Android 案例开发 实验报告

成绩	

实验名称 实验二 新增支出模块和查看我的支出模块的设计与调试

专业班级 软件 **172** 班级学号 **201715030208**

姓 名 朱洪龙 实验日期 **2019/03/20**

（报告内容包括：实验题目、实验环境、实验步骤、实验结果、遇到的问题和解决办法、实验小结等。

【实验题目】

1. 首先创建新增支出模块和查看我的支出模块的布局文件(新增收入和我的收入与此类似)
2. 输入新增支出类和我的支出类代码
3. 最后实现保存和取消按钮的点击事件（即创建将相关数据写入数据库的 DAO 类）

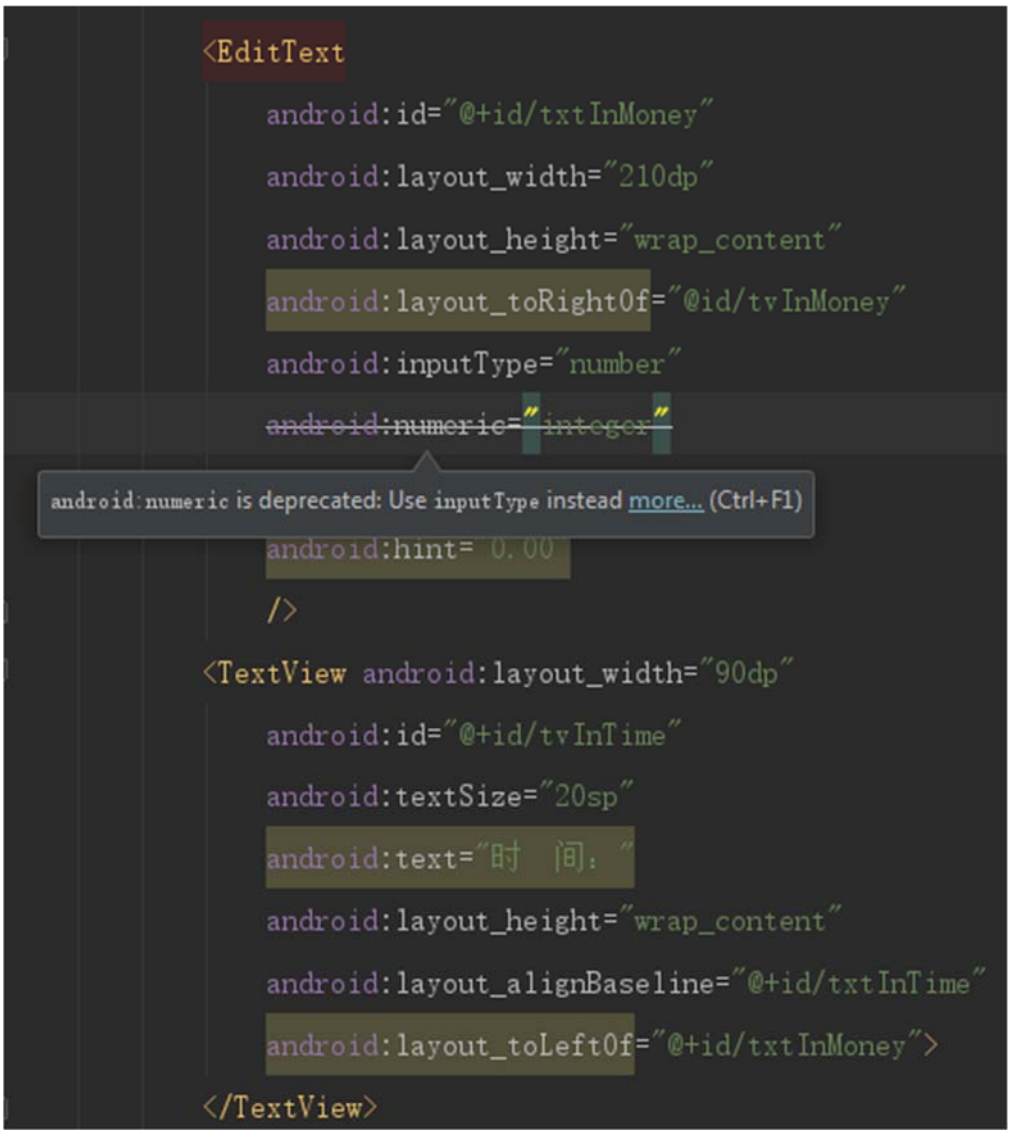
【实验环境】

OS:Windows 10
IDE:Android Studio 2.3.2
AVD:Nexus 5X API 23
minSdkVersion:21(android 5.0)
targetSdkVersion:23(android 6.0)

