中山大学移动信息工程学院本科生实验报告

课程名称：移动应用开发　　任课教师：郑贵锋

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、 实验内容

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：

1. 图1界面为应用启动后看到的第一个界面(数据在素材中有给出)
2. 点击右下方的悬浮按钮可以切换到购车如图 2
3. 上面两个列表点击任意一项后，可以看到详细的信息如图 3

图 1

图 2

图 3

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. 商品详清界面顶部

   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 Relative Layout 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以条分割线， 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

   这个没什么说的，内容和样式都很清楚。

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。(此处为了附加功能魔改)
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。

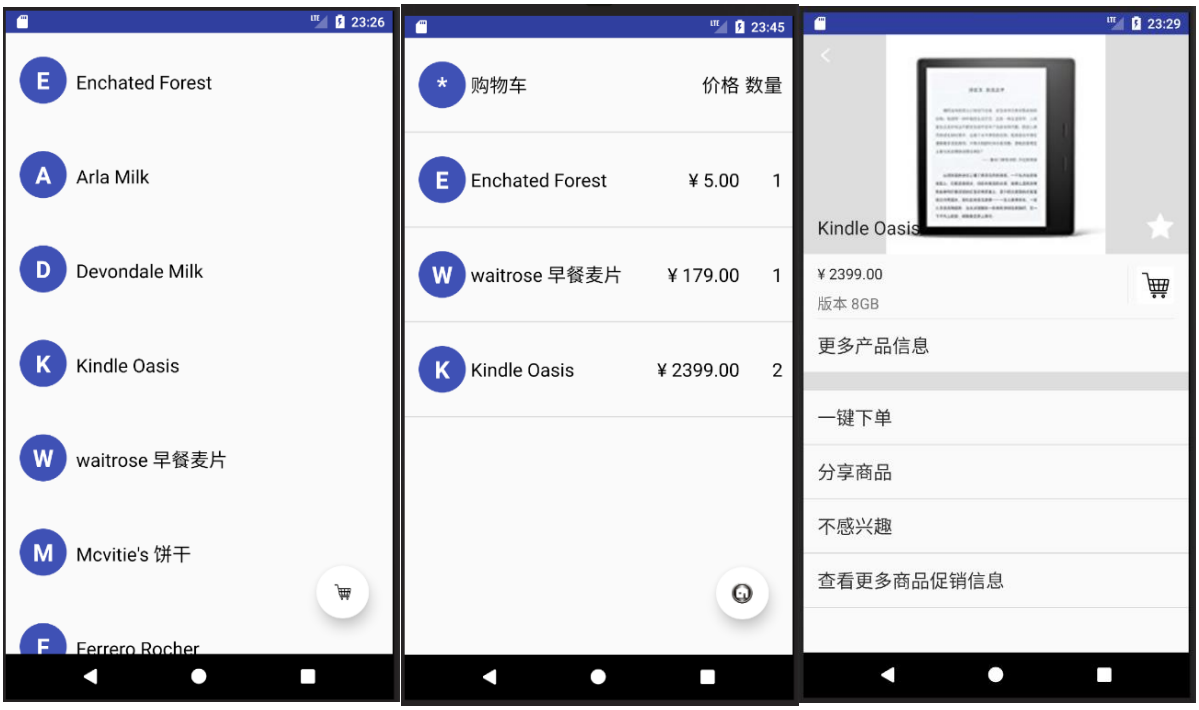
4. 商品详清界面中点击返回圈标会返回上—层，点击星标会切换状态，如果原先是空心星星，则会变成实心星星；如果原先是实心星星，则会变成空心星星。点击购物车圈标会将该商品添加到购物车中并弹出 Toast 提示“商品已添加到购物车”。

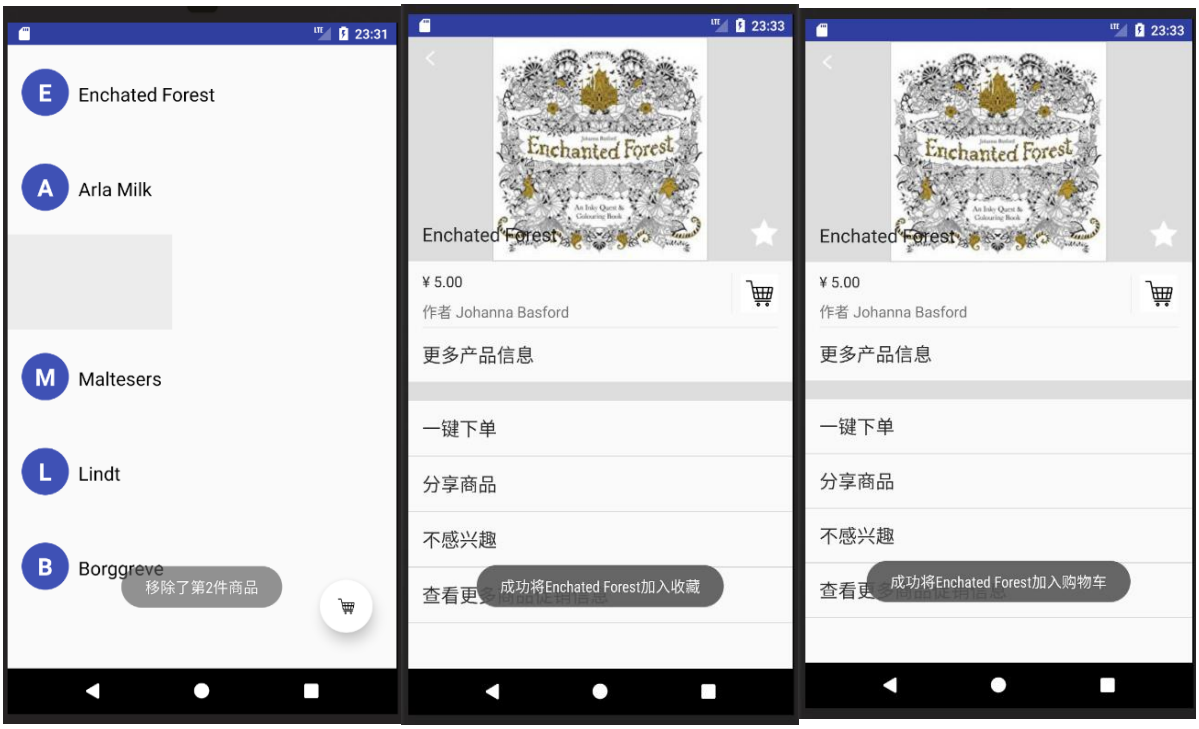
   注不要求判断购物车是否已有该商品即如果已有一件该商品添加之后则显示两个即可。未退出商品详细界面时点击多次购物车图标可以只添加一件商品也可以添加多件到购物车中(此处再次魔改)

