

# 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：郑贵峰

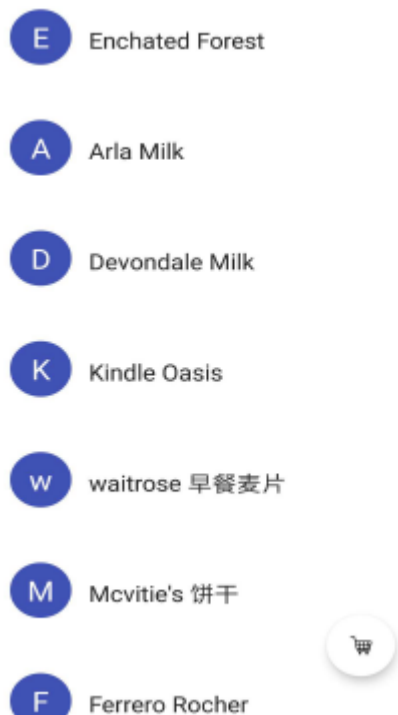
年级	15 级	专业 ( 方向 )	软件工程 ( 移动信息工程 ) 互联网方向
学号	15352211	姓名	林苗
电话	13763360840	Email	554562948@qq.com
开始日期	2017.10.22	完成日期	2017.10.26

### 一、 实验题目

1. 复习事件处理
2. 学习 Intent、Bundle 在 Activity 跳转中的应用
3. 学习 RecyclerView、ListView 以及各类适配器的用法

### 二、 实现内容

本次实验模拟实现一个商品表，有两个界面，第一个界面用于呈现商品，如下所示：



点击右下方的悬浮按钮可以切换到购物车：



上面两个列表点击任意一项后，可以看到详细的信息：



实验要求：

布局方面：

1. 商品表界面

每一项为一个圆圈和一个名字，圆圈与名字均竖直居中。圆圈为名字的首字母，首

字母要处于圆圈的中心，首字母为白色，名字为黑色，圆圈的颜色自定义即可，建议用深色的颜色，否则白色的首字母可能看不清。

2. 购物车列表界面  
在商品表界面的基础上增加一个价格，价格为黑色。
3. 商品详情界面顶部



顶部占整个界面的三分之一。每个商品的图片与这块 view 等高。返回图标处于这块 View 的左上角，商品名字处于左下角，星标处于右下角，它们与边距都有一定距离，自己调出合适的距离即可。需要注意的是，返回图标与名字左对齐，名字与星标底边对齐，建议使用 RelativeLayout 实现。

4. 商品详情界面中部

¥ 132.59  
重量 300g



更多产品信息

使用的黑色 argb 编码值为#D5000000，稍微偏灰色一点的“重量”、“300g”的 argb 编码值为#8A000000。价格那一栏下面有一条分割线，argb 编码值为#1E000000，右边购物车符号的左边也有一条分割线，argb 编码值也是#1E000000，这条分割线要求高度与购物车符号的高度一致，并且竖直居中。字体的大小看着调就可以了。“更多资料”底部的分割线高度自定，argb 编码值与前面的分割线一致。

5. 商品详情页面底部

一键下单

分享商品

不感兴趣

查看更多商品促销信息

6. 这次的两个界面顶部都没有标题栏，要用某些方法把它们去掉。

逻辑方面要求：

1. 使用 RecyclerView 实现商品列表。点击商品列表中的某一个商品会跳转到该商品详情界面，呈现该商品的详细信息；长按商品列表中的第  $i$  个商品会删除该商品，并且弹出 Toast，提示“移除第  $i$  个商品”。
2. 点击右下方的 FloatingActionButton，从商品列表切换到购物车或从购物车切换到商品列表，并且 FloatingActionButton 的图片要做相应改变。可通过设置 RecyclerView 不可见，ListView 可见来实现从商品列表切换到购物车。可通过设置 RecyclerView 可见，ListView 不可见来实现从购物车切换到商品列表。
3. 使用 ListView 实现购物车。点击购物车的某一个商品会跳转到商品详情界面，呈现该商品的详细信息；长按购物车中的商品会弹出对话框询问是否移除该商品，点击确定则移除该商品，点击取消则对话框消失。



对话框中的商品为被长按的商品。

- 商品详情界面中点击返回图标会返回上一层，点击星标会切换状态，如果原先是空心星星，则会变成实心星星；如果原先是实心星星，则会变成空心星星。点击购物车图标会将该商品添加到购物车中并弹出 Toast 提示：“商品已添加到购物车”。  
注：不要求判断购物车是否已有该商品，即如果已有一件该商品，添加之后则显示两个即可。未退出商品详情界面时，点击多次购物车图标可以只添加意见商品也可以添加多件到购物车中。

