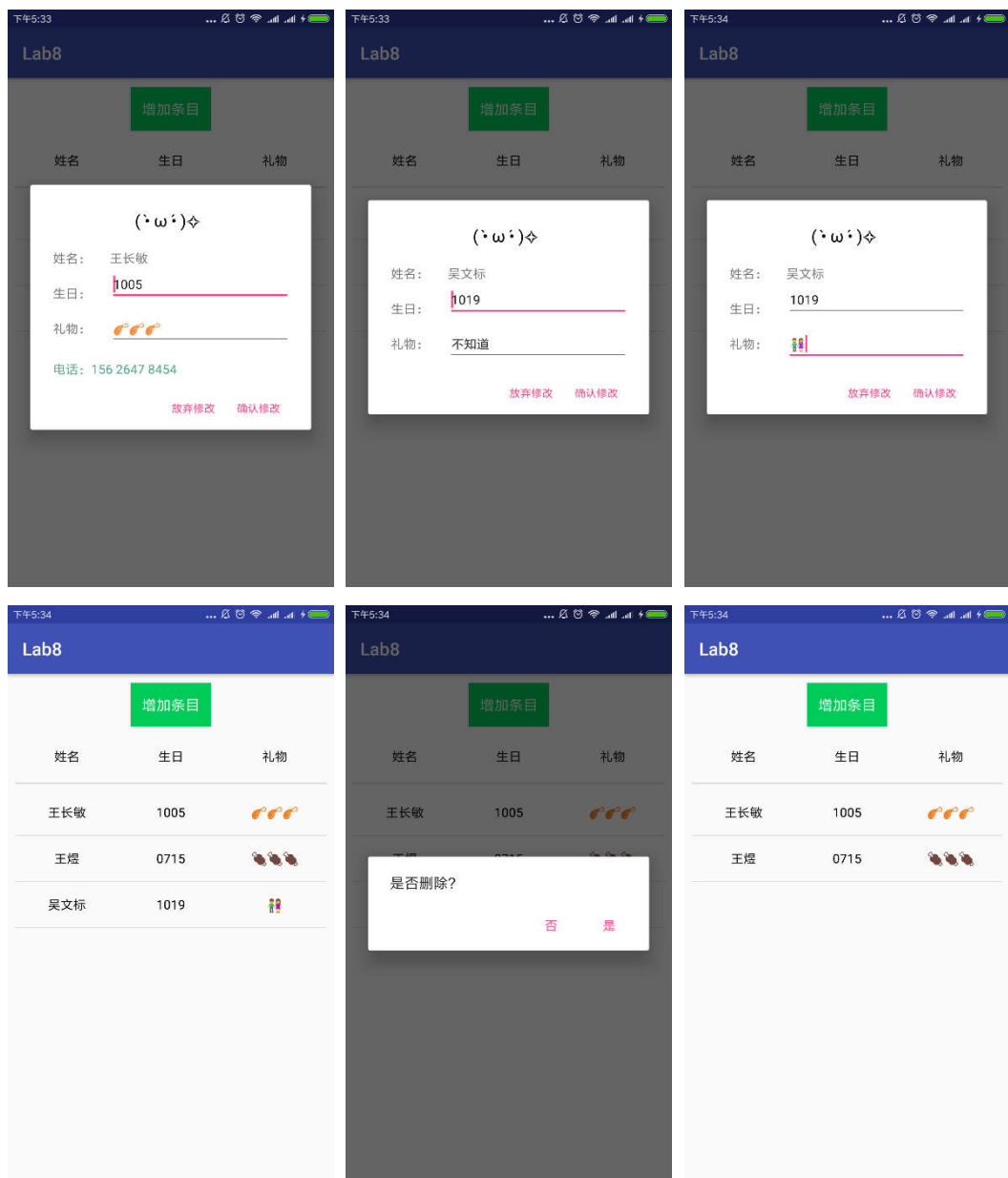
## 四、 课堂实验结果

### (1) 实验截图

首次打开页面以及增加条目后界面如下：



点击条目、修改条目和长按删除：

