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

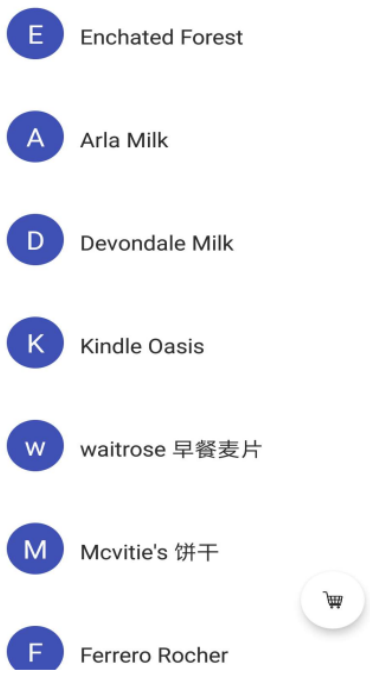
姓名	学号	班级	电话	邮箱
张子豪	15352427	15M1	15989046143	ahzzh1998@163.com

1.实验题目

Intent、Bundle的使用以及RecyclerView、ListView的应用

2.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：

上面两项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，返回图标位于这块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。这一块建议使用RelativeLayout实现，以便熟悉RelativeLayout的使用。中部底部样式都很清楚，没什么可说的。

4、特别提醒，这次的两个页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“。

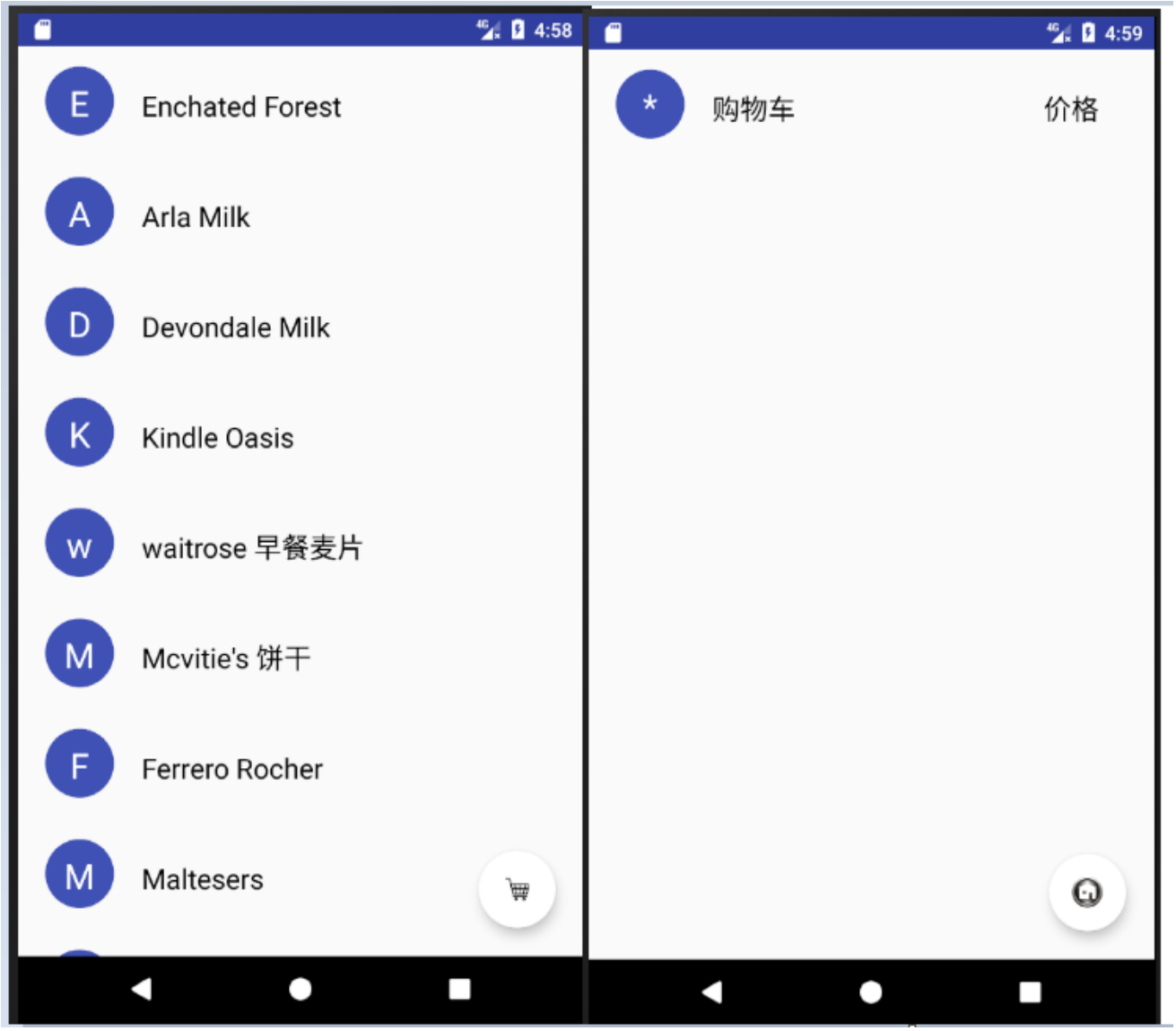
2、点击右下方的FloatingActionButton，从商品列表切换到购物车或从购物车切换到商品列表，并且FloatingActionButton的图片要做相应改变。