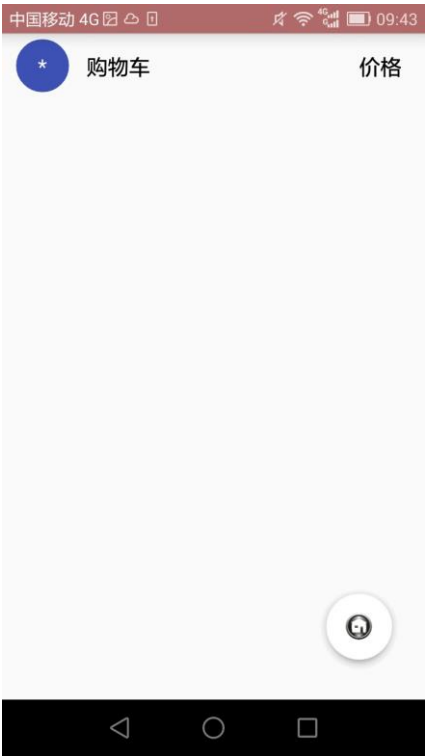
### 三、 课堂实验结果

#### (1) 实验截图

商品列表界面



购物车初始界面



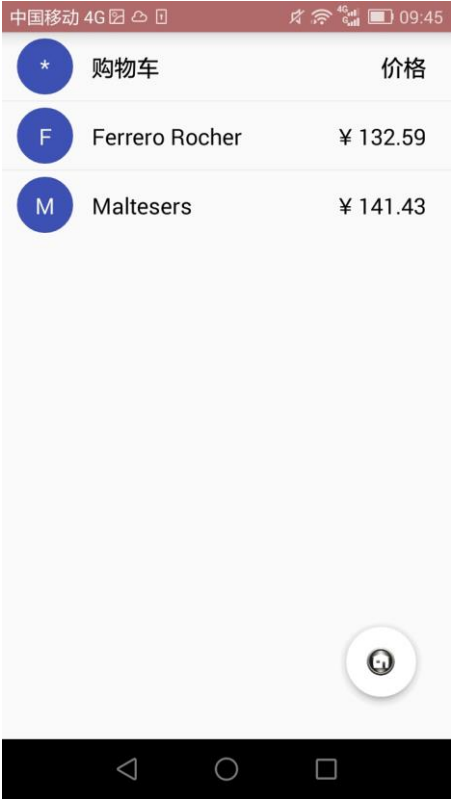
单击商品



选择添加到购物车，弹出 Toast 信息



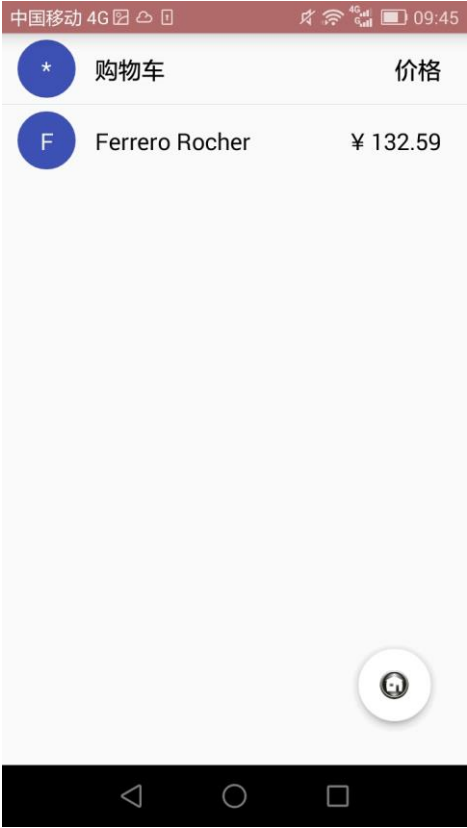
购物车列表



在购物车列表中长按商品



点击确定，购物车中只剩下一个商品

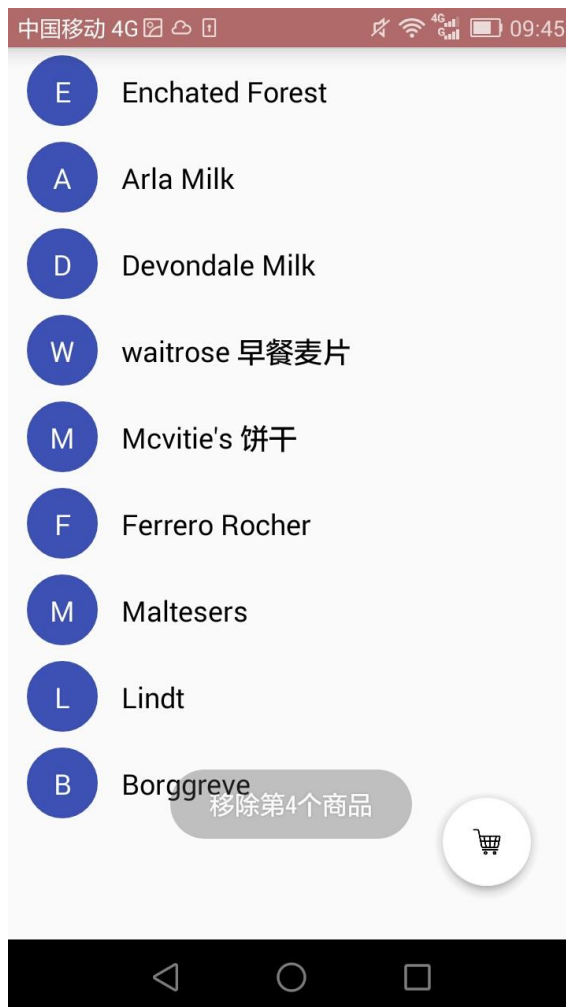




点击空心星星




商品列表中长按商品，弹出 Toast 信息



## (2) 实验步骤及关键代码

首先新建商品的类

 Info

定义以下属性

```
public class Info implements Serializable {

    private String name; //名字
    private String price; // 价格
    private String type; //类型
    private String info; //信息
    private String background; //背景
}
```

定义构造函数

```
public Info(String name, String price, String type, String info, String background) {
    this.name = name;
    this.price = price;
    this.type = type;
    this.info = info;
    this.background = background;
}
```

定义设置、获得属性函数，以 name 为例

```
public String getName() { return name; }
```


```
public void setName(String name) { this.name = name; }
```

同理定义价格、类型、信息、背景的相关设置和获得函数。

定义一个获得大写首字母的函数

```
public char getFirstLetter() {
    char first = name.charAt(0);
    if (first >= 97 && first <= 122) {
        first -= 32;
    }
    return first;
}
```

Mainactivity 的布局

 activity\_main.xml

整体使用 RelativeLayout

```
<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:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.linm.lab3.MainActivity">
```

购物车列表 ListView

```
<ListView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/list">

</ListView>
```

商品列表 RecyclerView

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/recycle_view"
    >
</android.support.v7.widget.RecyclerView>
```

浮动元素

需要先在 gradle 中添加依赖

```
compile 'com.android.support:design:25.4.0'
```

在 activity\_main.xml 中:

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/convert"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/shoplist"
    app:backgroundTint="#FFFFFF"
    app:backgroundTintMode="src_atop"
    app:rippleColor="#FFFFFF"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    >
</android.support.design.widget.FloatingActionButton>
```

在 MainActivity 中，创建商品信息为全局静态变量，便于其它类访问

```
