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	2015	专业（方向）	互联网
学号	15352194	姓名	梁杰鑫
电话	15113959962	Email	Alcanderian@gmail.com
开始日期	2017.10.20	完成日期	2017.10.27

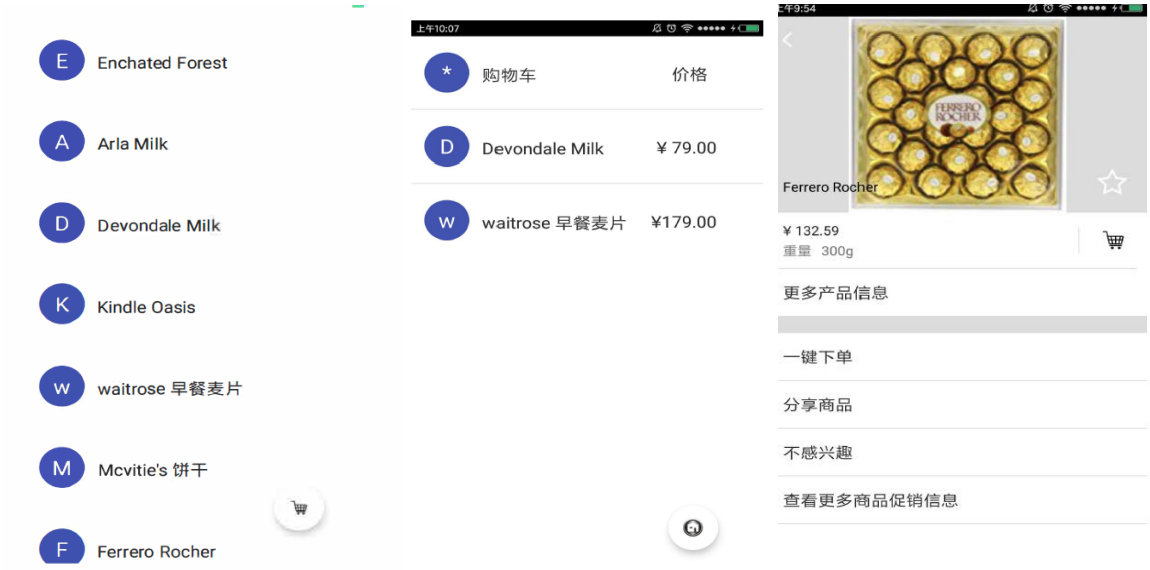
一、 实验题目

本次实验模拟实现一个商目表，有购物车及详细信息功能。
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 以及各尧置配器的用法

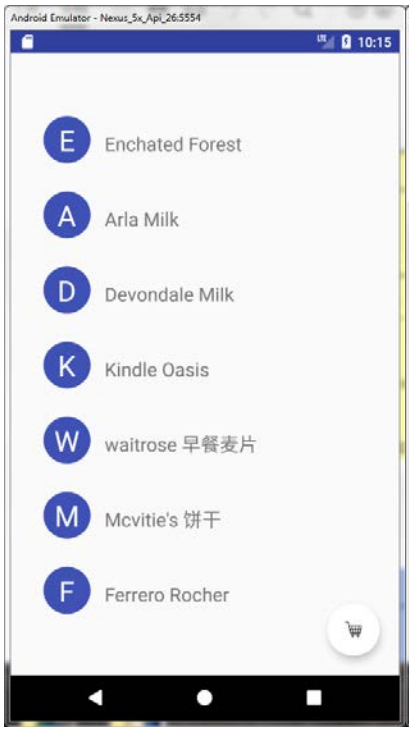
二、 实现内容

本次实验模拟实现一个商目表，有两个界面，第一个界面用于呈现商昂，如下左所示。
点击右下方的悬浮按钮可以切换到购物车，如下中图所示。
这两个列表随意点击一项可以进入详细信息页面。



