

# **Android 应用软件设计**

## **E 1 Hello Android**

**学号：SA17225433**

**姓名：杨帆**

**报告撰写时间：2018/10/11**

# 1.主题概述

## 1. 搭建 android studio 开发环境。

安装 android studio IDE 和 JDK、android SDK，主要进行安装和环境变量的配置。SDK 可以选择自己下载配置，或通过 as 内部工具 android SDK manager 进行管理。

## 2. 新建 Android Application 工程 SCOS，并在 src 包下定义 es.source.code.activity 子包。

新建工程项目，创建 activity 包，用于存放 activity 类

## 3. 在 es.source.code.activity 下定义一个名为 SCOSEntry 的 Activity 类，在 AndroidManifest.xml 中将该 Activity 设为 SCOS 的入口。

Android 应用的入口 activity 是通过设置 manifest 里的 activity 标签来决定的，具体方法见下文的实现过程。

## 4. 在 SCOS 工程的 res->layout 下新建 entry.xml 作为 SCOSEntry 的视图定义，将该视图定义为 RelativeLayout 布局，并在布局中使用 ImageView 嵌入 SCOS 的 LOGO 图片，要求 LOGO 为 png 格式（LOGO 每个组自己设计，要求适配手机或平板的屏幕大小）。

图片使用 ImageView 组件来显示，可以在 layout 里设置 view 的 src 来展示，也可以在 activity 类中调用 ImageView 类的 setImageResource()方法来实现。

## 5. 调试运行 SCOS，要求 SCOSEntry 正确显示 LOGO。

## 6. 调试正确后，打包&签名并发布正式版应用 V1.0，发布包名为 SCOS1.0.apk

这里使用了 android.keystore 创建了一个自己的 key，并签名。生成了 release 版本的 app，并安装测试成功。

## 2.假设

本应用总体目标是实现一款订餐软件，用来代替传统纸质订餐和电话订餐。由于这是第一次作业，只实现了创建应用和主页并展示 logo 的功能，没有涉及到一些具体业务情况的假设。

## 3. 实现或证明

### 1. Github:

<https://github.com/saaaaaail/SCOS>

### 2. APK:

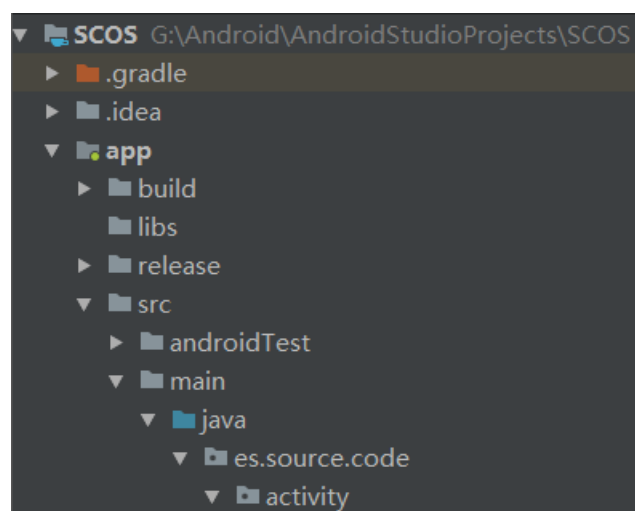


SCOS1.0.apk

### 3. 搭建 android studio 开发环境:

分别下载 android studio IDE 和 JDK，安装并配置 JDK 的环境变量，然后打开 android studio 软件，联网下载 SDK。

### 4. 新建 Android Application 工程 SCOS，并在 src 包下定义 es.source.code.activity 子包。



### 5. 设置入口程序:

在之前的创建应用过程中已经设置好了入口程序，这里说明一下入口程序的设置方法。在 Androidmanifest 文件中,将 intent-filter 标签(包含如下两句)放入需要设置为入口的 activity 对应的标签中即可。如图所示:

```
<activity
    android:name=".activity.SCOSEntry"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

### 6. 在入口 activity 中添加一个 logo 展示:

在 SCOSEntry.java 文件的布局文件 entry.xml 文件中添加 ImageView 标签，并设置 android:id, android:layout\_width, android:layout\_height 等属性，在 android:src 属性中添加 LOGO.png 文件@drawable/ic\_logo\_v2

ic\_logo\_v2.png

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="es.source.code.activity.SCOSEntry">

    <ImageView
        android:id="@+id/ic_logo_v2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:layout_centerInParent="true"
        android:src="@drawable/ic_logo_v2"/>

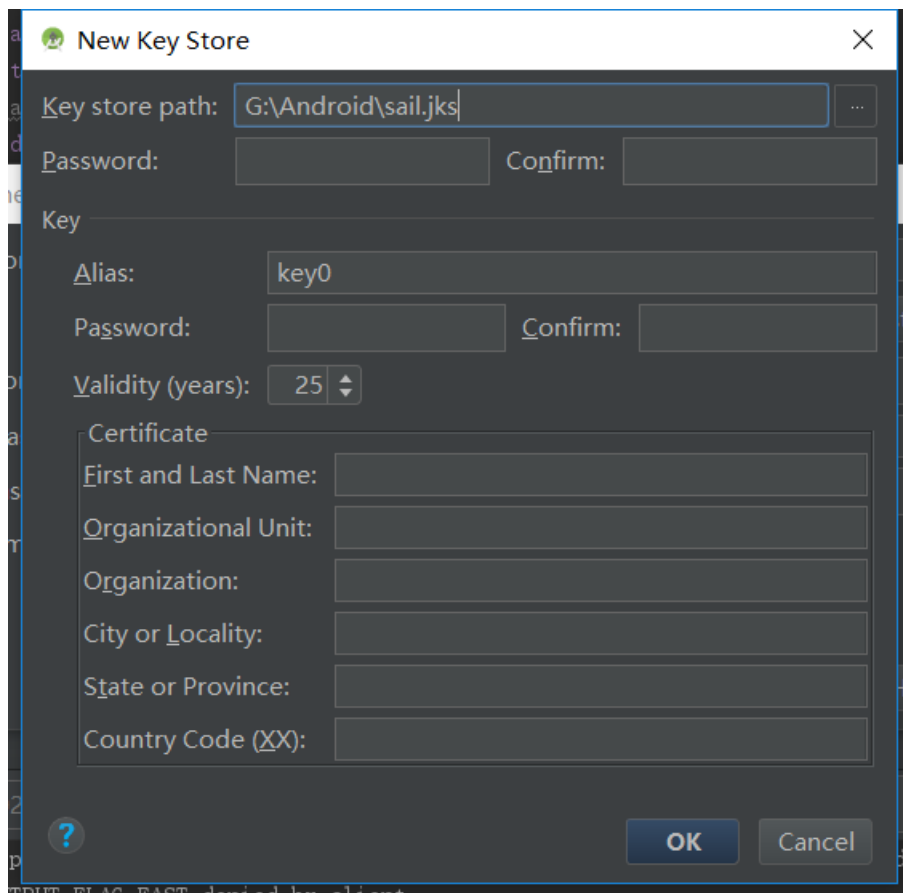
</RelativeLayout>
```

