中山大学数据科学与计算机学院本科生实验报告

(2017 年秋季学期)

课程名称：手机应用平台开发　　任课教师： 刘宁

年级	15 级	专业（方向）	软件工程
学号	15331236	姓名	马朝露
电话	15521122675	Email	1073627941@qq.com
开始日期	2017/12/7	完成日期	2017/12/7

一、 实验题目

数据存储（二）—— 生日备忘录

二、 实现内容

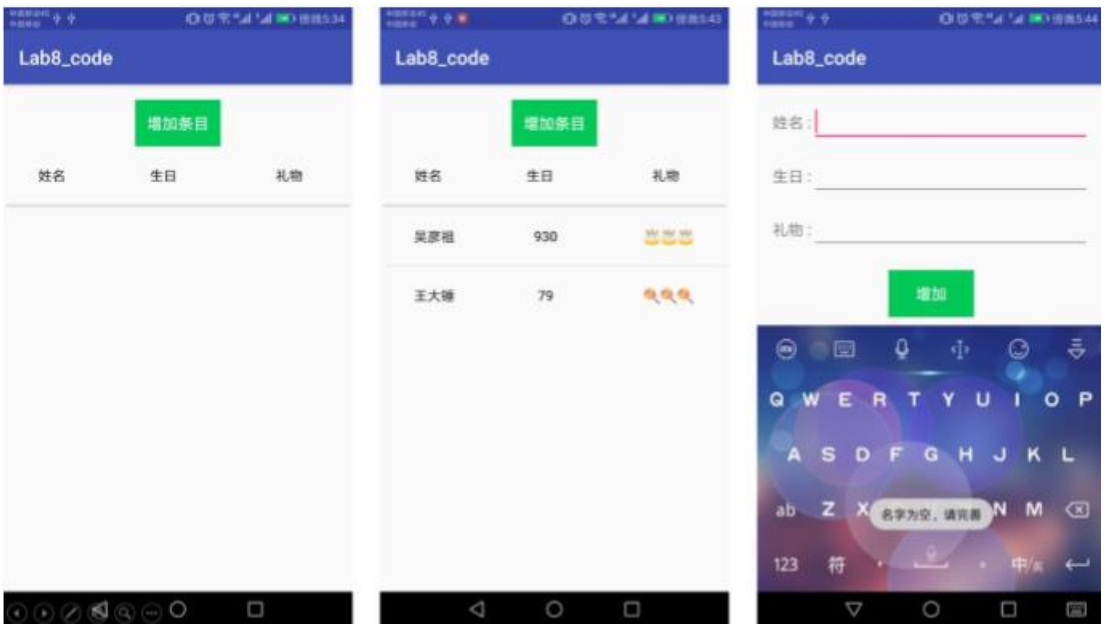
1、 实验目的

学习 SQL 数据库的使用

学习 ContentProvider 的使用

复习 Android 界面编程

2、 实验内容



从左至右：初始界面，添加一部分条目，名字不能为空

从左至右：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。

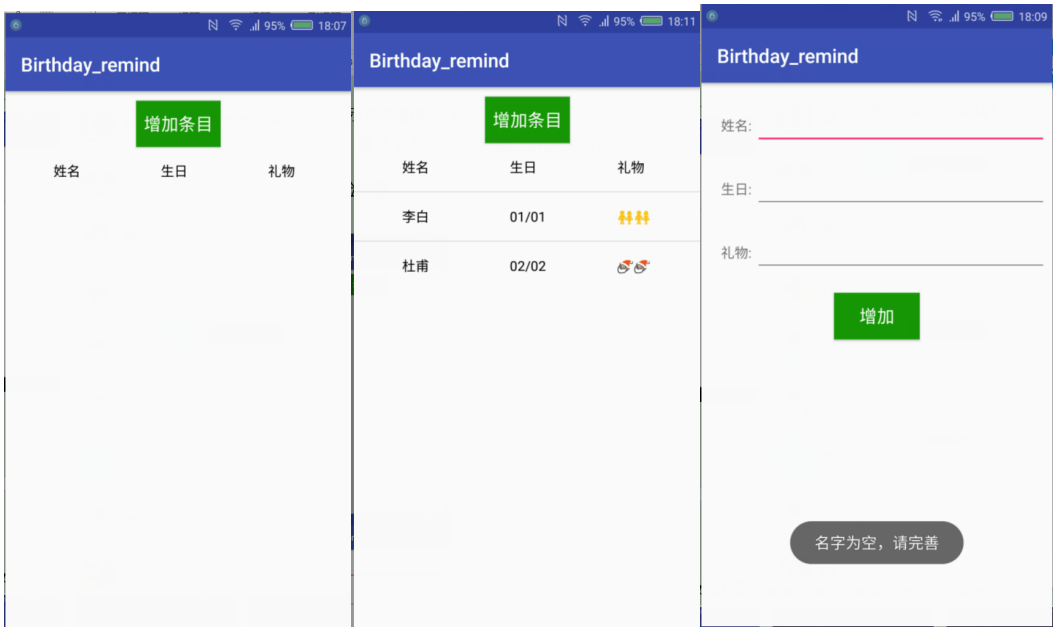
长按条目：

弹出对话框显示是否删除条目；

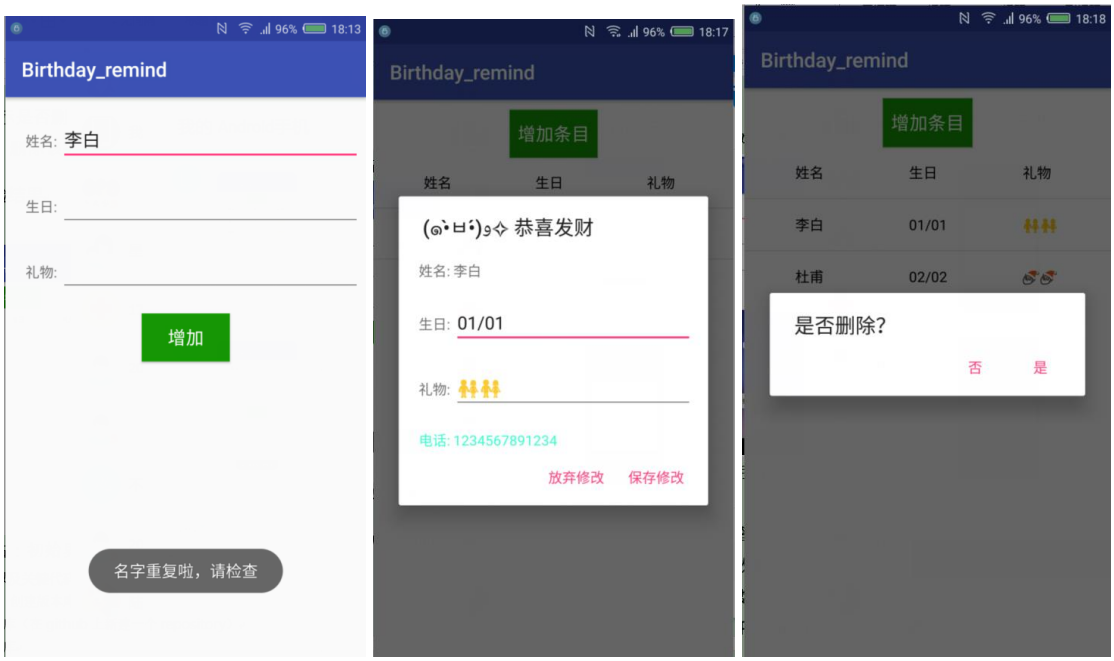
点击“是”按钮，删除该条目，并更新主界面生日列表。

三、 课堂实验结果

（1） 实验截图



从左至右：初始界面，添加一部分条目，名字不能为空



从左至右：名字不能重复，点击条目显示信息（可修改），长按删除条目。

(2) 实验步骤以及关键代码

【1】 主界面的布局

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.zhaoluma.birthday_remind.Birthday_remind">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/add"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="10dp"
        android:text="增加条目"
        android:textColor="#FFFFFF"
        android:textSize="18dp"
        android:background="@color/coloraddbutton"
        />
    <ListView
        android:layout_width="385dp"
        android:layout_height="wrap_content"
        android:id="@+id/birthdaylist"
        app:layout_constraintTop_toBottomOf="@+id/add"
        >
    </ListView>
</android.support.constraint.ConstraintLayout>
```