三、 课堂实验结果

（1） 实验截图



初始页面

滚动动画

移除商品

详细信息

购物车

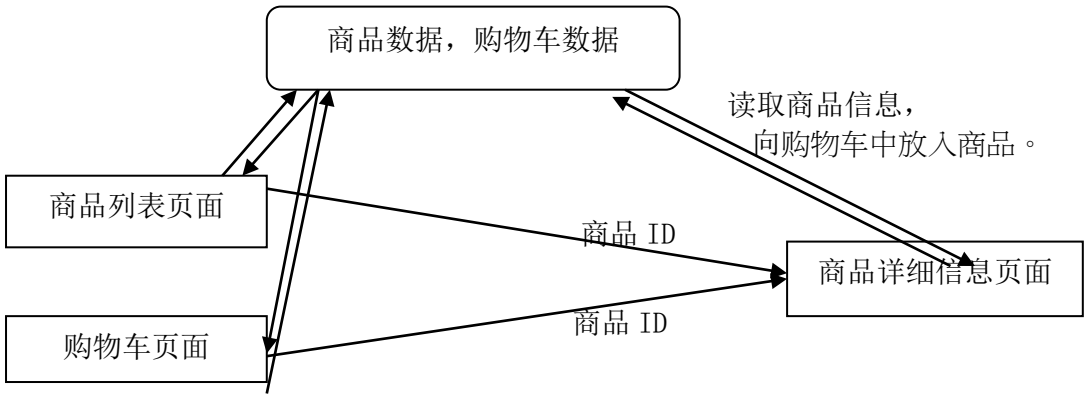
删除购物车中商品

（2） 实验步骤以及关键代码

1. 总体思路

虽然这个实验中的数据传输有很多种方法，但是，对于一个商城软件来说，我们应该是设置基于网络的数据库来获取数据的，所以这次实验中我们应该怎么呢？方法有很多，在 resource 中设置字符串，构建 Parcelable 接口等等。基于数据与应用解耦的思路，本次实验中采用了全局变量的模式进行数据储存，各个页面与数据之间是独立的。然后我们构建一个管理数据的 Class 即可。

数据流图如下：



使用这样的模型，使得数据与应用分离，所有页面公用数据库，可以避免页面之间繁琐的数据传输。

2. 数据管理 Class

与数据库表单的模型保持一致结构，类成员定义如下：

```
public class ItemData {

    public ArrayList<Map<String, Object>> data;
    public ArrayList<Integer> stared;
    public ArrayList<Integer> itemid;
    public ArrayList<Integer> imgid;
```

在这张表 itemid 是 INDEX，data，stared 和 imgid 为数据内容。

我们知道，在数据库中我们可以通过 INDEX 快速索引到数据所在的行，进行数据读取。所以我们的商品详细信息只需要得到一个 itemid，在读取数据库中数据即可。

该类的方法如下：

```
    public ItemData(boolean what) {...}

    public void remove(int pos) {...}

    public void add(Map<String, Object> m, int st, int iid, int im_id) {...}

    public int getIndex(int iid) { return itemid.indexOf(iid); }
}
```

其中 getIndex 就是根据 itemid 去索引行号。

构造函数中的 what 决定了构建购物车还是商品列表，购物车中不放入任何商品信息，商品列表会初始化我需要的商品信息。

3. 改造 Application 类

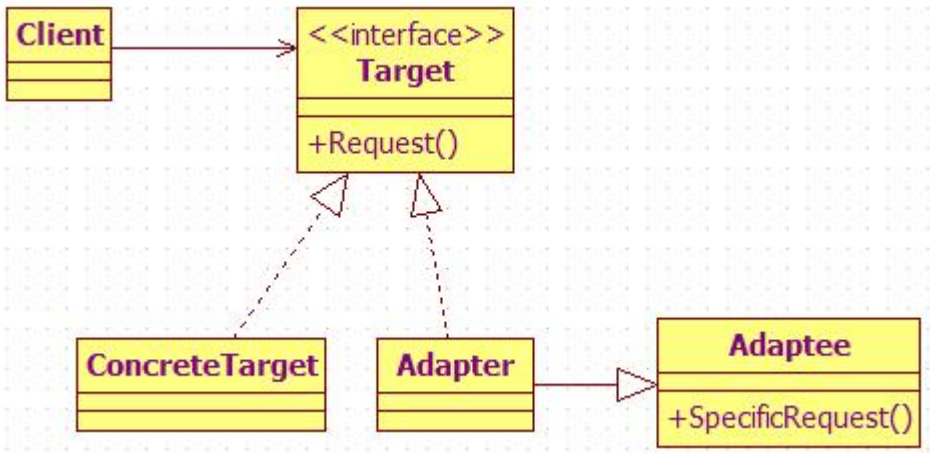
```
public class MyShopApp extends Application {
    public ItemData good_data;
    public ItemData cart_data;
    public MyLvAdapter lvad;
    @Override
    public void onCreate() {
        super.onCreate();
        good_data = new ItemData(true);
        cart_data = new ItemData(false);
    }
}
```