3、使用ListView实现购物车。点击购物车的某一商品会跳转到商品详情页面，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会返回上一层，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”。

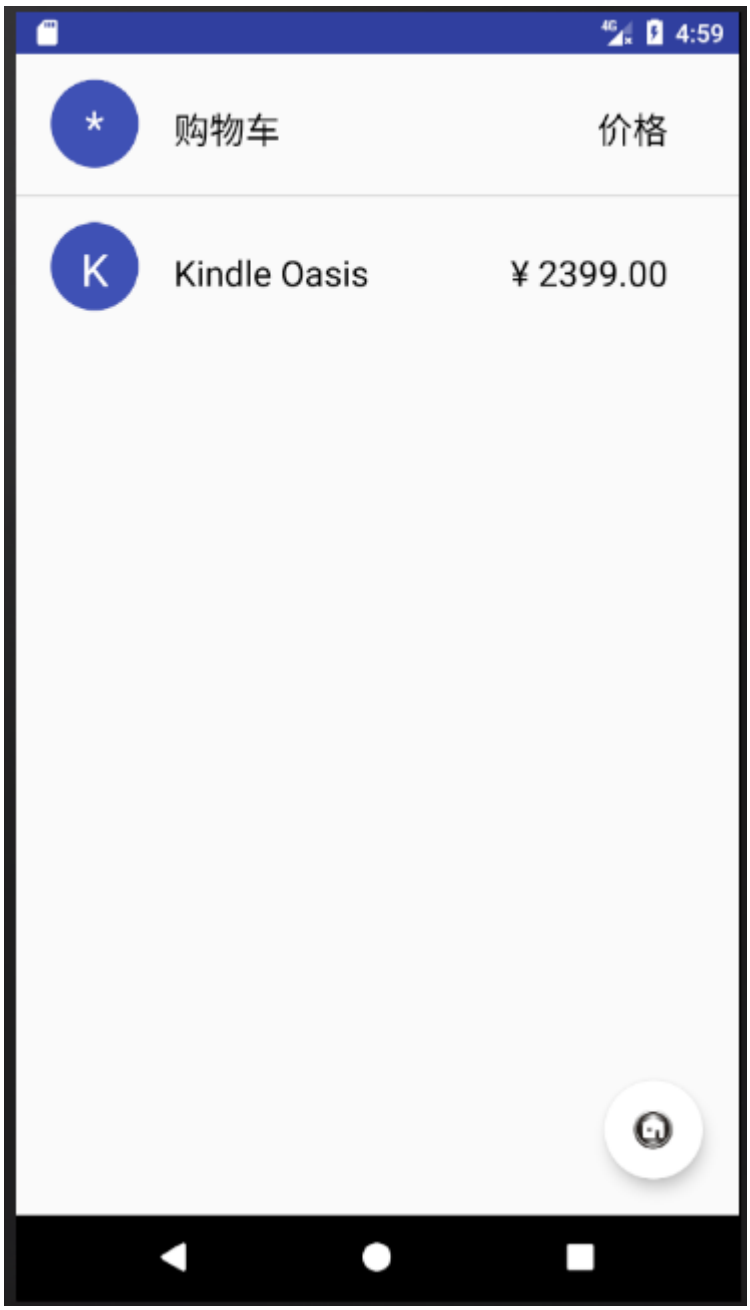
3.课堂实验结果

- 实验截图

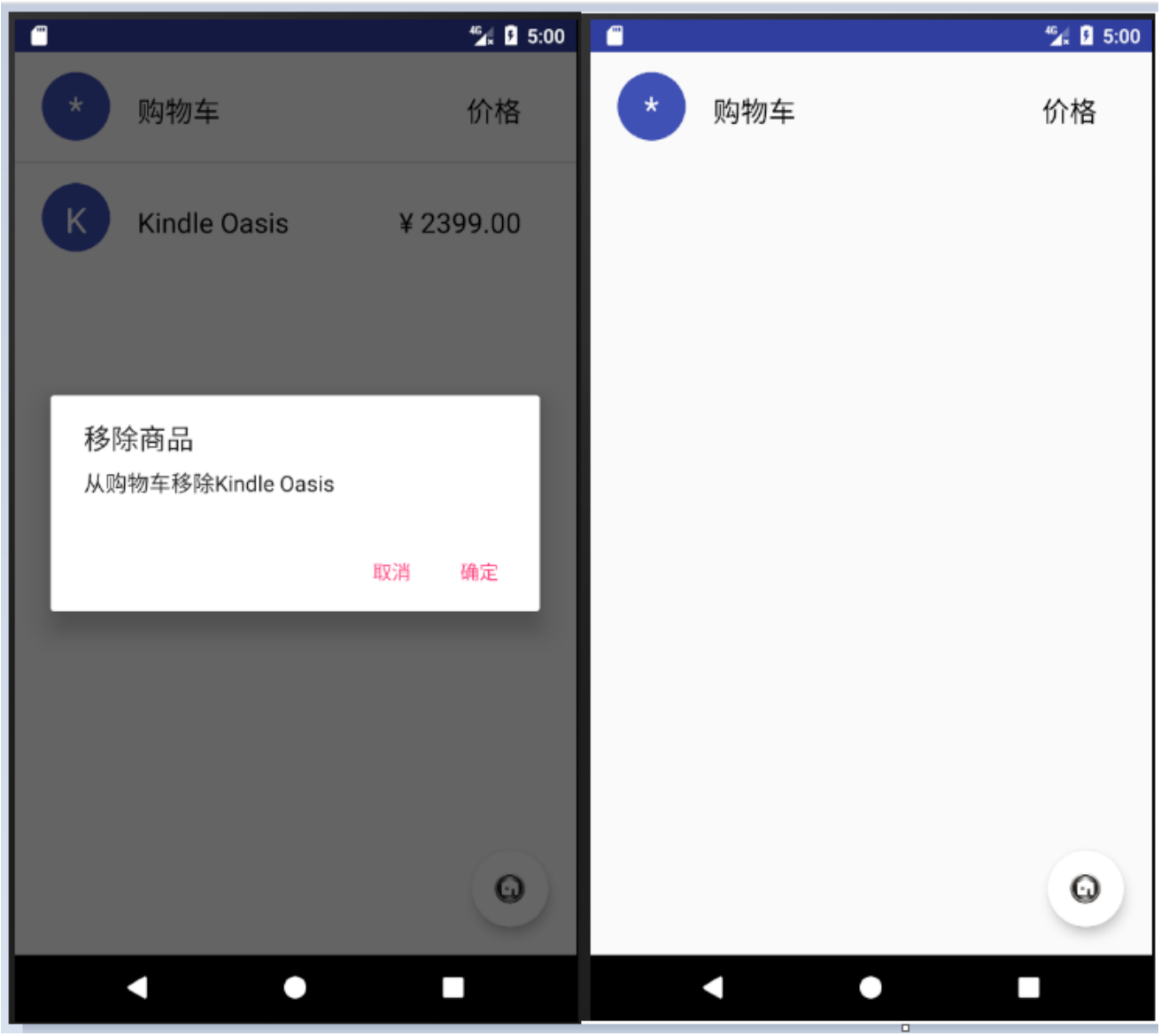


商品列表、购物车如上所示

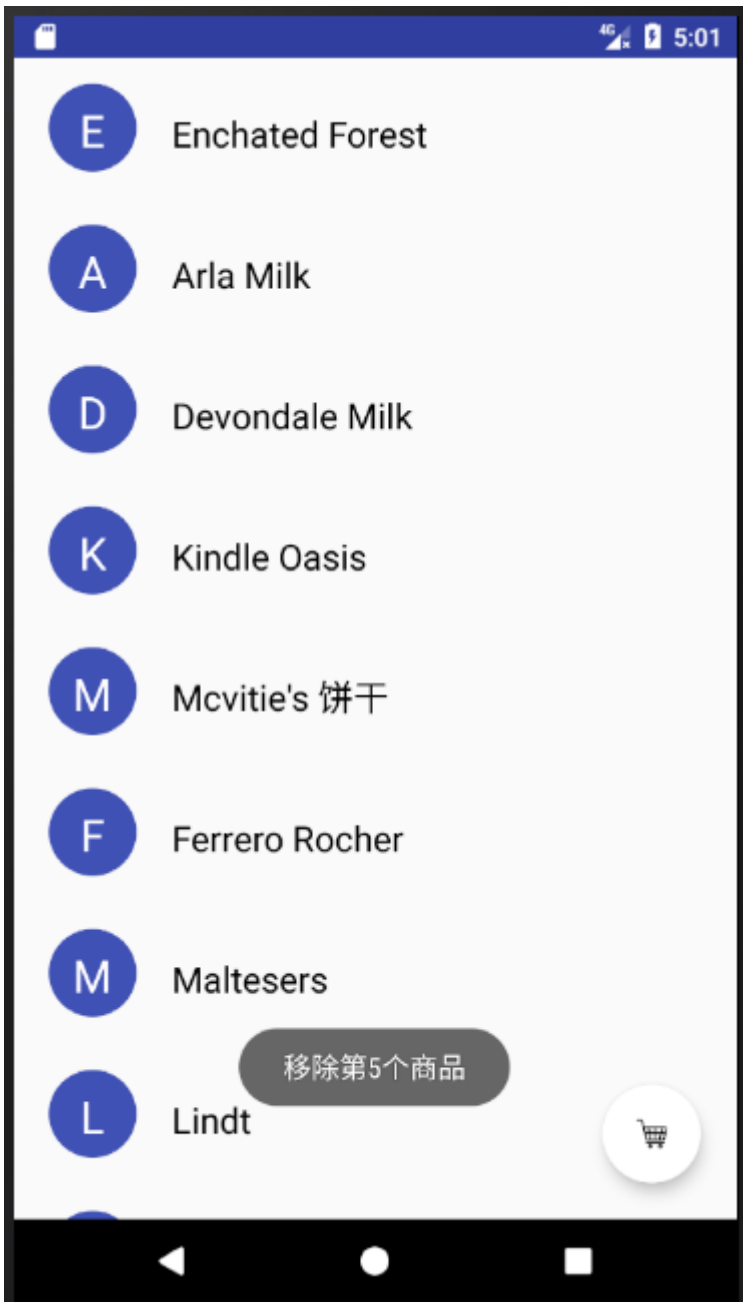
点击任意一项进入商品详情，切换星标状态，如上所示



点击添加到购物车后访问购物车，如上所示



长按删除并确定，如上所示



商品列表界面长按移除商品，如上所示

- 实验步骤以及关键代码

  构建商品列表与购物车的布局，这里以商品列表为例

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

<android.support.design.widget.FloatingActionButton
        android:id="@+id/shoplist"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:src="@mipmap/shoplist"
        app:backgroundTint="@color/white"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="20dp"
        android:layout_alignParentRight="true"
        android:layout_marginRight="20dp"/>
```

使用RecyclerView需要自定义数据适配器，代码如下

```
public class RecyclerAdapter extends RecyclerView.Adapter<RecyclerAdapter.mViewHolder>{

    private Context mContext;
    private List<Map<String,String>> mdata;
    private OnItemClickListener mOnItemClickListener;