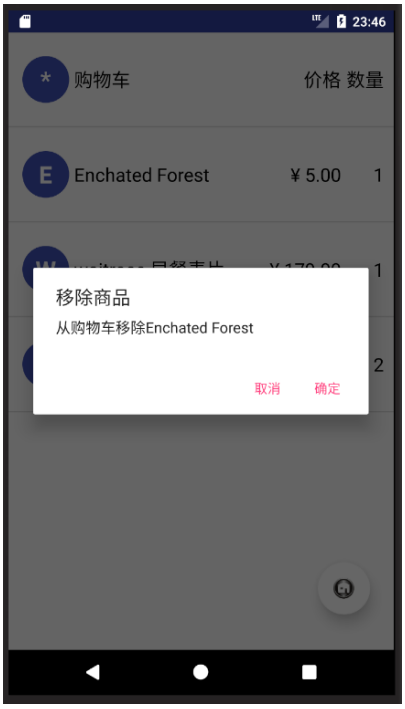
四、 课堂实验结果

（一） 实验截图：

1. 虚拟机上效果如下：





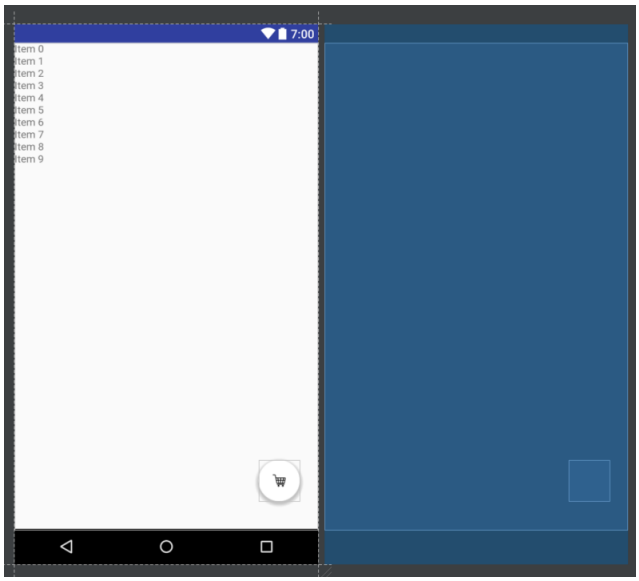


（二） 关键步骤：

实验中修改了一下使用了 3 个界面相互转跳用，此处所有数据存储在 Good 商品类中，由 Data 类进行管理，这两个类的实现见课后部分 Good 实现了界面中传参数。

1. 首先是主界面

   i. 首先是主界面问题，主界面是商品列表界面，需要使用 RecyclerView 实现，此外还需要一个 FloatingActionButton。此处使用在 Design 下拖入控件，此时 Android studio 会自动添加需要的 build，以下是主界面布局如下：



此处需要的这两个控件都需要在 build.gradle 添加依赖，不过 Android studio 会在你引入控件的时候自动添加依赖：

```
compile 'com.android.support:recyclerview-v7:26.+'
compile 'com.android.support:design:26.+'
```

这样就创建了 RecyclerView，然后因为在 java 文件中再填充数据，所以 id 是一定要设的。在 java 文件中获得这个 RecyclerView 之后，使用 Adapter 为这个 ListView 填充数据，此处因为 RecyclerView 所以我们需要自定义一个 Adapter。

ii. RecyclerView 要求必须使用 ViewHolder 模式，一般我们在使用过程中，都需要去建立一个新的 ViewHolder 然后作为泛型传入 Adapter。那么想要建立通用的 Adapter，必须有个通用的 ViewHolder。（此处定义在我们自定义的 Adapter 中以防止冲突到之后的 ListView）

```
public static class ViewHolder extends RecyclerView.ViewHolder {
    private View mConvertView;//存储 list_Irem
    private SparseArray<View>mView;//存储 list_Item 的子 View
    public ViewHolder(Context context,View view,ViewGroup viewGroup){
        super(view);
        mConvertView=view;
        mView = new SparseArray<View>();
    }
    //获取 viewHolder 实例
    public  static ViewHolder get (Context context,ViewGroup viewGroup,int layoutId){
        View itemView = LayoutInflater.from(context).inflate(layoutId,viewGroup,false);
        ViewHolder viewHolder = new ViewHolder(context,itemView,viewGroup);
        return viewHolder;
    }
    //viewHolder 尚未将子 View 缓存到 SparseArray 数组中时仍然需要通过 findViewByld()创建
View 对象如果已缓存直接返回
    public <T extends View>T getView(int viewId){
        View view=mView.get(viewId);
        if(view == null){
            view=mConvertView.findViewById(viewId);
            mView.put(viewId,view);
        }
        return (T)view;
    }
}
```

有了 Adapter 用的 ViewHolder 之后，Adapter 也可以开始写。

iii. 我们的每次使用过程中，针对的数据类型肯定是不同的，那么这里最好是要引入泛型代表， 但是因为此处为了简化一下直接实例化，为我们之后需要用到的一个数据类，此时也省的重载。

Adapter 扮演着两个角色。一是，根据不同 ViewType 创建与之相应的的 Itemlayout，二是，访问数据集合并将数据绑定到正确的 View 上。这就需要我们重写以下两个函数:

public ViewHolder onCreateViewHolder(final ViewGroup viewGroup,int viewType)

public void onBindViewHolder(final ViewHolder viewHolder, int position)