首先继承 Application，增加成员函数，分别是商品列表，购物车和一个 ListViewAdpater，这个适配器是通知购物车刷新用的，下面会讲到他的用法。

4. 自定义适配器

适配器模式是一种设计模式，使用的情况是：提供了统一的数据接口，但是无法规定统一的数据类型。

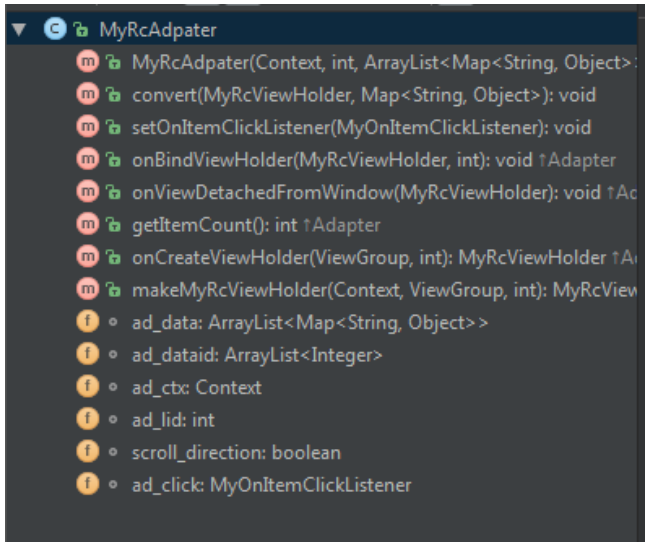
由此可见适配器的作用是：在不对每一种数据都进行子类化以匹配它们的接口的情况下，将数据转换为接口可以接受的格式。



在 Recycleview 中，必须自定义它的适配器以适配我们自定义的数据结构。另外，还需要自定义 ViewHolder 来管理 RecycleView 中子布局的内容。

- RecyclerView

RecyclerView.Adpater 的定义如下：



其中 data 与 dataid 用于存放数据及其索引号，lid 是 RecyclerView 中子 View 的布局 id。scroll_direction 用于记录滚动的方向，决定滚动时播放的动画。

MyRcApater 最重要的是 onBindViewHolder 和 convert 函数。

```
    @Override
    public void onBindViewHolder(final MyRcViewHolder vh, int pos) {
        convert(vh, ad_data.get(pos));
        if (ad_click != null) {
            vh.v_me.setOnClickListener((v) -> {
                    ad_click.onClick(vh.getAdapterPosition());
            });
            vh.v_me.setOnLongClickListener((v) -> {
                    ad_click.onLongClick(vh.getAdapterPosition());
                    return true;
            });
        }
        if (scroll_direction) {
            vh.itemView.setAnimation(AnimationUtils.loadAnimation(ad_ctx, R.anim.rotate_in_top));
        } else {
            vh.itemView.setAnimation(AnimationUtils.loadAnimation(ad_ctx, R.anim.rotate_in_down));
        }
    }
```

onBindViewHolder 在绑定 ViewHolder 时对 ViewHolder 进行相关的处理，这里是设置点击函数和设置进入页面的动画。

而 conevert 是一个空函数，它是需要重载的，在 MainActivity 中，对它进行了重载：

```
        MyRcAdpater rc_ad = new MyRcAdpater(MainActivity.this, R.layout.view_item,
                new ArrayList<>(my_app.good_data.data), new ArrayList<>(my_app.good_data.itemid)) {
            @Override
            public void convert(MyRcViewHolder vh, Map<String, Object> m) {
                TextView icon = vh.getView(R.id.item_icon);
                TextView name = vh.getView(R.id.item_name);
                TextView price = vh.getView(R.id.item_price);
                name.setText(m.get("name").toString());
                icon.setText(m.get("first").toString());
                price.setText("");
            }
        };
```

重载 convert 的必要性在于：我们设配器存放数据的结构式一个 Map，适配器是不会知道 Map 里装着什么数据的，所以对于特定内容的 Map，我们需要重载不同的 convert，将 Map 里的数据拿出来给 ViewHolder 中的数据。
加入我们这次的购物车也是用 RecycleView，那购物车的 convert 应该将 Map 中的价格也取出来传给 price。