调试完成后，打开应用显示如图：

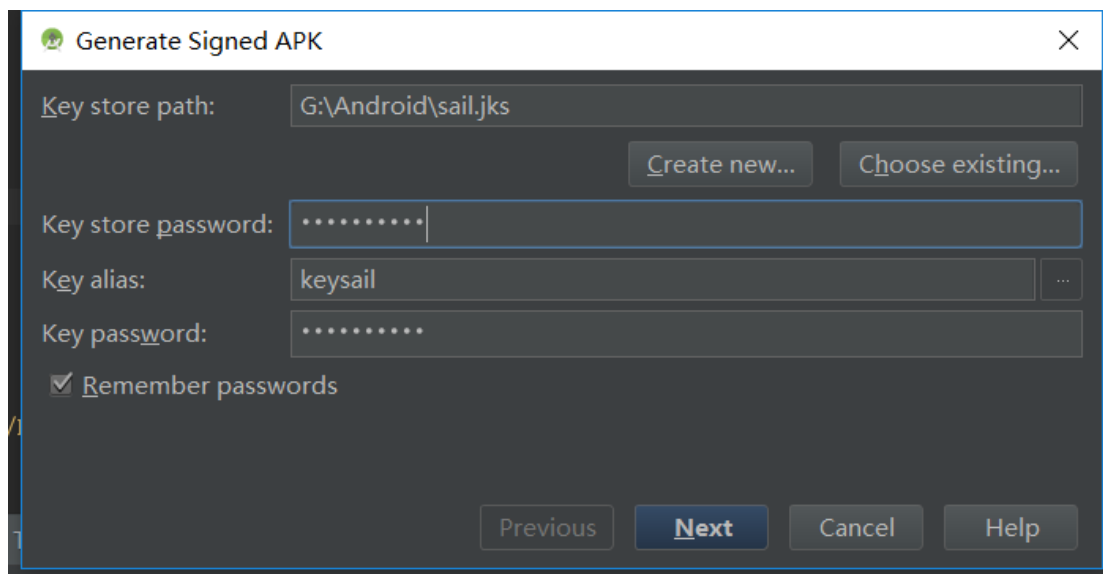


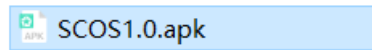
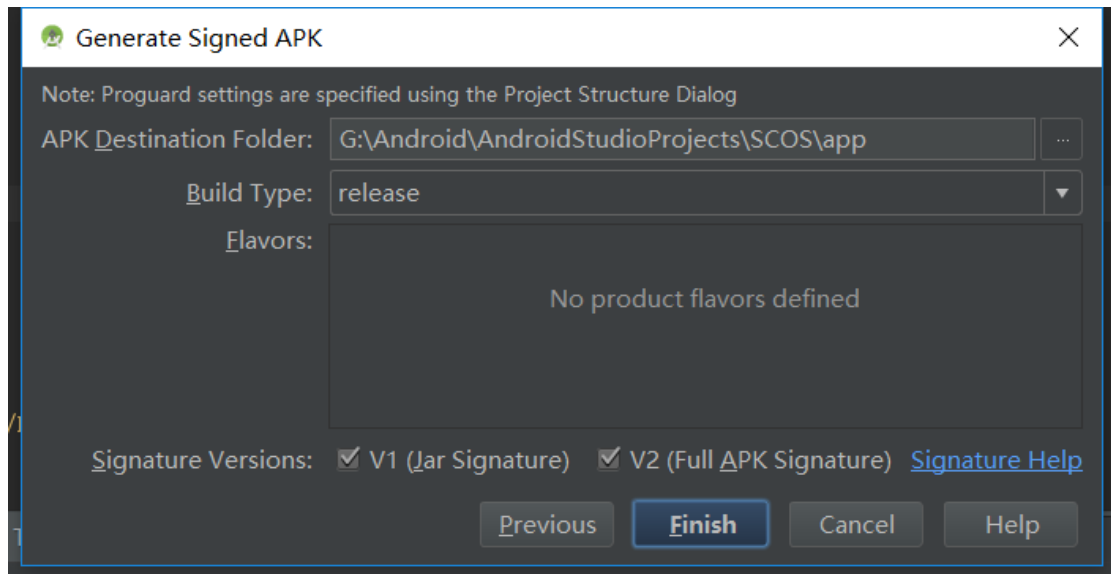
## 7. 签名并打包发布正式版本:

这里我们使用 AndroidStudio 中的 Build 栏下 Generate Signed APK 来创建一个 android.keystore。



而后返回上一个界面，输入密码，点击 next，设置输出目录，选中签名版本（V1,V2），Finish 即可生成到指定目录。





## 4. 结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。如 Android 程序调试方法之间的优缺点比较（文字、图标、代码辅助等）

本次练习让我熟悉了 android 的开发环境，在指导书的帮助下快速上手了第一个项目，使用了 android 中的 ImageView 组件，并成功使用 ImageView 加载了 LOGO.png，显示了欢迎界面。

当在生成签名 apk 的时候，如果勾选了 V2 不勾选 V1，会造成之后安装应用时权限校验失败。



## 5.参考文献

1. [使用 yeelogo 设计 SCOS 的 LOGO](#)
2. [github: 参考潘梦泽学长的项目书写风格](#)

# **Android 应用软件设计**

## **E 2 Intent and Intent Filter**

**学号：SA17225433**

**姓名：杨帆**

**报告撰写时间：2018/10/11**

# 1.主题概述

1.在 E1 的 SCOS 工程包 `es.source.code.activity` 下定义一个新 Activity 类 `MainScreen.java`。在 `MainScreen` 的屏幕布局中设计 SCOS 系统的主功能导航，导航项含：点菜，查看订单，登录/注册，系统帮助。要求导航项有图标和文字，并且可点击。在 `AndroidManifest.xml` 中添加 `MainScreen` 并定义其 `IntentFilter` 属性，添加一个 Action 值为“`scos.intent.action.SCOSMAIN`”和一个 Category 值为“`scos.intent.category.SCOSLAUNCHER`”

2.修改 `SCOSEntry` 的代码，实现用户向左滑动时，从 `SCOSEntry` 屏幕跳转到 `MainScreen` 屏幕，并使用 `Intent` 向 `MainScreen` 类传递一个 `String` 数据值“`FromEntry`”，`MainScreen` 接受 `SCOSEntry` 的 `String` 值，并作判断是否和 `FromEntry` 相等，如果相等正常显示屏幕，如果不等隐藏点菜和查看订单

3.在 `AndroidManifest` 中定义 `Permission` 值为“`scos.permission.ACCESSSCOS`”，并将 `Perssion level` 设置为“`dangerous`”，修改 `AndroidManifest` 中 `MainScreen` 的属性 `android:permission` 值为“`scos.permission.ACCESSSCOS`”再次测试从 `SCOSEntry` 跳转到 `MainScreen` 屏幕

4.在 `es.source.code.activity` 下定义一个 Activity 类，`LoginOrRegister.java`，并在其布局文件中添加登录名输入框和登录密码输入框，添加按钮登录和返回按钮，并添加按钮点击事件，以及跳转，传值判断是登录成功还是返回以隐藏或显示。

5.新建一个工程为 `TestSCOS`，在该工程中添加 Activity 类 `TestMain`，在 `Test` 屏幕中添加按钮，点击按钮跳转到 SCOS 的 `MainScreen`。