【实验步骤、程序调试过程中所有遇到的问题和解决办法】

1. 创建新增支出布局文件

整体布局都差不多，最外层为垂直线性布局，里面分为三部分（标题、内容标签项、按钮），分别用线性布局包裹，而里面都是采用相对布局来设置。




```
<EditText
    android:id="@+id/txtInMoney"
    android:layout_width="210dp"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/tvInMoney"
    android:inputType="number"
    android:numeric="integer"
    android:hint="0.00"
    />
<TextView android:layout_width="90dp"
    android:id="@+id/tvInTime"
    android:textSize="20sp"
    android:text="时  间："
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/txtInTime"
    android:layout_toLeftOf="@+id/txtInMoney">
</TextView>
```

android:numeric="integer"这条已废弃，可直接使用 inputType 声明输入的类型

```
<TextView android:layout_width="90dp"
    android:id="@+id/tvType"
```

```
        android:textSize="20sp"
        android:text="类　别："
        android:layout_height="wrap_content"
        android:layout_alignBottom="@id/spType"(这个可以删除，有下面两个足以定位)
       android:layout_alignBaseline="@id/spType"
        android:layout_alignStart="@id/tvTime">
</TextView>
<Spinner android:id="@+id/spType"
        android:layout_width="210dp"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/tvType"
        android:layout_below="@id/txtTime"
        android:entries="@array/outtype"
        />
```

android:singleLine=”false”可不写，已默认为 false

2. 创建我的支出布局文件

整体两部分（标题和显示数据的 ListView），设置空间的占比为 0.06:0.94，垂直线性布局，标题居中；显示数据的部分设置为总是画出滚动条。

3. 新增支出类和我的支出类代码

（1）在 Activity 中创建 AddOutaccount 类，onCreate 中设置布局文件并获取所有组件，设置时间框中的时间为本地时间（使用 Calendar 和 StringBuffer 或 StringBuilder 都可），接着设置时间框的点击事件，在 showDialog()方法和 onCreateDialog()方法已经弃用了，查了官方文档说明是改用了继承自 Fragment 的 DialogFragment(只不过现在这样写还是可以运行的)


```
txtTime.setOnClickListener((arg0) -> {
        // TODO Auto-generated method stub
        showDialog(DATE_DIALOG_ID);// 显示日期选择对话框
}
```

'showDialog(int)' is deprecated more... (Ctrl+F1)

```
@Override
protected Dialog onCreateDialog(int id){// 重写onCreateDialog方法
    switch (id) {
        case DATE_DIALOG_ID:// 弹出日期选择对话框
            return new DatePickerDialog(this, mDateSetListener, mYear, mMonth,
                    mDay);
    }
    return null;
}
```

自己亲自试了官方给出的例子：

```
import android.app.Activity;
import android.app.DialogFragment;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.os.Bundle;

/**
 * Created by mayn on 2019/3/9.
 */

public class Demo extends Activity{
    int mStackLevel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        showDialog();
    }

    void showDialog() {
        mStackLevel++;

        // DialogFragment.show() will take care of adding the fragment
        // in a transaction.  We also want to remove any currently showing
        // dialog, so make our own transaction and take care of that here.
        FragmentTransaction ft =
getFragmentManager().beginTransaction();
        Fragment prev =
getFragmentManager().findFragmentByTag("dialog");
        if (prev != null) {
            ft.remove(prev);
        }
        ft.addToBackStack(null);

        // Create and show the dialog.
        DialogFragment newFragment =
MyDialogFragment.newInstance(mStackLevel);//上转型
        newFragment.show(ft, "dialog");
    }
}
```

```java
package com.livejq.www.dialog_demo;

import android.app.DialogFragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class MyDialogFragment extends DialogFragment {

    int mNum;
    /**
     * Create a new instance of MyDialogFragment, providing "num"
     * as an argument.
     */
    static MyDialogFragment newInstance(int num) {
        MyDialogFragment f = new MyDialogFragment();

        // Supply num input as an argument.
        Bundle args = new Bundle();
        args.putInt("num", num);
        f.setArguments(args);

        return f;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mNum = getArguments().getInt("num");

        // Pick a style based on the num.
        int style = DialogFragment.STYLE_NORMAL, theme = 0;
        switch ((mNum-1)%6) {
            case 1: style = DialogFragment.STYLE_NO_TITLE; break;
            case 2: style = DialogFragment.STYLE_NO_FRAME; break;
            case 3: style = DialogFragment.STYLE_NO_INPUT; break;
            case 4: style = DialogFragment.STYLE_NORMAL; break;
            case 5: style = DialogFragment.STYLE_NORMAL; break;
            case 6: style = DialogFragment.STYLE_NO_TITLE; break;
            case 7: style = DialogFragment.STYLE_NO_FRAME; break;
            case 8: style = DialogFragment.STYLE_NORMAL; break;
        }
        switch ((mNum-1)%6) {
```

```
            case 4: theme = android.R.style.Theme_Holo; break;
            case 5: theme = android.R.style.Theme_Holo_Light_Dialog;
break;
            case 6: theme = android.R.style.Theme_Holo_Light; break;
            case 7: theme = android.R.style.Theme_Holo_Light_Panel;
break;
            case 8: theme = android.R.style.Theme_Holo_Light; break;
        }
        setStyle(style, theme);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_my_custom_dialog,
container, true);

        return v;
    }

}
```