OnItemClickListener

```
        rc_ad.setOnItemClickListener(new MyOnItemClickListener() {
            @Override
            public void onClick(int position) {
                MyRcAdpater rc_ad = (MyRcAdpater) rv_items.getAdapter();
                Intent it_item = new Intent();
                it_item.setClass(MainActivity.this, DetailActivity.class);
                it_item.putExtra("itemid", rc_ad.ad_dataid.get(position));
                startActivity(it_item);
            }
```

在短按的时候，创建 Intent，将所选的商品的 itemid 传给 DetailAcivity。

```
        @Override
        public void onLongClick(int position) {
            MyRcAdpater rc_ad = (MyRcAdpater) rv_items.getAdapter();
            rc_ad.ad_dataid.remove(position);
            rc_ad.ad_data.remove(position);
            rc_ad.notifyItemRemoved(position);
            rc_ad.notifyItemRangeChanged(position, rc_ad.ad_data.size());
            Toast.makeText(rc_ad.ad_ctx,
                    "移除第 " + position + "个商品", Toast.LENGTH_SHORT).show();
        }
    });
```

在长按的时候删除 item，不要忘记调用 notify。

RecyclerView.ViewHolder

```
public class MyRcViewHolder extends RecyclerView.ViewHolder {
    public SparseArray<View> v_basket;
    public View v_me;

    public MyRcViewHolder(View v_item) {
        super(v_item);
        v_me = v_item;
        v_basket = new SparseArray<>();
    }

    public <T extends View> T getView(int v_id) {
        View v = v_basket.get(v_id);
        if (v == null) {
            v = v_me.findViewById(v_id);
            v_basket.put(v_id, v);
        }
        return (T) v;
    }
}
```

在这个 ViewHolder 中采用了一种缓存的机制：
在成员中增加一个 SparseArray 作为缓存数组，当 getView 函数被调用时，先检查数组中是否存在 v_id 所对应的元素，如果有则直接回传，如果没有再实例化这个 View 并放到缓存中。
这样的好处在于不需要每次都调用 findViewById 这种耗时长的函数，快速生成 View。

- ListViewApdater

```
public class MyLvAdapter extends BaseAdapter {
    Context lv_ctx;
    ArrayList<Map<String, Object>> lv_data;
    ArrayList<Integer> lv_dataid;
```

用于购物车的 ListView 适配器，对它的 getView 函数，我们同样采取了检查 View 是否已经存在的机制，但是和上面的有所不同。

```
    @Override
    public View getView(int i, View v, ViewGroup vg) {
        View ret;
        MyLvViewHolder lv_vh;
        if (v == null) {
            ret = LayoutInflater.from(lv_ctx).inflate(R.layout.view_item, null);
            lv_vh = new MyLvViewHolder();
            lv_vh.name = ret.findViewById(R.id.item_name);
            lv_vh.first = ret.findViewById(R.id.item_icon);
            lv_vh.price = ret.findViewById(R.id.item_price);
            ret.setTag(lv_vh);
        } else {
            ret = v;
            lv_vh = (MyLvViewHolder) ret.getTag();
        }
        lv_vh.name.setText(lv_data.get(i).get("name").toString());
        lv_vh.first.setText(lv_data.get(i).get("first").toString());
        lv_vh.price.setText(lv_data.get(i).get("price").toString());

        return ret;
    }
```

同样我们也需要设置 ListView 的点击函数，因为 BaseApdater 自带点击函数接口，我们不需要另外定义接口。

短按接口和商品列表基本一致，这里不再复述，重点看长按函数。

```
        lv_items.setOnItemLongClickListener((parent, view, position, id) -> {
                if (position != 0) {
                    AlertDialog.Builder ad_builder;
                    ad_builder = new AlertDialog.Builder(MainActivity.this);
                    ad_builder
                            .setTitle("移除商品")
                            .setMessage("从购物车移除"
                                    + my_app.cart_data.data.get(position).get("name").toString()
                                    + "？")
                            .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int i) {
                                }
                            })
                            .setPositiveButton("确定", (dialog, i) -> {
                                    my_app.cart_data.remove(position);
                                    ((MyLvAdapter) lv_items.getAdapter()).notifyDataSetChanged();
                            }).create().show();
                }
                return true;
        });
    }
```

