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	15 级	专业（方向）	移动信息工程（移动互联网）
学号	15352049	姓名	陈新堉
电话	13670468594	Email	619156737@qq.com
开始日期	2017.10.21	完成日期	2017.10.24

一、 实验题目

Intent，Bundle 的使用以及 RecyclerView 和 ListView 的应用

二、 实现内容

实现一个 商品表，有两个界面：
界面 1：商品列表和购物车：

点击悬浮按钮切换到购物车：

界面二：商品的详细信息，点击商品列表或购物车某一项可以进入

布局要求：

1. 商品表界面：

   每一项包括圆圈和商品名称。圆圈内商品首字母处于圆圈中心，首字母白色，商品名称黑色。

2. 购物车界面：

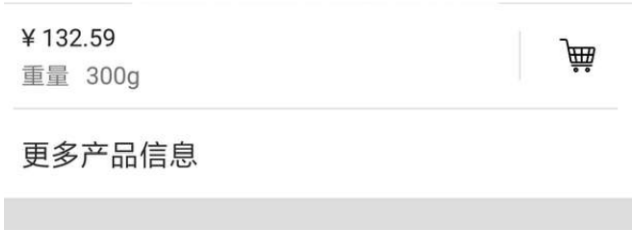
   在商品表界面的基础上添加价格信息。

3. 商品详情页面顶部：

   占整个屏幕 1/3，包括星标，返回按钮，商品名称，商品图片。使用 RelativeView，星标底部和商品名称对齐，商品名称和返回按钮左对齐。

4. 商品详情中部：

   包括黑色价格信息和灰色的其他信息。购物车图标，垂直分界线和水平分界线。



5. 商品详情页面底部：

   使用 ListView

6. 顶部没有标题行

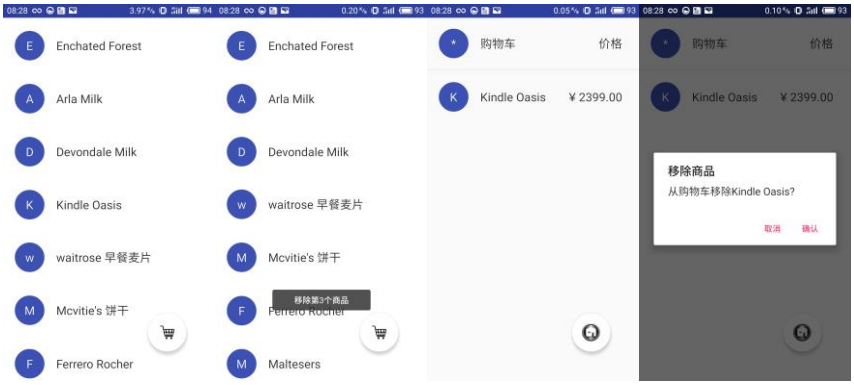
逻辑要求：

1．商品表页面使用 RecyclerView，单击进入详情页面，长按删除并有 Toast 提示。
2．购物车页面使用 ListView，单击进入详情页面，长按弹出对话框询问是否删除，取消对话框消失，确定删除对应的项。

3．商品表和购物车通过 FloatingActionButton 切换，切换后 FloatingActionButton 图案要切换。
4．详情页面返回按钮可以返回商品表或购物车页面。
5．详情页面的星标点击会切换空心和实心。
6．详情页面点击购物车会把商品添加到购物车。

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1． 商品表和购物车的布局

商品表使用 RecyclerView:

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/recycler_view"
    ndroid:layout_width="368dp"
    android:layout_height="wrap_content"
    android:visibility="visible"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"/>
```

购物车使用 ListView

```
<ListView
        android:id="@+id/shoppinglist"
        android:layout_height="wrap_content"
        android:layout_width="0dp"
        android:visibility="gone"
        tools:layout_editor_absoluteY="0dp"
        tools:layout_editor_absoluteX="0dp" />
```

FloatingActionButton 来切换购物车和商品表

```
<android.support.design.widget.FloatingActionButton
        android:id="@+id/switch_button"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_marginRight="50dp"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_marginBottom="50dp"
        app:borderWidth="0dp"
        android:src="@drawable/shoplist"
        android:backgroundTint="@color/white"
        android:background="#00000000"
        app:rippleColor="#00000000"
        android:layout_marginEnd="50dp" />
```

2. 商品表实现逻辑

获取 RecyclerView:

```
final RecyclerView goodsRecyclerView = (RecyclerView) findViewById(R.id.recycler_view);
goodsRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```

构造用于 RecyclerView 的 Adapter:

```
final CommonAdapter goodslistAdapter = new CommonAdapter(this,
R.layout.item, GoodsList)
{
  @Override
  public void convert(ViewHolder holder, Map<String, Object> s) {
    TextView name = holder.getView(R.id.name);
    name.setText(s.get("name").toString());
    TextView abbr = holder.getView(R.id.abbr);
    abbr.setText(s.get("abbr").toString());
   }
};
```