5:16
弹出会话窗口
请输入密码：
请输入密码
登录
取消

其大概就是继承 DialogFragment 并设置对话框的布局风格主题。首先通过设置一个静态方法传入一个参数（父类有设置参数的方法），来生成 style 和 theme 风格主题，之后就是重写 onCreate 和 onCreateView（设置布局）方法了。在主窗口中，每个 DialogFragment 都当做一个事务来处理（每个事务都有唯一的标签，相当于 ID）,开始时判断有无我要设置的会话的标签，有则从事务链中移除，然后通过上转型生成 DialogFragment 对象并调用其 show 方法弹出对话框。相比于以前，这种对话框的设置更加的模块化和有自主性。

回到主题：mDateSetListener 为 DatePickerDialog.OnDateSetListener，是一个当用户点击了需要设置的那个日期时所调用的方法，里面执行了 onDateSet 方法。

（2）在 Activity 中创建 Outaccountinfo 类，此类主要将查询到的数据放入 ListView 中并设置 setOnItemClickListener 事件

4. 实现保存和取消按钮的点击事件

取消按钮点击即为恢复所有输入框的默认值（大多为空），保存按钮则将所有输入框中的值设置到所创建的模板中并调用相关的操作对象使用其写入的方法将数据存入数据库中，最后给出提示。这里唯一要注意的是如何取出 Spinner 组件中的数据，其对象可调用 getSelectedItem()方法。