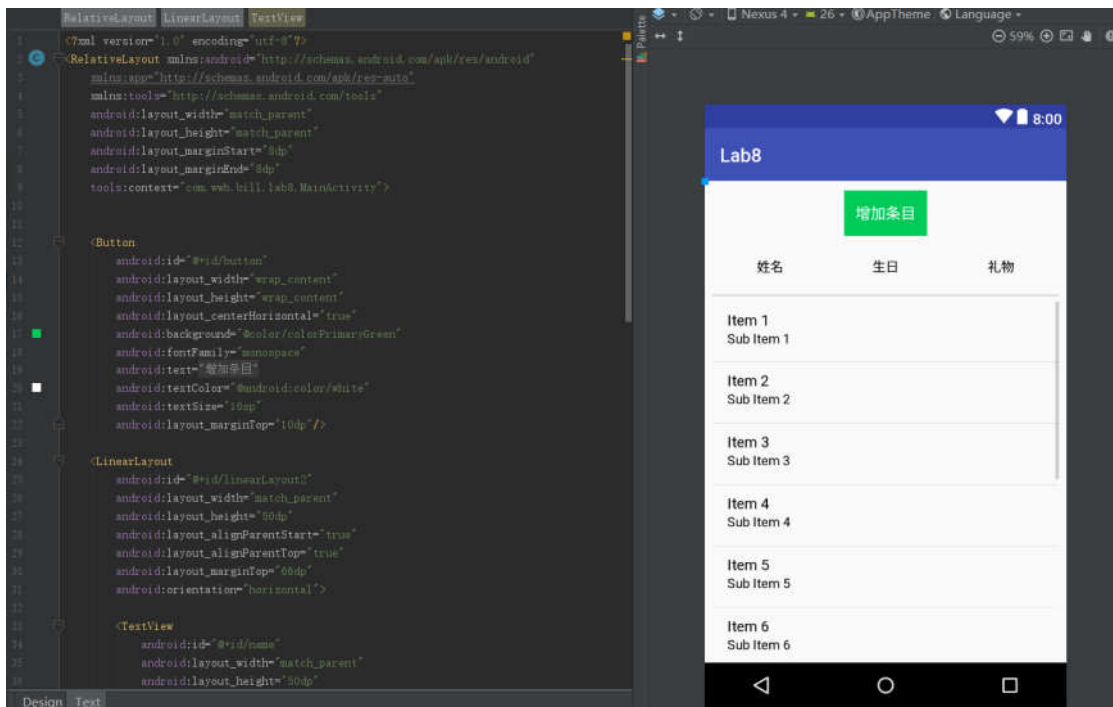


## (2) 实验步骤以及关键代码

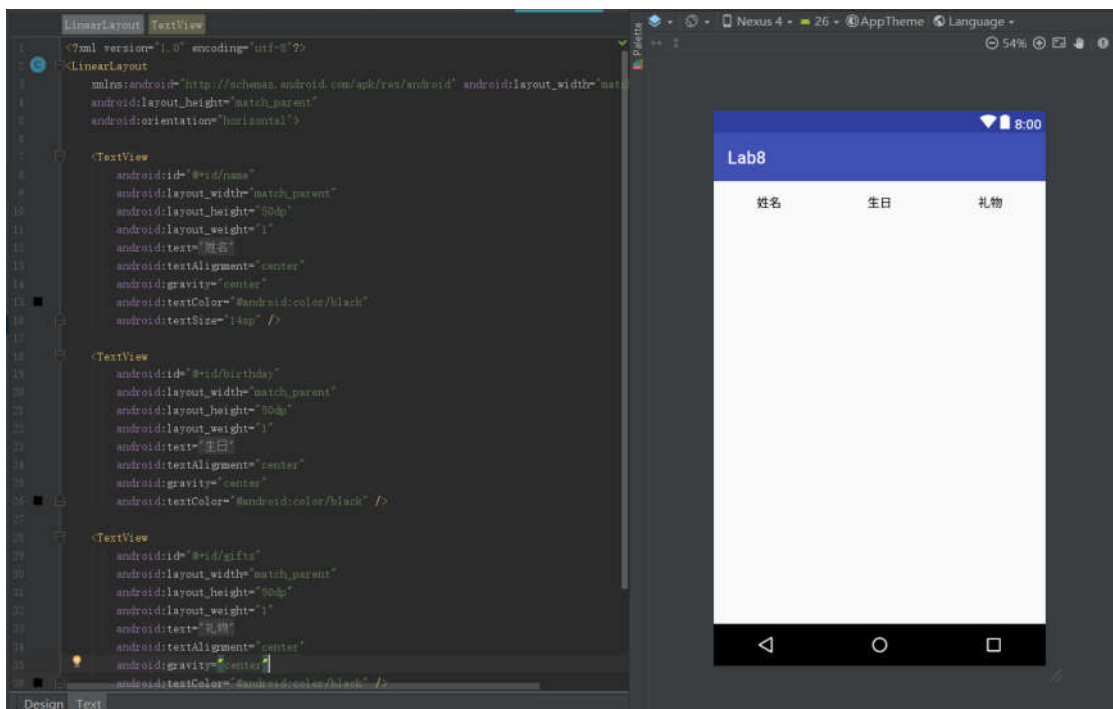
步骤 1: 新建 project，布局界面实现。

1) 页面布局及其部分代码