## 2.假设

本应用总体目标是实现一款订餐软件，用来代替传统纸质订餐和电话订餐。此次为第二次作业，本次作业添加了主屏幕界面能够进行点餐，查看订单，用户账户，帮助等功能，实现了欢迎界面 **Entry** 左滑向主界面的跳转，并设置了权限值，添加了用户账户界面，能由主界面点击按钮跳转，完成登录注册会携带用户操作信息返回主界面，最后新建了一个测试工程，用以在其他项目中跳转到当前项目的主界面。

## 3.实现或证明

### 1.Github:

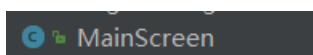
<https://github.com/saaaaaail/SCOS>

### 2. APK:




SCOS1.5.apk

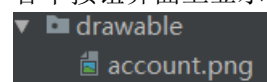
### 3. 在 E1 的 SCOS 工程包 es.source.code.activity 下定义一个新 Activity 类 mainScreen.java。



### 4. 在 mainScreen 的屏幕布局中设计 SCOS 系统的主功能导航，导航项含：点菜，查看订单，登录/注册，系统帮助。要求导航项有图标和文字，并且可点击。

MainScreen.java 的布局文件为  activity\_main\_screen.xml，该布局文件使用线性布局，

分别使用四个按钮 Button 组件表示：点菜，订单，用户账户，系统帮助。为四个按钮定义相同的大小属性，但是 android:id 设置为不同来标识不同的按钮，使用 android:text 属性设置各个按钮界面上显示的名称。按钮的图标需要事先下载好.png 文件。



然后使用 `android:drawableLeft="@drawable/order"` 的方式将图标添加在左边。

```
<Button
    android:id="@+id/btn_order"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dip"
    android:textSize="20dp"
    android:textColor="@color/white"
    android:background="@drawable/btn_nor_down"
    android:gravity="center"
    android:text="点菜啦"
    android:paddingLeft="15dp"
    android:drawableLeft="@drawable/order"/>
```

### 5. 在 AndroidManifest 中添加 mainScreen 并定义 IntentFilter 属性，添加 Action 值 scos.intent.action.SCOSMAIN 和 Category 值 scos.intent.category.SCOSLAUNCHER


```
<activity
    android:name=".activity.MainScreen"
    android:permission="scos.permission.ACCESSSSCOS"
    android:launchMode="singleTop"
    android:theme="@style/AppTheme.NoActionBar"
    android:exported="true">
    <intent-filter>
        <action android:name="scos.intent.action.SCOSMAIN" />
        <!-- 不加DEFAULT会有问题 -->
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="scos.intent.category.SCOSLAUNCHER" />
    </intent-filter>
</activity>
```

6.修改 SCOSEntry 的代码，实现用户向左滑动时，从 SCOSEntry 屏幕跳转到 MainScreen 屏幕，并使用 Intent 向 MainScreen 类传递一个 String 数据值“FromEntry”，MainScreen 接受 SCOSEntry 的 String 值，并作判断是否和 FromEntry 相等，如果相等正常显示屏幕，如果相等隐藏点菜和查看订单。

Android 提供了监听手势的类 GestureDetector，通过重写该类的手势监听下的 onFling (MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)方法，即判断其中 e1 手指触碰屏幕的位置与 e2 手指离开屏幕的位置的横坐标的差值大于 50mm 即认为手指从右向左滑动了屏幕，若满足以上条件则进行跳转到 MainScreen 类，并使用 Intent 传递 **FromEntry**。

```
detector = new GestureDetector( context: this, new GestureDetector.SimpleOnGestureListener() {
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        //如果从右向左滑超过50mm
        if (e1.getX() - e2.getX() > 50) {
            showToast(getResources().getString(R.string.toast_welcome));
            Intent intent = new Intent(mContext, MainScreen.class);
            intent.putExtra(Const.IntentKey.FROM, Const.IntentValue.FROM_VALUE);
            startActivity(intent);
            //设置切换动画，从右至左滑动
            overridePendingTransition(R.anim.right_in, R.anim.left_out);
            finish();
        }
        return false;
    }
});
```

7. 在 es.source.code.activity 下定义一个 Activity 类，LoginOrRegister.java，并在其布局文件中添加登录名输入框和登录密码输入框，添加按钮登录和返回按钮，并添加按钮点击事件，以及跳转，传值判断是登录成功还是返回以隐藏或显示。

定义这个 activity 和其布局文件为  activity\_login\_or\_register.xml，在其中添加两个 EditText 组件用来输入用户名和密码，设定不同的 id 值，设计两个 Button 组件分别标识登录和返回，并分别设计点击监听事件，使用 Intent 传递 String 返回值，使用 setResult 方法向 MainScreen 返回结果。

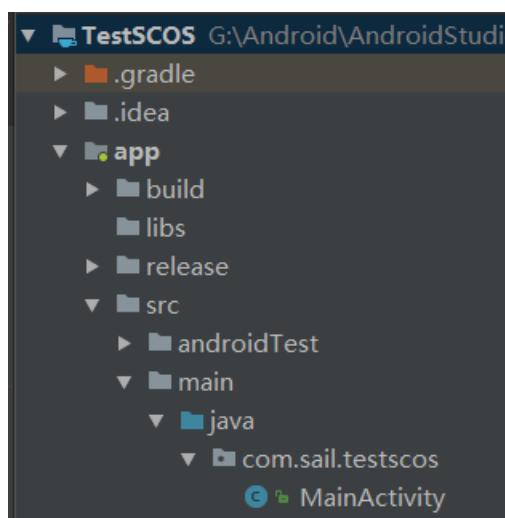
```
Intent intent = new Intent(mContext, MainScreen.class);
intent.putExtra(Const.IntentKey.LOGIN_STATUS, Const.IntentValue.LOGIN_SUCCESS);
setResult(Const.ResultCode.LOGIN_OR_REGISTER, intent);
finish();
}
```

而在 MainScreen 方法中通过 onActivityResult 方法获得返回值，并判断返回值，决定是否隐藏点菜和订单按钮。

```
switch (resultCode) {
    case Const.ResultCode.LOGIN_OR_REGISTER: {
        if (Const.IntentValue.LOGIN_SUCCESS.equals(intent.getStringExtra(Const.IntentKey.LOGIN_STATUS))) {
            Order_btn.setVisibility(View.VISIBLE);
            List_btn.setVisibility(View.VISIBLE);
        }
    }
}
```

8.新建一个工程为 TestSCOS，在该工程中添加 Activity 类 TestMain，在 Test 屏幕中添加按钮，点击按钮跳转到 SCOS 的 MainScreen。

