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）	移动互联网
学号	15352286	姓名	任萌
电话	13763361232	Email	352600801@qq.com
开始日期	2017 年 10 月 20 日	完成日期	2017 年 10 月 24 日

一、 实验题目

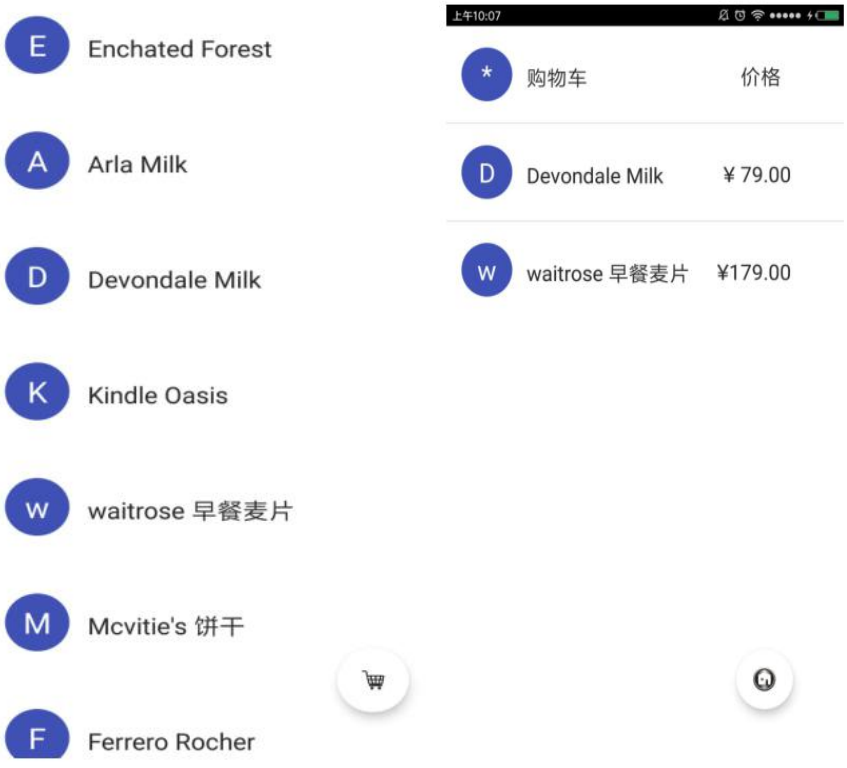
实验三：Intent、Bundle 的使用和 ListView 的应用

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，点击右下角的悬浮按钮可以切换到购物车，如下图所示：



上面两项列表点击任意一项后，可以看到详细的信息：

布局要求：

（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）购物车列表内部：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。

（4）商品详情界面中部和底部：内容和样式等见实验文档说明。

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 的图片要相应改变。可通过设置 RecyclerView 和 ListView 的可见状态来实现切换。

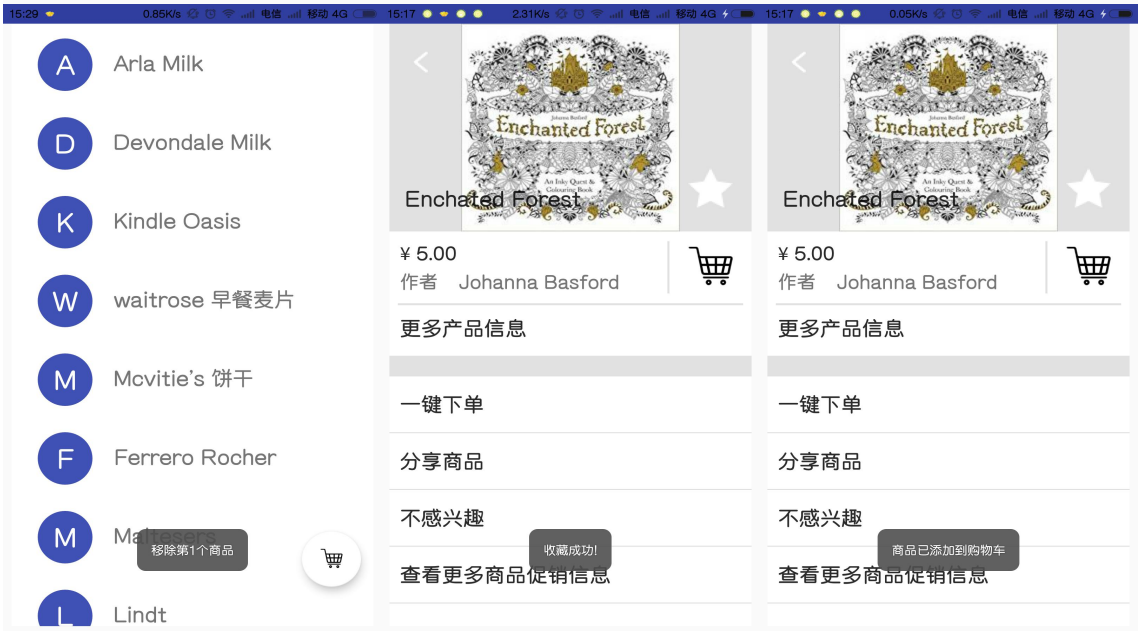
（3）使用 ListView 实现购物车。点击购物车的某一个商品会跳转到商品的详情界面，呈现该商品的详细信息；长按购物车中的商品会弹出对话框询问是否移除该商品，点击确定则移除该商品，点击取消则对话框消失。