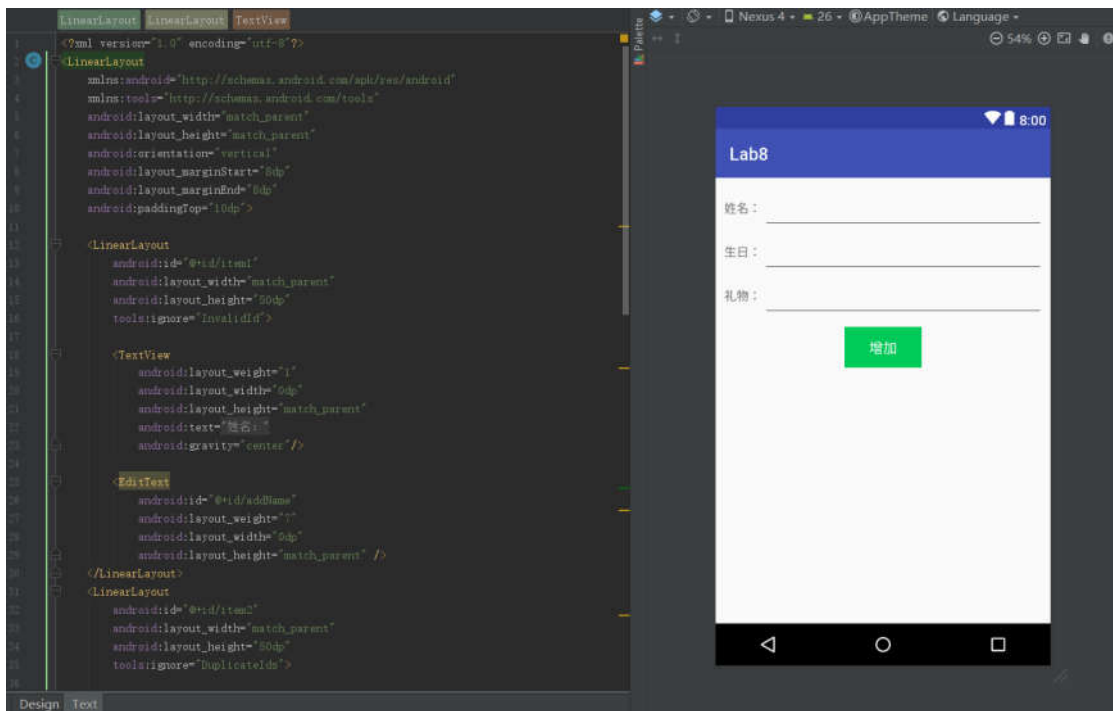
Activity\_main.xml:



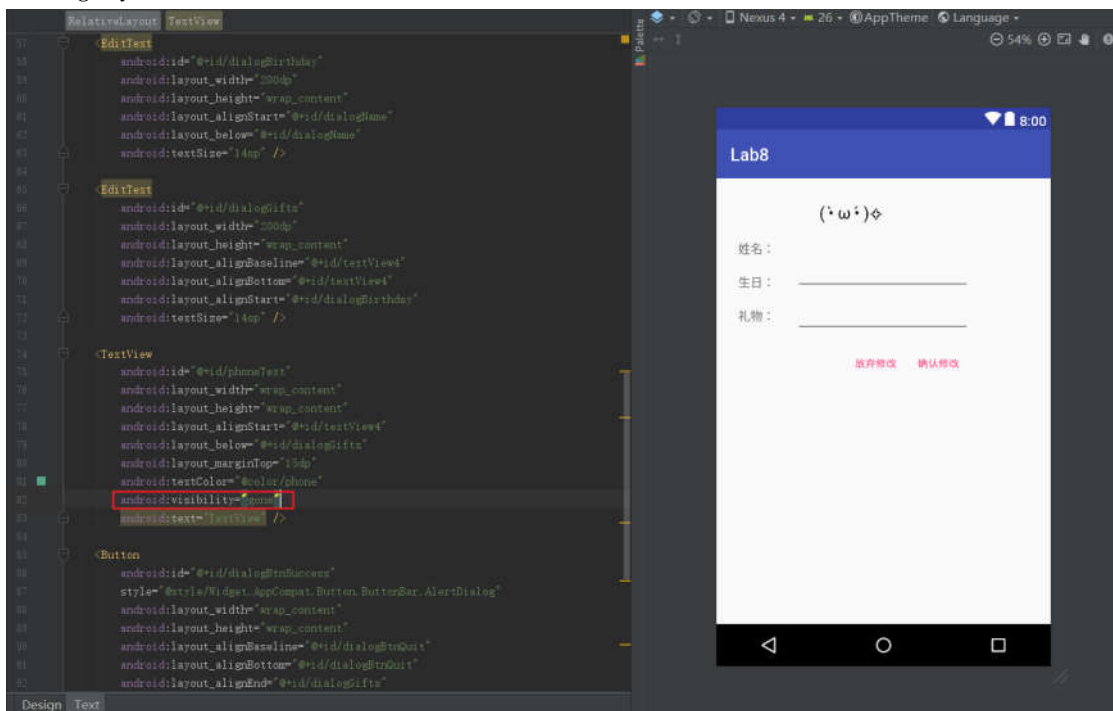
Item.xml:



Activity\_additem.xml



Dialoglayout.xml:



## 步骤 2: Activity 创建

### 1) 清单文件 AndroidManifest.xml

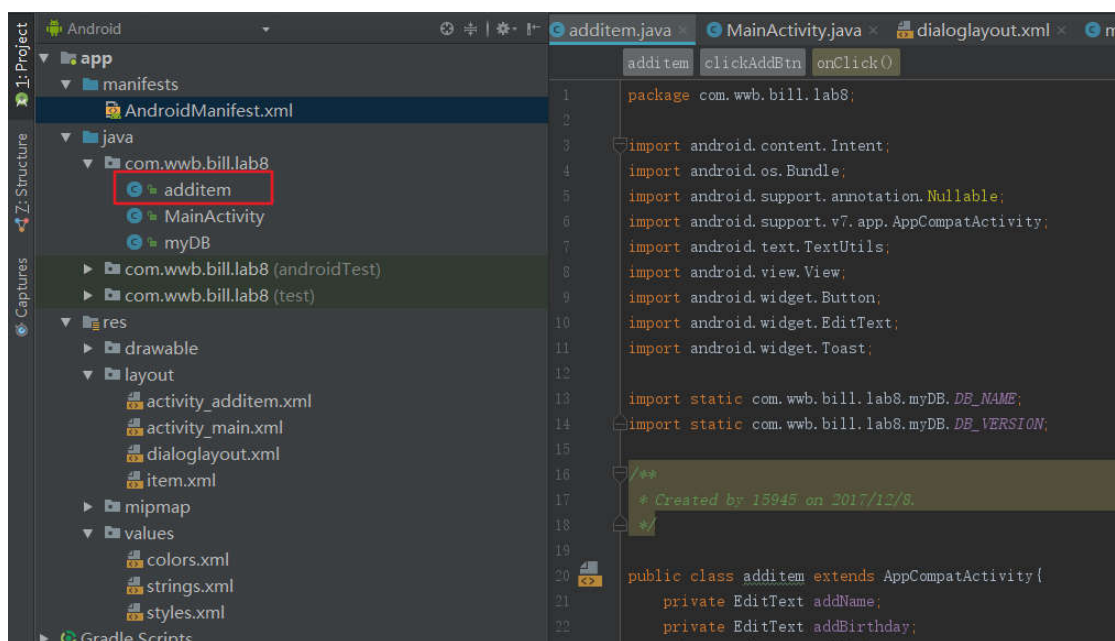
```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.wwb.bill.lab8">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="Lab8"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportRtl="true"
11         android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19         <activity android:name=".additem">
20         </activity>
21     </application>
22     <uses-permission android:name="android.permission.READ_CONTACTS" />
23     <uses-permission android:name="android.permission.CALL_PHONE" />
24 </manifest>

```

框出来部分为需要创建的 activity 和 app 申请的权限。

2) 创建 activity\_additem 的 java 文件。



```

1  package com.wwb.bill.lab8;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.support.annotation.Nullable;
6  import android.support.v7.app.AppCompatActivity;
7  import android.text.TextUtils;
8  import android.view.View;
9  import android.widget.Button;
10 import android.widget.EditText;
11 import android.widget.Toast;
12
13 import static com.wwb.bill.lab8.myDB.DB_NAME;
14 import static com.wwb.bill.lab8.myDB.DB_VERSION;
15
16 /**
17  * Created by 15945 on 2017/12/8.
18  */
19
20 public class additem extends AppCompatActivity {
21     private EditText addName;
22     private EditText addBirthday;

