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）	移动互联网
学号	15352071	姓名	戴斯铭
电话	13727023544	Email	908660116@qq.com
开始日期	2017.10.20	完成日期	2017.10.22

一、实验题目

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

二、实验目的

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

三、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: 布局方面的要求:

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。圆圈中为名字的首字母，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1/3。每个商品的图片在商品数据中已给出,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。这一块建议大家使用RelativeLayout实现,以便熟悉RelativeLayout的使用。

4、商品详情界面中部

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。注意，价格那一栏的下边有一条分割线，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聊天符号的高度一致，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己定，argb编码值与前面的分割线一致。

5、商品详情页面底部

6、特别提醒，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。

逻辑方面的要求：
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。

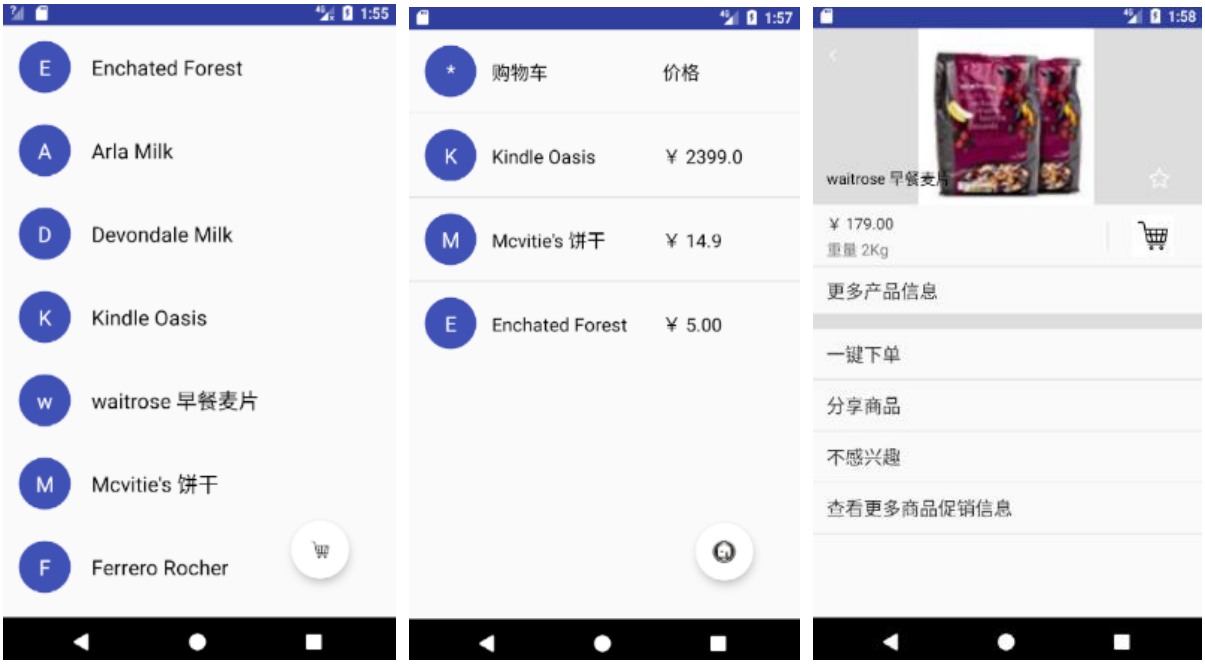
注意对话框中的商品名为被长按的商品。

4、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。未退出商品详细界面时,点击多次购物车图标可以只添加一件商品也可以添加多件到购物车中。

四、课堂实验结果

（1）布局方面，实现界面如下：

商品表界面　　购物车界面　　商品详情界面



实现过程：

①商品表界面：

实验要求说得很清楚，商品表界面需要使用 RecyclerView 来实现。由于实验文档讲得并不是特别详细，因此我选择在百度搜索，因此从搜索到的第一个博客中对 RecyclerView 有了一个十分详尽的了解。说到这里就有疑问，我们已经有了 ListView 和 GridView，为什么还要使用 RecyclerView 呢？那是因为整体上看 RecyclerView 架构，提供了一种插拔式的体验，高度的解耦，异常的灵活，通过设置它提供的不同 LayoutManager，ItemDecoration，ItemAnimator 实现令人瞠目的效果。（博客地址：http://blog.csdn.net/lmj623565791/article/details/45059587）