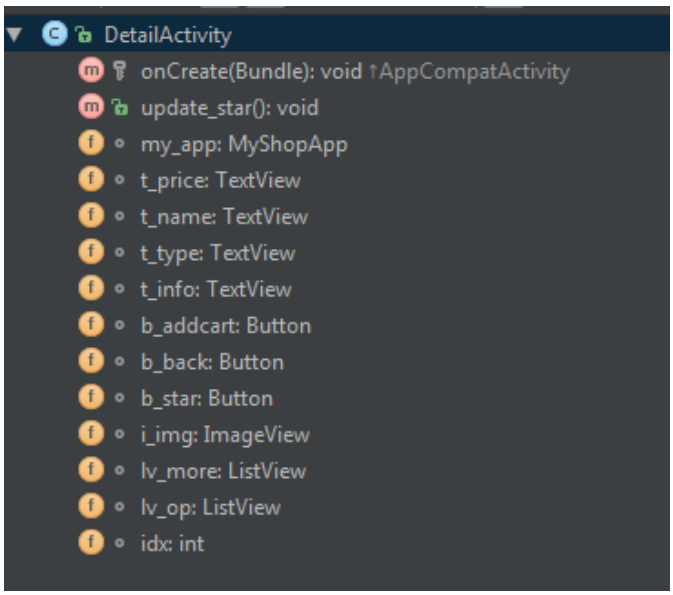
因为 ListViewAdpater 中的 data 是引用 my_app.cart_data 的，所以我们直接使用 cart_data 去删除购物车商品即可。

5. **列表切换显示**

```
fab_mode.setOnClickListener((v) → {
        if (view_mode == 1) view_mode = 2;
        else view_mode = 1;
        if (view_mode == 1) {
            lv_items.setVisibility(View.GONE);
            rv_items.setVisibility(View.VISIBLE);
            fab_mode.setImageResource(R.mipmap.shoplist);
        } else {
            lv_items.setVisibility(View.VISIBLE);
            rv_items.setVisibility(View.GONE);
            fab_mode.setImageResource(R.mipmap.mainpage);
        }
});
```

设置一个变量 mode，当 mode==1 是商品列表显示，mode==2 时是购物车显示。

6. **详细信息页面**



详细页面元素比较多，成员变量中有很多布局元素。

```
33          my_app = (MyShopApp) getApplication();
34          int iid = getIntent().getIntExtra("itemid", 0);
35
```

```
48          t_price.setText(my_app.good_data.data.get(idx).get("price").toString());
49          t_name.setText(my_app.good_data.data.get(idx).get("name").toString());
50          t_type.setText(my_app.good_data.data.get(idx).get("type").toString());
51          t_info.setText(my_app.good_data.data.get(idx).get("info").toString());
52          i_img.setImageResource(my_app.good_data.imgid.get(idx));
```

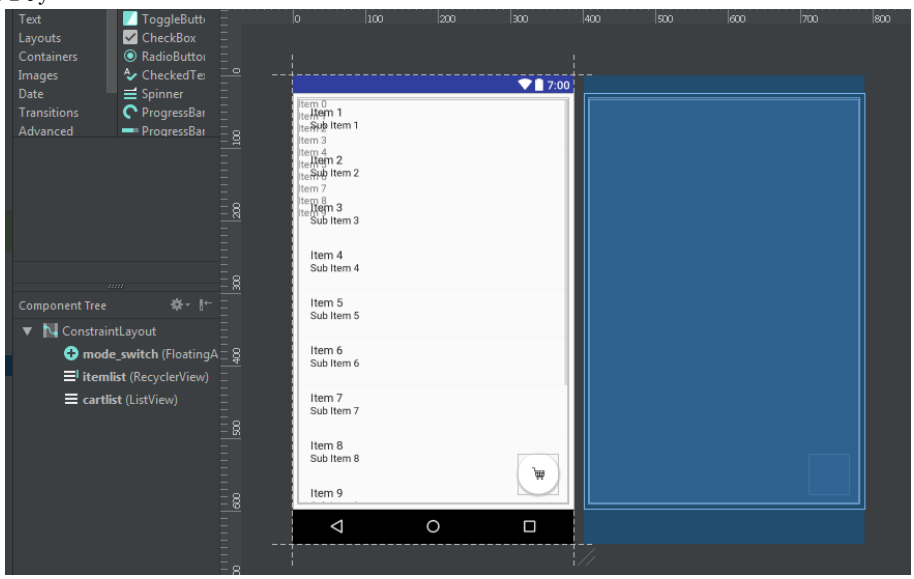
这两段代码是页面接受 itemid 参数，从索引中得到数据下标，然后从商品数据中取出数据。
为什么不直接传下标呢？因为我们不知道参数是购物车传过来还是商品列表传过来的，如果应用庞大，就可能还有很多个页面会调用这个页面。我们没理由去判断到底是那个页面传给我的参数，所以统一采用 itemid 可以简化逻辑，增加可扩展性。

```
public void update_star() {
    if (my_app.good_data.stared.get(idx) == 0) {
        b_star.setBackgroundResource(R.mipmap.empty_star);
    } else {
        b_star.setBackgroundResource(R.mipmap.full_star);
    }
}
```