另外还需要重写另一个方法，像 ListView-Adapter 那样，同样地告诉 RecyclerView-Adapter 列表 Items 的总数这就需要 public int getItemCount()

```
public class CommonAdapter extends RecyclerView.Adapter<CommonAdapter.ViewHolder> {
    private Context mContext;
    private List<Good> mDatas;
    private int mLayoutId;
    //构造函数
    public CommonAdapter(Context mContext,int mLayoutId,List<Good>mDatas) {
        this.mLayoutId=mLayoutId;
```

```
        this.mContext = mContext;
        this.mDatas = mDatas;
    }
    @Override
    public ViewHolder onCreateViewHolder(final ViewGroup viewGroup,int viewType) {
        ViewHolder viewHolder=ViewHolder.get(mContext,viewGroup,mLayoutId);
        return  viewHolder;
    }
    //将数据与界面进行绑定的操作
    @Override
    public void onBindViewHolder(final ViewHolder viewHolder, int position) {
        convert(viewHolder,mDatas.get(position));
        if(mOnItemClickListener !=null){
            viewHolder.itemView.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View view){
                    mOnItemClickListener.onClick(viewHolder.getAdapterPosition());
                }
            });
            viewHolder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View view) {
                    mOnItemClickListener.onLongClick(viewHolder.getAdapterPosition());
                    return false;
                }
            });
        }
    }
    //获取数据的数量
    @Override
    public int getItemCount() {
        return mDatas.size();
    }
}
```

此处我们需要重写适配器中的 convert 方法进行数据绑定，此处因为我们直接实现了专用的 Adapter 所以此时可以直接在 Adapter 类中写上这个方法。

```
public void convert(ViewHolder viewHolder, Good good){
    TextView first=viewHolder.getView(R.id.first);
    first.setText(good.getGoodName_First());
    TextView name=viewHolder.getView(R.id.name);
    name.setText(good.getGoodName());
}
```

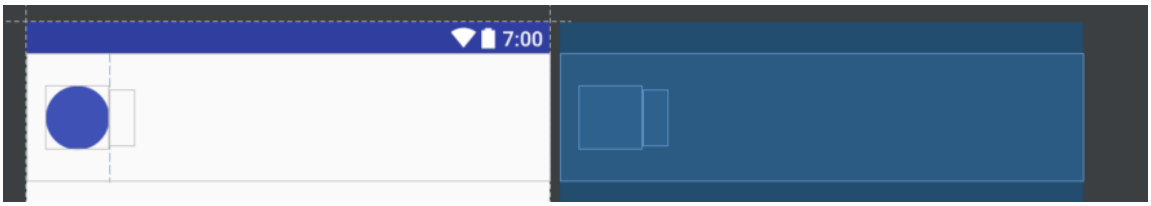
RecyclerView 没有 On ltemClicklistener 方法需要在适配器中实现。实现的方法为：在 Adapter 中设置一个监听器当 itemView 被点击时候调用该监听器并且将 itemView 的 position 作为参数传递出去。

```
//注册点击事件
public interface OnItemClickListener{
    void onClick(int position);
    void onLongClick(int position);
```

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

此处对应的点击实现包括 public void onBindViewHolder (final ViewHolder viewHolder, int position)中的 onClick 和 onLongClick。

iv. 此时我们可以在 main 中创建 RecyclerView，我们先要创建一个用来代表每一项的 item 的 xml 如下，此处的圆圈+首字母使用 TextView+自定义 Background 实现（和上次实验的 button background 基本一致不在展示）




```
<?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:background="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:id="@+id/items"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingTop="25dp"
    android:paddingBottom="25dp"
    android:paddingLeft="15dp"
    android:paddingRight="15dp"
    >
    <TextView
        android:id="@+id/first"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/buttonsytle"
        android:textAlignment="center"
        android:textColor="@android:color/white"
        android:textSize="25sp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:textColor="@android:color/black"
        android:textSize="20sp" />
</LinearLayout>
```

此时我们可以开始创建 RecyclerView 并进行动画设置

```
mRecyclerView = (RecyclerView)findViewById(R.id.myRecyclerView); //初始化 RecyclerView

mRecyclerView.setLayoutManager(new LinearLayoutManager(this));//这里用线性显示
mRecyclerView.setHasFixedSize(true);
//初始化适配
//mRecyclerView.setAdapter(commonAdapter);
commonAdapter = new CommonAdapter(MainActivity.this,R.layout.items,data.getGood_list());
animationAdapter=new ScaleInAnimationAdapter(commonAdapter);

animationAdapter.setDuration(1000); //设置开始动画
mRecyclerView.setAdapter(animationAdapter);//填充数据
mRecyclerView.setItemAnimator(new OvershootInLeftAnimator());
mRecyclerView.getItemAnimator().setRemoveDuration(300); //设置移除延时
```

到此为止我们主界面的 RecyclerView 创建和准备工作完成。

然后我们需要为 RecyclerView 创建对应的点击事件打开新的界面和删除，此处打开新的界面传入两个类数据都是基于商品类实现的商品列表和购物车列表：

```
commonAdapter.setOnItemClickListener(new CommonAdapter.OnItemClickListener(){
    @Override
    public void onClick(int position){
        if(position >= 0) {
            Intent i = new Intent(MainActivity.this, Good_Info.class);
            Bundle bundle = new Bundle();
            bundle.putInt("position", position);
            bundle.putParcelableArrayList("data", data.getGood_list());
            bundle.putParcelableArrayList("cart", data.getCart_list());
            i.putExtra("mainActivity", bundle);
            startActivityForResult(i, 0);
        }
    }
    @Override
    public void onLongClick(int position) {
        if (position >= 0) {
            data.removeGood_list_index(position);
            animationAdapter.notifyItemRemoved(position);
            mRecyclerView.setItemViewCacheSize(data.getGood_list().size());
            Toast.makeText(MainActivity.this, "移除了第"+String.valueOf(position)+"件商品", Toast.LENGTH_SHORT).show();
        }
    }
});
```