public static List<Info> Infos = new ArrayList<Info>() {{
    add(new Info("Enchanted Forest", "¥ 5.00", "作者", "Johanna Basford", "1"));
    add(new Info("Arla Milk", "¥ 59.00", "产地", "德国", "2"));
    add(new Info("Devondale Milk", "¥ 79.00", "产地", "澳大利亚", "3"));
    add(new Info("Kindle Oasis", "¥ 2399.00", "版本", "8GB", "4"));
    add(new Info("waitrose 早餐麦片", "¥ 179.00", "重量", "2Kg", "5"));
    add(new Info("McVitie's 饼干", "¥ 14.90", "产地", "英国", "6"));
    add(new Info("Ferrero Rocher", "¥ 132.59", "重量", "300g", "7"));
    add(new Info("Maltesers", "¥ 141.43", "重量", "118g", "8"));
    add(new Info("Lindt", "¥ 139.43", "重量", "249g", "9"));
    add(new Info("Boraggrove", "¥ 28.90", "重量", "640g", "10"));
}}
```

下面开始商品界面 RecyclerView 的编写

RecyclerView 每一项元素的布局

 info.xml

使用 LinearLayout 线性布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/FirstLetter"
        android:textSize="18sp"
        android:layout_marginStart="12dp"
        android:textColor="#FFFFFF"
        android:background="@drawable/d"
        android:gravity="center"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:id="@+id/name"
        android:textSize="18sp"
        android:layout_weight="1"
        android:textColor="#000000"
        android:padding="15dp" />
</LinearLayout>
```


在 gradle 文件夹下的 build.gradle 中添加依赖

```
allprojects {  
    repositories {  
        jcenter()  
        //lab3 add start  
        maven {  
            url "https://maven.google.com"  
        }  
        //lab 3 add end  
    }  
}
```

在 main 文件夹中的 build.gradle 中添加依赖

```
compile 'com.android.support:recyclerview-v7:25.4.0'  
compile 'com.android.support:cardview-v7:25.4.0'
```

自定义 RecyclerView 的 ViewHolder

 ViewHolder

使用一个 SparseArray 数组存储 list\_item 的子 view

```
public class ViewHolder extends RecyclerView.ViewHolder {  
  
    private SparseArray<View> mViews; //存储 list_Item 的子 View  
    private View mConvertView; //存储 list_Item
```

构造函数

```
public ViewHolder(Context context, View itemView, ViewGroup parent) {  
    super(itemView);  
    mConvertView = itemView;  
    mViews = new SparseArray<View>();  
}
```

获取 ViewHolder 实例