（4）商品详情界面中点击返回图标返回上一层，点击星标会切换状态，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”。

四、 课堂实验结果

（1） 实验截图


Enchated Forest
Arla Milk
Devondale Milk
Kindle Oasis
waitrose 早餐麦片
Mcvitie's 饼干
Ferrero Rocher
Maltesers
购物车
价格
Enchated Forest
¥ 5.00
作者 Johanna Basford
更多产品信息
一键下单
分享商品
不感兴趣
查看更多商品促销信息


Arla Milk
Devondale Milk
Kindle Oasis
waitrose 早餐麦片
Mcvitie's 饼干
Ferrero Rocher
Lindt
移除第1个商品
收藏成功!
商品已添加到购物车
Enchated Forest
¥ 5.00
作者 Johanna Basford
更多产品信息
一键下单
分享商品
不感兴趣
查看更多商品促销信息

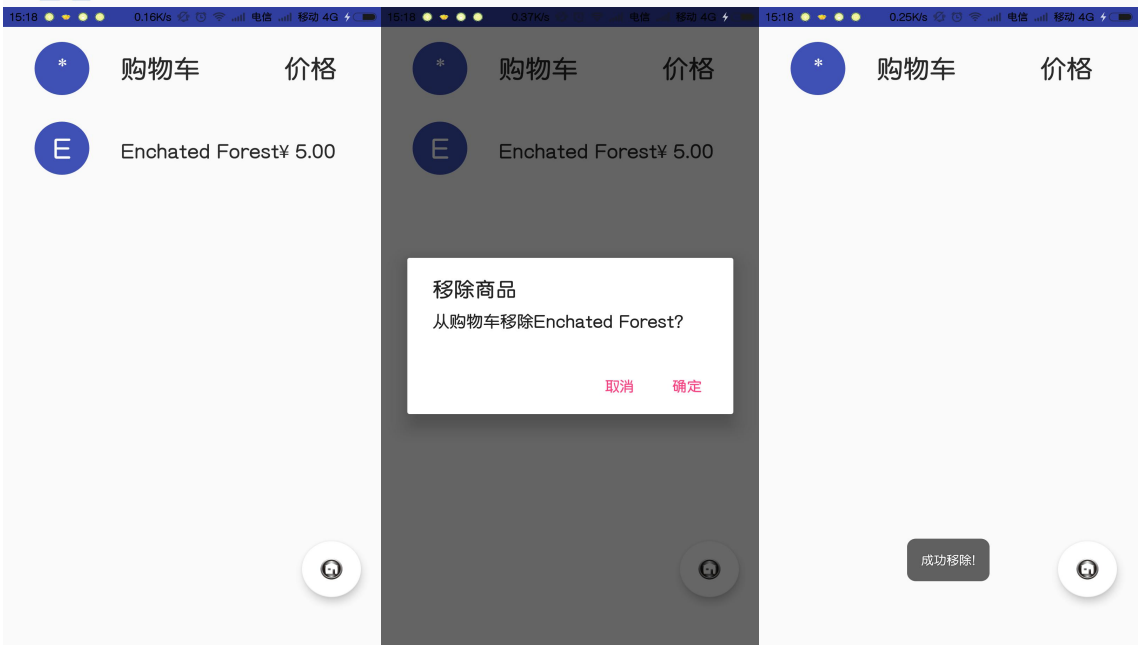

购物车
价格
Enchated Forest¥ 5.00
移除商品
从购物车移除Enchated Forest?
取消
确定
成功移除!

图 1 为主界面；图 2 为点击了悬浮按钮后切换到购物车界面，此时还没有添加商品，故为空；图 3 为点击商品后跳转到商品详情界面；图 4 为长按时删除第 i 个商品；图 5 为点击收藏按钮后空心星星变为实心；图 6 为点击购物车图标后，弹出提示成功添加到购物车；图 7 为添加了商品后返回购物车界面；图 8 为弹出对话框询问是否删除商品；图 9 为成功从购物车中删除了商品。

结果分析：

1. 整体布局效果基本符合实验文档的要求。

2. 初始界面是商品列表界面，在点击 item 时弹出该商品的详情界面，长按 item 时从商品列表中删除该商品，并弹出提示信息。

3. 进入商品详情界面后，显示效果符合文档要求，同时点击返回按钮时可以返回商品列表主菜单，点击收藏按钮时实心星星和空心星星切换，点击购物车按钮时添加该商品到购物车，并弹出提示信息。