下面就开始实现使用 RecyclerView 来完成的商品表界面。首先，我们需要在布局中添加 RecyclerView，如下：

```
<android.support.v7.widget.RecyclerView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/mRecycleView">

</android.support.v7.widget.RecyclerView>
```

接下来，由于在 RecyclerView 需要自定义实现 RecyclerView.Adapter 并且为其提供数据集合，同时实现时也必须遵循 ViewHolder 设计模式。如实验文档所说，ViewHolder 通常出现在适配器中，是为了当 ListView，RecyclerView 在滚动时能够快速设置值，而不需要每次都创建对象。在本次实验中，我创建了一个叫 HomeAdapter 的类，并且将 Adapter 和 ViewHolder 的实现写在了一起。

首先，需要自定义一个 ViewHolder，如下图：

```
class MyViewHolder extends ViewHolder{
    TextView name;
    TextView first_letter;
    public MyViewHolder(View view){
        super(view);
        name = (TextView)view.findViewById(R.id.name);
        first_letter = (TextView)view.findViewById(R.id.first_letter);
    }
}
```

可以看到，类 MyViewHolder 中有两个 TextView，通过 findViewById 来创建 view 对象。接下来，需要获取 MyViewHolder 的实例，如下图：

```
@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
    MyViewHolder holder = new MyViewHolder(LayoutInflater.from(context).inflate(R.layout.home,parent,false));
    return holder;
}
```

图中使用了 LayoutInflater，我们在这里便是使用 LayoutInflater 来加载布局，主要就是为了在滚动时能够快速设置并且加载。其次，可以看到 inflate 方法中加载的 layout 名称为 home，这是我为了商品表界面而设置的一个布局。实现时，只需要完成 RecyclerView 中一个值的布局，之后只需要通过设置和加载就可以得到所有对象的布局了。下面就讲一个 home 是如何实现的。

首先，我们需要实现一个 TextView 来充当商品表的首字母圆圈，因此需要先设计一个蓝色圆圈形状，如下：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid
        android:color="#3F51B5"/>
    <size
        android:height="50dp"
        android:width="50dp"/>
</shape>
```

接下来，再在 home 布局中，将此圆形作为 TextView 的背景；同时，由于 TextView 中的文字需要居中，因此需要设置 gravity 属性为 center，其余属性则是将其在界面中的位置摆放好。实现如下：

```
<TextView
    android:id="@+id/first_letter"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginStart="20dp"
    android:layout_marginTop="15dp"
    android:layout_marginBottom="15dp"
    android:background="@drawable/shape"
    android:text="E"
    android:textColor="@color/white"
    android:gravity="center"
    android:textSize="20sp"/>
```

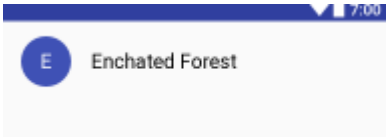
然后是实现商品名称，同样使用一个 TextView 来实现即可，直接上代码：

```
<TextView
    android:id="@+id/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginStart="20dp"
    android:layout_gravity="center_vertical"
    android:text="Enchated Forest"
    android:textSize="20sp"
    android:textColor="@color/black"
    />
```

由于商品名称需要竖直居中，因此需要设置 layout_gravity 属性为 center_vertical。其次，由于 RecyclerView 中的每一项都有固定宽高，因此需要在这里设置 LinearLayout 的属性，如下：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="80dp">
```

最终可得到效果如下：



之后的其他项只需要通过加载同样的布局便可以得到了。接下来继续实现 HomeAdapter 类的内容。自定义 Adapter 需要提供一个数据列表才能够填充数据，一般是 List 类型，以现在的商品为例，我需要创建一个商品类，则 List 数据列表中存储的便是商品类，再将这个 List 传入 HomeAdapter 即可作为数据填充。实现商品类如下：

```
public class ShoppingItem {
    private String commodity; //商品名称
    private String price;
    private String Info;

    public ShoppingItem(String commodity, String price, String Info){
        this.commodity = commodity;
        this.price = price;
        this.Info = Info;
    }
    public String getCommodity() { return commodity; }
    public String getPrice(){ return price;}
    public String getInfo(){ return Info;}
}
```