```
//获取 viewHolder 实例  
public static ViewHolder get(Context context, ViewGroup parent, int layoutId) {  
    View itemView = LayoutInflater.from(context).inflate(layoutId, parent, false);  
    ViewHolder holder = new ViewHolder(context, itemView, parent);  
    return holder;  
}
```


如果 ViewHolder 尚未将子 View 缓存到 SparseArray 数组中时, 仍然需要通过 findViewById() 创建 View 对象, 如果已创建, 通过 id 获取 View 返回

```

//通过id获取view
public <T extends View> T getView(int viewId) {
    View view = mViews.get(viewId);
    if (view == null) {
        //创建view
        view = mConvertView.findViewById(viewId);
        //将view存入mViews
        mViews.put(viewId, view);
    }
    return (T) view;
}

```

自定义 RecyclerView.Adapter

 CommonAdapter

定义成员变量

```

public abstract class CommonAdapter<T> extends RecyclerView.Adapter<ViewHolder>{
    protected Context mContext;
    protected int mLayoutId;
    protected List<T> mDatas;
    private OnItemClickListener mOnItemClickListener = null;
}

```

构造函数

```

public CommonAdapter(Context context, int layoutId, List<T> datas) {
    mContext = context;
    mLayoutId = layoutId;
    mDatas = datas;
}

```

创建 Item 视图，返回相应的 ViewHolder

```

@Override
//创建Item视图，返回相应的ViewHolder
public ViewHolder onCreateViewHolder(final ViewGroup parent, int viewType) {
    ViewHolder viewHolder = ViewHolder.get(mContext, parent, mLayoutId);
    return viewHolder;
}

```

绑定数据到正确的 Item 视图上

```

@Override
public void onBindViewHolder(final ViewHolder holder, int position) {
    //绑定数据到正确的Item视图上
    convert(holder, mDatas.get(position));
}

```

重写获得列表总数的方法

```

@Override
//获得列表项总数
public int getItemCount() { return mDatas.size(); }

```

删除列表项的方法

```
//删除列表项
public void removeItem(int position) {
    mDatas.remove(position);
    notifyItemRemoved(position);
}
```

RecyclerView 没有 onItemClick 方法，需要在适配器中实现。在 Adapter 中设置一个监听器，当 itemView 被点击的时候，调用该监听器并将 itemView 的 position 作为参数传递出去。

添加接口及函数

```
public interface OnItemClickListener {
    void onClick(int position);
    void onLongClick(int position);
}

public void setOnItemClickListener(OnItemClickListener onItemClickListener) {
    this.mOnItemClickListener = onItemClickListener;
}
```

在 onBindViewHolder() 中添加

```
if (mOnItemClickListener != null) {
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mOnItemClickListener.onClick(holder.getAdapterPosition());
        }
    });
    holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
            mOnItemClickListener.onLongClick(holder.getAdapterPosition());
            return true;
        }
    });
}
```

定义完 ViewHolder 和 Adapter 之后，我们就可以回 MainActivity 中定义 RecyclerView 了。

在 onCreate() 中，定义商品列表的 arraylist

```
//商品列表list
final List<Map<String, Object>> data = new ArrayList<>();
```

定义首字母的数组

```
//首字母list
char[] FirstLetter = new char[Infos.size()];
for (int i = 0; i < Infos.size(); i++) {
    char x = Infos.get(i).getFirstLetter();
    FirstLetter[i] = x;
}
```

定义名字的数组

```
//名字list
String[] name = new String[Infos.size()];
for (int i = 0; i < Infos.size(); i++) {
    String x = Infos.get(i).getName();
    name[i] = x;
}
```

定义价格的数组

```
//价格list
String[] price = new String[Infos.size()];
for (int i = 0; i < Infos.size(); i++) {
    String x = Infos.get(i).getPrice();
    price[i] = x;
}
```

商品列表的 data 中，加入 string 到对象的映射

```
//商品列表list添加
for (int i = 0; i < Infos.size(); i++) {
    Map<String, Object> temp = new LinkedHashMap<>();
    temp.put("FirstLetter", FirstLetter[i]);
    temp.put("name", name[i]);
    data.add(temp);
}
```

在主界面 activity\_main.xml 中，定义 RecyclerView 的布局

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/recycle_view"
    >
</android.support.v7.widget.RecyclerView>
```

将 java 文件中使用的 RecyclerView 与布局中的 id 绑定起来，设置为线性布局

```
mRecyclerView = (RecyclerView)findViewById(R.id.recycle_view);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```



重写适配器方法进行数据绑定

```
commonAdapter = new CommonAdapter<Map<String, Object>>(this, R.layout.info, data) {  
    @Override  
    public void convert(ViewHolder holder, Map<String, Object> s) {  
        TextView name = holder.getView(R.id.name);  
        name.setText(s.get("name").toString());  
        TextView first = holder.getView(R.id.FirstLetter);  
        first.setText(s.get("FirstLetter").toString());  
    }  
};
```

设置适配器

```
mRecyclerView.setAdapter(commonAdapter);
```

在 commonAdapter 的 setOnItemClickListener 中实现商品单击和长按的事件处理