List 的显示需要配置好每项的布局：布局文件为 item.xml，里面有两个 TextView，分别用来显示首字母，商品名。

CommonAdapter 是自定义的一个 Adapter，需要新建 Java 类文件‘CommonAdapter.java’，继承自 Android 内置的 RecyclerView.Ada-pter，定义和构造函数如下：

```
public abstract class CommonAdapter extends RecyclerView.Adapter<ViewHolder> {
    protected Context mContext;
    protected int mLayoutId;
    protected List<Map<String, Object>> mDatas;
    protected LayoutInflater mInflater;
    private OnItemClickListener mOnItemClickListener = null;

    public CommonAdapter(Context context, int layout, List<Map<String, Object>> datas){
        mContext = context;
        mLayoutId = layout;
        mDatas = datas;
        mInflater = LayoutInflater.from(context);
    }
```

之后需要重载 onCreateViewHolder，onBindViewHolder，getItemCount。

还要声明一个 convert 的抽象函数，在初始化 CommonAdapter 对象时重载。
为了使 RecyclerView 有对于点击和长按的监听器，还需要定义 OnClickItemListener 的接口以及 setOnItemClickListener 函数。
CommonAdapter 的成员变量有一个 ViewHolder 用来保存 Adapter 对应的 RecyclerView。ViewHolder 也是自定义一个的 java 类‘ViewHolder.java’，继承自 RecyclerView.ViewHolder

```
public class ViewHolder extends RecyclerView.ViewHolder{
    private SparseArray<View> mViews;//存储 list 的子 view
    private View mConvertView;//存储
    private  Context mContext;
```

Adapter 要填充列表内容，需要一个 List 来告知要填充的内容

```
List<Map<String, Object>> GoodsList = new ArrayList<>();
```

在 OnCreate 函数外面定义函数来填充商品列表 GoodsList:

```
private void initGoodsList()
{
 String[] goodName = new String[]{"Enchated Forest", "Arla Milk", "Devondale Milk",
                         "Kindle Oasis", "waitrose 早餐麦片",
                         "Mcvitie's 饼干", "Ferrero Rocher",
                         "Maltesers","Lindt","Borggreve"};
 for(int i = 0; i < 10; i++)
 {
   Map<String, Object> temp = new LinkedHashMap<>();
   temp.put("abbr", goodName[i].substring(0,1));
   temp.put("name", goodName[i]);
   GoodsList.add(temp);
  }
```

处理单击和长按事件:

```
goodslistAdapter.setOnItemClickListener(new CommonAdapter.OnItemClickListener() {
  @Override
  public void onClick(int position) {
     String chose_name = GoodsList.get(position).get("name").toString();
     Intent intent = new Intent(MainActivity.this, detail.class);
     intent.putExtra("goodsName", chose_name);
     startActivityForResult(intent,1);
   }

   @Override
   public boolean onLongClick(int position) {
     String index = Integer.toString(position);
     GoodsList.remove(position);
     goodslistAdapter.notifyItemRemoved(position);
     Toast.makeText(MainActivity.this,"移除第"+index+"个商品",Toast.LENGTH_SHORT).show();
     return true;
    }
});
```

长按的函数需要返回 true，告诉其他 listener 这个 item 已经被长按处理过了。

3. 购物车实现逻辑

获取 ListView，并为其使用 SimpleAdapter 为其填充数据:

```
final ListView shoppingListView = (ListView) findViewById(R.id.shoppinglist);
simpleAdapter = new SimpleAdapter(this, ShoppingList,R.layout.shoppinglist_layout,new
               String[]{"abbr","name", "price"},new int[]{R.id.abbr,R.id.name,R.id.price});
shoppingListView.setAdapter(simpleAdapter);
```

List 的显示需要配置好每项的布局：布局文件为 shoppinglist_layout.xml，里面有三个 TextView，分别用来显示首字母，商品名和价格。
填充 Adapter 需要一个 List:

```
List<Map<String, Object>> ShoppingList = new ArrayList<>();
```

初始化购物车列表:

```
private void initShoppingList()
```

```
{
  String[] goodName = new String[]{"购物车"};
  String[] goodPrice = new String[]{"价格"};
 Map<String, Object> temp0 = new LinkedHashMap<>();
 temp0.put("abbr", "*");
 temp0.put("name", goodName[0]);
 temp0.put("price", goodPrice[0]);
 ShoppingList.add(temp0);
}
```