4. 在商品列表界面点击悬浮按钮可以切换到购物车界面，这里可以显示所有被添加到购物车中的商品，同时长按某个商品时，会弹出对话框询问是否从购物车中移除。

（2） 实验步骤以及关键代码

1) 定义 Data 类

本次实验基于的对象是商品，它拥有图片、名字、价格、类型和产品信息等具体内容，故为了后面不同界面间数据传送的便利，我将其封装成了一个 Data 类。这个类中包含四个变量 name、price、type、info，同时定义了该类的构造函数和 get()函数。

```
public class Data {
    private String name;
    private String price;
    private String type;
    private String info;

    public Data(String name, String price, String type, String info) {
        this.name = name;
        this.price = price;
        this.type = type;
        this.info = info;
    }

    public String getName() {
        return name;
    }
    public String getPrice() { return price; }
    public String getType() {
        return type;
    }
    public String getInfo() {
        return info;
    }
}
```

2) 主页面

利用 RecyclerView 实现商品列表的展示。首先定义一个 RecyclerView 的实例 mRecyclerView 和它的适配器 homeAdapter，再使用 list 定义商品链表。

```
protected RecyclerView mRecyclerView;
protected List<Data> list = new ArrayList<Data>();
protected HomeAdapter homeAdapter;
```

由于在 RecyclerView 中必须要自定义实现 RecyclerView.Adapter 并为其提供数据集合，故我们需要重写 RecyclerView 的适配器。首先使用 data 存储商品数据，定义构造函数和接口函数，然后创建 Item 视图，并返回相应的 ViewHolder。

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

这里我是自定义了一个 ViewHolder 来为商品列表服务，由文档要求中主页的商品列表展示效果可知，该视图主要包含两个 TextView 对象，一个用于显示首字母，一个用于显示商品名称。通过 findViewById 实现与布局文件中的控件绑定。

```
class MyViewHolder extends RecyclerView.ViewHolder {
     TextView name;
     TextView firstletter;

     public MyViewHolder(View view) {
         super(view);
         name = (TextView) view.findViewById(R.id.name);
         firstletter = (TextView) view.findViewById(R.id.firstletter);
     }
 }

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

在创建了视图后，接下来的任务就是绑定数据到正确的 item 视图上。RecyclerView 会循环执行绑定函数，根据商品列表中的数据变动不断更新视图内容，故我们直接在每一次的调用时对控件进行赋值即可。利用 position 获得数据在 list 中所处的位置，substring(0,1)获取首字母。由于 RecyclerView 中没有定义 Click 触发事件，故在对两个 TextView 赋值后我们还要重定义点击函数，让他们在被点击时返回数据在 list 中的位置，同时长按时有返回值 false。

```
@Override
public void onBindViewHolder(final MyViewHolder holder, final int position) {
    holder.name.setText(data.get(position).getName());
    holder.firstletter.setText(data.get(position).getName().substring(0,1).toUpperCase());

    if(mOnItemClickListener != null) {
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            /* 获取Item位置 */
            public void onClick(View v) {
                mOnItemClickListener.onClick(position);
            }
        });
        holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                mOnItemClickListener.onLongClick(position);
                return false;
            }
        });
    }
}
```

最后，我们还需要重写 getItemCount 方法来告诉 Recyclerview.Adapter 列表 Items 的总数。

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

回到主页表文件，使用四个 String 数组定义每个商品的名字、价格、类型和产品信息，然后用一个 for 循环将它们添加到数据链表中。然后将 mRecyclerView 和布局文件中的 shop 绑定，再为其添加适配器 homeAdapter。

```
final String[] Name = new String[]{"Enchated Forest", "Arla Milk", "Devondale Milk",
                                   "Kindle Oasis", "waitrose 早餐麦片", "Mcvitie's 饼干",
                                   "Ferrero Rocher", "Maltesers", "Lindt", "Borggreve"};
final String[] Price = new String[]{"¥ 5.00", "¥ 59.00", "¥ 79.00", "¥ 2399.00", "¥ 179.00",
                                    "¥ 14.00", "¥ 132.59", "¥ 141.43", "139.43", "28.90"};
final String[] Type = new String[]{"作者", "产地", "产地", "版本", "重量",
                                   "产地", "重量", "重量", "重量", "重量"};
final String[] Info = new String[]{"Johanna Basford", "德国", "澳大利亚", "8GB", "2Kg",
                                   "英国", "300g", "118g", "249g", "640g"};
```

```
mRecyclerView = (RecyclerView) findViewById(R.id.shop);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
homeAdapter = new HomeAdapter(list, Ex3.this);
mRecyclerView.setAdapter(homeAdapter);
```

当某个商品被点击时，需要从主页面跳转到商品详情界面，这里使用 intent 进行不同界面间的消息传递，利用 putExtra 函数将需要传递的值以自定义的名字（如 Name）传递给下一个进程，我们利用此方法依次传递商品名、价格、类型和信息。startActivityForResult 是启动一个 activity，并再其结束后接收一个返回值，这里的 1 并没有实际的数值意义，只是用来比较和返回值是否相同。这里我使用 1 表示没有向购物车添加物品时的返回值，0 表示需要更新购物车列表时的返回值，具体在 [商品详情] 部分会进一步说明。