```

步骤 3: 保存生日信息 SQLite 数据库的实现

1) 新建 SQLite 数据库 myDB 类

myDB 类继承自 SQLiteOpenHelper 类, SQLiteOpenHelper 是 SQLiteDatabase 的一个帮助类, 用来管理数据的创建和版本更新。一般的用法是定义一个类继承 SQLiteOpenHelper, 并实现两个回调方法, onCreate(SQLiteDatabase db) 和 onUpgrade(SQLiteDatabase, int oldVersion, int newVersion) 来创建和更新数

数据库，这里我们不需要更新数据库版本所以只需要实现 onCreate(SQLiteDatabase db)。

相关常量、变量：

```
14     private static final String DB_NAME = "Contacts.db";
15     public static final String TABLE_NAME = "Contacts";
16     private static final int DB_VERSION = 1;
```

主体代码：

```
18     private myDB(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
19         super(context, name, factory, version);
20     }
21
22     //数据库的构造函数，传递三个参数的
23     public myDB(Context context, String name, int version) {
24         this(context, name, factory: null, version);
25     }
26
27     //数据库的构造函数，传递一个参数的，数据库名字和版本号都写死了
28     public myDB(Context context) {
29         this(context, DB_NAME, factory: null, DB_VERSION);
30     }
31
32     @Override
33     public void onCreate(SQLiteDatabase sqLiteDatabase) {
34         String CREATE_TABLE = "create table " + TABLE_NAME + "(_id integer primary key , name text, birthday text, gifts text)";
35         sqLiteDatabase.execSQL(CREATE_TABLE);
36     }
37
38     @Override
39     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
40
41     }
```

分析：

```
public myDB(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
    super(context, name, factory, version);
}
```

SQLiteOpenHelper 子类必须要的一个构造函数。通过 super 调用父类的构造函数。

Super、this 区别：

```
18     public myDB(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
19         super(context, name, factory, version);
20     }
21
22     //数据库的构造函数，传递三个参数的
23     public myDB(Context context, String name, int version) {
24         this(context, name, factory: null, version);
25     }
26
27     //数据库的构造函数，传递一个参数的，数据库名字和版本号都写死了
28     public myDB(Context context) {
29         this(context, DB_NAME, factory: null, DB_VERSION);
30     }
```

this 通常指当前对象，super 则指父类的。

## 2) 数据库操作方法实现



```

43 //查询方法
44 public String[] getPoint(int position) {
45     String[] str = new String[3];
46     SQLiteDatabase db = getReadableDatabase();
47     Cursor cursor = db.rawQuery(
48         sql: "select * from Contacts limit ?,?",
49         new String[] { String.valueOf(position),
50             String.valueOf(1) });
51     while (cursor.moveToNext()) {
52         str[0] = cursor.getString(1);
53         str[1] = cursor.getString(2);
54         str[2] = cursor.getString(3);
55     }
56     return str;
57 }
58
59 public boolean hasName(String name) {
60     SQLiteDatabase db = getReadableDatabase();
61     Cursor cursor = db.rawQuery( sql: "select * from Contacts", selectionArgs: null);
62     while (cursor.moveToNext()) {
63         String strName = cursor.getString(1);
64         if(strName.equals(name)) {
65             return true;
66         }
67     }
68     return false;
69 }
70
71 //插入方法
72 public void insert(String name, String birthday, String gifts) {
73     SQLiteDatabase db = getWritableDatabase();
74     ContentValues values = new ContentValues();
75     values.put("name", name);
76     values.put("birthday", birthday);
77     values.put("gifts", gifts);
78     db.insert(TABLE_NAME, nullColumnHack: null, values);
79     db.close();
80 }
81
82 //更新方法
83 public void update(String name, String birthday, String gifts) {
84     SQLiteDatabase db = getWritableDatabase();
85     ContentValues values = new ContentValues();
86     String whereClause = "name = ?"; //主键列名 = ?
87     String[] whereArgs = { name }; //主键的值
88     values.put("name", name);
89     values.put("birthday", birthday);
90     values.put("gifts", gifts);
91     db.update(TABLE_NAME, values, whereClause, whereArgs);
92     db.close();
93 }
94
95 //删除方法
96 public void delete(String name) {
97     SQLiteDatabase db = getWritableDatabase();
98     String whereClause = "name = ?"; //主键列名 = ?
99     String[] whereArgs = { name }; //主键的值
100     db.delete(TABLE_NAME, whereClause, whereArgs);
101     db.close();
102 }