update_star 用于更新收藏的状态，收藏的状态是持久化在数据库中的，所以并不是退出页面就变回空心星星。

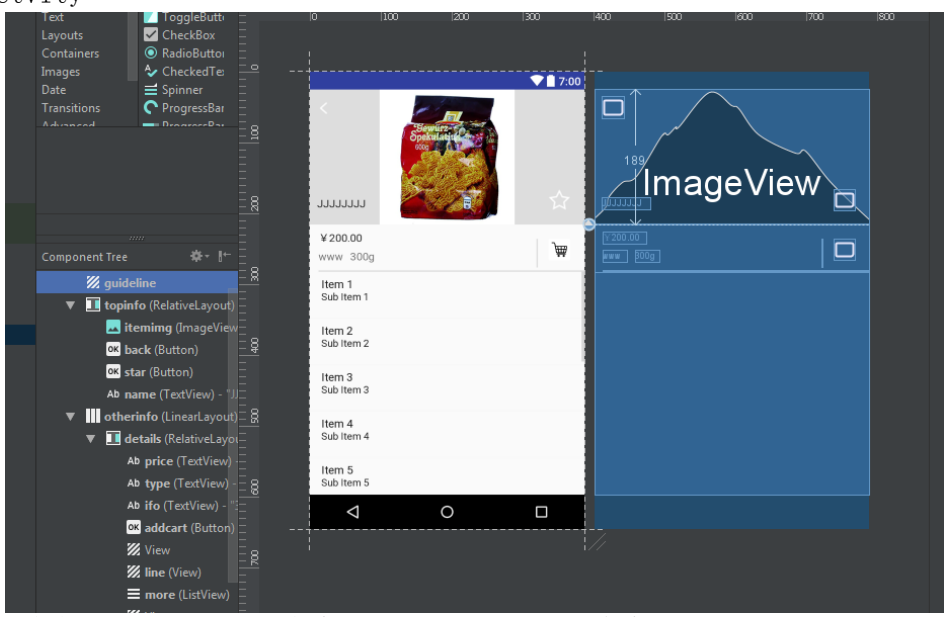
7. 布局

- MainActivity



因为比较简单就不再详细讲述。

- DetailActvity



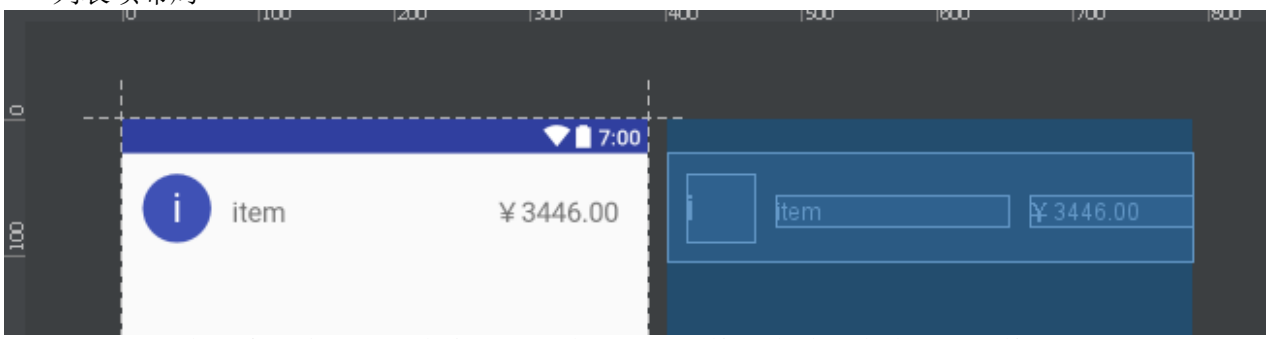
可以看到引导线在 189px 的位置，灰色的先挑我们用 View 来实现。返回按钮，星星，价格，购物车等元素我们部署在一个 relativeLayout 中。

```
<TextView
    android:id="@+id/price"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="12dp"
    android:layout_marginTop="10dp"
    android:text="￥200.00"
    android:textColor="@color/detialblack"
    android:textSize="15sp" />

<TextView
    android:id="@+id/type"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/price"
    android:layout_below="@+id/price"
    android:layout_marginTop="8dp"
    android:text="www"
    android:textColor="@color/detialdarkgray"
    android:textSize="15sp" />
```

RelativeLayout 的布局方法非常简单而且好用。
我们通过 marginStart 和 layout_below 等属性设置元素的相对位置即可。

- 列表项布局



列表项的布局有三个元素，我们在商品列表中不设置价格，在购物车中设置价格，就可以做到布局的复用。

（3） 实验遇到困难以及解决思路

1. **如何商品数据，如何使收藏信息持久化？**
   解决方案，分离数据和页面，采用全局变量存放数据？
2. **不知道是那个页面调用详细商品页面，怎么知道从那里取数据？**
   采用统一的参数，统一从商品信息取数据而不是判断哪个页面调用。
3. **如何保证详细页面的图片占据 1/3 高度？**
   设置 1/3 位置引导线，将图片的布局约束在引导线上。

四、 课后扩展

设置动画

首先写好动画文件

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:duration="300"
        android:fromDegrees="90"
        android:pivotX="0"
        android:pivotY="0"
        android:toDegrees="0" />
</set>
```

向下转动。

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:duration="300"
        android:fromDegrees="-90"
        android:pivotX="0"
        android:pivotY="100%"