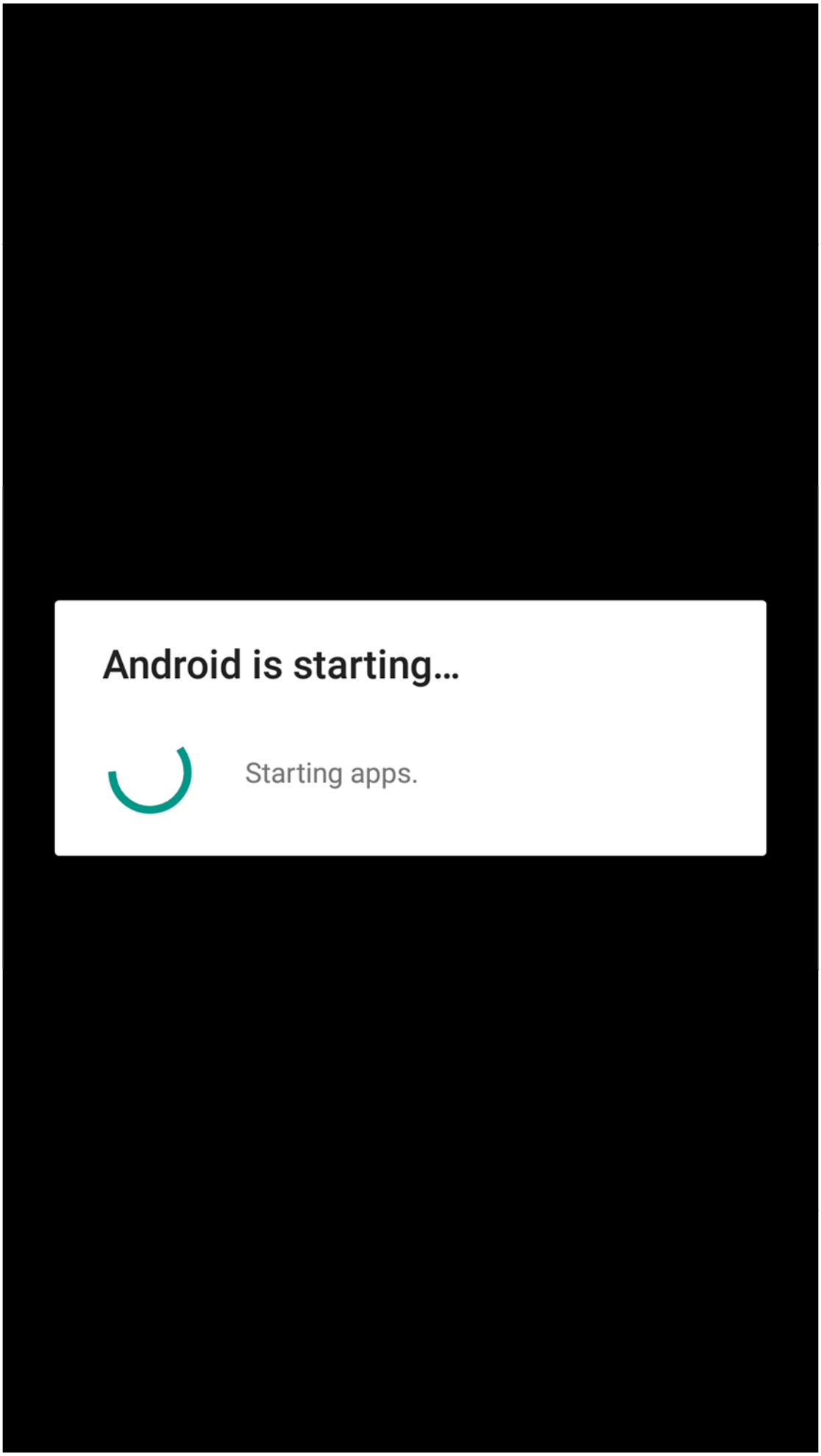
5. 查看我的支出

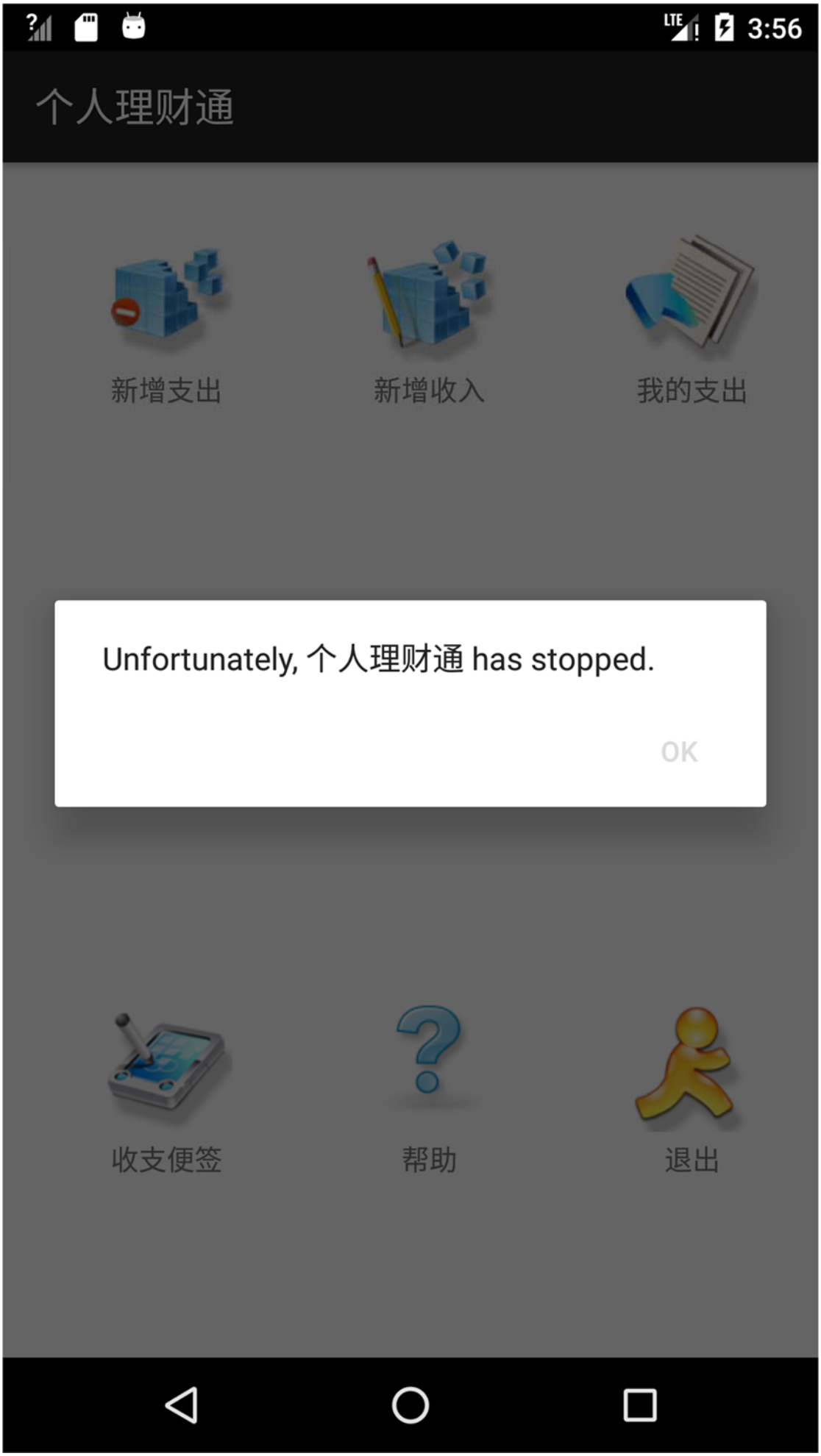
在这里首先实现显示数据，然后再讲对 ListView 设置 setOnItemClickListener 事件。这里将显示数据写成一个方法 ShowInfo()，记得我的支出设置的布局文件里有 ListView，所以可以使用适配器，首先获取需要获取显示的数据（对象数组的形式）来构造一个适配器，接着再对 ListView 对象设置 setAdapter。获取数据即调用数据库操作对象 DAO 中的方法返回一个集合数据（每个数据都是一个支出模板），然后再把它们遍历出来并进行格式化处理（拼接）,保存到事先创建的根据返回的集合大小所设的字符数组中。