处理单击事件：

```
shoppingListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
 @Override
 public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
 final int pos= position;
 if(pos != 0)
 {
   shoppinglist_alertdialog.setPositiveButton("确认", new DialogInterface.OnClickListener() {
     @Override
     public void onClick(DialogInterface dialog, int which) {
     ShoppingList.remove(pos);
     simpleAdapter.notifyDataSetChanged();}
     }).setMessage("从购物车移除"+ShoppingList.get(pos).get("name")+"?")
       .create()
       .show();
 }
 return true;
 }
});
```

注意第 0 项是表头，点击不能有反应。单击的响应是调用一个 AlertDialog 显示对话框。

处理长按事件：

```
shoppingListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
  @Override
  public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
    if(i != 0) {
      String chose_name = ShoppingList.get(i).get("name").toString();
      Intent intent = new Intent(MainActivity.this, detail.class);
      intent.putExtra("goodsName", chose_name);
      startActivityForResult(intent, 1);
  }
 }
});
```

4. FloatingActionButton 实现逻辑

```
final FloatingActionButton SwitchBtn = (FloatingActionButton)findViewById(R.id.switch_button);
SwitchBtn.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
   if(goodsRecyclerView.getVisibility() == View.VISIBLE)
   {
     goodsRecyclerView.setVisibility(View.GONE);
     shoppingListView.setVisibility(View.VISIBLE);
     SwitchBtn.setImageResource(R.drawable.mainpage);
   }
   else if(shoppingListView.getVisibility() == View.VISIBLE)
  {
     goodsRecyclerView.setVisibility(View.VISIBLE);
     shoppingListView.setVisibility(View.GONE);
     SwitchBtn.setImageResource(R.drawable.shoplist);//如何设置图片为 background?
   }
 }
});
```

通过判断当前哪个列表课件来判断如何重新设置列表的可见性。

get/setVisibility 可以或取或改变 visibility 性质，

setImageResource 可以改变嵌入按钮的图片，

FloatingActionButton 继承于 ImageButton，因此使用的方法类似。

5. Activity 切换

启动显示详情页面的 activity: detail，在单机时间处理时使用：

```
String chose_name = ShoppingList.get(i).get("name").toString();
Intent intent = new Intent(MainActivity.this, detail.class);
intent.putExtra("goodsName", chose_name);
startActivityForResult(intent, 1);
```

使用 putExta 把被点击的商品名传到 detail。

从 detail 返回时，需要得到 detail 传来的信息，在 OnCreate 外部重载 onActivityResult:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
 if(requestCode == 1)
 {
  if(resultCode == RESULT_OK)
  {
   String rev_name = data.getStringExtra("name");
   String rev_price = data.getStringExtra("price");
   if(rev_name != null && rev_price != null)
   {
     Map<String,Object> temp = new LinkedHashMap<>();
     temp.put("abbr", rev_name.substring(0,1));
     temp.put("name", rev_name);
     temp.put("price", rev_price);
     ShoppingList.add(temp);
     simpleAdapter.notifyDataSetChanged();
    }
   }
  }
}
```

6. 详情页面布局

详情页面外部使用 LinearLayout 垂直布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  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.chan.android_lab3.detail"
    android:orientation="vertical">
    <RelativeLayout…
    </RelativeLayout>

    <LinearLayout…
    </LinearLayout>
    <View…
    <ListView…
</LinearLayout>
```

线性布局内部有用来实现顶部的 RelativeLayout，实现中部的 LinearLayout 和实现底部列表的 ListView，他们之间使用权重来使得顶部占 1/3。还有一个 View 是用来实现分割线。

```
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:background="@color/grey">
        <ImageView…
        <ImageButton…
        <TextView…
        <ImageButton…
    </RelativeLayout>
```

顶部里面包括两个 ImageButton 实现返回和星标。一个 TextView 显示商品名称，一个 ImageView 显示图片。

```
51      <LinearLayout
52          android:layout_width="match_parent"
53          android:layout_height="0dp"
54          android:layout_weight="2"
55          android:orientation="vertical">
56          <LinearLayout…
100         </LinearLayout>
101         <View…
106         <TextView…
115     </LinearLayout>
```

中部包括一个 LinearLayout 显示价格和其他信息，TextView 显示固定的更多产品信息。

```
56          <LinearLayout
57              android:layout_width="match_parent"
58              android:layout_height="0dp"
59              android:layout_weight="1">
60              <LinearLayout…
86              </LinearLayout>
87              <View…
92              <ImageButton…
100         </LinearLayout>
```

展开上面的 LinearLayout，里面还再嵌套了 LinearLayout，和一个 ImageButton 显示购物车图标。

```
60          <LinearLayout
61              android:layout_width="0dp"
62              android:layout_height="match_parent"
63              android:layout_weight="4"
64              android:orientation="vertical">
65              <TextView…
75              <TextView…
86          </LinearLayout>
```