因此，完成 HomeAdapter 构造函数的内容：

```
public HomeAdapter(List<ShoppingItem> mDatas, Context context){
    this.mDatas = mDatas;
    this.context = context;
}
```

这里的 mDatas 便是数据列表，而 context 是指上下文，代表访问整个 Android 应用的接口。实际上，如实验文档所说的，Adapter 扮演着两个角色，一个是根据不同的 ViewType 创建与之相应的 Item-layout，二是访问数据集合并且将数据绑定到正确的 view 上。因此我们需要重写两个函数，需要注意的是，onCreateViewHolder 已在上面贴出，作为获取 MyViewHolder 的实例对象。接下来是重写 onBindViewHolder 函数。

```
@Override
public void onBindViewHolder(final MyViewHolder holder, final int position){
    holder.name.setText(mDatas.get(position).getCommodity());
    holder.first_letter.setText(mDatas.get(position).getCommodity().substring(0,1));
```

同时，我们还需要重写另一个方法，告知 RecyclerView 列表 Items 的总数：

```
@Override
public int getItemCount() { return mDatas.size(); }
```

至此，RecyclerView 的基本布局已经实现了，在 java 文件中调用 setLayoutManager()以及 setAdapter()方法即可使用。目前缺少相应的逻辑功能，如长按或点击等，这部分会在之后的逻辑方面进行介绍。

②购物车界面

购物车界面需要使用 ListView 进行实现。如实验文档所说，尽管 ListView 有常用的 Adapter 来填充数据，比如 SimpleAdapter,ArrayAdapter，但这两个的功能十分有限。因此我使用的是自定义

Adapter 来作为 ListView 的 Adapter。Adapter 需要自己提供数据列表，这个部分在 RecyclerView 实现中已提过，不再重复。当然，需要在布局中添加 ListView。

在我的实现中，购物车 ListView 的 Adapter 命名为 ShopAdapter。自定义的 Adapter 需要继承 BaseAdapter，共计有四个方法是需要重写的，可从实验文档中得到相应的介绍：

int getCount();获得数据项列表的长度，也就是一共有多少个数据项。

Object getItem(int i);获得某一个数据项。

long getItemId(int i);获得数据项的位置。

View getView(int i, View view, ViewGroup viewGroup);获得数据项的布局样式，最重要的一个方法。

前三个方法的实现很简单，直接截图：

```
public class ShopAdapter extends BaseAdapter {
    private Context context;
    private List<ShoppingItem> mDatas;

    public ShopAdapter(Context context, List<ShoppingItem> mDatas){
        this.context = context;
        this.mDatas = mDatas;
    }
    @Override
    public int getCount(){
        if(mDatas == null){
            return 0;
        }
        return mDatas.size();
    }
    @Override
    public Object getItem(int i){
        if(mDatas == null){
            return 0;
        }
        return mDatas.get(i);
    }
    @Override
    public long getItemId(int i) { return i; }
```

接下来重点实现 getView 函数。getView 一共有三个参数，其中 i 指的是当前在加载第 i 项的列表项；viewGroup 是列表项 View 的父视图，用于调整列表项的宽高；而 view 指的是一个列表项的视图。实验文档中，一共介绍了两种 getView 的写法，在这里当然是选择对应用运行最为有效的写法。

在该函数中，首先需要声明一个 View 变量和 ViewHolder 变量。然后，在当 view 为空时才加载布局，并且创建一个 ViewHolder，获得布局中的两个控件；否则，让 convertView 等于 view，然后从中取出 ViewHolder 即可。如下：

```
public View getView(int i, View view, ViewGroup viewGroup){
    View convertView;
    ViewHolder viewHolder; //新声明一个View变量和ViewHolder变量
    //当view为空时才加载布局，并且创建一个ViewHolder，获得布局中的两个控件。
    if(view == null){
        //通过inflate方法加载布局，context这个参数需要使用adapter的Activity传入
        convertView = LayoutInflater.from(context).inflate(R.layout.shoplist,null);
        viewHolder = new ViewHolder();
        viewHolder.name = (TextView)convertView.findViewById(R.id.name);
        viewHolder.first_letter = (TextView)convertView.findViewById(R.id.first_letter);
        viewHolder.price = (TextView)convertView.findViewById(R.id.price);
        convertView.setTag(viewHolder);//用setTag方法将处理好的viewHolder放入view中
    } else{ //否则，让convertView等于view，然后从中取出ViewHolder即可。
        convertView = view;
        viewHolder = (ViewHolder)convertView.getTag();
    }
```