收入模块的设计与上述类似。

【实验结果】

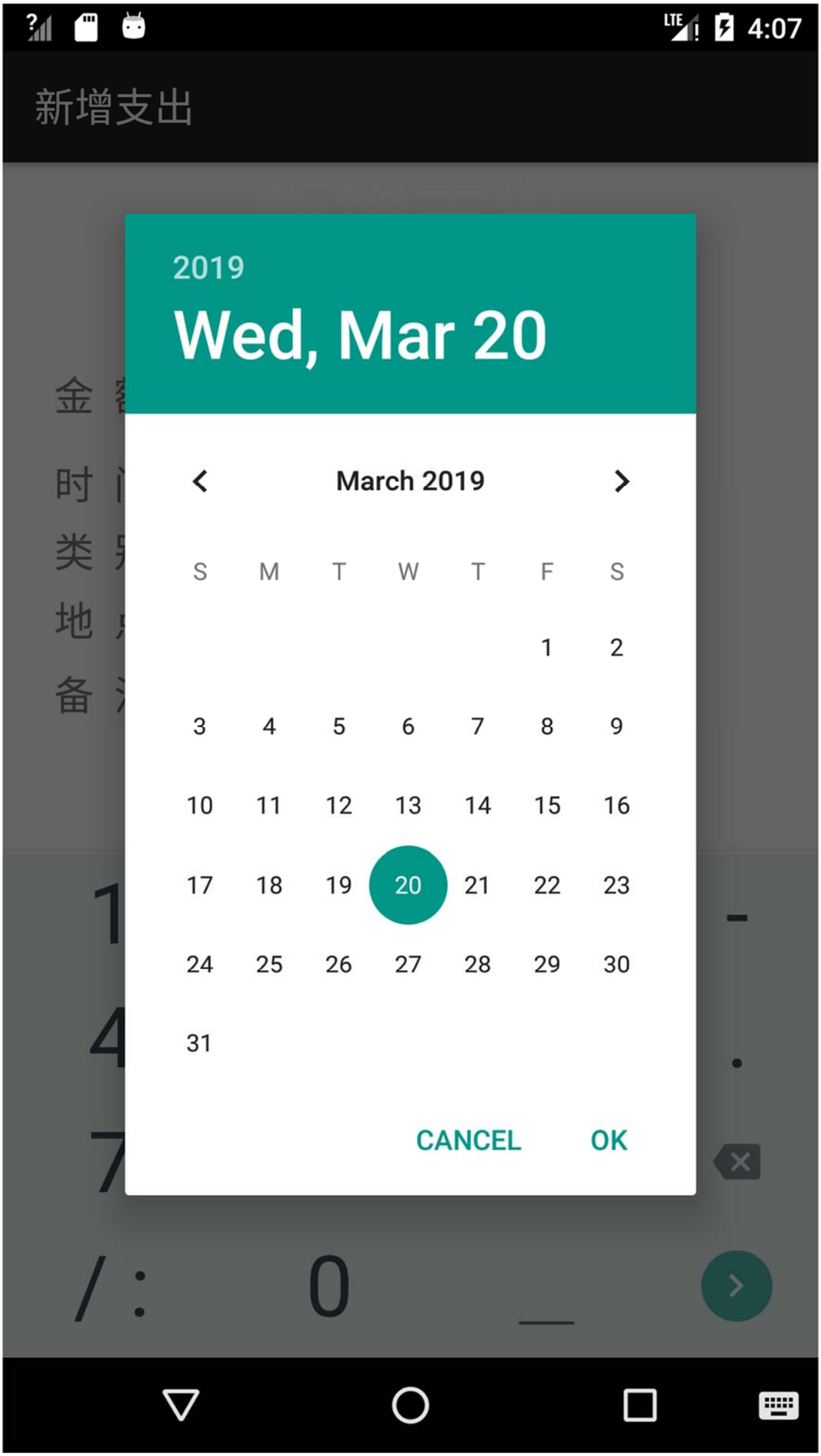
支出模块：


Android is starting…
Starting apps.