v. 此时主界面还差最后一步就完工了就是 FloatingActionButton 的点击事件和对于转跳之后回传数据的处理。

FloatingActionButton 的点击事件：

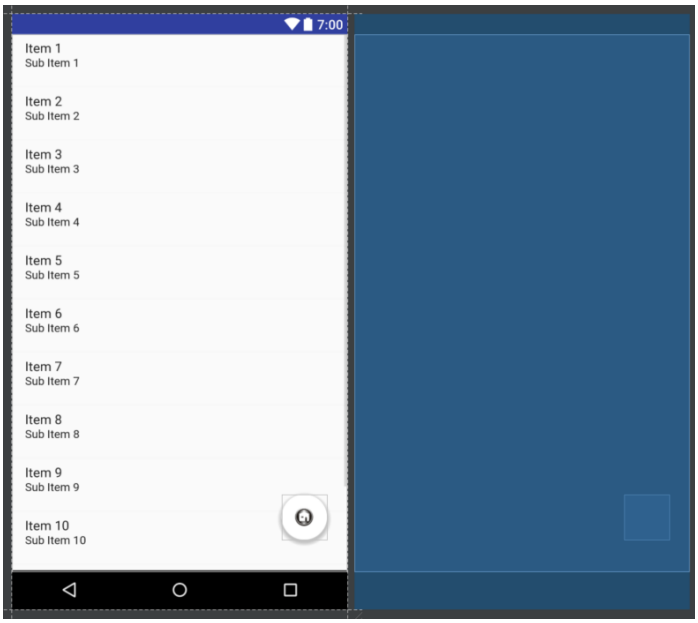
```
floatingActionButton = (FloatingActionButton) findViewById(R.id.floatingActionButton); //初始化按钮
floatingActionButton.setOnClickListener(new View.OnClickListener() {
    @Override //跳转至 购物车界面
    public void onClick(View view) {
```

```
        Intent i = new Intent(MainActivity.this, Shoppingcart.class);
        Bundle bundle = new Bundle();//传输数据
        bundle.putParcelableArrayList("data",data.getGood_list());
        bundle.putParcelableArrayList("cart",data.getCart_list());
        i.putExtra("mainActivity",bundle);
        startActivityForResult(i,1);
    }
});
```

转跳后回传数据的处理：

```
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent); //数据回传
    if (resultCode == RESULT_OK) {
        Bundle extras = intent.getExtras();
        ArrayList<Good> tmp= extras.getParcelableArrayList("data");
        ArrayList<Good> tmp1= extras.getParcelableArrayList("cart");
        data.getGood_list().clear();
        animationAdapter.notifyDataSetChanged();
        mRecyclerView.removeAllViews();
        data.setGood_list(tmp);
        data.setCart_list(tmp1);
        init();
        init_listener();
    }
}
```

2. 第二个部分购物车界面。
   i. 首先还是购物车界面的 Layout，此处和之前的主界面是完全不同的两个界面用于实现附加功能。此处需要在 tools:context 添加对应的 activity 的类，同时 manifest 要注册这个类。



此处还需 Item 需要的 xml 文件，但是跟之前的类似不在展示。

ii. 此处使用 ListView 的自定义 Adapter 进行填充数据，具体看课后实现部分。

iii. 首先我们从主界面中接收传过来的数据，然后增加上购物车那个特有的一行

```
private void get_set_data(){ //接收数据
    Bundle extras = getIntent().getBundleExtra("mainActivity");
    if (extras != null) {
        ArrayList<Good> tmp = extras.getParcelableArrayList("data");
        ArrayList<Good> tmp1 = extras.getParcelableArrayList("cart");
        data.setGood_list(tmp);
        data.setCart_list(tmp1);
    }
    //购物车时增加一列空
    if (data.getCart_list().size() == 0 || !data.getCart_list().get(0).getGoodName().equals("购物车")) {
        Good good = new Good("购物车", "价格", " ", " ");
        good.setGoodFisrt("*");
        data.addCart_list(0, good);
    }
}
```

iv. 之后初始化控件和 ListView

```
private void init(){//初始化
    builder = new AlertDialog.Builder(this);
    listView = (ListView) this.findViewById(R.id.myListView);
    myAdapter = new MyAdapter(this, data.getCart_list(), 0);
    listView.setAdapter(myAdapter);
    floatingActionButton = (FloatingActionButton) findViewById(R.id.floatingActionButton1);
}
```

v. 然后注册监听事件，此处传递到商品信息界面的也是两个 good 类 list

```
floatingActionButton.setOnClickListener(new View.OnClickListener() { //按钮转跳
    @Override
    public void onClick(View view) { //设置数据
        Intent i = new Intent();
        Bundle bundle = new Bundle();
        bundle.putParcelableArrayList("data", data.getGood_list());
        bundle.putParcelableArrayList("cart", data.getCart_list());
        i.putExtras(bundle);
        setResult(RESULT_OK, i);
        finish();
    }
});
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {//单击进入详细
        if (data.getCart_list().get(i).getGoodName().equals("购物车")) {
        } else {//数据传输
            Intent intent = new Intent(Shoppingcart.this, Good_Info.class);
            Bundle bundle = new Bundle();
            bundle.putInt("position", i);
```

```
            bundle.putParcelableArrayList("data", data.getGood_list());
            bundle.putParcelableArrayList("cart", data.getCart_list());
            intent.putExtra("shoppingcart", bundle);
            startActivityForResult(intent, 2);
        }
    }
});
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {//长按提示删除
    @Override
    public boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l) {
        if (data.getCart_list().get(i).getGoodName().equals("购物车")) {
        } else {
            alertdialog_build(i);
            AlertDialog dialog = builder.create(); //完成创建 AlertDialog 并显示
            dialog.show();
        }
        return true;
    }
});
```