接着再从 ViewHolder 中取出对应的对象，进行赋值，再将处理好的 convertView 返回即可：

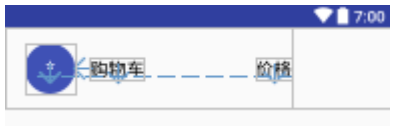
```
viewHolder.name.setText(mDatas.get(i).getCommodity());
if(i == 0){
    viewHolder.first_letter.setText("*");
}else{
    viewHolder.first_letter.setText(mDatas.get(i).getCommodity().substring(0,1));
}
viewHolder.price.setText(mDatas.get(i).getPrice());

return convertView;
```

当然，实现上述代码的前提是还需要实现 ViewHolder 类，如下：

```
private class ViewHolder{
    public TextView name;
    public TextView first_letter;
    public TextView price;
}
```

当然，ListView 也和 RecyclerView 类似，需要先完成列表其中一项的布局，在 getView 函数中可以看到，我将这个 layout 命名为 shoplist，由于实现比较简单，因此不贴出 xml 文件。最终可以得到以下基本项的布局：



在 MainActivity 中调用 setAdapter 即可使用。同理，设置点击事件会在之后的部分进行讲解。

③商品详情界面

· 商品详情的界面布局需要使用 RelativeLayout 来实现，这部分与其他布局的实现形式基本一致。首先从网上可以查询到各种属性，而我主要使用如下属性：

当然，可能根据兄弟组件定位会更为方便，但我在这次实现中仅使用根据父容器来定位，从而能更好的统一布局。整个界面我共分为三个部分的 RelativeLayout 来实现。

首先是实现详情界面的最上面部分，具体如下：

可以看到，这部分一共分为 5 个部分：总体背景色为#1E000000，此部分通过设置 RelativeLayout 的背景色即可实现：

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="170dp"
    android:background="#1E000000">
```

接下来是实现四个控件，分别是两个 ImageButton，一个 ImageView 和一个 TextView。这部分的布局只要通过 RelativeLayout 提供的属性即可进行设置，因此不再贴出 xml 代码。

接下来是商品详情界面的中间部分：

这个界面一共有五个部分，分别是一个 ImageButton，两个 TextView 和两条颜色为#1E000000 的横线及竖线。前三个控件的实现很简单，只要注意将位置摆放好即可。下面讲一下两条线如何实现。在我的实现中，都是使用 ImageView 来完成。直接通过设置宽度和高度，再添加背景色即可完成。以横线为例，实现如下：

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:background="#1E000000"
    android:layout_alignParentTop="true"
    android:layout_marginTop="59dp" />
```

最后实现商品详情界面的最后一部分，同样是放在 RelativeLayout 中实现。

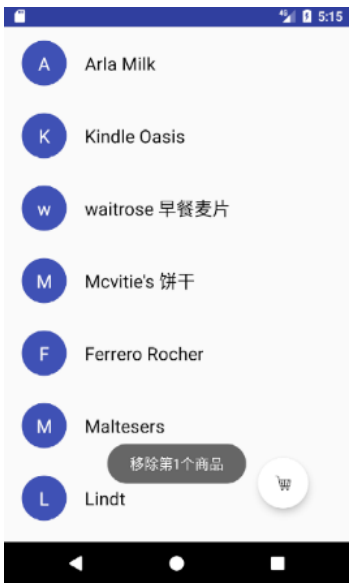
这个部分共分为三个部分，一个 TextView，一条颜色为#1E000000 的粗横线，以及一个 ListView 来实现下方的四个部分。前面两个控件的实现很简单，同样只需要注意位置的定位即可。需要注意的是 ListView。从图中可以看到，这次要实现的 ListView 比较简单，列表项内容仅有文字而没有其他的，因此一开始，我打算使用 ArrayAdapter 来实现。