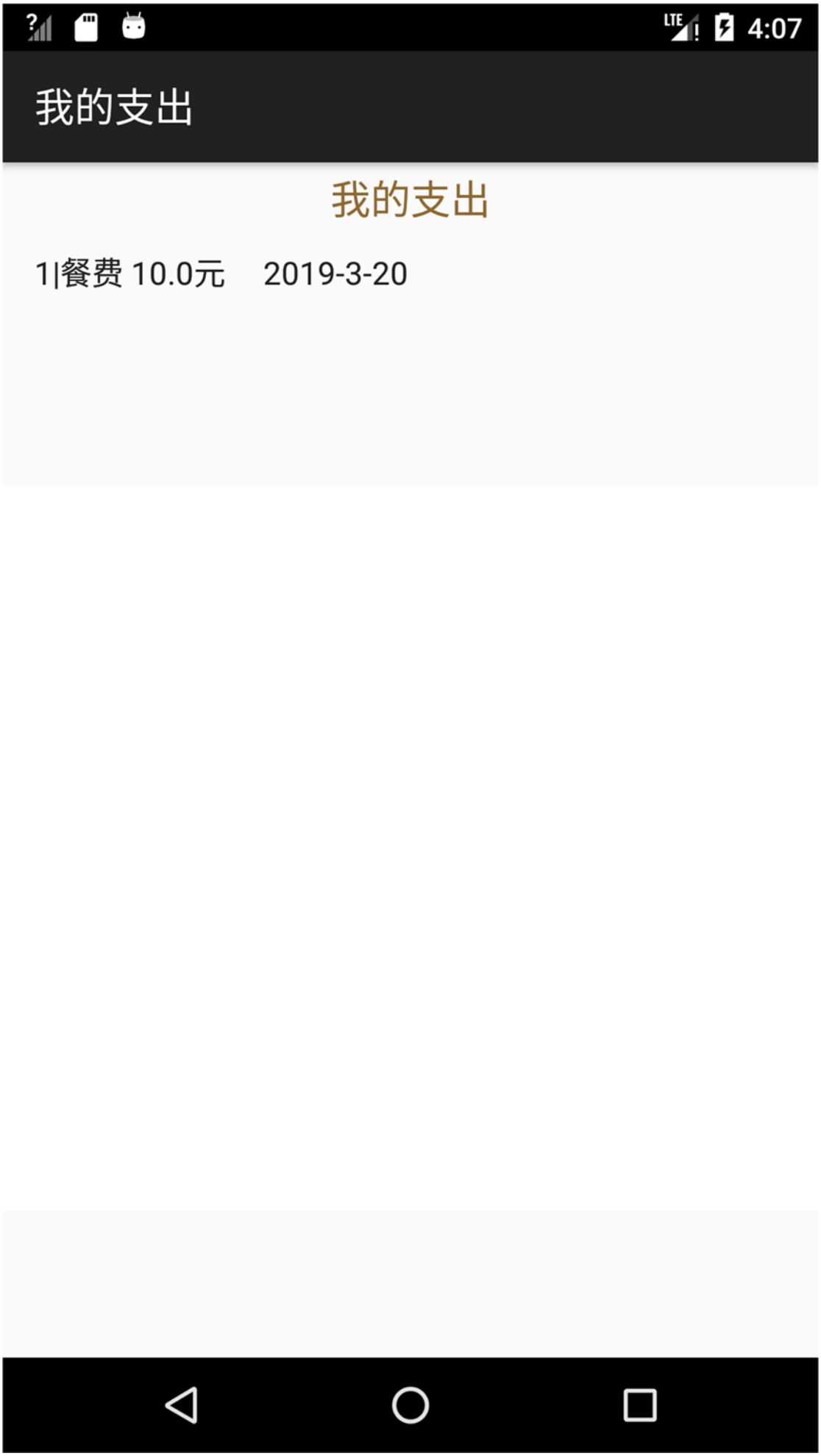

3:56
个人理财通
新增支出
新增收入
我的支出
Unfortunately, 个人理财通 has stopped.
OK
收支便签
帮助
退出

发现是布局文件支出和收入倒置了，修正后：


4:06
新增支出
新增支出
金 额:
0.00
时 间:
2019-3-20
类 别:
餐费
地 点:
备 注:
1
2
3
-
4
5
6
.
7
8
9
/ :
0
_