```
Intent intent = new Intent(Ex3.this, detail.class);
intent.putExtra("Name", list.get(position).getName());
intent.putExtra("Price", list.get(position).getPrice());
intent.putExtra("Type", list.get(position).getType());
intent.putExtra("Info", list.get(position).getInfo());
startActivityForResult(intent,1);
```

在长按某商品时，使用 Toast 弹出提示消息，利用重写的 onLongClick 函数返回的 position 值获取需要删除的商品在 list 中的位置，使用 remove 方法从 list 中移除该项，再调用 notifyDataSetChange 函数更新整个 Recyclerview。

```
public void onLongClick(int position) {
    Toast.makeText(Ex3.this,"移除第" + String.valueOf(position+1) + "个商品",Toast.LENGTH_SHORT)
                .show();
    list.remove(position);
    homeAdapter.notifyDataSetChanged();
}
```

3) 悬浮按钮部分

在主页面和购物车页面的右下角有一个悬浮按钮，其功能是可以实现两个界面的显示切换，同时按钮图标也要实现购物车和主页图标的切换。这里我设置了一个 Tag 用来控制状态转换，当 Tag 为 0 时，表示此时是购物车界面，要切换到主页界面，故设置图标为主页图标，同时设置主页部分的布局为可见，购物车部分的布局为不可见，并更新 Tag 为 1；当 Tag 为 1 时同理。

补充说明：刚开始的时候我使用了两个布局文件和两个 class 来分别实现主页和购物车界面，但是在后面的实验过程中，我发现这样在返回购物车时会建立一个新的 activity，不仅无法回到同一个购物车，而且多次运行后还会造成内存爆炸。于是在同学的建议下，我对这两个部分进行了整合，只使用一个布局文件，通过设置部分布局不可见的方式实现界面切换。

```
/* 悬浮按键 */
final ImageButton f = (ImageButton) findViewById(R.id.floating);
f.setTag("0");
f.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (f.getTag() == "0") {
            f.setTag("1");
            f.setImageResource(R.mipmap.home);
            shoplist.setVisibility(View.VISIBLE);
            mRecyclerView.setVisibility(View.INVISIBLE);
        } else {
            f.setTag("0");
            f.setImageResource(R.mipmap.car);
            shoplist.setVisibility(View.INVISIBLE);
            mRecyclerView.setVisibility(View.VISIBLE);
        }
    }
});
```

4) 自定义 ListView 适配器

重写 ListView 适配器的过程与 Recyclerview 类似，而且因为 ListView 本身功能就很强大，所以工作量还要更少一点（比如不用自定义 Click 事件触发函数）。首先还是定义一个 list 来表示购物车内的商品链表，然后定义构造函数，重写 getCount、getItem、getItemId 等函数获取列表项的相关内容。

不过在购物车界面，最重要的还是视图加载的部分。若当前 view 为空，则要在新布局中加载数据模块，利用 MyViewHolder 找到布局中首字母、名字和价格的控件，并将它与当前的 view 绑定起来，实现界面的实现；若 view 已经存在，说明购物车界面在之前已经被建立了，这时候只需要把已经设置好的 view 加载出来即可。最后更新 holder 中每个控件的数据值。

```
@Override
public View getView(int position, View view, ViewGroup viewGroup) {
    View convertView;
    MyViewHolder holder;
    if (view == null) {
        convertView = LayoutInflater.from(context).inflate(R.layout.shoplistdetail, null);
        holder = new MyViewHolder();
        holder.firstletter = (TextView) convertView.findViewById(R.id.icon);
        holder.name = (TextView) convertView.findViewById(R.id.iname);
        holder.price = (TextView) convertView.findViewById(R.id.iprice);
        convertView.setTag(holder);
    } else {
        convertView = view;
        holder = (MyViewHolder) convertView.getTag();
    }
    holder.firstletter.setText(carthings.get(position).getName()
                               .substring(0,1).toUpperCase());
    holder.name.setText(carthings.get(position).getName());
    holder.price.setText(carthings.get(position).getPrice());
    return convertView;
}
```

5) 购物车界面

与主页界面类似，首先将购物车部分的布局设置为可见，然后将购物车列表与布局文件中的 ListView 控件绑定，并为它设置适配器 shopAdapter。

```
final LinearLayout shoplist = (LinearLayout) findViewById(R.id.shoplist);
shoplist.setVisibility(View.INVISIBLE);

mListView = (ListView) findViewById(R.id.caritem);
shopAdapter = new ShopAdapter(carlist, Ex3.this);
mListView.setAdapter(shopAdapter);
```

当某个商品被点击时，与主页部分代码相同，利用 intent 进行数据传递，实现从购物车界面到商品详情界面的跳转。（此处不再重复贴码）当长按购物车中商品时，利用对话框 AlertDialog 弹出一个消息提示，若确定移除该商品，则从 carlist 中删除 position 位置处的数据，并弹出成功移除的提示信息；否则弹出 Toast 提示信息，对话框消失。