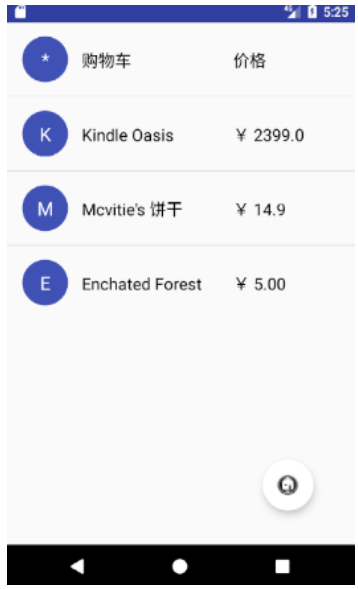
但后面发现，使用 ArrayAdapter 有很大的局限性。首先，我们可以提供的布局里只能有一个 TextView，而连一个 LinearLayout 也不可以有，这说明我不能够调整列表项的宽高；虽然 android 自己有提供一些列表项的布局，但是使用后发现，列表项的高度不适合。之后想使用 SimpleAdapter，发现实现起来很简单，能够实现比 ArrayAdapter 更多的功能。但为了之后的可重用性，我仍然使用自定义的 Adapter，命名为 DetailAdapter。由于实现与之前基本相同，因此不再进行解释。