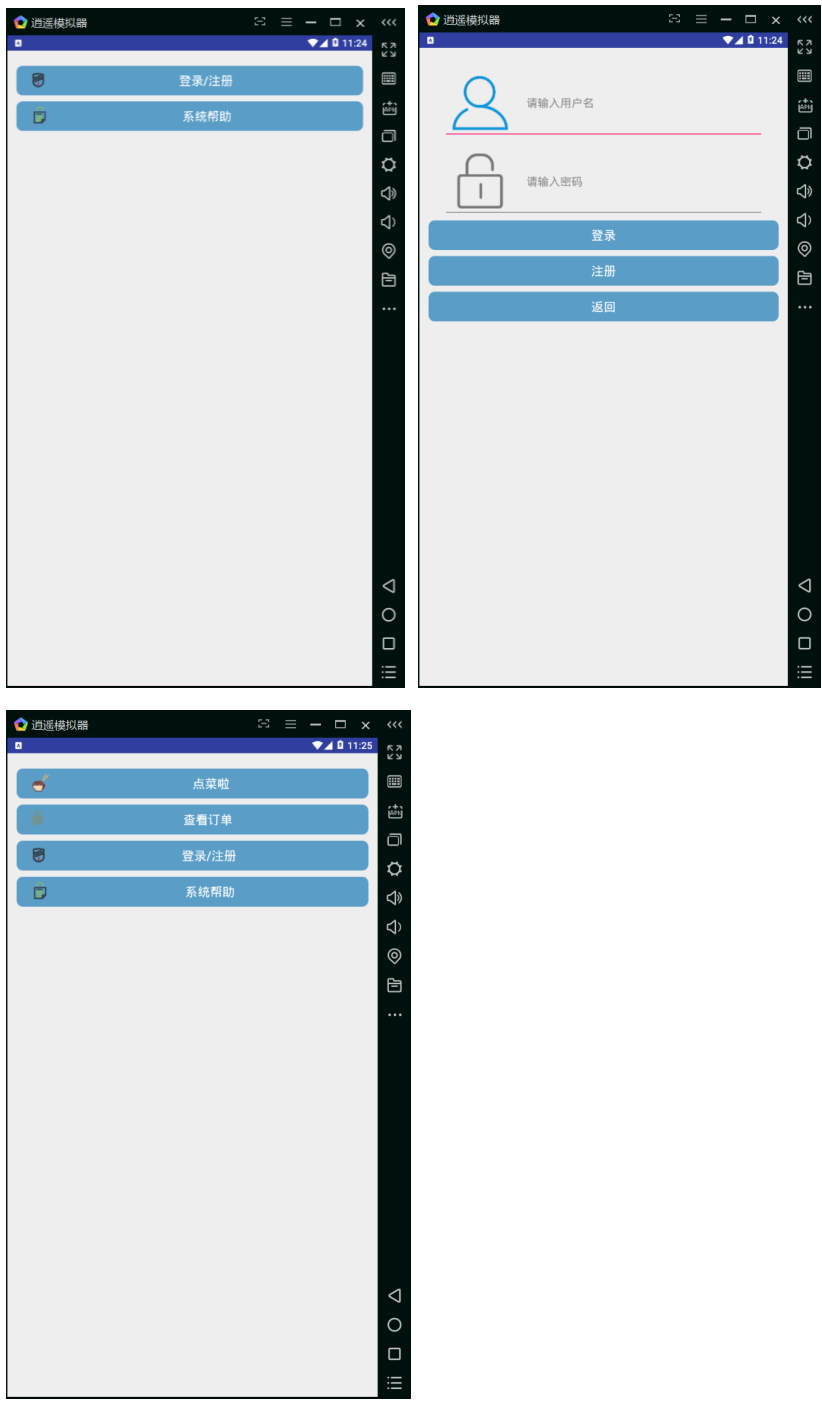
新建 TestSCOS 工程。



使用 Intent 从当前工程跳转到 SCOS 工程下的 MainScreen，需要 SCOS 工程的包名和 MainScreen 的类名。

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String pckName = "es.source.code";
        String clsName = "es.source.code.activity.MainScreen";
        Intent intent = new Intent();
        intent.addCategory(Intent.CATEGORY_LAUNCHER);
        ComponentName cn = new ComponentName(pckName, clsName);
        intent.setComponent(cn);
        startActivity(intent);
    }
});
```

调试完成后，打开应用显示如图：





## 9. 查阅资料总结 android:protectionlevel 的不同类型，并说明如何使用。

Android protectionLevel 分 4 个级别：normal、dangerous、signature、signatureOrSystem，  
normal：这是最低风险的权限，如果应用声明了此权限，也不会提示安装应用的用户授权（例如，如果声明了定位权限，则应用到定位功能时，会明确提示用户，是否授予定位权限，但是 protectionLevel 为 normal 的不会明确提示，直接默认授予），系统直接默认该应用有此权限；

dangerous：这种级别的权限风险更高，拥有此权限可能会访问用户私人数据或者控制设备，给用户带来负面影响，这种类型的权限一般不会默认授权，例如可能访问私人数据（例如短信，联系人），或者控制设备（使用摄像头，定位）。

signature：这种权限级别，只有当发请求的应用和接收此请求的应用使用同一签名文件，并且声明了该权限才会授权，并且是默认授权，不会提示用户授权

signatureOrSystem：表示将权限授予具有相同数字签名的应用程序或者只有 Android 系统包类才可访问，也不会提升用户授权。

## 10. 查阅资料总结 IntentFilter 如何测试 Action、Category、Data，并说明 IntentFilter 用法。

IntentFilter 主要用于启动 activity 的隐式调用：需要 Intent 能匹配目标组件的 IntentFilter 中所设置的过滤信息。如果不匹配将无法启动目标 Activity。

**Action 匹配规则：**Intent 中的 Action 必须能够和 Activity 过滤规则中的 Action 匹配。(这里的匹配是完全相等)。一个过滤规则中有多个 action，那么只要 Intent 中的 action 能够和 Activity 过滤规则中的任何一个 action 相同即可匹配成功。

**Category 的匹配规则：**如果 Intent 中的存在 category，那么这些 category 都必须和 Activity 过滤规则中的 category 相同，才能和这个 Activity 匹配。Intent 中的 category 数量可能少于 Activity 中配置的 category 数量，但是 Intent 中的这 category 必须和 Activity 中配置的 category 相同才能匹配。通俗的讲就是，比如 Intent 中有 3 个 category，activity 的过滤规则中有 5 个 category，那 intent 中的 3 个 category 需要是 activity 的过滤规则中有 5 个 category 中的 3 个，若有任意一个未出现在这 5 个里面，匹配就失败。

**Note:** 只通过 category 匹配是无法匹配到 AActivity 的。因为 category 属性是一个执行 Action 的附加信息。所以只靠 category 是无法匹配的，需要同时匹配 Action。

**data 的匹配规则：**类似于 action 匹配，但是 data 有更复杂的结构。