    public void DeleteData(int position){//删除数据
        mdata.remove(position);
    }
    public interface OnItemClickListener{//注册点击监听
        void OnClick(View v,int position);
        void OnLongClick(View v,int position);
    }
    public void setItemClickListener(OnItemClickListener listener){//构造函数
        mOnItemClickListener=listener;
    }
    public RecyclerAdapter(Context  context,List<Map<String,String>> datas){//构造函数
        this.mContext=context;
        this.mdata=datas;
    }
    @Override
    public mViewHolder onCreateViewHolder(ViewGroup parent,int viewType){//创建ViewHolder
        View v=LayoutInflater.from(mContext).inflate(R.layout.recycler_item,parent,false);
        mViewHolder holder=new mViewHolder(v,mOnItemClickListener);
        return holder;
    }
    @Override
    public void onBindViewHolder(final mViewHolder holder, final int position){
        //将ViewHolder与数据绑定
        holder.icon.setText(mdata.get(position).get("icon"));
        holder.text.setText(mdata.get(position).get("name"));
    }
    @Override
    public int getItemCount(){ return mdata.size();}

    public class mViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener,View.OnLongClickListener{//自定义ViewHolder

        TextView icon;
        TextView text;
        OnItemClickListener mListener;

        public mViewHolder(View v,OnItemClickListener tmp) {
            super(v);
            icon=(TextView)v.findViewById(R.id.icon);
            text=(TextView)v.findViewById(R.id.text);
            this.mListener=tmp;
            v.setOnClickListener(this);
            v.setOnLongClickListener(this);
        }
        @Override
        public void onClick(View v) {

            if(mListener!=null){
```

```
                mListener.OnClick(v,getPosition());
            }
        }
        @Override
        public boolean onLongClick(View v) {
            if(mListener!=null){
                mListener.OnLongClick(v,getPosition());
            }
            return true;
        }
    }
}
```

实现RecyclerView，使用RecyclerView需要使用setLayoutManager定义布局，如纵向排列，九宫格等；setAdapter使用数据适配器，将数据与item_view绑定，最后放入RecyclerView。

```