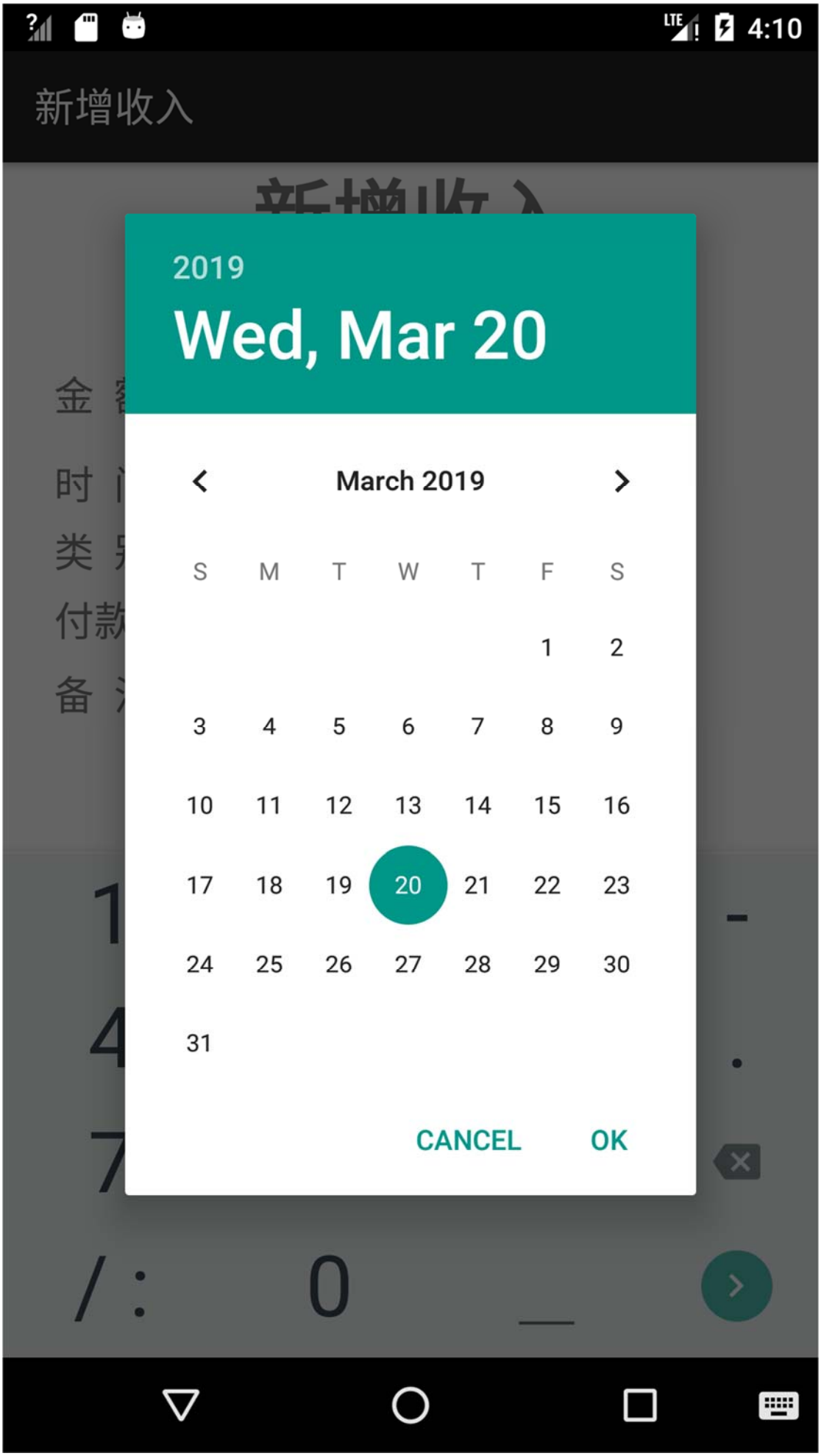

4:07
新增支出
2019
Wed, Mar 20
March 2019
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
CANCEL
OK


4:07
新增支出
新增支出
金 额:
10
时 间:
2019-3-20
类 别:
餐费
地 点:
canteen
备 注:
test
〖新增支出〗数据添加成功!
保存
取消