```
<data android:scheme="axe"
      android:host="axe"
      android:port="axe"
      android:path="axe"
      android:pathPattern="axe"
      android:pathPrefix="axe"
      android:mimeType="axe"/>
```

data 的结构：

## 4.结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。如 Android 程序调试方法之间的优缺点比较（文字、图标、代码辅助等）

本次作业让我创建了三个 activity，并且实现了以监听手势滑动的方式进行跳转，以点击按钮的方式进行跳转，并学会了使用 Intent 进行跳转时，添加变量，传递变量给目标 activity，学会了使用 Intent 的 startActivityforResult 方法进行有返回值的跳转方式，在 onActivityResult 方法中获得返回标签。学会了自定义 activity 的权限，能够通过权限限制其他 activity 的访问，如果其他 activity 要想访问带有权限的 activity，需要在这个 activity 的属性中添加 permission 属性，并等于对应的权限。

当使用 TestSCOS 跳转到 MainScreen 时，会报错 permission denied，需要在 TestSCOS 的 AndroidManifest 文件中去添加权限，使用 uses-permission 标签。

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

## 5.参考文献

- 1.[github:参考潘梦泽学长代码书写规范，使用并修改其资源文件](#)
- 2.[Android 的屏幕切换动画—左右滑动切换](#)
- 3.[Android 开发之漂亮 Button 样式：按钮风格使用了此博客提供的样式](#)
- 4.[Android 实现自定义带文字和图片的 Button](#)
- 5.[Android 之利用正则表达式校验邮箱、手机号、密码、身份证号码等](#)
- 6.[Android 从一个 APP 跳转到另一个 APP 的主界面或某页面，并传递数据](#)
- 7.[android 自定义 permission android:protectionLevel 说明](#)
- 8.[Android 自定义 action 与 permission!!!](#)
- 9.[Android 基础总结十一:intent-filter 的 action, category, data 匹配规则](#)

# **Android 应用软件设计**

## **E 3 Adapter View**

**学号：SA17225433**

**姓名：杨帆**

**报告撰写时间：2018/10/11**

# 1.主题概述

1.使用 GridView 替换已有的按钮导航：点菜、查看订单、登录/注册、系统帮助；导航项可以点击；每个导航项都有图标和文字，登录注册与 E2 相同。

2.新建 User 类。

3.修改 LoginOrRegister 代码：添加“注册”按钮；点击登录时，保存 User 对象，传递 LoginSuccess;点击注册按钮时，保存 User 对象，传递 RegisterSuccess;点返回与 E2 相同。

4.修改 mainScreen 代码：新建 User 对象；保存由 LoginOrRegister 传回的 User 对象。

5.新建 FoodView 类,添加 activity\_food\_view.xml 布局:使用 viewPager&tablayout 显示“冷菜”、“热菜”、“海鲜”、“酒水”导航；支持左右滑动；使用 RecyclerView 加载食物列表；添加菜单项“已点菜品”“查看订单”“呼叫服务”。

6.新建 FoodOrderView 类，添加 activity\_food\_order\_view.xml 布局:使用 viewPager&tablayout 显示“已下单菜”“未下单菜”；支持左右滑动；已下单菜使用 RecyclerView 显示菜名，价格，数量；未下单菜使用 RecyclerView 显示菜名，价格，数量，备注，“退点”按钮；；页面下方显示菜品总数、订单总价、结账&提交订单按钮。

7.新建 FoodDetailed 类，添加 activity\_food\_detailed.xml 布局：显示图片，菜名，菜价，备注输入框；添加按钮判断菜品是否已点显示“退点”“点菜”；支持左右滑动屏幕。

8. mainScreen 向 FoodView 跳转时，向 FoodOrderView 跳转时传递 user 对象。

9. 从 FoodView 向 FoodOrderView 跳转时，传递 user 对象，并跳转到指定页面。

10. 在 FoodOrderView 点击结账时，弹出提示语句。

## 2.假设

本应用总体目标是实现一款订餐软件，用来代替传统纸质订餐和电话订餐。本次为第三次作业，主要实现各个菜品与页面的加载的功能，通过使用各种 **adapter** 实现点餐页面导航项的左右滑动切换，每个导航项下菜品的加载，实现订单页面“已下单菜”、“未下单菜”的左右滑动切换，以及底下菜品的加载，点菜与退菜的实现，详单的页面实现所有菜品的左右滑动加载，以及页面之间跳转到准确的 **view** 页。

### 3.实现或证明

#### 1. Github:

<https://github.com/saaaaaail/SCOS>

#### 2. APK:



SCOS2.0.apk

3. 使用 **GridView** 替换已有的按钮导航：点菜、查看订单、登录/注册、系统帮助；导航项可以点击；每个导航项都有图标和文字，登录注册与 E2 相同[1][2][4]。

初始化使用 GridView 九宫格布局替换导航按钮，

```
<GridView
    android:layout_centerInParent="true"
    android:id="@+id/gl"
    android:numColumns="2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"/>
```

新建 Function 类，存储每项按钮的图标、名称、标识属性，

```
public Function(int img,String name,String tag) {
    this.img=img;
    this.name=name;
    this.tag=tag;
}
```

新建 FunctionAdapter 类，传入按钮数据，使用 ViewHolder 获得按钮视图，并加载图标、名称、标识。

```
public FunctionAdapter(Context mContext, List<Function>functionList) {
    this.functionList=functionList;
    this.mContext=mContext;
    inflater = LayoutInflater.from(mContext);
}
```

```
viewHolder.Function_btn.setText(function.getName());
Drawable drawable = mContext.getResources().getDrawable(function.getImg()); // 获取res下的图
drawable.setBounds( left: 0, top: 0, drawable.getMinimumWidth(), drawable.getMinimumHeight()
viewHolder.Function_btn.setCompoundDrawables(drawable, top: null, right: null, bottom: null);
```

#### 4. 新建 User 类。

```
public class User {
    private String userName;
    private String password;
    private Boolean oldUser;
```

5. 修改 LoginOrRegister 代码：添加“注册”按钮；点击登录时，保存 User 对象，传递 LoginSuccess;点击注册按钮时，保存 User 对象，传递 RegisterSuccess;点返回与 E2 相同。

```
<Button
    android:id="@+id/btn_register"
    style="@style/TableButton"
    android:layout_margin="5dip"
    android:text="注册"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

添加注册按钮。

点击登录与点击注册按钮，使用相同的传值方法，使用 SharedPreferences 类保存 User 对象 [8][9]

```
sharedPreferenceUtil = SharedPreferencesUtil.getInstance(mContext);
sharedPreferenceUtil.setUser(loginUser);
//但不起传值的作用
```

6. 新建 FoodView 类，添加 activity\_food\_view.xml 布局：使用 viewPager&tablayout 显示“冷菜”、“热菜”、“海鲜”、“酒水”导航；支持左右滑动；使用 RecyclerView 加载食物列表；添加菜单项“已点菜品”“查看订单”“呼叫服务”。

新建 BaseFoodFragment 用来表示四页导航项，新建 FragmentViewPagerAdapter 适配器，编写 BaseFoodFragment 的四个子类 Fragment，new 四个子类 Fragment 并添加到 FragmentViewPagerAdapter 适配器中，[3][10][12]

```
ColdFoodFragment coldFoodFragment = new ColdFoodFragment();
HotFoodFragment hotFoodFragment = new HotFoodFragment();
SeaFoodFragment seaFoodFragment = new SeaFoodFragment();
DrinkFoodFragment drinkingFragment = new DrinkFoodFragment();

fragmentList.add(coldFoodFragment);
fragmentList.add(hotFoodFragment);
fragmentList.add(seaFoodFragment);
fragmentList.add(drinkingFragment);
```

重写 getItem 方法，返回对应位置的 Fragment，

```
return fragmentList.get(position)
```

重写 getCount()返回总页数，重写 getPageTitle 方法返回每页的标题。

```
getCount() { return fragmentList.size(); } return fragmentList.get(position).getFoodTag();
```