继续展开上面的 LinearLayout，里面两个 TextView 显示商品价格和商品信息。

7. 详情页面接收来自 MainActivity 的信息

```
Bundle extras = getIntent().getExtras();
if(extras != null)
{
  data = extras.getString("goodsName");
  for(int i = 0; i < Informations.size(); i++)
  {
   if(Informations.get(i).get("goodName").toString().equals(data))
   {
     item_NO = i;
   }
  }
}
```

data 是通过读取 MainActivity 通过 Intent 传过来商品名称，通过在一个 List 找到对应的价格，图片等信息。

8. 信息显示实现

显示图片，使用数组保存所有图片的 ID:

```
int[] imageID = {R.drawable.enchatedforest, R.drawable.arla, R.drawable.devondale, R.drawable.kindle,
R.drawable.waitrose, R.drawable.mcvitie, R.drawable.ferrero, R.drawable.maltesers,
R.drawable.borggreve};
ImageView goodsimage = (ImageView)findViewById(R.id.image_detail);
int resId = imageID[item_NO];
goodsimage.setImageResource(resId);
```

显示各种文本信息:

价格信息:

```
TextView PriceView = (TextView)findViewById(R.id.price_in_detail);
PriceView.setText(Informations.get(item_NO).get("Price").toString());
```

重量等其他信息：

```
final TextView InfoView = (TextView)findViewById(R.id.info);
InfoView.setText(Informations.get(item_NO).get("infoType").toString()+" "+Informations.get(item_NO).get("info").toString());
```

商品名称：

```
TextView goods_name = (TextView)findViewById(R.id.goods_name);
goods_name.setText(data);
```

9. 按钮实现逻辑

返回按钮：

```
ImageButton BackButton =
(ImageButton)findViewById(R.id.back_button);
BackButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
   Intent intent = new Intent();
   intent.putExtra("name",choiceName);
   intent.putExtra("price", choicePrice);
   setResult(RESULT_OK, intent);
   finish();
 }
});
```

设置了一个按钮的监听器，返回按钮被按下，把一个有商品名称和价格的 Extra 的 Intent 通过 setResult 返回，调用 finish 销毁当前 activity。

星标：

```
final ImageButton Star = (ImageButton)findViewById(R.id.Star);
Star.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
   if(Star.getTag().toString().equals("0"))
   {
     Star.setTag("1");
     Star.setImageResource(R.drawable.full_star);
   }
   else if(Star.getTag().toString().equals("1"))
   {
     Star.setTag("0");
     Star.setImageResource(R.drawable.empty_star);
   }
 }
});
```

和之前 FloatingActionButton 的实现类似，但是由于没有方法得到 star 使用的图片来源的 ID，于是只能多加一个 tag 属性来判断 star 现在是空心还是实心。

10. 返回到主 Activity

```
@Override
public void onBackPressed(){
 Intent intent = new Intent();
 intent.putExtra("name",choiceName);
 intent.putExtra("price", choicePrice);
 setResult(RESULT_OK, intent);
 finish();
}
```

之前返回按钮已经得到一种回到 mianActivity 的方法。通过重载 onBackPressed 函数可以实现在手机物理返回键被按下时销毁当前 activity。具体操作和返回按钮一致。

（3） 实验遇到困难以及解决思路

1. 为 recyclerView 添加动画后长按闪退，发现是 notify 函数写错：

```
GoodsList.remove(position);
goodslistAdapter.notifyItemChange(position);
```

上面是出错的代码，因为是 remove 一个项，因此需要提示的是 ItemRemove。

```
GoodsList.remove(position);
goodslistAdapter.notifyItemRemoved(position);
```

2.星标和返回按钮一直有个灰色的背景，只要把颜色设置为透明颜色就行。颜色编码使用 argb 模式，a 是透明度，为 0 时就是透明的。

3. 由于把自定的 ViewHolder 命名为 ViewHolder，于是在使用时要确定编译器把这个 ViewHolder 关联到自己的类。

四、 课后实验结果

使用 wasabeef 的 github 上面提供的其他动画：

在设置 recyclerView 的 adapter 的地方修改使用的动画：

```
AlphaInAnimationAdapter animationAdapter = new AlphaInAnimationAdapter(goodslistAdapter);
animationAdapter.setDuration(1000);
goodsRecyclerView.setAdapter(animationAdapter);
goodsRecyclerView.setItemAnimator(new LandingAnimator());
```

这里进入的动画效果过是列表由浅变深，删除的效果是向屏幕外扩散。

五、 实验思考及感想

这次实验花费的时间很多，主要是对于 RecylerView 的使用，自定义 Adapter，ViewHolder 还很不懂，加上实验报告在这两方面没有很完整，需要自己去查找很多的资料。这次的实验主要参考了一些博客，《第一行代码》。实际上对自定义 ViewHolder，Adapter 还是需要再熟悉一下。

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。