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）	软件工程
学号	15352441	姓名	钟丹彬
电话	18927386861	Email	2229690353@qq.com

一、 实验题目

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

二、 实现内容

本次实验模拟实现一个商品表，有两个界面。一个用于呈现商品，一个用于查看商品的详细信息。

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。

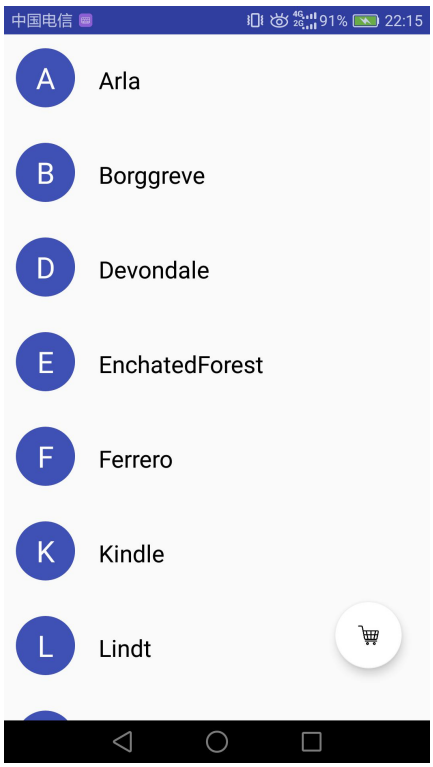
3、使用ListView实现购物车。点击购物车的某一个商品会跳转到商品详情界面，呈现该商品的详细信息；长按购物中的商品会弹出对话框询问是否移除该商品，点击确定则移除该商品，点击取消则对话框消失。

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。

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1. 建立商品类

```
public  class Good {
    private String name;          //商品名称
    private String nameId;        //商品首字母大写
    private String price;         //商品价格
    private String message;       //商品补充信息
    private int image;            //商品图片
    private int id;             //商品编号

      //实例化商品
    public Good(int id, String name, String nameId,
                        String price, String message, int image){
       this.id = id;
       this.name = name;
       this.nameId = nameId;
```

```
        this.price = price;
        this.message = message;
        this.image = image;
    }
    public int getId() {return id;}
    public String getName(){return name;}
    public String getnameId(){return nameId;}
    public String getPrice(){ return price;}
    public String getMessage() {return message;}
    public int getImage() {return image;}
}
```

2. 创建一个继承自 ArrayAdapter 的适配器自定义，将泛型指定为 Good 类,新建 Good_listAdapter :

```
public class Good_listAdapter extends ArrayAdapter<Good> {
    private int resourceId;
    public Good_listAdapter(Context context, int textViewResourceId,
                            List<Good> objects){
        super(context,textViewResourceId,objects);
        resourceId = textViewResourceId;
    }
    @Override
    public View getView(int position,View convertView,ViewGroup parent){
        //获取当前 Good 实例
        Good good = getItem(position);
        View view =
LayoutInflater.from(getContext()).inflate(resourceId,parent,false);
        //获取布局文件中控件实例
            TextView fruitImage = (TextView)view.findViewById(R.id.fruit_image);
        TextView fruitName = (TextView)view.findViewById(R.id.fruit_name);
        TextView fruitPrice = (TextView) view.findViewById(R.id.fruit_price);
        //将商品类实例的信息传入空间
            fruitImage.setText(good.getnameId());
        fruitName.setText(good.getName());
        fruitPrice.setText(good.getPrice());
        return view;
    }
}
```