并使用 tabLayout.setupWithViewPager(viewPager)方法绑定 tablayout 和 viewPager。

新建 Food 类，存储菜品数据



```
public class Food {
    // 菜名
    private String foodName;

    // 价格
    private int price;

    // 库存;
    private int store;

    //备注
    private String remark;

    //类别
    private int category;

    // 是否点单
    private boolean order;

    // 图片资源ID;
    private int imgId;
}
```

然后新建 FoodRecyclerViewAdapter 类继承 RecyclerView.Adapter 用来实现 RecyclerView 加载菜品，FoodRecyclerViewAdapter 类需要传入菜品 Food 数据，创建 ViewHolder 继承实现 RecyclerView.ViewHolder，[5]

```
public class ViewHolder extends RecyclerView.ViewHolder {
    public final View mView;
    public final ImageView imageView;
    public final TextView foodNameView;
    public final TextView foodStoreView;
    public final TextView foodPriceView;
    public final TextView foodOrderBtn;

    public final LinearLayout FoodItem;
    public Food mItem;

    public ViewHolder(View view) {
        super(view);
        mView = view;
        imageView = (ImageView) view.findViewById(R.id.iv_food);
        foodNameView = (TextView) view.findViewById(R.id.tv_food_name);
        foodStoreView = (TextView) view.findViewById(R.id.tv_food_store);
        foodPriceView = (TextView) view.findViewById(R.id.tv_food_price);
        foodOrderBtn = (TextView) view.findViewById(R.id.btn_order);
        FoodItem = (LinearLayout) view.findViewById(R.id.food_item);
    }
}
```

使用 viewHolder 获得单条 item（布局为  fragment\_food.xml）中的所有组件视图并初始化（包括一个 ImageView 显示图片，三个 TextView 分别显示菜名，菜价，库存，以及

一个点菜按钮），在 onBindViewHolder 方法中使用 viewholder 填充 food 对象的数据到视图中。

```
holder.mItem = mValues.get(position);
holder.foodNameView.setText(mValues.get(position).getFoodName());
Glide.with(mContext).load(mValues.get(position).getImgId()).into(holder.imageView);
holder.foodStoreView.setText("剩余: "+mValues.get(position).getStore()+"份");
holder.foodPriceView.setText(mValues.get(position).getPrice()+"元");
holder.foodOrderBtn.setText(holder.mItem.isOrder() ? "退点" : holder.mItem.getStore() > 0 ? "点菜" : R.string.btn_empty);
```

然后在 BaseFoodFragment 中实例化 FoodRecyclerViewAdapter 类，传入 foods 数据，初始化并绑定 RecyclerView 与资源 ID，recyclerView.setAdapter(foodRecyclerViewAdapter) 设置适配器成功。


**7.新建 FoodOrderView 类，添加 activity\_food\_order\_view.xml 布局：**使用 viewPager& tablayout 显示“已下单菜”“未下单菜”；支持左右滑动；已下单菜使用 RecyclerView 显示菜名，价格，数量；未下单菜使用 RecyclerView 显示菜名，价格，数量，备注，“退点”按钮；；页面下方显示菜品总数、订单总价、结账&提交订单按钮。

此类实现方式与 FoodView 类似。均是使用 viewPager+Fragment 实现“已下单菜”“未下单菜”左右滑动，然后实现 OrderRecyclerViewAdapter 来添加菜品数据，并在 BaseOrderFragment 中设置 adapter。

**8.新建 FoodDetailed 类，添加 activity\_food\_detailed.xml 布局：**显示图片，菜名，菜价，备注输入框；添加按钮判断菜品是否已点显示“退点”“点菜”；支持左右滑动屏幕 [3][20][21][22]。

使用 viewPager+继承 pageradapter 实现单个视图的切换：

新建 FoodDetailedAdapter 类，继承实现 pageradapter，传入视图切换的基本 item 视图资源

ID  item\_food\_detailed.xml 和 food 对象 list。

```
public FoodDetailedAdapter(int resourceID, List<Food> foods, Context mContext) {
    // this.views = views;
    ID=resourceID;
    this.foods = foods;
    this.mContext = mContext;
    initData();
}
```

首先根据 Food 对象的数目，新建基本 item 视图与之一一对应。

```
private void initData() {
    views = new ArrayList<>();
    LayoutInflater inflater = LayoutInflater.from(mContext)
    for (int i=0;i<foods.size();i++) {
        views.add(inflater.inflate(ID, root: null));
    }
}
```

然后重写 instantiateItem 方法，绑定 item 布局中的 Button、ImageView、TextView 对应布局文件中的资源 ID，并填充 food 对象中的数据。最后添加该条初始化好的 view。

```
container.addView(views.get(position));
```

在 FoodDetailed 类的布局文件 `activity_food_detailed.xml` 中需要添加“点菜”“退点”功能 Button。在 FoodDetailed 类中实例化 FoodDetailedAdapter，传入每一项的布局 ID `item_food_detailed.xml` 和 food 对象 list，然后使用 viewPager 设置适配器。

### 9.发 MainScreen 向 FoodView 跳转时，向 FoodOrderView 跳转时传递 user 对象。从 FoodView 向 FoodOrderView 跳转时，传递 user 对象，并跳转到指定页面。[17]

实现 MainScreen 点击按钮跳转到 FoodView 和 FoodOrderView，由于适配器获得了各个 Function 按钮的视图，其点击监听事件在 FunctionAdapter 中实现，

```
viewHolder.Function_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(callBack!=null){
            callBack.onClick(function, position);
        }
    }
});
```

无法被 MainScreen 获得，因此需要自定义一个包含点击方法的接口，在 FunctionAdapter 中构建 setCallBack 方法，并获得这个接口对象，[6]

```
public void setCallBack(CallBack<Function> callBack) { this.callBack=callBack; }
```

然后在 MainScreen 类中使用 functionAdapter 调用该方法，并向其点击方法中传入 Function 对象，这当点击的时候，就能向 adapter 中的按钮传递数据，并调用这个实现接口的点击方法。

```
functionAdapter.setCallBack(new CallBack<Function>() {
    @Override
    public void onClick(Function function, int position) {
        doFunction(function);
    }
});
```

实现 FoodView 向 FoodDetailed 类的跳转，同样需要使用接口的回调方法，获得接听事件，向 FoodDetailed 跳转需要在 adapter 实现该条 LinearLayout 布局的点击监听事件。

```
holder.foodOrderBtn.setOnClickListener((v) -> {
    if (null != mBtnListener) {
        // Notify the active callbacks interface (the activity, if the
        // fragment is attached to one) that an item has been selected.
        mBtnListener.onClick(holder.mItem);
    }
});
```