然后设置一下上面需要 Alertdialog 此处建立的代码和上一次的差不多，但是此次只需要确定和取消。

```
public void alertdialog_build(final int position) {
    //alertdialog 初始化
    builder.setTitle("移除商品 ");
    final String tmp = data.getCart_list().get(position).getGoodName();
    builder.setMessage("从购物车移除" + tmp);//设置显示文本
    builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
        @Override //设置确定按钮动作
        public void onClick(DialogInterface dialogInterface, int i) {
            data.removeCart_list_index(position);
            myAdapter.notifyDataSetChanged();
            Toast.makeText(getApplicationContext(), "成功删除 " + tmp, Toast.LENGTH_SHORT).show();
        }
    });
    builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
        @Override //设置取消按钮动作
        public void onClick(DialogInterface dialogInterface, int i) {
            Toast.makeText(getApplicationContext(), "您选择了取消", Toast.LENGTH_SHORT).show();
        }
    });
    builder.setCancelable(true); //允许取消
}
```

vi. 之后设置一下对于商品信息类回传数据的处理就大功告成

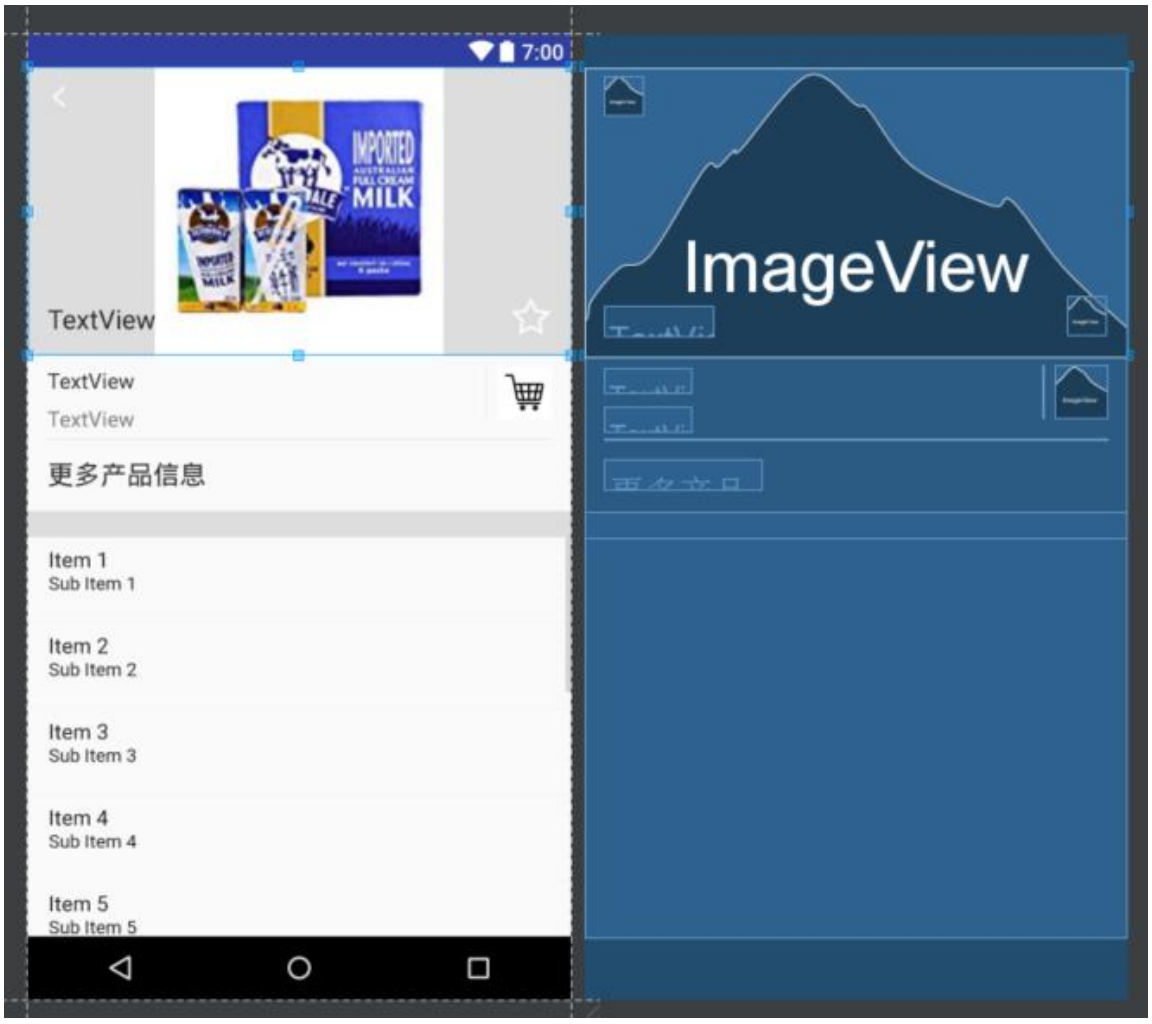
```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) { //接收返回数据
    super.onActivityResult(requestCode, resultCode, intent);
    if (resultCode == RESULT_OK) {
```

```
        Bundle extras = intent.getExtras();
        ArrayList<Good> tmp= extras.getParcelableArrayList("data");
        ArrayList<Good> tmp1= extras.getParcelableArrayList("cart");
        data.setGood_list(tmp);
        data.setCart_list(tmp1);
        myAdapter = new MyAdapter(this, data.getCart_list(), 0);
        listView.setAdapter(myAdapter); //接收返回数据重新建表
    }
}
```