至此，布局基本实现完毕，接下来介绍逻辑功能方面的实现。

（2）逻辑方面，实现结果如下：

①商品界面逻辑：





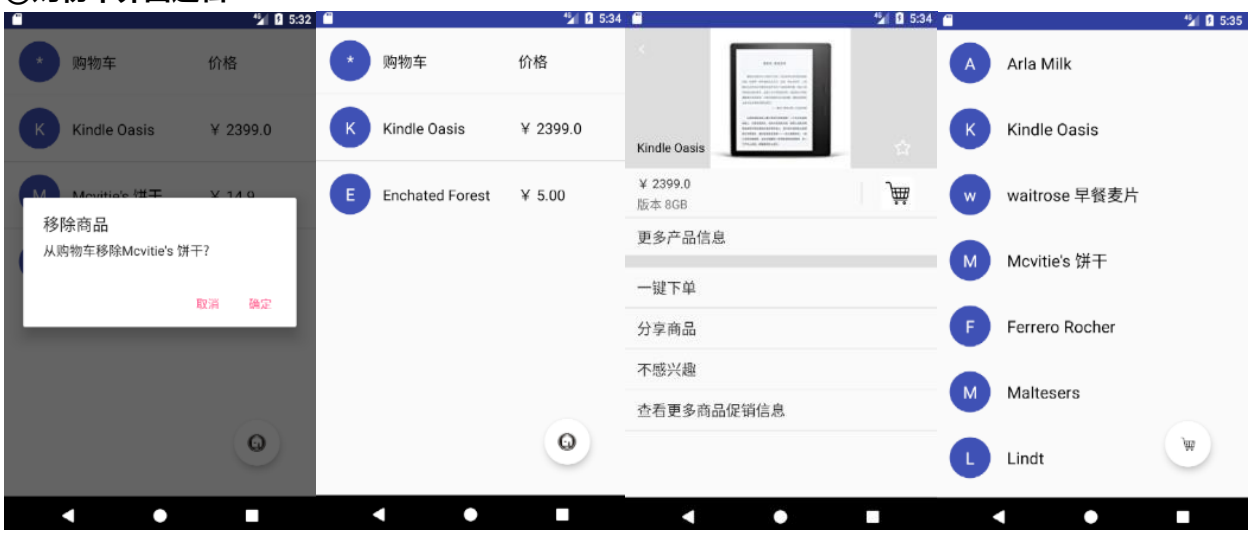
上述三张图分别表示实现了商品界面的三个功能：

长按列表项后删除该列表项，并且弹出 Toast 信息；

直接点击列表项，进入对应商品的详情界面；

点击右下方的购物车悬浮按钮，进入购物车。

②购物车界面逻辑：



上述四张图分别表示实现了购物车界面的三个功能：

长按列表项后显示对话框，点击确定后该列表项被移除，如图二，点击取消后对话框消失；

直接点击列表项，进入对应商品的详情界面；

点击右下方商品列表悬浮按钮，回到商品列表界面。

③商品详情界面

商品界面需要实现的功能，通过图片很难展示出来，这里仅提供文字表述：

点击左上方 back 按钮，回到商品列表或者购物车界面；

点击右侧星星按钮，如果原为空心，则变为实心，否则变为空心；

点击购物车按钮，购物车中会增加该商品。

具体实现：

由于界面之间具有相通性，因此是根据具体功能来进行接下来的解读。

首先先来实现商品界面右下方的浮动按钮。根据实验文档添加依赖后，在主界面中添加 FloatingActionButton。需要注意的两个属性是 backgroundTint 和 backgroundTintMode。前一个属性是指浮动按钮上的背景，而后一个属性则是按钮模式，选择 src_atop 则说明将提供给按钮的 src 内容放在顶层，而 backgroundTint 的内容放在后面。

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@mipmap/shoplist"
    app:backgroundTint="@color/white"
    app:backgroundTintMode="src_atop"
    app:rippleColor="@color/white"
    android:layout_marginRight="8dp"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginLeft="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="8dp"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginBottom="8dp"
    app:layout_constraintHorizontal_bias="0.878"
    app:layout_constraintVertical_bias="0.943" />
```

接着，设置这个 Button 的点击事件。由于这个按钮位于主界面中，因此点击事件在 MainActivity 的 java 文件中进行实现。要实现商品列表和购物车的切换，可以通过设置属性 Visibility 来实现。位于商品界面时，购物车的 ListView 为不可见；位于购物车界面时，商品界面的 RecyclerView 不可见。点击按钮时，通过获取当前两个 view 的状态，进行判断并且设置 visibility 属性即可。实现如下：

```
change = (FloatingActionButton) findViewById(R.id.fab);
change.setOnClickListener((view) -> {
        if(mRcyclerView.getVisibility() == View.VISIBLE &&
                shopListView.getVisibility() == View.INVISIBLE){
            mRcyclerView.setVisibility(View.INVISIBLE);
            shopListView.setVisibility(View.VISIBLE);
            change.setImageResource(R.mipmap.mainpage);
        }
        else if(mRcyclerView.getVisibility() == View.INVISIBLE &&
                shopListView.getVisibility() == View.VISIBLE){
            mRcyclerView.setVisibility(View.VISIBLE);
            shopListView.setVisibility(View.INVISIBLE);
            change.setImageResource(R.mipmap.shoplist);
        }
});
```

除了有切换界面的功能，浮动按钮本身的图标也需要改变，这里使用 setImageResource 的方法即可实现。

接下来是实现长按商品列表项后实现删除以及弹出相应的 toast 信息。在具体实现前，由于 recyclerView 比较特殊，没有设置 OnItemClickListener 的方法，因此需要自己在对应的 Adapter 中实现。首先添加接口以及函数：

```
public interface OnItemClickListener{
    void onClick(int position);
    void onLongClick(int position);
}
public void setOnItemClickListener(OnItemClickListener mOnItemClickListener){
    this.mOnItemClickListener = mOnItemClickListener;
}
```

接下来需要在 onBindViewHolder()中添加：

```
if(mOnItemClickListener!=null){
    holder.itemView.setOnClickListener((v) → {
            mOnItemClickListener.onClick(holder.getAdapterPosition());
    });
    holder.itemView.setOnLongClickListener((v) → {
            mOnItemClickListener.onLongClick(holder.getAdapterPosition());
            //removeData(position);
            return false;
    });
}
```

通过在设置，可以实现列表项与点击事件的绑定。完成上述步骤后，接下来可以在 MainActivity 文件中进行两个点击事件的具体实现了。下面就开始具体点击事件的实现。

首先需要对 RecyclerView 进行最基本的声明以及初始化，并且调用 Adapter。

```
mRcyclerView = (RecyclerView)findViewById(R.id.mRecycleView);
mRcyclerView.setLayoutManager(new LinearLayoutManager(this));
mRcyclerView.setAdapter(homeAdapter = new HomeAdapter(mDatas, this));
```

接下来先讲讲如何实现具体的功能，首先是长按事件：