3. 新建一个继承于 RecyclerView.Adapter 的适配器 Good_recyclerAdapter，将泛型指定为 Good_recyclerAdapter.ViewHolder。ViewHolder 是定义在 Good_recyclerAdapter 的一个内部类。

```
    public class Good_recyclerAdapter extends
RecyclerView.Adapter<Good_recyclerAdapter.ViewHolder>{
        //私有元素 Good 型 List
          private List<Good> mGoodList;

          //继承自 RecyclerView.ViewHolder
        static class ViewHolder extends RecyclerView.ViewHolder{
           View fruitView;
           TextView fruitId;
           TextView fruitName;
               //构造 ViewHolder，传入 RecycleView 子项的最外层布局
           public ViewHolder(View view){
              super(view);
              fruitView = view;
                    //获取布局中的实例
              fruitId = (TextView) view.findViewById(R.id.fruit_image);
              fruitName = (TextView) view.findViewById(R.id.fruit_name);
           }
        }

          //Good_recyclerAdapter 构造函数，将数据源传入
        public Good_recyclerAdapter(List<Good> goodList){
           mGoodList = goodList;
        }
        @Override

          //修改 ViewHolder，处理点击、长按事件
        public ViewHolder onCreateViewHolder(ViewGroup parent,int viewType){
           View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item,parent,false);
           final ViewHolder holder = new ViewHolder(view);

               //处理点击事件
           holder.fruitView.setOnClickListener(new View.OnClickListener(){
```

```
            @Override
            public void onClick(View v){

                Activity CurrentActivity = (Activity)v.getContext();

                        //利用 Intent 隐式跳转到商品详细信息界面
                Intent intent = new Intent("com.example.vincent.lab3.ACTION_START");
                        //返回当前被点击的位置
                int position = holder.getAdapterPosition();
                        //通过 putExtra 将被点击的对应商品信息传入商品详细信息界面
                intent.putExtra("good_id", mGoodList.get(position).getId());
                intent.putExtra("good_name", mGoodList.get(position).getName());
                intent.putExtra("good_price", mGoodList.get(position).getPrice());
                intent.putExtra("good_message", mGoodList.get(position).getMessage());
                intent.putExtra("good_image", mGoodList.get(position).getImage());
                intent.putExtra("good_nameid", mGoodList.get(position).getnameId());
                //界面跳转
                        CurrentActivity.startActivityForResult(intent,1);
            }
        });

            //处理长按事件
        holder.fruitView.setOnLongClickListener(new View.OnLongClickListener(){
            @Override
            public boolean onLongClick(View v){
                int position = holder.getAdapterPosition();
                Activity CurrentActivity = (Activity)v.getContext();
                //弹出 Toast 信息提示商品被移除
                        Toast.makeText(CurrentActivity,
                    "商品"+ mGoodList.get(position).getName()+"已被移除
",Toast.LENGTH_SHORT)
                    .show();

                        //将该商品移出 mGoodList
                mGoodList.remove(position);
                        //重新加载 Good_recylerAdapter
                notifyDataSetChanged();
                return false;
            }
        });
```

```
        return holder;
    }
    ……
}
```

4. 考虑到要将商品在商品详细信息界面无跳转的将商品添加至购物车，在不熟悉 java 语言的情况下，我决定将购物车对应的 Good_listAdapter shoplist 变成全局静态变量，以便在任何界面都能对购物车列表进行操作。类似的，为了确定某种商品是否被收藏，同样让商品收藏状态变成全局静态变量。建立一个 Data 类，专门用于存放全局变量和对其进行操作。

```
public class Data {
    //购物车内商品信息
    private static List<Good> shoplist = new ArrayList<>();
    public static Good_listAdapter listadapter;
    //商品对应收藏状态
    private static boolean Tag[] =
                    {false,false,false,false,false,false,false,false,false,false};
    //返回购物车列表
    public static List<Good> getShoplist(){
        return shoplist;
    }
    //向购物车内添加商品
    public static void setData(Good good){
        Data.shoplist.add(good);
    }
    //从购物车移除商品
    public static void removeData(int position){
        Data.shoplist.remove(position);
    }
    //返回商品的收藏状态
    public static boolean getTag(int i) {return Tag[i];}
    //改变商品的收藏状态
    public static void setTag(int i) { Tag[i] = !Tag[i];}
}
```

5. 编写商品列表 java 文件

i. 将购物车适配器实例化

```
//将购物车适配器实例化
Data.listadapter = new Good_listAdapter(this,R.layout.item,Data.getShoplist());
 Data.listadapter.notifyDataSetChanged();
```

ii. 设置商品列表显示

```
//初始化商品列表
initFruits();
//获取布局中 Recycler 实例
final RecyclerView recyclerView = (RecyclerView)findViewById(R.id.recycle_view);
//创建一个 LinearLayoutManager 对象，将其设置到 RecyclerView 中
LinearLayoutManager layoutManager = new LinearLayoutManager(this);
recyclerView.setLayoutManager(layoutManager);
//创建 Good_recyclerAdapter 实例，将商品列表传入
Good_recyclerAdapter adapter = new Good_recyclerAdapter(goodList);
//完成适配器设置
recyclerView.setAdapter(adapter);
```