        android:toDegrees="0" />
</set>
```

向上转动。

```java
rv_items.addOnScrollListener(new RecyclerView.OnScrollListener() {
    @Override
    public void onScrolled(RecyclerView rv, int dx, int dy) {
        super.onScrolled(rv, dx, dy);
        MyRcAdpater rc_ad = (MyRcAdpater) rv_items.getAdapter();
        rc_ad.scroll_direction = dy > 0;
    }
});
```

为适配器自定义滚动的监听器，判断滚动方向。

```java
@Override
public void onBindViewHolder(final MyRcViewHolder vh, int pos) {
    convert(vh, ad_data.get(pos));
    if (ad_click != null) {
        vh.v_me.setOnClickListener((v) -> {
                ad_click.onClick(vh.getAdapterPosition());
        });
        vh.v_me.setOnLongClickListener((v) -> {
                ad_click.onLongClick(vh.getAdapterPosition());
                return true;
        });
    }
    if (scroll_direction) {
        vh.itemView.setAnimation(AnimationUtils.loadAnimation(ad_ctx, R.anim.rotate_in_top));
    } else {
        vh.itemView.setAnimation(AnimationUtils.loadAnimation(ad_ctx, R.anim.rotate_in_down));
    }
}
```

前面说过的，在 onbindViewHolder 时设置动画。

```
    @Override
    public void onViewDetachedFromWindow(MyRcViewHolder vh) {
        super.onViewDetachedFromWindow(vh);
        vh.itemView.clearAnimation();
    }
```

最后便是在离开窗口时解除动画。

五、 感想

这是一个综合性比较强的项目，在一开始的时候遇到了不少问题，比如无法理解设配器模式、不知道如何设计数据传输方式、如何运用 RelativeLayout 等等。

要解决这些问题只能自己去多查找资料，去理解安卓的设计模式。总的来说这次的实验中学会了不少以前没想掉过的设计模式或者技巧。安卓本身自带的 API 固然是不能满足多种多样的展示效果的，这时候高度定制的 RecyclerView 应该就是大多数开发者所使用的列表部署工具了。可以说在这次试验之后，真正打开了安卓的大门。