```
@Override
public void onLongClick(int position){
    int i = position + 1;
    Toast.makeText(MainActivity.this, "移除第"+i+"个商品", Toast.LENGTH_SHORT).show();
    homeAdapter.removeData(position);
}
```

其中可以看到一个 removeData 函数，这是我在 HomeAdapter 中实现的用于删除列表项的函数，其具体内容如下：

```
public void removeData(int position)
{
    mDatas.remove(position);
    notifyItemRemoved(position);
}
```

接下来是实现点击后跳转到商品详情界面。这个部分的实现也是这次实验的重点。我们需要使用 intent，bundle 来实现 activity 之间的跳转以及消息的传递。这个部分的完成我是参考理论课上所讲的知识来实现的。

首先，当然是需要创建一个新的 Activity 文件，我将其取名为 DetailActivity。在 DetailActivity 中初始化需要的各个控件，即将之前实现的详情界面的 xml 文件中的控件添加进去。特别是需要有数据修改的比如图片、商品名称、以及价格、信息这四个部分。ListView 部分直接在 java 文件中调用即可，不再贴出。

接着是使用 Intent 来实现界面跳转以及信息传递。如下：

```
@Override
public void onClick(int position){
    Intent intent = new Intent(MainActivity.this, DetailActivity.class);
    intent.putExtra("name", mDatas.get(position).getCommodity());
    intent.putExtra("price",mDatas.get(position).getPrice());
    intent.putExtra("Info", mDatas.get(position).getInfo());
    startActivityForResult(intent, 0);
}
```

可以看到，新建一个 intent 对象，建立 MainAcitivity 和 DetailAcitivity 之间的联系。随后使用 putExtra 方法将需要传递的数据放进去。接着需要调用 startActivityForResult 来进行 intent 的传递，括号中的第二位是指 requestCode，即请求码，用来标记数据是由 MainActivity 传递的。对应的，在 DetailActivity 里要接受消息，如下：

```
Intent intent = getIntent();
Bundle extras = intent.getExtras();
//接受从MainActivity传来的信息
String name = null;
String price = null;
String info = null;
if(extras != null){
    name = extras.getString("name");
    price = extras.getString("price");
    info = extras.getString("Info");
}
goodname.setText(name);
goodprice.setText(price);
goodInfo.setText(info);
```

由上可知，接受到的消息可用于设置商品详情页面中的控件的具体信息。此外，由于页面中需要有不同商品的图片，因此我使用 switch(name)case 语句来确定使用哪张图片，部分代码如下：

```
switch (name){
    case "Enchated Forest":
        pic.setImageResource(R.drawable.enchartedforest);
        break;
    case "Arla Milk":
        pic.setImageResource(R.drawable.arla);
        break;
    case "Devondale Milk":
        pic.setImageResource(R.drawable.devondale);
        break;
```

同样的，在商品详情界面左上角有一个 back 按钮，需要在点击其之后回到原来的界面，这个实现很简单，只要调用 finish 方法即可，实现如下：（位于 DetailActivity.java）

```
back = (ImageButton)findViewById(R.id.back);
back.setOnClickListener((v) -> { finish(); });
```

接下来，需要实现购物车方面的功能。长按列表项，会弹出对话框询问是否删除商品，这个地方的实现也很简单，直接贴出代码：

```
shopListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener(){
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int position, long id){
        if(position==0) return true;
        ShoppingItem p = (ShoppingItem) parent.getItemAtPosition(position);
        String name = "从购物车移除" + p.getCommodity();
        final AlertDialog.Builder alertDialog = new AlertDialog.Builder(MainActivity.this);
        alertDialog.setTitle("移除商品").setMessage(name+"?").setNegativeButton("取消",
                (dialogInterface, i) → {
                        try{
                            Field field = dialogInterface.getClass().getSuperclass().getDeclaredFi
                            field.setAccessible(true);
                            field.set(dialogInterface, true);
                        } catch (Exception e){
                            e.printStackTrace();
                        }
                }).setPositiveButton("确定",
                (dialogInterface, i) → {
                        shopAdapter.removeData(position);
                }).show();
        //alertDialog.show();
        return true;
    }
});
```

当然，有个需要注意的地方是，一般而言，setOnItemLongClick 一般返回 false，但这样会导致一个问题，那就是如果先触发的是长按事件而 return false 的话，代表当前事件需要继续向下传递，就会导致短按事件就被响应了。因此在这里需要修改为 return true。接下来实现点击事件，与商品界面相同，需要将商品的信息传递到 DetailActivity 中，实现如下：