RecyclerView mRecyclerView=(RecyclerView) findViewById(R.id.RecyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
final RecyclerAdapter recyclerAdapter= new RecyclerAdapter(this, listems);
recyclerAdapter.setItemClickListener(new RecyclerAdapter.OnItemClickListener() {
    @Override
    public void OnClick(View v, int position) {
            Intent intent=new Intent("android.intent.action.ProductActivity");
            Bundle bundle=new Bundle();
            bundle.putInt("key",position);
            intent.putExtras(bundle);
            startActivity(intent);
    }
    @Override
    public void OnLongClick(View v,int position){
        recyclerAdapter.notifyItemRemoved(position);
        recyclerAdapter.DeleteData(position);
        Toast.makeText(MainActivity.this,"移除第"+(position+1)+"个商
品",Toast.LENGTH_SHORT).show();
    }
});
mRecyclerView.setAdapter(recyclerAdapter);
```

购物车页面需要用ListView实现，由于ListView可以使用Android自带的数据适配器，因此操作起来方便很多，SimpleAdapter的五个参数分别为

整个android应用程序接口

生成一个Map(String ,Object)列表选项

界面布局的id 表示该文件作为列表项的组件

该Map对象的哪些key对应value来生成列表项

来填充的组件Map对象key对应的资源一依次填充组件，顺序有对应关系