```
final AlertDialog.Builder alertDialog = new AlertDialog.Builder(Ex3.this);
alertDialog.setTitle("移除商品").setMessage("从购物车移除"+carlist.get(position).getName()+"?")
            .setPositiveButton("确定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int which) {
        if (shopAdapter.getCount() != 0 ) {
            carlist.remove(position);
            shopAdapter.notifyDataSetChanged();
            Toast.makeText(getApplicationContext(), "成功移除!", Toast.LENGTH_SHORT).show();
        }
    }
}).setNegativeButton("取消", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        Toast.makeText(getApplicationContext(), "你选择了[取消]", Toast.LENGTH_SHORT).show();
    }
}).show();
```

6) 商品详情界面

商品详情界面是一个新的 activity。我新定义了一个布局文件，一个 detail 文件和一个针对商品详细信息的适配器。

6.1 布局文件

我使用的主体框架是一个线性布局嵌套两个相对布局，根据实验文档要求，利用 weight 设置上面的图片部分占据整个界面的 1/3，返回按钮、图片、Name 和星星处于一个相对布局中，利用 alignParent...实现与父组件之间的相对位置关系。然后下方的两行属于第二个相对布局，利用 alignTop 和 alignButtom 实现购物车图标与左边的商品信息对齐。添加一个 TextView 控件，让其宽度与父组件对齐，并且设置背景为浅灰色，实现分隔线。最后利用 ListView 实现底部的四行显示。

6.2 自定义适配器

与之前类似，新定义一个 DetailAdapter 实现数据绑定。首先编写 onCreateViewHolder 函数，创建 Item 视图，将 view 与布局文件进行绑定，并返回相应的 ViewHolder。

```
public DetailAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(context).inflate(R.layout.detail, parent, false);
    DetailAdapter.MyViewHolder holder = new DetailAdapter.MyViewHolder(view);
    return holder;
}
```

重写 getItemCount 函数：

```
@Override
public int getItemCount() { return 1; }
```

之后和 mRecyclerView 类似，我们需要自定义一个 ViewHolder 来为商品信息服务。由文档要求中主页的商品列表展示效果可知，该视图主要包含五个商品对象，分别是商品的图片、名字、价格、类型和产品信息，还有返回、收藏、购物车三个按键对象和一个 listView 对象。分别通过 findViewById 实现这些变量与布局文件中的对应控件绑定。

```
class MyViewHolder extends RecyclerView.ViewHolder {
    TextView tv1, tv2, tv3 ,tv4;
    ImageView img;
    ImageButton back, star, addcar;
    ListView listView;

    public MyViewHolder(View view) {
        super(view);
        tv1 = (TextView) view.findViewById(R.id.d_name);
        tv2 = (TextView) view.findViewById(R.id.d_price);
        tv3 = (TextView) view.findViewById(R.id.d_type);
        tv4 = (TextView) view.findViewById(R.id.d_info);
        img = (ImageView) view.findViewById(R.id.pic);

        back = (ImageButton) view.findViewById(R.id.back);
        star = (ImageButton) view.findViewById(R.id.star);
        star.setTag("0");  // 初始为empty_star
        addcar = (ImageButton) view.findViewById(R.id.d_car);

        listView = (ListView) view.findViewById(R.id.message);
    }
}
```

然后就是最重要的部分了：绑定数据到正确的 item 视图上。首先接收传递进来的商品信息，并对对应控件赋值。

```
final Intent intent = act.getIntent();
act.setResult(1,intent);

holder.tv1.setText(Name);
holder.tv2.setText(Price);
holder.tv3.setText(Type);
holder.tv4.setText(Info);
```

根据 Name 进行图片加载。利用 if..else 语句进行选择，给 img 加载对应的产品图片。

```
// 加载每个商品对应的图片
if (Name.equals("Enchated Forest")) holder.img.setImageResource(R.drawable.pic1);
else if (Name.equals("Arla Milk")) holder.img.setImageResource(R.drawable.pic2);
else if (Name.equals("Devondale Milk")) holder.img.setImageResource(R.drawable.pic3);
else if (Name.equals("Kindle Oasis")) holder.img.setImageResource(R.drawable.pic4);
else if (Name.equals("waitrose 早餐麦片")) holder.img.setImageResource(R.drawable.pic5);
else if (Name.equals("Mcvitie's 饼干")) holder.img.setImageResource(R.drawable.pic6);
else if (Name.equals("Ferrero Rocher")) holder.img.setImageResource(R.drawable.pic7);
else if (Name.equals("Maltesers")) holder.img.setImageResource(R.drawable.pic8);
else if (Name.equals("Lindt")) holder.img.setImageResource(R.drawable.pic9);
else if (Name.equals("Borggreve")) holder.img.setImageResource(R.drawable.pic10);
```

当点击了购物车按钮时，利用 Toast 弹出提示信息，同时利用 intent 将要添加商品数目、名字、价格、类型和信息传递给原始调用 detail 这个 activity 的活动（商品列表&购物车页面），利用 setResult(0,intent)中的 0 表示购物车中的内容发生了改变，在主页面 onActivityResult 函数对返回值进行接收。