3. 商品信息界面

i. 商品信息界面 xml 的建立，此处使用 RelativeLayout，同时和之前一样也需要组成绑定对应的 activity 类：

此处因为数据较多直接展示一下布局结果：



同样的这里也有一个 listView,所以也需要一个 Item 的 xml，不过这个简单不展示。

ii. 然后设计商品信息界面对应的 activity 类，首先是初始化和建立那个固定的 listView，这里为了使得少写一个适配器，直接把这个固定的 listView 当商品装进去。

```
private void init(){ //初始化
    String []hits ={"一键下单","分享商品","不感兴趣","查看更多商品促销信息"};
    hit=new ArrayList<>();
    for(int i=0;i<4;i++){//假装商品数据装进去
```

```
        Good good = new Good();
        good.setGoodName(hits[i]);
        hit.add(good);
    }
    imageView=(ImageView)findViewById(R.id.imageView);
    star = (ImageView)findViewById(R.id.imageView2);
    cart = (ImageView)findViewById(R.id.imageView3);
    back = (ImageView)findViewById(R.id.imageView4);
    name = (TextView)findViewById(R.id.info_name);
    price = (TextView)findViewById(R.id.info_price);
    info = (TextView)findViewById(R.id.info_info);
    list = (ListView)findViewById(R.id.myListView1);
}
```

iii. 然后是接收数据，此处要分别判断数据是来自主界面还是购物车界面。

```
private  void get_set_data(){
    Bundle extras_main = getIntent().getBundleExtra("mainActivity"); //获取数据  主界面
    Bundle extras_shop = getIntent().getBundleExtra("shoppingcart");//购物车
    if (extras_main != null||extras_shop != null) {
        if(extras_main != null) {  //主界面
            flag = true;            //设置 flag
            id=extras_main.getInt("position"); //获得数据位置
            ArrayList<Good>tmp = extras_main.getParcelableArrayList("data"); //获得数据
            ArrayList<Good>tmp1 = extras_main.getParcelableArrayList("cart");
            data.setGood_list(tmp);//设置数据
            data.setCart_list(tmp1);
        }
        else{
            flag=false;//购物车//设置 flag
            id=extras_shop.getInt("position");//获得数据位置
            ArrayList<Good>tmp = extras_shop.getParcelableArrayList("data");//获得数据
            ArrayList<Good>tmp1 = extras_shop.getParcelableArrayList("cart");
            data.setGood_list(tmp);//设置数据
            data.setCart_list(tmp1);
        }
        set_info();//设置界面
    }
}
```

iv. 然后根据传进来的数据设置对应的商品信息,此处对于图片的处理设置比较麻烦一点，同时因为能够记录收藏情况所以那个星星也要根据传入的收藏情况进行处理。

```
private void set_info(){
    String names; //获商品名称
    if(flag)names=data.getGood_list_index(id).getGoodName();
    else names=data.getCart_list_index(id).getGoodName();
    //匹配图片
    if(names.equals("Enchated Forest"))imageView.setImageResource(R.mipmap.enchatedforest);
    else if(names.equals("Arla Milk"))imageView.setImageResource(R.mipmap.arla);
```

```
    else if(names.equals("Devondale Milk"))imageView.setImageResource(R.mipmap.devondale);
    else if(names.equals("Kindle Oasis"))imageView.setImageResource(R.mipmap.kindle);
    else if(names.equals("waitrose 早餐麦片"))imageView.setImageResource(R.mipmap.waitrose);
    else if(names.equals("Mcvitie's 饼干"))imageView.setImageResource(R.mipmap.mcvitie);
    else if(names.equals("Ferrero Rocher"))imageView.setImageResource(R.mipmap.ferrero);
    else if(names.equals("Maltesers"))imageView.setImageResource(R.mipmap.maltesers);
    else if(names.equals("Lindt"))imageView.setImageResource(R.mipmap.lindt);
    else if(names.equals("Borggreve"))imageView.setImageResource(R.mipmap.borggreve);
    //设置 价格 物品名称 物品提示
    if(flag) {//商品列表转跳时
        name.setText(data.getGood_list_index(id).getGoodName());
        price.setText(data.getGood_list_index(id).getGoodPrice());
        if (data.getGood_list_index(id).getStar() == 1)
            star.setImageResource(R.mipmap.full_star);
        else star.setImageResource(R.mipmap.empty_star);
        info.setText(data.getGood_list_index(id).getGoodTypes() + " " + data.getGood_list_index(id).getGoodInfo());
    }else{//购物车列表转跳时
        name.setText(data.getCart_list_index(id).getGoodName());
        price.setText(data.getCart_list_index(id).getGoodPrice());
        if (data.getCart_list_index(id).getStar() == 1)
            star.setImageResource(R.mipmap.full_star);
        else star.setImageResource(R.mipmap.empty_star);
        info.setText(data.getCart_list_index(id).getGoodTypes() + " " + data.getCart_list_index(id).getGoodInfo());
    }
    MyAdapter myAdapter=new MyAdapter(this,hit,1); //设置下面 4 个 items
    list.setAdapter(myAdapter);
}
```