```
ListView listView=(ListView)findViewById(R.id.shopcar);
simpleAdapter=new SimpleAdapter(this,list,R.layout.list_item,new String[]
{"icon","name","price"},new int[]{R.id.icon,R.id.text,R.id.price});
listView.setAdapter(simpleAdapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if(position!=0){
            Intent intent=new Intent("android.intent.action.ProductActivity");
            Bundle bundle=new Bundle();
            bundle.putInt("key",m.get(list.get(position).get("name")));
            intent.putExtras(bundle);
            startActivity(intent);
        }
    }
});
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, final int position,
long id) {
        if(position!=0){
            AlertDialog.Builder talk1=new AlertDialog.Builder(MainActivity.this);
            talk1.setTitle("移除商品");
            talk1.setMessage("从购物车移除"+name[m.get(list.get(position).get("name"))]);
            talk1.setNegativeButton("取消", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                }
            });
            talk1.setPositiveButton("确定", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    list.remove(position);
                    simpleAdapter.notifyDataSetChanged();//更新数据适配器
                }
            });
            talk1.show();
        }
        return true;
    }
});
```

使用FloatingButton切换显示的View，可以通过setVisibility函数来实现，以从购物车切换回商品列表为例，GONE代表这个View不可见也不占用体积，VISIBLE代表可见。

```
findViewById(R.id.shopcar).setVisibility(View.GONE);//设置购物车不可见
findViewById(R.id.mainpage).setVisibility(View.GONE);//设置mainpage不可见
findViewById(R.id.RecyclerView).setVisibility(View.VISIBLE);//设置商品列表可见
findViewById(R.id.shoplist).setVisibility(View.VISIBLE);//设置购物车小图标可见
```

商品详情：

布局使用RelativeLayout，RelativeLayout使用起来大体与约束布局相同，使用layout_alignParent可以选定该view是否与父容器的某一方向对齐，layout_align可以选定该view的start是否与另外一个view的top/bottom/left/right对齐，margin调节间距，这些部分基本足以满足这次实验要求。

在商品详情中需要读取MainActivity传过来的Intent来确定是哪个商品被点击，并为对应的view赋值

```
Bundle bundle=this.getIntent().getExtras();
int position=bundle.getInt("key");
pos=position;//获得被点击商品的下标
ImageView imageView=(ImageView)findViewById(R.id.product_img);
imageView.setImageDrawable(getResources().getDrawable(img[position]));//图片赋值
TextView product_name=(TextView)findViewById(R.id.product_name);
product_name.setText(name[position]);
TextView product_price=(TextView)findViewById(R.id.price);
product_price.setText(price[position]);
TextView product_type=(TextView)findViewById(R.id.type);
product_type.setText(type[position]);
TextView product_information=(TextView)findViewById(R.id.information);
product_information.setText(information[position]);
```

对于星星图标的点击事件，可以用一个int类型的flag来记录，初始化为1，每次点击乘-1，这样flag就在1与-1这两个状态之间变换，对应实心与空心。

- 实验遇到困难以及解决思路
    - 写RecyclerVIew的时候对数据适配器理解的不是很清楚，照着pdf写出了很多问题，理解了一下适配器的原理后重写了一份代码，这次实验ViewHolder只需要绑定两个TextView，所以可以在自定义的ViewHolder类中放两个TextView成员，绑定数据直接使用setText即可。
    - ListView更新数据过程中，没有使用notifyDataSetChanged更新数据适配器，导致程序崩溃、错误的点击事情等问题。
    - 使用广播机制时静态注册receiver无法接收到Intent，最后使用了动态注册。

4.课后实验结果

广播机制：

在商品详情这个Activity(ProductActivity)中点击购物车图片后会将这个商品添加到购物车中，这就涉及到两个Activity之间的信息交互，使用广播机制就可以解决这个问题。在ProductActivity中监听点击事件，点击事件发生后new一个Intent，将含有待传输数据的Boundle放入这个Intent，然后使用sendBroadcast函数发送出去。在onCreate中使用registerReceiver动态注册监听事件，onReceive中处理接受到的Intent，onDestroy中取消注册。