```
commonAdapter.setOnItemClickListener(new CommonAdapter.OnItemClickListener) {
```

商品列表单击

```
@Override  
//商品列表单击  
public void onClick(int position) {  
    Intent intent = new Intent(MainActivity.this, InfoActivity.class);  
    Bundle bundle = new Bundle();  
    String productname = data.get(position).get("name").toString();  
    bundle.putString("name", productname); //属性为name, 数据为productname  
    intent.putExtras(bundle);  
    startActivity(intent); //将intent传入  
}
```

商品列表长按

```
@Override  
//商品列表长按  
public void onLongClick(int position) {  
    if (position < Infos.size()) {  
        commonAdapter.removeItem(position);  
        Toast.makeText(MainActivity.this, "移除第" + (position + 1) + "个商品", Toast.LENGTH_SHORT).show();  
    }  
}
```

接下来用 ListView 实现购物车

商品列表每一项的布局设置

 shoplistinfo.xml

使用 LinearLayout 布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/FirstLetter"
        android:textSize="18sp"
        android:layout_marginStart="12dp"
        android:textColor="#FFFFFF"
        android:background="@drawable/d"
        android:gravity="center"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/name"
        android:textSize="18sp"
        android:layout_weight="1"
        android:textColor="#000000"
        android:padding="15dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/price"
        android:textSize="18sp"
        android:layout_weight="1"
        android:textColor="#000000"
        android:padding="15dp"
        android:gravity="end"
        android:layout_marginEnd="5dp"/>
</LinearLayout>
```

在 MainActivity 中定义全局静态变量 shoplist 和它的适配器 simpleListAdapter

```
public static List<Map<String, Object>> shoplist = new ArrayList<Map<String, Object>>() {{
    Map<String, Object> t = new LinkedHashMap<>();
    t.put("FirstLetter", "*");
    t.put("name", "购物车");
    t.put("price", "价格");
    add(t);
}};

public static SimpleAdapter simpleListAdapter;
```

与 xml 的布局绑定

```
final ListView LV = (ListView) findViewById(R.id.list);
```

绑定适配器

```
simpleListAdapter = new SimpleAdapter(this, shoplist, R.layout.shoplistinfo,  
    new String[] {"FirstLetter", "name", "price"}, new int[] {R.id.FirstLetter, R.id.name, R.id.price});  
LV.setAdapter(simpleListAdapter);
```

实现购物车单击事件

```
LV.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
        if (i!=0) {  
            Intent intent = new Intent(MainActivity.this, InfoActivity.class);  
            Bundle bundle = new Bundle();  
            String productname = shoplist.get(i).get("name").toString();  
            bundle.putString("name", productname);  
            intent.putExtras(bundle);  
            startActivity(intent);  
        }  
    }  
});
```

实现购物车长按事件

```
LV.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {  
    @Override  
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int position, long id) {  
        if (position!=0) {  
            AlertDialog.Builder message = new AlertDialog.Builder(MainActivity.this);  
            message.setTitle("移除商品");  
            message.setMessage("从购物车移除" + shoplist.get(position).get("name").toString());  
            message.setPositiveButton("确定", (dialogInterface, i) -> {  
                shoplist.remove(position);  
                simpleListAdapter.notifyDataSetChanged();  
            });  
            message.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
                public void onClick(DialogInterface dialogInterface, int i) {  
                }  
            });  
            message.create().show();  
        }  
        return true;  
    }  
});
```

一开始将购物车的 ListView 设为不可见

```
LV.setVisibility(View.GONE);
```

接着设置浮动图标，实现购物车和主界面的切换

首先绑定 id

```
final FloatingActionButton convert = (FloatingActionButton) findViewById(R.id.convert);
```

定义一个 tag 为 false


```
private boolean tag1 = false;
```

实现 onClick 函数

```
convert.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if (!tag1) {  
            convert.setImageResource(R.mipmap.mainpage);  
            tag1 = true;  
            LV.setVisibility(View.VISIBLE); //将购物车列表设为可见  
            mRecyclerView.setVisibility(View.GONE); //商品列表不可见  
        } else {  
            convert.setImageResource(R.mipmap.shoplist);  
            tag1 = false;  
            LV.setVisibility(View.GONE); //购物车列表不可见  
            mRecyclerView.setVisibility(View.VISIBLE); //商品列表可见  
        }  
    }  
});
```

接下来实现商品介绍界面

布局

 activity\_info.xml

整体使用 LinearLayout 布局

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:orientation="vertical"  
    tools:context="com.example.linm.lab3.InfoActivity">
```

## 顶部使用 RelativeLayout

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:id="@+id/Top"
    android:layout_weight="1"
    android:background="#1E000000">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/img"
    />
    <Button
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:id="@+id/Back"
        android:layout_marginStart="15dp"
        android:layout_marginTop="10dp"
        android:background="@mipmap/back" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Name"
        android:textSize="25sp"
        android:text="hahahha"
        android:layout_marginTop="135dp"
        android:layout_alignLeft="@+id/Back"
        android:textColor="#000000" />

    <Button
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:id="@+id/star"
        android:layout_marginLeft="10dp"
        android:layout_marginStart="310dp"
        android:layout_alignBottom="@+id/Name"
        android:background="@mipmap/empty_star" />

</RelativeLayout>
```