v. 注册不同按钮对应的点击事件

```
private void init_listener(){
    star.setOnClickListener(new View.OnClickListener() { //收藏按钮监听
        @Override
        public void onClick(View view) {
            if(flag) {//商品列表转跳时
                if (data.getGood_list_index(id).getStar() == 1) {//判断是否收藏
                    star.setImageResource(R.mipmap.empty_star);
                    Toast.makeText(Good_Info.this,"成功将"+data.getGood_list_index(id).getGoodName()+"取消收藏",Toast.LENGTH_SHORT).show();
                } else {
                    star.setImageResource(R.mipmap.full_star);
                    Toast.makeText(Good_Info.this,"成功将"+data.getGood_list_index(id).getGoodName()+"加入收藏",Toast.LENGTH_SHORT).show();
                }
                data.setGood_list_Stat(id); //同步购物车列表收藏
                for(int i=0;i<data.getCart_list().size();i++){
                    if(data.getGood_list_index(id).getGoodName().equals(data.getCart_list_index(i).getGoodName())){
                        data.setCart_list_Stat(i);
                    }
                }
            }else{//购物车列表转调试
                if (data.getCart_list_index(id).getStar() == 1) {//判断是否收藏
                    star.setImageResource(R.mipmap.empty_star);
```

```
                    Toast.makeText(Good_Info.this,"成功将"+data.getCart_list_index(id).getGoodName()+"取消收藏",Toast.LENGTH_SHORT).show();
                } else {
                    star.setImageResource(R.mipmap.full_star);
                    Toast.makeText(Good_Info.this,"成功将"+data.getCart_list_index(id).getGoodName()+"加入收藏",Toast.LENGTH_SHORT).show();
                }
                data.setCart_list_Stat(id); //同步商品列表收藏
                for(int i=0;i<data.getGood_list().size();i++){
                    if(data.getCart_list_index(id).getGoodName().equals(data.getGood_list_index(i).getGoodName())){
                        data.setGood_list_Stat(i);
                    }
                }
            }
        }
    });
    cart.setOnClickListener(new View.OnClickListener() { //购物车按钮
        @Override
        public void onClick(View view) {
            if(flag){
                int incart= -1;  //确定购物车里有没有
                for(int i=0;i<data.getCart_list().size();i++)
                    if(data.getGood_list_index(id).getGoodName().equals(data.getCart_list_index(i).getGoodName()))
                        incart=i;
                if(incart== -1) //没有加入购物车
                    data.addCart_list(data.getGood_list_index(id));
                else //有数量+1
                    data.setCart_list_Cnt(incart,data.getCart_list_index(incart).getCnt()+1);
                Toast.makeText(Good_Info.this,"成功将"+data.getGood_list_index(id).getGoodName()+"加入购物车",Toast.LENGTH_SHORT).show();
            }
            else {
                data.setCart_list_Cnt(id,data.getCart_list_index(id).getCnt()+1);
                Toast.makeText(Good_Info.this,"成功将"+data.getCart_list_index(id).getGoodName()+"加入购物车",Toast.LENGTH_SHORT).show();
            }
        }
    });
    back.setOnClickListener(new View.OnClickListener() { //返回按钮监听
        @Override
        public void onClick(View view) {  //数据返回
            Intent i = new Intent(Good_Info.this, Shoppingcart.class);
            Bundle bundle = new Bundle();
            bundle.putParcelableArrayList("data",data.getGood_list());
            bundle.putParcelableArrayList("cart",data.getCart_list());
            i.putExtras(bundle);
            setResult(RESULT_OK,i);
            Toast.makeText(Good_Info.this,"back",Toast.LENGTH_SHORT).show();
            finish();
        }
    });
}
```

至此界面全部完成，可以上虚拟机运行。

（三） 实验中遇到的困难和解决思路：

1. RecyclerView 的时候一开始想使用模板类的方式，结果写了完提示不能实例化，于是直接将模板类的方式改成了数据类的方式直接实现不在使用抽象。

2. 实验中开始使用 RecyclerView 的时候写好了其专用适配器之后，发现运行就立马崩溃，然后一直找不到问题，最后调用 debug 模式，找崩溃点，然后发现是因为在 build 文件中加了实验文档中的一句导致的认不到 TextView 模块导致模块加载时候报错，崩溃。删掉就好了。这问题完全是实验文档坑人了。

3. 在测试运行的时候发现 RecyclerView，添加了动画之后，如果删除之后快速点击被删除的部分会导致崩溃，（给出的 apk 也有这个问题），此时再次使用 debug,发现删除之后快速点击删除部位会导致返回的点击位置为-1,所以此时需要在点击是多加一个判断 position>=0 即可。

4. 购物车页面使用 alertdialog 的时候出现崩溃，debug 发现是提示说有控件没有没定义宽度，然后去找了一遍 xml 确认所有都定义宽度，此时再次怒改 build 把所有自己加上的的附加项全部删除，让 Android studio 自行添加需要部分，此时完美解决。所以又是实验文档的问题。

5. 运行的时候发现部分列表项宽度不对，会自动变化，之后修改了一下对于的 Items 的 xml，修改了高度不在 match parent 而是随内容，然后使用 padding 固定宽度。

五、 课后实验结果

1. 实验中我们将商品数据封装成商品类然后，我们在两个页面间传递就是在传递这个类的一个 list，所以此处需要用到 Parcelable 接口。
首先是这个类的实现没什么好说的 一堆 get、set 所以就展示一部分

```
public class Good  implements Parcelable{
    private String name;
    private String price;
    private String info;
    private String types;
    private String first;
    private int cnt;
    private int star;
    public Good(){}
    public Good(String name,String  price,String info,String types){ //初始化
        setGoodName(name);
        setGoodInfo(info);
        setGoodTypes(types);
        setGoodPrice(price);
        cnt=0;
        star=0;
    }
    public void setGoodName(String name){//设置商品名称
        this.name=name;
```

```
        first= String.valueOf(this.name.charAt(0)).toUpperCase();
    }
    //省略一堆
    public String getGoodName(){
        return name;
    }
   //省略一堆
}
```

之后我们需要定义 Parcelable 接口：

```
@Override
public int describeContents() {
    return 7;
}
@Override
public void writeToParcel(Parcel parcel, int flags) {
    parcel.writeString(name);
    parcel.writeString(price);
    parcel.writeString(info);
    parcel.writeString(types);
    parcel.writeString(first);
    parcel.writeInt(cnt);
    parcel.writeInt(star);
}
public static final Parcelable.Creator<Good> CREATOR =new Parcelable.Creator<Good>() {
    @Override
    public Good createFromParcel(Parcel source){
        Good good = new Good();
        good.name=source.readString();
        good.price=source.readString();
        good.info=source.readString();
        good.types=source.readString();
        good.first=source.readString();
        good.cnt=source.readInt();
        good.star=source.readInt();
        return good;
    }
    @Override
    public Good[] newArray(int size){
        return new Good[size];
    }
};
```

此时就可以使用 putParcelableArrayList 进行传参了。此外定义了一个 Data 来维护数据商品类，Data 类中存放了两个列表购物车列表和商品列表，和商品数据等。展示部分代码，其他代码类似。

```
public Good getGood_list_index(int i){return good_list.get(i);}
//删除商品列表中的一个
public void removeGood_list_index(int i){    good_list.remove(i);}
//返回商品列表
public ArrayList<Good> getGood_list() {    return good_list;}
```