```xml
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="50dp"
    android:id="@+id/item">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal"
        android:layout_marginLeft="50dp"
        >
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:id="@+id/name"
            android:text="姓名"
```

ardcoded string "姓名", should use @string resource more... (Ctrl+F1)

```xml
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:id="@+id/birthday"
            android:text="生日"
            android:textColor="#000000"/>
        <TextView
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:id="@+id/gift"
```

使用 listview 中的 simpleadpter，其中名为 item 的 layout 是 listview 每一个 item 的布局

【2】 实现 member 类

```java
package com.example.zhaoluma.birthday_remind;

/**
 * Created by zhaoluma on 2017/12/7.
 */

public class member {
    private String name;
    private String birthday;
    private String gift;

    member(String n, String b, String g) {
        name = n;
        birthday = b;
        gift = g;
    }

    public void setName(String n) {name = n;}
    public String getName() {return name;}

    public void setBirthday(String b) {birthday = b;}
    public String getBirthday() {return birthday;}

    public void setGift(String g) {gift = g;}
    public String getGift() {return gift;}
}
```

实现 member 类主要是为了方便数据库的操作，同时使代码更加整洁

【3】 实现数据库

```java
public class myDB extends SQLiteOpenHelper{
    private static final String DB_NAME = "Contacts.db";
    private static final String TABLE_NAME = "Contacts";
    private static final int DB_VERSION = 1;
    public  myDB(Context context) {

        //第三个参数CursorFactory指定在执行查询时获得一个游标实例的工厂类,设置为null,代表使用系统默认的工厂类
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_TABLE = "create table " + TABLE_NAME
                + " (_id integer primary key autoincrement, "
                + "name text , "
                + "birthday text , "
                + "gift text);";
        db.execSQL(CREATE_TABLE);
    }


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

    }
```

默认构造函数以及必须要重载的函数 onCreate 与 onUpdate 的实现，本次实验没有用到 onUpdate

```
public boolean Query(String n) {   // 查询操作，用于判断姓名是否重复
    SQLiteDatabase db = getWritableDatabase();
    Cursor cursor = db.rawQuery("select * from "+ TABLE_NAME + " where name=?",
            new String[]{n});
    if (!cursor.moveToFirst()) return false;
    else {
        cursor.close();
        db.close();
        return true;
    }
}

public void insert(member m) {
    if(!Query(m.getName())) {
        SQLiteDatabase db = getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put("name", m.getName());
        values.put("birthday", m.getBirthday());
        values.put("gift", m.getGift());
        db.insert(TABLE_NAME, null, values);
        db.close();
    }
}

public void update(String n, String b, String g) {
    SQLiteDatabase db = getWritableDatabase();
    String whereClause = "name = ?"; //主键列名，这里是根据名字修改
    String[] whereArgs = {n}; // 主键的值
    ContentValues values = new ContentValues();
    values.put("name", n);
    values.put("birthday", b);
    values.put("gift", g);
    db.update(TABLE_NAME, values, whereClause, whereArgs);
    db.close();
}

public void delete(String n) {
    SQLiteDatabase db = getReadableDatabase();
    String whereClause = "name = ?"; //主键列名，这里是根据名字删除
    String[] whereArgs = {n}; // 主键的值
    db.delete(TABLE_NAME, whereClause, whereArgs);
    db.close();
}
```

增删改查的实现，本次实验查询主要是用在防止插入时重复，所以在插入前先要判断是否已经存在。查询使用了 cursor，当能找到这个名字时返回 true

```
public ArrayList<member> getAllmember() {
    SQLiteDatabase db = getWritableDatabase();
    Cursor cursor = db.rawQuery("select * from "+TABLE_NAME, null);
    ArrayList<member> all = new ArrayList();

    if (cursor != null) {
        if (cursor.moveToFirst()) {
            do {
                String n = cursor.getString(cursor.getColumnIndex("name"));
                String b = cursor.getString(cursor.getColumnIndex("birthday"));
                String g = cursor.getString(cursor.getColumnIndex("gift"));
                member m = new member(n, b, g);
                all.add(m);
            } while (cursor.moveToNext());
        }
    }
    cursor.close();
    db.close();
    return all;
}
```

除了增删改查外，数据库类中要实现一个获取全部成员的函数，以便在主界面中显示数据库内容。查找数据库内所有成员，逐行加入到 list 中，最后返回 list。

【4】 实现 mainactivity