往下界面使用 LinearLayout

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="2"
    android:orientation="vertical"
>
```

价格、类型、数据、购物车图标、竖线、横线、“更多产品信息”、分割线使用相对布局

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/price"
    android:layout_marginStart="15dp"
    android:layout_marginTop="10dp"
    android:textSize="20sp"
    android:textColor="#D5000000"
/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/info1"
    android:layout_alignStart="@+id/price"
    android:layout_below="@+id/price"
    android:textSize="15sp"
    android:layout_marginTop="8dp"
    android:textColor="#8A000000" />
```

<TextView

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/info1"
    android:layout_alignStart="@+id/price"
    android:layout_below="@+id/price"
    android:textSize="15sp"
    android:layout_marginTop="8dp"
    android:textColor="#8A000000" />
```

<TextView

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/info2"
    android:layout_alignBottom="@+id/info1"
    android:layout_marginStart="15dp"
    android:textSize="15sp"
    android:layout_toEndOf="@+id/info1"
    android:textColor="#8A000000" />
```

<Button

```
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:id="@+id/shopcar"
    android:layout_alignParentEnd="true"
    android:layout_alignTop="@+id/price"
    android:layout_marginEnd="15dp"
    android:layout_marginTop="8dp"
    android:background="@mipmap/shoplist" />
```

<View

```
    android:layout_width="2dp"
    android:layout_height="40dp"
    android:layout_alignTop="@+id/shopcar"
    android:layout_marginEnd="18dp"
    android:layout_toStartOf="@+id/shopcar"
    android:background="#1E000000" />
```

```

<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:id="@+id/line"
    android:layout_alignStart="@+id/info1"
    android:layout_below="@+id/info1"
    android:layout_marginTop="10dp"
    android:background="#1E000000" />

<ListView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/more"
    android:layout_below="@+id/line">
</ListView>

<View
    android:layout_width="match_parent"
    android:layout_height="18dp"
    android:layout_below="@+id/more"
    android:background="#1E000000" />

</RelativeLayout>

```

底部的使用 ListView 实现“一键下单”“分享商品”“不感兴趣”“查看更多商品促销信息”。

```

<ListView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/listview">

</ListView>

```

接着考虑 InfoActivity.java 文件

### Info

先在 AndroidManifest.xml 中加入该 activity

```

<activity android:name=". InfoActivity">
</activity>

```

建立从名字到物品类的映射

```
private Map<String, Info> mProductDetailsMap = new HashMap<>();
```

在 onCreate 函数中，将名字与物品类一一对应

```

for(int i=0; i<MainActivity.Infos.size(); i++) {
    mProductDetailsMap.put(MainActivity.Infos.get(i).getName(), MainActivity.Infos.get(i));
}

```



从传进来的 bundle 中获取名字，映射到 p

```
Bundle bundle = getIntent().getExtras();
String ProductName = bundle.getString("name");
final Info p = mProductDetailsMap.get(ProductName);
```

设置信息

```
//设置名字
TextView name = (TextView) findViewById(R.id.name);
name.setText(p.getName());
//设置价格
TextView price = (TextView) findViewById(R.id.price);
price.setText(p.getPrice());
//设置类型
TextView type = (TextView) findViewById(R.id.info1);
type.setText(p.getType());
//设置信息
TextView info = (TextView) findViewById(R.id.info2);
info.setText(p.getInfo());

//设置图片
ImageView img = (ImageView) findViewById(R.id.img);
if (p.getBackground().equals("1")) {
    img.setImageResource(R.drawable.enchantedforest);
}
else if (p.getBackground().equals("2")) {
    img.setImageResource(R.drawable.arla);
}
else if (p.getBackground().equals("3")) {
    img.setImageResource(R.drawable.devondale);
}
else if (p.getBackground().equals("4")) {
    img.setImageResource(R.drawable.kindle);
}
else if (p.getBackground().equals("5")) {
    img.setImageResource(R.drawable.waitrose);
}
else if (p.getBackground().equals("6")) {
    img.setImageResource(R.drawable.mcvitie);
}
```

```

else if (p.getBackground().equals("8")) {
    img.setImageResource(R.drawable.maltesers);
}
else if (p.getBackground().equals("9")) {
    img.setImageResource(R.drawable.lindt);
}
else if (p.getBackground().equals("10")) {
    img.setImageResource(R.drawable.borggreve);
}

```

实现返回图标的单击事件

```

//设置返回图标
Button back = (Button) findViewById(R.id.Back);
back.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});

```

使用 ListView 展示剩余信息

在 more.xml 中设置每一项的布局

```

<?xml version="1.0" encoding="utf-8"?>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/Final"
    android:textSize="18sp"
    android:textColor="#D5000000"
    android:padding="12dp"
    xmlns:android="http://schemas.android.com/apk/res/android" />