```
// 添加商品到购物车
holder.addcar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast.makeText(context,"商品已添加到购物车",Toast.LENGTH_SHORT).show();
        cnt++;

        intent.putExtra("cnt",cnt);
        intent.putExtra("name",Name);
        intent.putExtra("price",Price);
        intent.putExtra("type",Type);
        intent.putExtra("info",Info);
        // 回调
        act.setResult(0,intent);
    }
});
```

（主页面）当接收到的返回值为 0 时，使用 bundle 接收传回的数据，利用一个 for 循环将数据依次添加到购物车列表中。

```
// 向购物车中添加商品
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    if(resultCode == 0) {
        Bundle bud = intent.getExtras();
        String name = bud.getString("name");
        String price = bud.getString("price");
        String type = bud.getString("type");
        String info = bud.getString("info");
        int cnt = bud.getInt("cnt",0);
        for(int i = 0; i < cnt; i++){
            carlist.add(new Data(name, price, type, info));
        }
        shopAdapter.notifyDataSetChanged();
    }
}
```

返回实现部分：当返回图标被点击时，调用 finish 函数来结束这个 activity，同时返回上一级活动。

```
// 返回
holder.back.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        act.finish();
    }
});
```

星星转换实现部分：利用 tag 实现实心和空心星星的切换，tag 初始化为 0，当 tag 为 0 时，表示要从空心切换为实心，更新 tag 的值为 1，利用 setImageResource 更改图片，同时弹出提示信息；tag 为 1 时同理。

```
// 星星转换
holder.star.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Object tag = holder.star.getTag();
        if(tag == "0") {
            holder.star.setTag("1");
            holder.star.setImageResource(R.mipmap.full_star);
            Toast.makeText(context, "收藏成功!", Toast.LENGTH_SHORT).show();
        }
        else {
            holder.star.setTag("0");
            holder.star.setImageResource(R.mipmap.empty_star);
            Toast.makeText(context, "你取消了收藏", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Listview 实现部分：利用一个字符串数组 message 定义底部的四个选项，然后使用 listView 自带的 SimpleAdapter 实现数据绑定。

```
// 下拉列表
final String[] message = new String[] {"一键下单","分享商品","不感兴趣","查看更多商品促销信息"};
final List<Map<String, Object> > data = new ArrayList<>();
for(int i = 0; i < 4; i++) {
    Map<String, Object> tmp = new LinkedHashMap<>();
    tmp.put("message", message[i]);
    data.add(tmp);
}
String[] from = new String[] {"message"};
int[] to = new int[] {R.id.message};
final SimpleAdapter simpleAdapter = new SimpleAdapter(context, data, R.layout.item2, from, to);
holder.listView.setAdapter(simpleAdapter);
```

6.3 主文件

定义好了适配器以后，主文件的实现就变得很简单了。为 RecyclerView 定义一个新的布局文件 rv，将其与 detail 绑定，然后利用 intent 接收从主页面或购物车页面传来的信息，获取被选择商品的名字、价格、类型和产品信息。使用这些信息初始化一个 adapter，然后通过 setAdapter 方法实现数据绑定。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.rv);

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

    // 接收信息
    Intent intent = getIntent();
    final String Name = intent.getStringExtra("Name");
    final String Price = intent.getStringExtra("Price");
    final String Type = intent.getStringExtra("Type");
    final String Info = intent.getStringExtra("Info");