```
Intent intent=new Intent();//发送广播
intent.setAction("action.refreshShopcar");
Bundle bundle=new Bundle();
bundle.putInt("key",pos);//pos为要传输的数据，即该商品在数组中的下标
intent.putExtras(bundle);
sendBroadcast(intent);
```

```
IntentFilter intentFilter=new IntentFilter();//动态注册
intentFilter.addAction( "action.refreshShopcar");
registerReceiver(mBroadcastReceiver,intentFilter);

@Override
protected void onDestroy(){//在onDestroy中销毁广播
    super.onDestroy();
    unregisterReceiver(mBroadcastReceiver);
}
```

```
private BroadcastReceiver mBroadcastReceiver=new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action=intent.getAction();
        if(action.equals("action.refreshShopcar")){//处理接收到的Intent
            Bundle bundle=intent.getExtras();
            int pos=bundle.getInt("key");
            Map<String,String> tmp=new HashMap<>();
            tmp.put("icon",String.valueOf(name[pos].charAt(0)));
            tmp.put("name",name[pos]);
            tmp.put("price",price[pos]);
            list.add(tmp);
            simpleAdapter.notifyDataSetChanged();
        }
    }
};
```

5.实验思考及感想

这次实验内容很多，所以做起来花了很多时间。个人认为实验文档中自定义适配器部分有一点问题

```
public abstract class CommonAdapter extends RecyclerView.Adapter
```

这样声明的话抽象类是没法实例化的，除非传一个模板进去，所以我在写这部分的过程中定义了实类。

这次实验中涉及到不同Activity之间的数据交互，使用广播机制就可以解决这个问题。在使用广播机制的时候，个人认为静态注册receiver是可以完成实验要求的，因为MainActivity并没有finish，仍然在栈里，但是在实现的时候静态注册receiver总是接收不到第二个Activity传来的Intent，很奇怪，可能是我代码写错了，最后还是使用了registerReceiver函数来进行动态注册。

在用ListView实现购物车的时候，我碰到了一个奇怪的bug，从**商品列表**点进**商品信息**后将该商品添加进**购物车**，在**购物车**中点击该商品进入**商品信息**，然后再次将这个商品添加进**购物车**，然后返回后点击ListView的任意一项，程序都会崩溃。找了很久没找到这个问题，问卢巧笑师姐后才明白，我在**商品信息**这个Activity中点击购物车图标后传递消息给上一个Activity，最开始的实现方法是直接在ListView对应的List<T>里面add数据，这样既无法更新，同时也会因为错误的List<T>及其对应的点击事件导致程序崩溃，需要在广播机制中接收到Intent的函数里更新数据适配器

```
simpleAdapter.notifyDataSetChanged();
```

这样就不会引起程序崩溃了，同时也能及时更新ListView。

另外我实现长按从购物车中删除商品的时候也是在List<T>中删除数据，然后重新定义适配器传过去，这样导致虽然在视觉上这个item已经被删去，但是由于在ListView中它依然存在，在这个空白的地方点击事件依然可以触发，这个bug同样是在长按事件的处理函数中调用simpleAdapter.notifyDataSetChanged()来更新数据。

最后

感谢卢巧笑师姐！！！为师姐疯狂打call！！！