然后在 BaseFoodFragment 中调用 adapter 实现这个接口的 onClick 方法

```

foodRecyclerViewAdapter.setOnListFragmentInteractionListener((item) → { //跳转详细页面
    int cate = item.getCategory();
    SharedPreferencesUtil spu = SharedPreferencesUtil.getInstance(mContext);
    List<Food> tmpfoods = spu.getAllFood(cate);

    //判断位置
    int i=0;
    for(Food food:tmpfoods) {
        if(food.getFoodName().equals(item.getFoodName())) {break;}
        i++;
    }
    int index = i;

    Intent intent = new Intent(mContext, FoodDetailed.class);
    intent.putExtra(AppGlobal.IntentKey.CURRENT_CATEGORY, cate);
    intent.putExtra(AppGlobal.IntentKey.CURRENT_DETAILED_POSITION, index);
    startActivity(intent);
});

```

为了准确定位 FoodDetailed 界面中的某一个 view，需要在跳转时传入点击的位置。

实现 FoodView 向 FoodOrderView 跳转需要实现一个菜单项，在 FoodView 中添加 toolbar 替代自带的 actionBar，在 activity\_food\_view.xml 布局中，添加 toolbar 组件，在 FoodView 类中绑定资源 ID，toolbar 通过 inflateMenu(R.menu.menu\_order)方法绑定菜单，然后定义菜单项的点击事件。[13][14][15][16]

```

toolbar.setOnMenuItemClickListener((item) → {
    int menuItemId = item.getItemId();
    switch(menuItemId) {
        case R.id.action_order:
            Intent intent = new Intent(mContext, FoodOrderView.class);
            intent.putExtra(AppGlobal.IntentKey.USER_INFO, userString);
            intent.putExtra(AppGlobal.IntentKey.CURRENT_INDEX, AppGlobal.Lable.UNORDER_LABLE);
            startActivity(intent);
            break;
        case R.id.action_bill:
            Intent intent1 = new Intent(mContext, FoodOrderView.class);
            intent1.putExtra(AppGlobal.IntentKey.USER_INFO, userString);
            intent1.putExtra(AppGlobal.IntentKey.CURRENT_INDEX, AppGlobal.Lable.ORDERED_LABLE);
            startActivity(intent1);
            break;
        case R.id.action_serve:
            break;
    }
    return true;
});

```

同时还需要设置 menu，解析 menu 布局到 menu。

```

public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_order, menu);
    return true;
}

```

为了能够实现点击“已点菜品”能跳转到“未下单菜”，点击“未点菜品”能跳转到“已下单

菜”，需要在跳转同时传入 `OrderedFragment` 和 `UnOrderFragment` 的标识值，使 `viewpager` 准确切换到跳转页面。

#### 10.在 `FoodOrderView` 点击结账时，弹出提示语句。

不同页面之间跳转传递的 `User` 对象使用 `SharedPreference` 存储，然后在跳转页面后再获得保存的 `User` 对象，保存 `User` 对象先使用 `Gson` 库将 `User` 对象解析为 `json` 字符串，然后保存[25]。

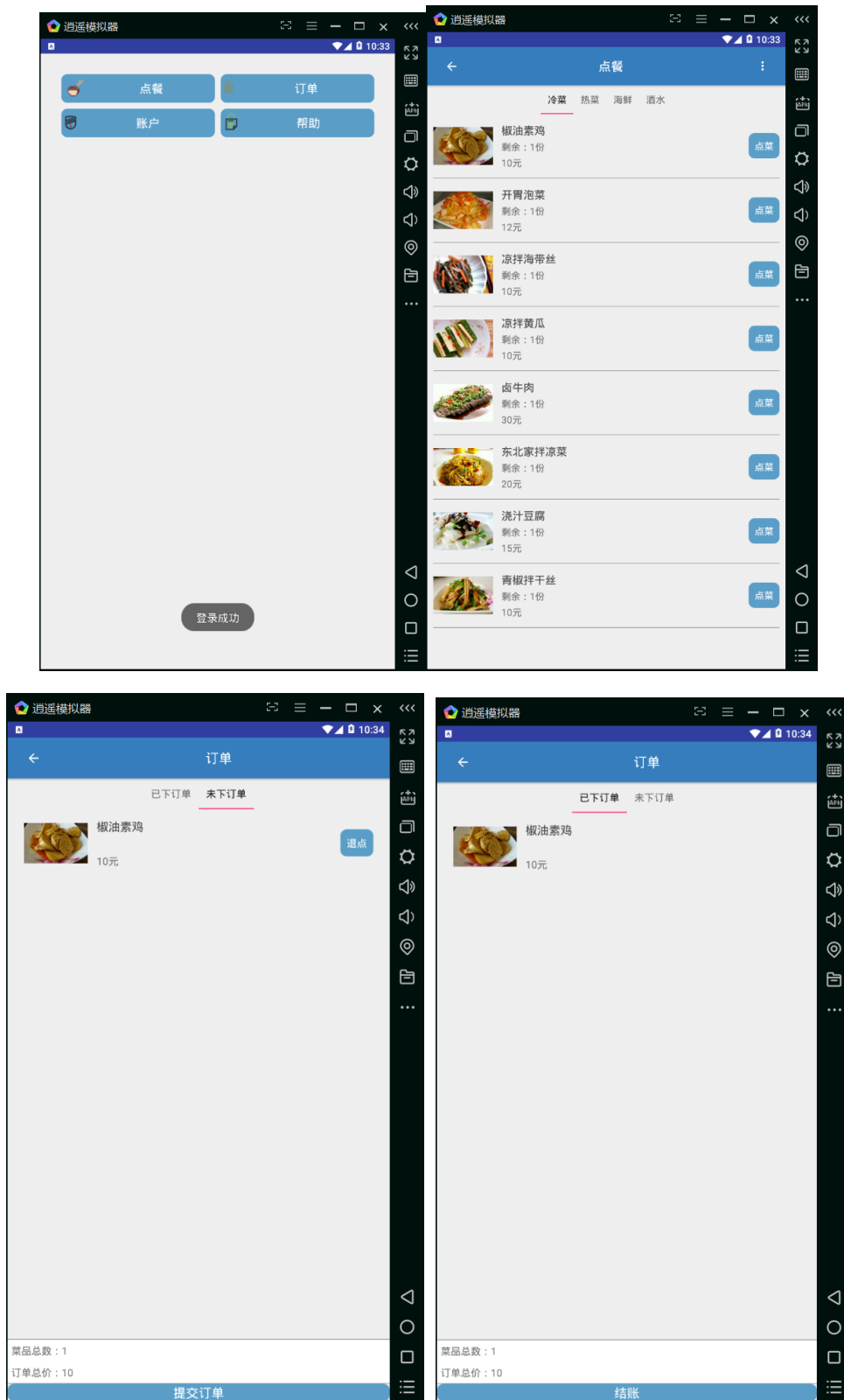
```
public void setUser(User user) {
    Gson gson = new Gson();
    sharedPreferences = mContext.getSharedPreferences(AppGlobal.SPKey.RESOURCE, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    System.out.println("app:"+user);
    String userString = gson.toJson(user);
    //仅仅起传值的作用
    System.out.println("app:"+user);
    editor.putString(AppGlobal.SPKey.USER, userString).apply();
}

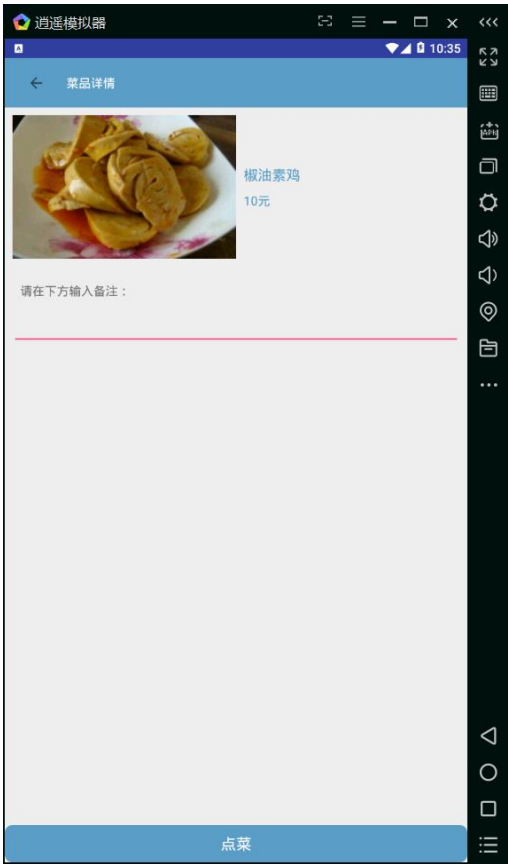
public User getUser() {
    SharedPreferences sharedPreferences = mContext.getSharedPreferences(AppGlobal.SPKey.RESOURCE, Context.MODE_PRIVATE);
    String userString = sharedPreferences.getString(AppGlobal.SPKey.USER, defValue: null);
    return new Gson().fromJson(userString, User.class);
}
```