    DetailAdapter detailAdapter = new DetailAdapter(Name, Price, Type, Info,
                                                    detail.this, detail.this);
    mRecyclerView.setAdapter(detailAdapter);
}
```

最后，为了让其可以正常运行，要在 manifests 的 AndroidManifest.xml 中对其进行注册，新添加一行该 class 的名字即可。

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

7) 去掉标题栏

在 style.xml 中把 DarkActionBar 改为 NoActionBar 即可。

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

（3） 实验遇到困难以及解决思路

问题 1： App 无法运行，一打开就闪退。

解决方法： 这是因为我一开始忘记对设置好的 Recyclerview 和 adapter 进行初始化和赋值操作，导致程序无法运行。补上了初始化后就一切正常了。

问题 2： 跳转到商品详情页面的时候会崩溃。

解决方法： 没有对商品详情的 activity 进行注册，在 manifests 中给该 activity 注册即可。

问题 3： 在商品详情界面点击返回时程序会崩溃。

解决方法： 在启动商品详情的 activity 时，设置的启动方式是 startActivityForResult，因此它需要接收一个返回值，所以我在 DetailAdapter 文件中增加了一句 act.setResult(1,intent);设置默认返回值为 1 后再点击返回就没事了。

问题 4： 在商品详情界面点击购物车按钮后返回，查看购物车，发现除了第一次之外，后面每次添加到购物车都会一次性添加多个商品。

解决方法： 检查代码后我发现是自己不小心给统计变量 cnt 设置为了 static，它和 C++里的 static 类似，属于全局变量，故每次不会自动清零。设置为 private 之后就好了。

问题 5： 第一次导入图片时报错说命名不合法。

解决方法： 图片文件的名字里不可以有大写字母，故改为全部小写时就没事了。

问题 6： 商品图片无法与 view 等高，总会与 view 的上下边框有一定的距离。

解决方法： 将图片由导入到 mipmap 文件夹，改为导入到 drawable 文件夹即可。这个查了百度后好像是因为 mipmap 文件夹一般都是用来设置图标等图片，会自动增加一圈透明的边框。

问题 7： 最后一个是困扰自己时间最长且最简单的问题，就是我的“back”和“star”图像显示不出来，都是一个正方形的白色块。

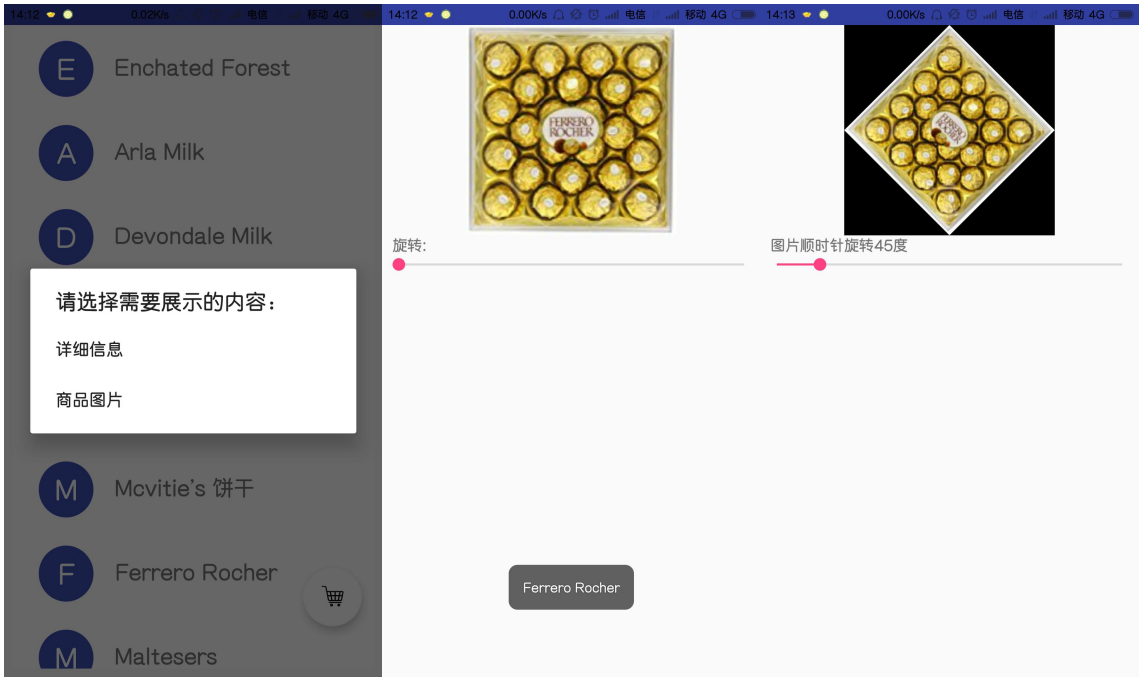
解决方法： 插入图片文件的时候，我选的 squre 会让图片自动背景成方形的白色背景，又因为“back”“star”这些图片本来就是白的，所以会看不见，只显示一个正方形的白。改为“none”时，图片的背景就是透明的，就可以正常显示了。

五、 课后实验结果

（1） 实验内容

① 自定义实现 ListView 的 Adapter
② 增加了商品图片旋转功能
③ 给 App 换了自定义图标

（2） 实验截图



结果分析：使用自定义的 shopAdapter 替代系统的 simpleAdapter，修改了 App 图标。同时在原来的功能基础上，增加了商品图片旋转界面，通过对话框提示进行功能选择，当选择的是[商品图片]时弹出新界面并弹出商品名字，然后通过滑动条来控制图片顺时针旋转角度。

（3） 实验步骤以及关键代码

1) 自定义实现 ListView 的 Adapter

在商品详情界面的下方有我最初用 SimpleAdapter 写的 listView，用来实现底部信息显示，在后期我使用自定义的 listViewAdapter 对其进行了进一步优化。代码部分和基础部分中的相同，故不再重复贴码。

2) 旋转商品图标界面

和 detail 类似，定义一个新的布局文件和一个新的 activity（也要注册）。通过矩阵类实现旋转操作。利用 intent 接收被选中的商品的名字，然后根据商品名给 bitmap 加载每个商品的对应图片，然后利用 createBitmap 函数在原图像区域建立一个旋转了一定角度的新图像。

对主页面的修改：在点击 item 部分，增加一个对话框，若选择了 [详细信息]，则和基础实验内容一样，显示商品的详细信息；若选择了 [商品图片]，则跳转到旋转界面，通过滑动条控制旋转角度，同时实时展示图像旋转了该角度后的样子。