```

在 InfoActivity 中

```

String[] operations1 = new String[] {"更多产品信息"};
ArrayAdapter<String> arrayAdapter1 = new ArrayAdapter<>(this, R.layout.more, operations1);
ListView listView1 = (ListView) findViewById(R.id.more);
listView1.setAdapter(arrayAdapter1);

String[] operations2 = new String[] {"一键下单", "分享商品", "不感兴趣", "查看更多商品促销信息"};
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(this, R.layout.more, operations2);
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(arrayAdapter);

```

实现星星图标的切换

```
final Button star = (Button) findViewById(R.id.star);
star.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!tag) {
            star.setBackgroundResource(R.mipmap.full_star);
            tag = true;
        } else {
            star.setBackgroundResource(R.mipmap.empty_star);
            tag = false;
        }
    }
});
```

实现单击购物车图标，将商品加入到购物车队列中

```
Button shopcart = (Button) findViewById(R.id.shopcar);
shopcart.setOnClickListener((v) -> {
    Map<String, Object> temp = new LinkedHashMap<>();
    temp.put("FirstLetter", p.getFirstLetter());
    temp.put("name", p.getName());
    temp.put("price", p.getPrice());
    MainActivity.shoplist.add(temp);
    MainActivity.simpleListAdapter.notifyDataSetChanged();
    Toast.makeText(InfoActivity.this, "商品已添加到购物车", Toast.LENGTH_SHORT).show();
});
```

实现动画

首先添加依赖

```
compile 'jp.wasabeef:recyclerview-animators:2.2.7'
compile 'com.android.support:support-core-utils:25.4.0'
```

在 MainActivity 中，将

```
mRecyclerView.setAdapter(commonAdapter);
```

改为

```
ScaleInAnimationAdapter animationAdapter = new ScaleInAnimationAdapter(commonAdapter);
animationAdapter.setDuration(2000);
mRecyclerView.setAdapter(animationAdapter);
mRecyclerView.setItemAnimator(new OvershootInLeftAnimator());
```

即可实现进入商品列表时的动画显示。

去掉标题栏

在 values 文件目录下的 styles.xml 中，将 parent 部分修改为

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
```

### (3) 实验遇到的困难及解决思路

- 1) 用 RecyclerView 实现商品界面的时候，一开始显示正常，但滑动之后，每一项商品就占据了一整个屏幕。

解决思路：可能是布局出问题。将 RecyclerView 布局里的 `android:layout_height` 改为 `wrap_content` 即可，使其根据内容自适应，而不是根据整个屏幕（原先为 `match_parent`）。

- 2) 单击商品名称时无法实现页面跳转，闪退。

解决思路：在确保传参正确的前提下，闪退有可能是忘了在 `AndroidManifest.xml` 中注册跳转的 activity。可以手动在文件中添加：

```
<activity android:name=".InfoActivity">
</activity>
```

或者选择另一种新建 activity 的方法：File->New->Activity->Empty Activity，填好 java 文件名和 xml 文件名之后点击 Finish 创建，这样子创建的 activity 会自动在 `AndroidManifest.xml` 中注册。

- 3) 在实现商品介绍界面点击购物车图标，将商品添加到购物车这一个功能时，不需要用到界面跳转，因此就无法通过 `startactivity` 来传递数据，而购物车列表 `shoplist` 又是 `MainActivity` 中的一个内部成员变量，不能直接引用。

解决思路：最终目的是将商品加入到 `shoplist` 中，既然无法通过 activity 跳转传递数据，那就考虑把 `shoplist` 定义成一个静态全局变量，则在商品页面介绍的 activity 中可对 `MainActivity.shoplist` 进行 add 操作。

- 4) 在购物车中点击商品，弹出对应商品详细介绍，继续点击当前页面的购物车图标，返回购物车页面，没有增加新商品，且点击原先的购物车的商品时会闪退。只有先切换到商品列表再切换回购物车页面时，才显示正常且点击商品不会闪退。

解决思路：查看报错信息，显示列表已经更新，但是适配器并没有更新，于是导致程序崩溃。然后我意识到，在商品详情页面点击商品购物车图标的事件处理中，我只是将商品 add 到 `shoplist` 中去，并没有更新适配器，而对应的适配器又是定义在 `MainActivity` 内部。因此和上面的思路类似，将适配器 `simpleListAdapter` 定义为静态全局变量，更新 `shoplist` 的同时，更新适配器：