4:07
我的支出
我的支出
1|餐费 10.0元 2019-3-20

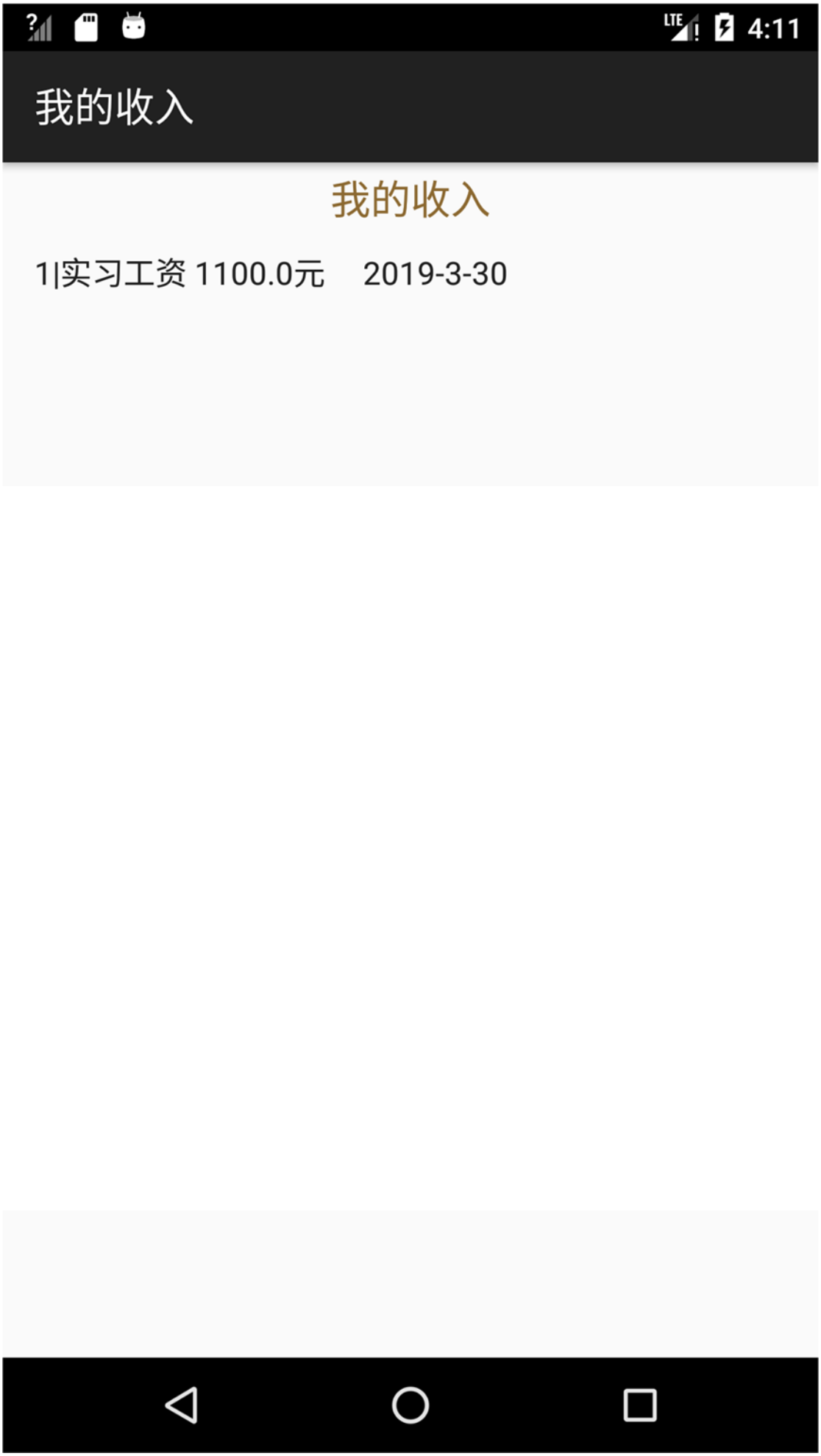
收入模块：


4:10
新增收入
新增收入
金 额:
1100
时 间:
2019-3-20
类 别:
生活费
付款方:
备 注:
保存
取消


4:10
新增收入
2019
Wed, Mar 20
March 2019
S M T W T F S
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
CANCEL
OK


4:11
新增收入
新增收入
金 额:
1100
时 间:
2019-3-30
类 别:
付款方:
备 注:
生活费
兼职
实习工资
奖金
其他
保存
取消


4:11
新增收入
新增收入
金 额:
1100
时 间:
2019-3-30
类 别:
实习工资
付款方:
company
备 注:
test
〖新增收入〗数据添加成功!
保存
取消


4:11
我的收入
我的收入
1|实习工资 1100.0元 2019-3-30

【实验小结】

这个实验做得很顺利，并没有大的错误，不过需要注意的地方很多。报告老实说做得有些仓促，不过理解的过程还是进行了很长时间的。难点应该就是编写数据库的操作对象中的方法了吧，其中有很多只有在大一实验课上使用过，平时是很少用的，只是简单的增删改查，本次的实验也是对其中的操作又再复习了一遍吧。