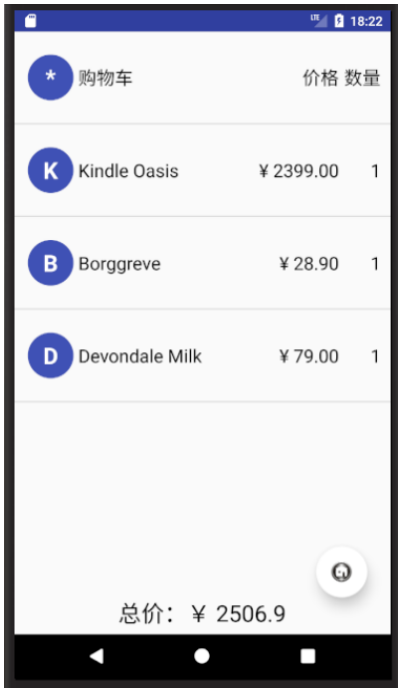
```
//重设商品列表
public void  setGood_list(ArrayList<Good> arrayList){good_list=arrayList;}
```

2. 实现自定义的 Adapter,自定义的 Adapter 需要继承 BaseAdapter 所以要重载下面 4 个方法
int getCount(); 获得购居项列表的长度，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

```
public class MyAdapter extends  BaseAdapter  {
    private Context context;
    private List<Good> list;
    private int choose;
    //同时适配两种界面 一种是购物车 一种是商品详情下方列表
    public MyAdapter(Context context, List<Good> list,int choose) {
        this.context = context;
        this.list = list;
        this.choose=choose;
    }
    @Override
    public int getCount() {//获得购居项列表的长度，也就是一共有多少个数据项。
        if (list == null) {
            return 0;
        }
        return list.size();
    }
    @Override
    public Object getItem(int i) {//获得某一个数据项。
        if (list == null) {
            return 0;
        }
        return list.get(i);
    }
    @Override
    public long getItemId(int i) {
        return i;
    }//获得数据项的位置。
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {//获得数据项的布局样式
        View convertView;
        ViewHolder viewHolder = new ViewHolder();
        if (view == null) {
            //创建 View
            if(choose == 0) {//购物车时
                convertView = LayoutInflater.from(context).inflate(R.layout.shopping_cart, null);
                viewHolder.goodFirst = (TextView) convertView.findViewById(R.id.first1);
                viewHolder.goodName = (TextView) convertView.findViewById(R.id.name1);
                viewHolder.goodPrice = (TextView) convertView.findViewById(R.id.price1);
                viewHolder.goodCnt = (TextView) convertView.findViewById(R.id.cnt);
            }
```

```
            else {//商品下方列表时
                convertView = LayoutInflater.from(context).inflate(R.layout.good_info_item, null);
                viewHolder.goodName = (TextView) convertView.findViewById(R.id.info_textView);
            }
            convertView.setTag(viewHolder);
        } else {
            convertView = view;
            viewHolder = (ViewHolder) convertView.getTag();
        }
        if(choose == 0) {//购物车时
            viewHolder.goodFirst.setText(list.get(i).getGoodName_First());
            viewHolder.goodName.setText(list.get(i).getGoodName());
            viewHolder.goodPrice.setText(list.get(i).getGoodPrice());
            viewHolder.goodCnt.setText(String.valueOf(list.get(i).getCnt()));
            if(list.get(i).getGoodName().equals("购物车"))//设置第一行
                viewHolder.goodCnt.setText("数量");
        }
        else{//商品下方列表时
            viewHolder.goodName.setText(list.get(i).getGoodName());
        }
        return convertView;
    }
    private class ViewHolder {
        public TextView goodFirst;
        public TextView goodName;
        public TextView goodPrice;
        public TextView goodCnt;
    }
}
```

3. 此处学着淘宝什么的做了一个视图切换，然后购物车加入结算功能。
先简单的展示一下效果：



代码的话主要就是修改一下 xml 布局文件，购物车计算部分比较简单就是一个加法而已，实现视图切换需要+图片显示需要重新设置 LayoutManager 和重新建立 Adapter 然后重新显示和建立时间监听：

```
private void change(){
    int scrollPosition = 0;
    // If a layout manager has already been set, get current scroll position.
    if (mRecyclerView.getLayoutManager() != null) {
        scrollPosition = ((LinearLayoutManager) mRecyclerView.getLayoutManager())
                .findFirstCompletelyVisibleItemPosition();
    }//布局切换
    if(flag) {
        mLayoutManager = new LinearLayoutManager(this);
        commonAdapter = new CommonAdapter(MainActivity.this, R.layout.items, data.getGood_list(),
flag);
    }else{
        mLayoutManager = new GridLayoutManager(this, 2);
        commonAdapter = new CommonAdapter(MainActivity.this, R.layout.item2, data.getGood_list(),
flag);
    }
    mRecyclerView.setLayoutManager(mLayoutManager);
    animationAdapter = new ScaleInAnimationAdapter(commonAdapter);
    mRecyclerView.setAdapter(animationAdapter);//填充数据
    Adapter_listener();//建立监听
    mRecyclerView.scrollToPosition(scrollPosition);
}
```

六、 实验思考及感想

这次实验比较多，有比较烦需要实现 3 种界面布局，虽然说，商品列表和购物车列表界面类似，但是他们是两个不同的 View，所以还是需要些两种适配器的。写适配器还要注意模板类当然此处偷懒没写。其他的地方基本没什么，主要就是布局和逻辑多了一点，主要就是逻辑要求很多。

实验中给出的样本 app 很多功能还是没有实现的比如那个收藏功能，基本退出就没。一开始实现的时候还是考虑这个然后使用了类进行传参，然后发现这个类传参还需要自定义接口，不过好在也是挺简单的百度看看就回了。

不得不吐槽一下实验文档，实验文档那个在 build 加东西，结果帮倒忙，出现一些无法理解的崩溃问题，最后还是使用 debug 功能直接找到报错行和报错信息，那个实验文档在 build 加的东西每次都是使得页面在一开始建立的时候就不成功，说实在我发现 Android Studio 在你拖入控件的时候会帮你加 build 文件并不需要手动加。这个真的坑了很久。