```
public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
    // 初始化
    Bitmap bitmap = ((BitmapDrawable)
                    (getResources().getDrawable(R.drawable.pic1))).getBitmap();

    if (Name.equals("Enchated Forest")) bitmap = ((BitmapDrawable)
                (getResources().getDrawable(R.drawable.pic1))).getBitmap();
    else if (Name.equals("Arla Milk")) bitmap = ((BitmapDrawable
                        (getResources().getDrawable(R.drawable.pic2))).getBitmap();
    else if (Name.equals("Devondale Milk")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic3))).getBitmap();
    else if (Name.equals("Kindle Oasis")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic4))).getBitmap();
    else if (Name.equals("waitrose 早餐麦片")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic5))).getBitmap();
    else if (Name.equals("Mcvitie's 饼干")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic6))).getBitmap();
    else if (Name.equals("Ferrero Rocher")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic7))).getBitmap();
    else if (Name.equals("Maltesers")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic8))).getBitmap();
    else if (Name.equals("Lindt")) bitmap = ((BitmapDrawable)
                         (getResources().getDrawable(R.drawable.pic9))).getBitmap();
    else if (Name.equals("Borggreve")) bitmap = ((BitmapDrawable)
                        (getResources().getDrawable(R.drawable.pic10))).getBitmap();
```

旋转部分代码：利用矩阵类实现以原图像为中心的顺时针旋转，在旋转过程中对图像进行适当的缩放。旋转角度通过进度条的 progress 控制。

```
matrix.setRotate(progress);
bitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(), matrix, true);
imageView.setImageBitmap(bitmap);
textView.setText("图片顺时针旋转"+progress+"度");
```

3) 自定义 App 图标

更改 mipmap 中的 ic_launcher 图标为自己自定义的图片即可。

（4） 实验遇到困难以及解决思路

问题 1：从商品列表页面跳转到图片旋转界面时，发现传递过去的 Name 没有被接收，同时程序会发生闪退的情况。

解决方法：发现是消息接收部分代码写错了地方，写回 onCreate 函数后一切就正常了。

问题 2：在给 Bitmap 进行 if..else 赋值时，报错该变量未进行初始化。

解决方法：随便选一个图片作为 Bitmap 的初始化图片即可。

六、 实验思考及感想

本次实验让我深刻体会到了安卓开发的艰难，从布局到 java 文件再到数据传递、界面跳转，每一个环节都花费了我大量的时间和精力。

首先是 RecyclerView 的布局，这是我第一次写完布局文件后无法直接看到界面显示效果，心里不免虚虚的，不过在成功运行代码之后，就可以检查自己的布局是否合理了。在经过第一次实验 ConstrictLayout 的洗礼后，首次使用 RelativeLayout 的我深深感受到了它的友好。它既有基本属

性 gravity，也有如 layout_alignParentLeft 这种根据父容器定位的属性，自然也有如 layout_toLeftOf 这种根据兄弟组件定义的属性，也有着像 LinearLayout 那些其他布局都有的 margin 和 padding 属性，使 RelativeLayout 布局变得方便无比。

本次实验的一大难点是要自定义 Recyclerview 和 Listview 的适配器，刚开始的时候完全不知道该从哪入手，看完实验文档后依然是一头雾水，后来在将近一天的资料查询和同学的帮助下，我才艰难实现了 homeAdapter 的构建。后来我发现 java 的类其实和 C++一模一样，都是继承一个基础类然后新定义一些变量，再重写一些方法，可能真正难住我的是对它所继承的基础类的不了解吧，因为不知道哪些函数有，哪些函数没有，又没有深入理解各个函数的功能和意义，所以才会不知所措、无从下手。

虽然实现的过程是痛苦的，但是不可否认，自定义了 adapter 之后使用确实方便了很多，以 listView 为例，因为它自带的 SimpleAdapter 功能有限，所以在这次实验中，比如要设计针对 Data 类的适配器时就会非常复杂，要分四个 String 进行数据绑定，而自定义了以后就可以直接对类对象进行操作，更好地实现了封装，也方便了我们的使用。

最后，本次实验还让我懂得了一些相似事物的区别，比如 bundle 和 intent，一个偏向数据存储，一个偏向消息传递；再比如 mipmap 和 drawable，以前我一直觉得这两个文件夹用来存放图片时没有什么太大的区别，但是这次实验被坑之后我才终于明白为什么 TA 在第一次实验课的时候就说图片最好存放在 drawable 文件夹下……总之，这次实验虽然充满艰辛，却让我收获颇丰，希望在下一次实验中自己可以做得更好。