```

分析:

A、getWritableDatabase 对比 getReadableDatabase

```
SQLiteDatabase db = getWritableDatabase();
```

```
SQLiteDatabase db = getReadableDatabase();
```

表面上看，getWritableDatabase 应该是取得一个用于写数据库的 SQLiteDatabase 实例，getReadableDatabase 是取得一个用于读数据库的 SQLiteDatabase 实例。

从 APIW 文档上解读 getWritableDatabase 取得的实例不是仅仅具有写的功能，而是同时具有读和写的功能，同样的，getReadableDatabase 取得的实例也是具对数据库进行读和写的功能。两者的区别在于：

getWritableDatabase 取得的实例是以读写的方式打开数据库，如果打开的数据库磁盘满了，此时只能读不能写，此时调用了 getWritableDatabase 的实例，那么将会发生错误（异常）

getReadableDatabase 取得的实例是先调用 getWritableDatabase 以读写的方式打开数据库，如果数据库的磁盘满了，此时返回打开失败，继而用 getReadableDatabase 的实例以只读的方式去打开数据库。

B、SQLiteDatabase 提供的 execSQL() 方法可以执行 insert、delete、update 和 CREATE TABLE 之类有更改行为的 SQL 语句；rawQuery() 方法用于执行 select 语句。

rawQuery() 方法的第一个参数为 select 语句；第二个参数为 select 语句中占位符参数的值，如果 select 语句没有使用占位符，该参数可以设置为 null。带占位符参数的 select 语句使用例子如下：

```
Cursor cursor = db.rawQuery(  
    sql: "select * from Contacts limit ?,?",  
    new String[] { String.valueOf(position),  
        String.valueOf(1) } );
```

Cursor 是结果集游标，用于对结果集进行随机访问，使用 moveToNext() 方法可以将游标从当前行移动到下一行，如果已经移过了结果集的最后一行，返回结果为 false，否则为 true。getXX(i) 方法可以获取数据库当前游标选中行的某一列的值，下标为 0 到 n。

```
while (cursor.moveToNext()) {  
    str[0] = cursor.getString(1);  
    str[1] = cursor.getString(2);  
    str[2] = cursor.getString(3);  
}
```

C、insert、update、delete 可以用 execSQL() 方法实现，该方法一开始只有一个参数为 SQL 语句，后来重载了有第二个参数占位符，实现起来比较简单，这里因为学习所以使用了第二种方法去实现 insert、update、delete，如下：

```
db.insert(TABLE_NAME, nullColumnHack: null, values);
```

nullColumnHack: 当 values 参数为空或者里面没有内容的时候, 我们 insert 是会失败的 (底层数据库不允许插入一个空行), 为了防止这种情况, 我们要在这里指定一个列名, 到时候如果发现将要插入的行为空行时, 就会将你指定的这个列名的值设为 null, 然后再向数据库中插入。

```
db.update(TABLE_NAME, values, whereClause, whereArgs);
```

```
db.delete(TABLE_NAME, whereClause, whereArgs);
```

其中 values 为键值对的方法存值:

```
ContentValues values = new ContentValues();

values.put("name", name);
values.put("birthday", birthday);
values.put("gifts", gifts);
```

whereClause, whereArgs 相当于带占位符的 sql 语句, 后者为占位符的值。

```
String whereClause = "name = ?"; //主键列名 = ?
String[] whereArgs = { name }; //主键的值
```

#### 步骤 4: 功能实现

##### 1) ListView 实现生日信息列表

```
public void updatelistview() {
    SQLiteDatabase db = helper.getWritableDatabase();
    final Cursor cr = db.query(TABLE_NAME, columns: null, selection: null, selectionArgs: null,
        having: null, orderBy: null,
        groupBy: null);
    String[] ColumnNames = cr.getColumnNames();
    // ColumnNames为数据库的表的列名, getColumnNames()为得到指定table的所有列名

    ListAdapter adapter = new SimpleCursorAdapter(context: this, R.layout.item,
        cr, ColumnNames, new int[] {0, R.id.name, R.id.birthday, R.id.gifts});
    // layout为listview的布局文件, 包括三个TextView, 用来显示三个列名所对应的值
    // ColumnNames为数据库的表的列名
    // 最后一个参数是int[]类型的, 为view类型的id, 用来显示ColumnNames列名所对应的值。view的类型为TextView
    listView.setAdapter(adapter);
}
```