```
public void onItemClick(AdapterView<?> parent, View view, final int position, long id){
    if(position == 0);
    else{
        Intent intent = new Intent(MainActivity.this, DetailActivity.class);
        intent.putExtra("name", shopDatas.get(position).getCommodity());
        intent.putExtra("price",shopDatas.get(position).getPrice());
        intent.putExtra("Info", shopDatas.get(position).getInfo());
        startActivityForResult(intent, 0);
    }
}
```

至此，商品界面以及购物车界面的功能基本实现了。下面再接着介绍商品详情界面的功能实现。第一个是 back 按钮，在上面已经说过，不再重复。接着是星星图标的切换事件。这个部分的实现可以使用设置 tag 来完成，每个 View 都可以设置 tag，通过 tag 可以用来判断 View 的状态。我将星星空心时设置,tag 为 0，实心时为 1，通过获取 tag 值来判断当前星星状态，从而可知需要切换哪个图标。实现如下：

```
star = (ImageButton)findViewById(R.id.star);
star.setTag("0");
star.setOnClickListener((v) → {
        Object tag = star.getTag();
        if(tag == "0"){
            star.setImageResource(R.mipmap.full_star);
            star.setTag("1");
        }else {
            star.setImageResource(R.mipmap.empty_star);
            star.setTag("0");
        }
});
```

最后是购物车图标的点击，通过点击该图标，可以将当前商品添加到购物车界面中。从这里可以知道，我们需要返回信息给 MainActivity。当然，首先需要在 DetailActivity 中设置按钮的点击事件，将消息传递给主 Activity。具体如下：

```
final Intent intent1 = new Intent();
setResult(0,intent1);
back = (ImageButton)findViewById(R.id.back);
back.setOnClickListener((v) → { finish(); });

car = (ImageButton) findViewById(R.id.car);
car.setOnClickListener((v) → {
        Toast.makeText(getApplicationContext(), "商品已添加到购物车", Toast.LENGTH_SHORT).show();
        intent1.putExtra("name", goodname.getText());
        intent1.putExtra("price",goodprice.getText());
        intent1.putExtra("Info", goodInfo.getText());
        setResult(1, intent1);
});
```

可以看到，图中使用 setResult 来进行消息传递，其设置的 resultCode 共有两个，分别为 0 和 1。为 0 时，此时不进行消息的传递；为 1 时，会将商品的详细信息传回给 MainActivity。既然 MainActivity 需要接收消息，则其需要实现一个 onActivityResult 方法。具体如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent){
    if(requestCode == 0 && resultCode == 1){
    //添加到购物车
        Bundle extras = intent.getExtras();
        String name = extras.getString("name");
        String price = extras.getString("price");
        String info = extras.getString("Info");
        ShoppingItem shop = new ShoppingItem(name, price, info);
        shopDatas.add(shop);
        shopAdapter.notifyDataSetChanged();
    }
}
```

通过设置此方法，得到需要的商品信息后，再将其包装成一个 ShoppingItem 类，从而可以添加到购物车的数据列表中，进而再调用 notifyDataSetChanged 来通知 Activity 更新 ListView。

至此，所有的基本功能实现完毕。

五、课后实验结果

在本次实验中，删除列表项时我使用了 notifyDataSetChanged()方法，这里会使得列表会使用默认的删除动画功能。想要试用新的动画效果，但是发现用了之后会出现闪退的情况。因此只能作罢。只能争取下次来实现啦，得抓紧写其他作业了。

六、实验思考及感想

这次实验和以往实验相比，突然觉得以前的实验简直是小儿科~ 这次需要实现的功能比较多，所以感觉做得很吃力。再加上实验文档解释得也不是特别明白，因此实际上都是各种百度看资料看解释。同时，也少不了和同学的沟通交流。

当然，这次实验也大大提高了我的自学能力。这对我们是有很大帮助的。