```
// 每次oncreate时根据数据库的变化更新listview
private void getdata() {
    ArrayList<member> al = mydb.getAllmember();
    for (int i = 0; i < al.size(); i++) {
        member m = al.get(i);
        Map<String, Object> map = new HashMap<>();
        map.put("name", m.getName());
        map.put("birthday", m.getBirthday());
        map.put("gift", m.getGift());
        mDatas.add(map);
        adapter.notifyDataSetChanged();
    }

}
```

在 oncreate 函数中，要将数据库中的所有内容加载到 listview 中去。这个函数是用于获取数据库中的数据

```
// 长按删除
final AlertDialog.Builder alertDialog = new AlertDialog.Builder(this);
birthdaylist.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        p = position;
        if (!mDatas.get(p).get("name").toString().equals("姓名")) {
            alertDialog.setTitle("是否删除？").setNegativeButton("否",null).
                    setPositiveButton("是", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            String dm = mDatas.get(p).get("name").toString();
                            mydb.delete(dm);
                            mDatas.remove(p);
                            adapter.notifyDataSetChanged();
                            Toast.makeText(Birthday_remind.this, "成功删除", Toast.LENGTH_SHORT).show();
                        }
                    }).create();
            alertDialog.show();
        }
        return true;
    }
});
```

长按删除

```
birthdaylist.setOnItemClickListener((parent, view, position, id) -> {
        p = position;     // 锁定点击对象
        if (!mDatas.get(p).get("name").toString().equals("姓名")) {
            layoutInflater = LayoutInflater.from(Birthday_remind.this);
            dialogview = layoutInflater.inflate(R.layout.dialogview,null);
            dialogname = (TextView)dialogview.findViewById(R.id.dialogname);
            dialogbirthday = (EditText)dialogview.findViewById(R.id.dialogbirthday);
            dialoggift = (EditText)dialogview.findViewById(R.id.dialoggift);
            phonetext = (TextView)dialogview.findViewById(R.id.phonetext);
            dialogphone = (TextView)dialogview.findViewById(R.id.dialogphone);
            dialogname.setText(mDatas.get(p).get("name").toString());
            dialogbirthday.setText(mDatas.get(p).get("birthday").toString());
            dialoggift.setText(mDatas.get(p).get("gift").toString());
            if (number(mDatas.get(p).get("name").toString()) != "") {
                dialogphone.setText(number(mDatas.get(p).get("name").toString()));
            } else {
                dialogphone.setText("无");
            }
```

```
        AlertDialog.Builder alertDialog1 = new AlertDialog.Builder(Birthday_remind.this);
        alertDialog1.setTitle("(๑•ㅂ•)و✧ 恭喜发财").setView(dialogview)
                .setNegativeButton("放弃修改",null)
                .setPositiveButton("保存修改", (dialog, which) -> {
                        mydb.update(mDatas.get(p).get("name").toString(),
                                dialogbirthday.getText().toString(),
                                dialoggift.getText().toString());
                        Map<String, Object> map = new HashMap<>();
                        map.put("name", mDatas.get(p).get("name").toString());
                        map.put("birthday", dialogbirthday.getText().toString());
                        map.put("gift", dialoggift.getText().toString());
                        mDatas.set(p,map);
                        adapter.notifyDataSetChanged();
                        Toast.makeText(Birthday_remind.this, "成功修改", Toast.LENGTH_SHORT).show();
                }).create();
        alertDialog1.show();
    }
```

点击弹出对话框 dialog，这里使用了自定义 view 的对话框

```
// 根据姓名查找通讯录内电话号码
public String number(String n) {
    // 使用ContentResolver查找联系人数据
    Cursor cursor = getContentResolver().query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
    // 遍历查询结果，找到所需号码
    while (cursor.moveToNext()) {
        // 获取联系人ID
        String contactId = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts._ID));
        // 获取联系人的名字
        String contactName = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));

        if (n.equals(contactName)) {
            //使用ContentResolver查找联系人的电话号码
            Cursor phone = getContentResolver().query(ContactsContract.CommonDataKinds
            .Phone.CONTENT_URI, null, ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = "
            + contactId, null, null);
            if (phone.moveToNext()) {
                String phoneNumber = phone.getString(phone.getColumnIndex(ContactsContract
                .CommonDataKinds.Phone.NUMBER));
                return phoneNumber;
            }
        }
    }
    return "";
}
```

根据姓名查找电话。首先要获取联系人数据，遍历查询结果，获取联系人 ID，姓名。如果当前的联系人姓名恰好等于我们要查找的，则根据当前联系人的 ID 去找电话号码，如果找到则返回 string 类型的电话号码。

【5】 实现 Add_item 类