每次增删改数据库就调用一次该函数即可刷新 listView 列表。

```
@Override
protected void onResume() {
    super.onResume();
    updatelistview();
}
```

##### 2) 增加条目页面功能

```

21  private Button addBtn;
22
23  final myDB helper = new myDB( context, this);
24
25  @Override
26  protected void onCreate(@Nullable Bundle savedInstanceState) {
27      super.onCreate(savedInstanceState);
28      setContentView(R.layout.activity_additem);
29
30      initUI();
31      addListener();
32  }
33
34  private void addListener() { addBtn.setOnClickListener(new clickAddBtn()); }
35
36  private void initUI() {
37      addName = (EditText) findViewById(R.id.addName);
38      addBirthday = (EditText) findViewById(R.id.addBirthday);
39      addGifts = (EditText) findViewById(R.id.addGifts);
40      addBtn = (Button) findViewById(R.id.addButton);
41  }
42
43  private class clickAddBtn implements View.OnClickListener {
44      @Override
45      public void onClick(View view) {
46          if(TextUtils.isEmpty(addName.getText())) {
47              Toast.makeText(getApplicationContext(), text: "名字为空, 请完善", Toast.LENGTH_SHORT).show();
48          } else if(helper.hasName(addName.getText().toString())) {
49              Toast.makeText(getApplicationContext(), text: "名字重复啦, 请检查", Toast.LENGTH_SHORT).show();
50          } else {
51              helper.insert(addName.getText().toString(), addBirthday.getText().toString(), addGifts.getText().toString());
52              Intent intent = new Intent();
53              intent.setClass( packageContext, additem.this, MainActivity.class);
54              startActivity(intent);
55          }
56      }
57  }
58
59  }
60

```

框选部分为数据库操作。

成功添加则跳转回到信息列表界面。

### 3) ListView 中 item 点击功能

```

private class clickItem implements android.widget.AdapterView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, final int position, long l) {
        LayoutInflater layoutInflater = LayoutInflater.from(MainActivity.this);
        View view_in = layoutInflater.inflate(R.layout.dialoglayout, (ViewGroup) findViewById(R.id.dialog));
        final AlertDialog.Builder alertDialog = new AlertDialog.Builder(context MainActivity.this);
        final AlertDialog dialog = alertDialog.setView(view_in).show();

        final TextView dialogName = (TextView) view_in.findViewById(R.id.dialogName);
        final EditText dialogBirthday = (EditText) view_in.findViewById(R.id.dialogBirthday);
        final EditText dialogGifts = (EditText) view_in.findViewById(R.id.dialogGifts);
        final Button quitBtn = (Button) view_in.findViewById(R.id.dialogBtnQuit);
        final Button successBtn = (Button) view_in.findViewById(R.id.dialogBtnSuccess);
        final TextView phoneText = (TextView) view_in.findViewById(R.id.phoneText);
        final String[] person = helper.getPoint(position);
        System.out.println(person);
        dialogName.setText(person[0]);
        dialogBirthday.setText(person[1]);
        dialogGifts.setText(person[2]);
        quitBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                dialog.dismiss();
            }
        });
        successBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                helper.update(person[0], dialogBirthday.getText().toString(), dialogGifts.getText().toString());
                updatelistview();
                dialog.dismiss();
            }
        });
    }
}

```

分析:

```

LayoutInflater layoutInflater = LayoutInflater.from(MainActivity.this);
View view_in = layoutInflater.inflate(R.layout.dialoglayout, (ViewGroup) findViewById(R.id.dialog));
final AlertDialog.Builder alertDialog = new AlertDialog.Builder(context MainActivity.this);
final AlertDialog dialog = alertDialog.setView(view_in).show();

```

设置 dialoglayout 为 alertDialog 的界面，并绑定 dialoglayout 的 ViewGroup，便可在当前 layout 操作 dialoglayout 控件，以下为绑定其中 view 的 id 方法

```

final TextView dialogName = (TextView) view_in.findViewById(R.id.dialogName);
final EditText dialogBirthday = (EditText) view_in.findViewById(R.id.dialogBirthday);
final EditText dialogGifts = (EditText) view_in.findViewById(R.id.dialogGifts);
final Button quitBtn = (Button) view_in.findViewById(R.id.dialogBtnQuit);
final Button successBtn = (Button) view_in.findViewById(R.id.dialogBtnSuccess);
final TextView phoneText = (TextView) view_in.findViewById(R.id.phoneText);