在 `FoodOrderView` 获得 `User` 对象，然后判断 `user != null && user.getOldUser` 为真，即为老用户。

```
submitButton.setOnClickListener((v) -> {
    if(currentPageIndex == AppGlobal.Lable.ORDERED_LABEL) {
        sup = SharedPreferenceUtil.getInstance(mContext);
        user = sup.getUser();
        if(user != null && user.getOldUser()) {
            showToast("您好，老顾客，本次你可享受7折优惠！！！！")
        }
    } else {
        showToast("暂时无法下单");
    }
});
```

程序调试结束后，界面显示如下：





## 4. 结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。如 Android 程序调试方法之间的优缺点比较（文字、图标、代码辅助等）

本次练习使用了各种各样的 adapter 类来部署数据，学会了使用 Gridview+BaseAdapter 实现了九宫格按钮的加载，学会了使用 Fragment+FragmentManager 来实现了导航页面的加载与左右滑动切换，学会了使用 RecyclerView+RecyclerView.adapter 来加载不同的 food 列表项，学会了使用 ViewPager+PagerAdapter 实现视图组件的左右滑动加载，学会了使用接口回调 adapter 中的组件的点击监听事件，学会了构建 SharedPreferences 工具类来快速存储数据。

碰到的问题也是最大的问题即从 adapter 中回调里面组件的点击监听事件到 fragment 中

首先创建一个回调接口

```
public interface OnListFragmentInteractionBtnListener {
    // TODO: Update argument type and name
    public void onClick(Food item);
}
```

然后在 adapter 类中定义一个 OnListFragmentInteractionBtnListener mListener 接口变量，并构建传入接口变量的方法。

```
public void setOnListFragmentInteractionBtnListener(OnListFragmentInteractionBtnListener mBtnListener) {
    this.mBtnListener=mBtnListener;
}
```

然后设置 adapter 中组件的监听事件，并判断接口变量是否为 null，不为 null 则调用接口类下的 onClick 方法

```
holder.foodOrderBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (null != mBtnListener) {
            // Notify the active callbacks interface (the activity, if
            // fragment is attached to one) that an item has been selected
            mBtnListener.onClick(holder.mItem);
        }
    }
});
```

最后在 Fragment 中实例化 adapter 并调用 setOnListFragmentInteractionBtnListener 方法，并实例化接口变量，同时向接口的 onClick 方法中传入 food 对象，并调用这个实现方法里的事件，实现接口回调。



```
foodRecyclerViewAdapter.setOnListFragmentInteractionBtnListener(new OnListFragmentInteractionBtnListener() {  
    @Override  
    public void onClick(Food item) { //添加食物  
        item.setOrder(!item.isOrder());  
        foodRecyclerViewAdapter.updateData(dataList);  
        SharedPreferencesUtil sup = SharedPreferencesUtil.getInstance(mContext);  
        sup.operateFood(item, item.isOrder());  
    }  
});
```

第二个错误为，设置了 **adapter** 后报出异常：

Attempt to invoke virtual method 'int android.view.View.getImportantForAccessibility()'

原因是 **FunctionAdapter** 类中 **getView()**方法中返回了 **null** 值。应该是返回配置好的视图

```
return convertView;
```

第三个错误为经常性的发生的 **setAdapter(adapter)**空指针 **nullPointer**

当设置完 **adapter** 后，报这个错误一般是 **adapter** 需要绑定的布局视图或者其中组件的视图资源 ID 绑定存在问题。

## 5.参考文献

- 1.[Android GridView 实现桌面效果](#)
- 2.[Android 基础入门教程——2.4.9 GridView\(网格视图\)的基本使用](#)
- 3.[github:参考潘梦泽学长代码书写风格，使用其食品图片资源，修改并使用其部分布局资源](#)
- 4.[Android 如何实现简单的手机桌面 GridView](#)
- 5.[Android ViewHolder 的基本使用](#)
6. [在适配器 Adapter 中回调他的点击事件到 activity 或者 fragment 当中](#)
- 7.[\[ 已 解 决 \]Attempt to invoke virtual method 'int android.view.View.getImportantForAccessibility\(\)'](#)
- 8.[Android-存储: SharedPreferences 使用及其存储类型](#)
- 9.[个人感觉好用的 sharedpreference 工具类写法](#)
- 10.[Android 导航条效果实现（六） TabLayout+ViewPager+Fragment](#)
- 11.[Basefragment 的封装使用](#)
- 12.[ViewPager 用法详细解析](#)
- 13.[Android 开发: Toolbar 基本使用和自定义 Toolbar](#)
- 14.[Android Toolbar 样式定制详解](#)
- 15.[Android 的 Toolbar\(含溢出菜单设置\[弹出菜单的使用\]\)的使用 PopMenu 的样式](#)
- 16.[Android Toolbar 使用系统原生返回键，并改变其颜色，自定义图片替换系统原生返回键](#)
- 17.[Android 开发从一个 activity 设置跳转到另一个 activity 中的一个 fragment 中的一个 viewpager 中的某一个页面](#)
- 18.[Android 错误之 setAdapter\(adapter\)空指针 nullPointer 解决办法](#)
- 19.[Android ViewPager Fragment 切换刷新数据，解决生命周期只走一次的问题](#)
- 20.[ViewPager 与 PagerAdapter 实现图片左右滑动效果](#)
- 21.[Android - 布局管理器 LayoutInflater 及 LayoutParams 动态设置宽高属性](#)
- 22.[Android 基础--ViewPager 的 PagerAdapter 的介绍](#)
- 23.[android 一个 BaseAdapter 的使用（LayoutInflater 加载自定义布局）](#)
- 24.[Glide 4.7.1 学习使用](#)
- 25.[Gson 的入门使用](#)