```
private myDB mydb;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_item);

    mydb = new myDB(getApplicationContext());
    additem = (Button)findViewById(R.id.additem);
    addname = (EditText)findViewById(R.id.addname);
    addbirthday = (EditText)findViewById(R.id.addbirthday);
    addgift = (EditText)findViewById(R.id.addgift);

    additem.setOnClickListener((v) -> {
            String name = addname.getText().toString();
            String birthday = addbirthday.getText().toString();
            String gift = addgift.getText().toString();

            if (name.isEmpty()) {
                Toast.makeText(getApplicationContext(),
                        "名字为空，请完善", Toast.LENGTH_SHORT).show();
            } else {
                if (mydb.Query(name)) {
                    Toast.makeText(getApplicationContext(),
                            "名字重复啦，请检查", Toast.LENGTH_SHORT).show();
                } else {
                    member m = new member(name, birthday, gift);
                    mydb.insert(m);
                    Intent intent = new Intent(Add_item.this, Birthday_remind.class);
                    startActivity(intent);
                }
            }

    });
```

这里注意可以重现 new 一个 db 而不一定要使用 mainactivity 中的 db，因为重新建立数据库是会判断是否重名，如果重名了，则不改变，所以我们一直操作的都是同一个数据库

【6】 添加访问通讯录的权限

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

```
public static void verifyContactsPermission(Activity activity) {
    try {
        int hascontactPermission = ActivityCompat.checkSelfPermission(activity,
                "android.permission.READ_CONTACTS");
        if (hascontactPermission != PackageManager.PERMISSION_GRANTED) {
            // 没有权限，去权限，会弹出对话框
            ActivityCompat.requestPermissions(activity, new String[] {"android.permission.READ_CONTACTS"},
                    1);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

四、 实验思考及感想

实验中遇到的问题：

【1】 第一次点击 item 能进入更改界面，第二次点击 item 则闪退

解决方法：dialog 使用了是自定义的 view，刚开始如下图的两句代码是放在点击 item 响应外的，也就是说 dialogview 在使用后释放了，所以只有第一次点击时能成功进入 dialog 界面，第二次点击时 dialogview 不存在，所以会闪退。将下图两句代码放入点击 item 响应事件内部，即每次点击就获取一个 dialogview

```
layoutInflater = LayoutInflater.from(Birthday_remind.this);
dialogview = layoutInflater.inflate(R.layout.dialogview,null);
```

【2】 不知道如何实现更新 listview，刚开始想的是可以根据姓名删除旧的，然后再插入更新后的，但是这样位置会错乱。

解决方法：发现了 ArrayList 有 set 函数，参数为当前的位置，和值。

【3】 手机通讯录权限问题。

解决方法：参考音乐播放器那次获取手机内存的权限，改一些参数使得安装 app 时，如果没有权限时弹出对话框，获取权限。

实验感想：

【1】 总体来说比较顺利，因为期中项目已经实现过数据库，所以这次使用数据库没有什么问题。

【2】 了解了 SQLite 数据库

SQLite 数据库特点 ： 更加适用于嵌入式系统，嵌入到使用它的应用程序中；占用非常少，运行高效可靠，可移植性好 ； 提供了零配置（zero-configuration）运行模式

SQLite 数据库不仅提高了运行效率，而且屏蔽了数据库 使用和管理的复杂性，程序仅需要进行最基本的数据操作，其他操作可以交给进程内部的数据库引擎完成

【3】 学习了 ContentProvider

ContentProvider（数据提供者）是在应用程序间共享数 据的一种接口机制

提供了更为高级的数据共享方法，应用程序可以指定需 要共享的数据，而其他应用程序则可以在不知数据来源、 路径的情况下，对共享数据进行查询、添加、删除和更 新等操作

许多 Android 系统的内置数据通过 ContentProvider 提供 给用户使用，例如通讯录、音视频文件和图像文件等