```

```

final String[] person = helper.getPoint(position);
System.out.println(person);
dialogName.setText(person[0]);
dialogBirthday.setText(person[1]);
dialogGifts.setText(person[2]);

```

数据库获取当前 item 数据，设置 dialoglayout 中 textView、EditView 值。



#### 4) Dialoglayout 界面按钮功能

```
quitBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        dialog.dismiss();
    }
});
successBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        helper.update(person[0], dialogBirthDay.getText().toString(), dialogGifts.getText().toString());
        updateListView();
        dialog.dismiss();
    }
});
```

其中, dismiss() 方法需作用在. show() 后的 alertDialog 中

```
final AlertDialog dialog = alertDialog.setView(view_in).show();
```

#### 5) 获取通讯录功能

```
// getContentResolver 方法读取联系人列表
Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
    projection: null, selection: null, selectionArgs: null, sortOrder: null);
cursor.getCount();
while(cursor.moveToNext()) {
    String name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
    System.out.println(name);
    if (name.equals(dialogName.getText().toString())) {
        System.out.println(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.ID)));
        String id = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.ID));
        int isHas = Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)));
        System.out.println("isHas: " + isHas);
        if (isHas == 1) {
            Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, projection: null,
                selection: ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + id + " and " +
                    ContactsContract.CommonDataKinds.Phone.TYPE + " = " + ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE, selectionArgs: null, sortOrder: null);
            String phoneNumber = "";
            while(phone.moveToNext()) {
                phoneNumber += phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
            }
            phoneText.setText(phoneNumber);
            phoneText.setVisibility(View.VISIBLE);

            //点击跳转到拨打电话界面
            final String phoneCall = phoneNumber;
            phoneText.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    call(phoneCall);
                }
            });
        }
        break;
    }
}
cursor.close();
```

首先在 AndroidManifest.xml 注册读取通讯录权限, 安装 app 后需手动赋予权限

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

分析:

```
Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI,
    projection: null, selection: null, selectionArgs: null, sortOrder: null);
cursor.getCount();
```

获取通讯录数据库的游标集。

```
while(cursor.moveToNext()) {
    String name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
    System.out.println(name);
    if (name.equals(dialogName.getText().toString())) {
```

移动游标，获取通讯录数据库姓名和 dialoglayout 的姓名进行对比，相等则继续下一步。

```
int isHas = Integer.parseInt(cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER)));
System.out.println("isHas:"+isHas);
if (isHas == 1) {
```

判断是否存在电话号码，存在则进行下一步。

```
String id = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));

//只查询手机电话
Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI, projection: null,
    selection: ContactsContract.CommonDataKinds.Phone.CONTACT_ID + ' = ' + id + " and " +
    ContactsContract.CommonDataKinds.Phone.TYPE+"="+ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE, selectionArgs: null,
    String phoneNumber = "";
while(phone.moveToNext()) {
    phoneNumber += phone.getString(phone.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
}
phoneText.setText("电话: "+phoneNumber);
phoneText.setVisibility(View.VISIBLE);
```

通过 id 查询手机电话然后设置 textView 的显示值。

## 6) ListView 中 item 长按删除功能

```
private class longClickItem implements AdapterView.OnItemClickListener {
    @Override
    public boolean onItemClick(AdapterView<?> adapterView, View view, final int position, long l) {
        final AlertDialog.Builder dialog = new AlertDialog.Builder(context: MainActivity.this);
        dialog.setMessage("是否删除?")
            .setNegativeButton(text: "否", listener: null)
            .setPositiveButton(text: "是", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    String name = helper.getPoint(position)[0];
                    helper.delete(name);
                    updatelistview();
                    //Toast.makeText(getApplicationContext(), name, Toast.LENGTH_SHORT).show();
                }
            }).show();
        return false;
    }
}
```

## 五、课后实验

### (1) 点击电话号码跳转拨打电话界面

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

```

//点击跳转到拨打电话界面
final String phoneCall = phoneNumber;
phoneText.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        call(phoneCall);
    }
});

private void call(String phone) {
    Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:"+phone));
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
}

```

## 六、 实验思考及感想

本次实验是对数据库的操作，这方面和最近学习的几门课程都相差不多所以理论基础还好并没有什么问题，不过实际操作还是会有一些小问题，比如这里会用到的 cursor 游标，然后对比起来 SQLite 比起其他数据库都好操作，所以用起来也挺顺手。然后就是对 alertDialog 的自定义用法更熟悉，是个很强大的东西。

作业要求：

1. 命名要求： 学号\_姓名\_实验编号，例如 15330000\_林 XX\_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。