```
MainActivity.simpleListAdapter.notifyDataSetChanged();
```

- 5) RecyclerView 单击时正常，长按也正常，但是按得不久会闪退。

解决思路：Recyclerview 的适配器是自定义的，观察到一开始照着 ppt 打的 `boolean onLongClick` 的返回值为 `false`。试着改变一下返回值为 `true`，就不会出现这个问题了。后来了解到，返回值如果为 `true`，则表示事件已经成功处理，而如果返回为 `false`，则表示没有处理完该事件，并且希望其它方法对其进行处理。而我一开始的返回值为 `false`，并且没有定义其它解决方法，因此程序无法解决按得不久的事件处理，所以会闪退。

- 6) 在商品列表删除商品之后，后面商品的详情会乱序了。

解决思路：原先是根据触摸屏的位置 `position` 来获取存储的第 `position` 个

对象的，商品列表删除一个对象之后，商品类队列 Infos 并没有删除该对象，如果仍然按照位置来寻找的话，就会出错。所以改进思路是不要用位置来作为 activity 的数据传递，而改为传递名字，建立名字和商品类的映射，这样就不会乱了。

- 7) 商品详情页面中，商品的图片会填满整个顶部背景。  
解决思路：原先是设置 RelativeLayout 的背景，改成添加一个 ImageView，然后使用 setImageResource 的方法设置图片。
- 8) 要求商品详情界面中，顶部占三分之一。  
解决思路：将页面分成两大模块，设置顶部的 RelativeLayout 的 layout\_weight 为 1，底部的 LinearLayout 的 layout\_weight 为 2。

## 四、 实验思考及感想

下面贴上一些这次实验学习到的知识。

### RelativeLayout 相对布局常用属性

- a)、第一类:属性值为 true 或 false  
android:layout\_centerHorizontal 水平居中  
android:layout\_centerVertical 垂直居中  
android:layout\_centerInParent 相对于父元素完全居中  
android:layout\_alignParentBottom 贴紧父元素的下边缘  
android:layout\_alignParentLeft 贴紧父元素的左边缘  
android:layout\_alignParentRight 贴紧父元素的右边缘  
android:layout\_alignParentTop 贴紧父元素的上边缘
- b)、第二类: 属性值必须为 id 的引用名 “@id/id-name”  
android:layout\_below 在某元素的下方  
android:layout\_above 在某元素的上方  
android:layout\_toLeftOf 在某元素的左边  
android:layout\_toRightOf 在某元素的右边  
android:layout\_alignTop 本元素的上边缘和某元素的的上边缘对齐  
android:layout\_alignLeft 本元素的左边缘和某元素的的左边缘对齐  
android:layout\_alignBottom 本元素的下边缘和某元素的的下边缘对齐  
android:layout\_alignRight 本元素的右边缘和某元素的的右边缘对齐
- c)、第三类: 属性值为具体的像素值，如 30dip, 40px  
android:layout\_marginBottom 离某元素底边缘的距离  
android:layout\_marginLeft 离某元素左边缘的距离  
android:layout\_marginRight 离某元素右边缘的距离  
android:layout\_marginTop 离某元素上边缘的距离

RelativeLayout 比一般的 LinearLayout 的好处在于它可以设置元素之间的相对位置，比如左对齐、下边缘对齐等等，一般的约束只能设置水平中心线对齐。

## ✚ 两个 Activity 之间的数据传递

### a) 直接使用 intent

在 MainActivity 设置数据:

```
Intent intent = new Intent(MainActivity.this, Activity2.class);
intent.putExtra("name", "Alice");
intent.putExtra("age", "18");
startActivity(intent);
```

在 Activity2 接受数据:

```
Intent intent = getIntent();
String name = intent.getStringExtra("name");
int age = intent.getIntExtra("age", 0);
```

### b) 使用 bundle

在 MainActivity 设置数据:

```
Intent intent = new Intent(MainActivity.this, Activity2.class);
Bundle bundle = new Bundle();
bundle.putString("name", "Alice");
bundle.putInt("age", 18);
intent.putExtras(bundle);
startActivity(intent);
```

在 Activity2 接受数据:

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
String name = bundle.getString("name");
int age = bundle.getInt("age");
```

## ✚ 各种 map 的使用

map 用于存储键值对, 根据键得到值, 一个键只能对应一个值。LinkedHashMap 根据输入的顺序来存储对象, 位置与输入顺序对应。而 HashMap 的对象位置与输入顺序不一致, 一般按照升序来处理。

通常用法:

```
Map<Integer,String> map1 = new HashMap<Integer,String>();
map1.put(6, "apple");
map1.put(3, "banana");
map1.put(2, "pear");
String a = map1.get(6); // a = "apple";
```

## 实验感想

虽然之前没有学过 java 代码, 但是一些机理和 c++ 还是很类似的, 比如类的定义、数组定义、list 的用法等等, 只不过是函数调用有些许差别。简单的代码调用, 却能实现比 c++ 丰富的多的功能, 比如弹出 Toast 对话框、实现长按短按监听等等。虽然对函数的参数设置不是很了解, 不过网上还是挺多资料的, 看看例子也能够举一反三。实验难度跨度比较大, 一开始也是毫无头绪, 后来参考了众多例子, 慢慢 debug, 写完整个代码之后, 条理也渐渐清晰了, 也试着优化一些看起来冗余的细节, 使整体架构更加简洁。从这次实验中学到不少东西, 收获颇丰。