iii. 设置购物车界面的点击事件

```
//设置购物车界面的点击事件，将相应的商品信息传入商品详细信息界面中并跳转
      listView.setOnItemClickListener(new AdapterView.OnItemClickListener(){
         @Override
         public void onItemClick(AdapterView<?> parent,View view,
                                              int position,long id){
            Good good = Data.getShoplist().get(position);
            Activity CurrentActivity = (Activity)view.getContext();
            Intent intent = new Intent("com.example.vincent.lab3.ACTION_START");

            intent.putExtra("good_id",Data.getShoplist().get(position).getId());
            intent.putExtra("good_name",Data.getShoplist().get(position).getName());
            intent.putExtra("good_price",Data.getShoplist().get(position).getPrice());

intent.putExtra("good_message",Data.getShoplist().get(position).getMessage());
            intent.putExtra("good_image",Data.getShoplist().get(position).getImage());
```

```
    intent.putExtra("good_nameid",Data.getShoplist().get(position).getnameId());
                CurrentActivity.startActivityForResult(intent,1);
}
```

iv. 设置购物车长按事件

```
            //设置长按后弹出的对话框
        final AlertDialog.Builder mbuilder = new AlertDialog.Builder(this);
            //设置长按事件
        listView.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent,
                                           View view, final int position, long id) {
                final String[] items = new String[]{"从购物车中移除"
                                    +Data.getShoplist().get(position).getName()+"?"};
                mbuilder.setTitle("移除商品")
                        .setNegativeButton("取消",new DialogInterface.OnClickListener(){
                            @Override
                            public void onClick(DialogInterface dialogInterface,int i){}
                        })

                        //点击确定按钮，将购物车列表相应商品移出，并刷新 listadapter
                        .setPositiveButton("确定",new DialogInterface.OnClickListener(){
                            @Override
                            public void onClick(DialogInterface dialogInterface,int i){
                                Data.removeData(position);
                                Data.listadapter.notifyDataSetChanged();
                            }
                        })
                        .setItems(items,new DialogInterface.OnClickListener(){
                            @Override
                            public void onClick(DialogInterface dialogInterface,int i){}
                        })
                        .show();
                            //注意，为了使长按事件和点击事件不冲突，返回值为 true
                return true;
            }
        });
```

v. 通过悬浮按钮的 tag 值判断显示商品列表还是购物车列表

```
        final FloatingActionButton FloatButton =
(FloatingActionButton)findViewById(R.id.fab);
        FloatButton.setOnClickListener(new FloatingActionButton.OnClickListener(){
            @Override
            public void onClick(View v){
                tag = !tag;
                if(tag == false){
                    FloatButton.setImageResource(R.mipmap.shoplist);
                    listView.setVisibility(View.GONE);
                    recyclerView.setVisibility(View.VISIBLE);

                }
                else{
                    FloatButton.setImageResource(R.mipmap.mainpage);
                    listView.setVisibility(View.VISIBLE);
                    recyclerView.setVisibility(View.GONE);
                }
            }
    });
```

6. 编写商品详细信息界面

i. 从商品列表界面接收商品数据

```
final int data_id = (int)getIntent().getSerializableExtra("good_id");
final String data_name = (String) getIntent().getSerializableExtra("good_name");
TextView name = (TextView)findViewById(R.id.good_name);
name.setText(data_name);
final String data_price = (String) getIntent().getSerializableExtra("good_price");
TextView price = (TextView) findViewById(R.id.price);
price.setText(data_price);
final String data_message = (String) getIntent().getSerializableExtra("good_message");
TextView message = (TextView) findViewById(R.id.weight);
message.setText(data_message);
final int data_image = (int) getIntent().getSerializableExtra("good_image");
ImageView image = (ImageView) findViewById(R.id.image);
image.setImageResource(data_image);
final String data_nameid = (String) getIntent().getSerializableExtra("good_nameid");
```

ii. 点击购物车，将商品添加至购物车列表

```
        ShopCar = (ImageView) findViewById(R.id.shopcar);
        ShopCar.setOnClickListener(new ImageView.OnClickListener(){
            @Override
            public void onClick(View v){
                Good new_shop = new
Good(data_id,data_name,data_nameid,data_price,data_message,data_image);
                Data.setData(new_shop);
                Toast.makeText(Message.this,"商品已添加到购物车",Toast.LENGTH_SHORT)
                        .show();
                 Data.listadapter.notifyDataSetChanged();
            }
        });
```

iii. 点击返回按钮，返回上一界面

```
        Back = (ImageView) findViewById(R.id.back);
        Back.setOnClickListener(new ImageView.OnClickListener(){
            @Override
            public void onClick(View v){
                returnData();
            }
            private void returnData(){
                Intent intent = new Intent();
                setResult(RESULT_OK,intent);
                finish();
            }
        });
```

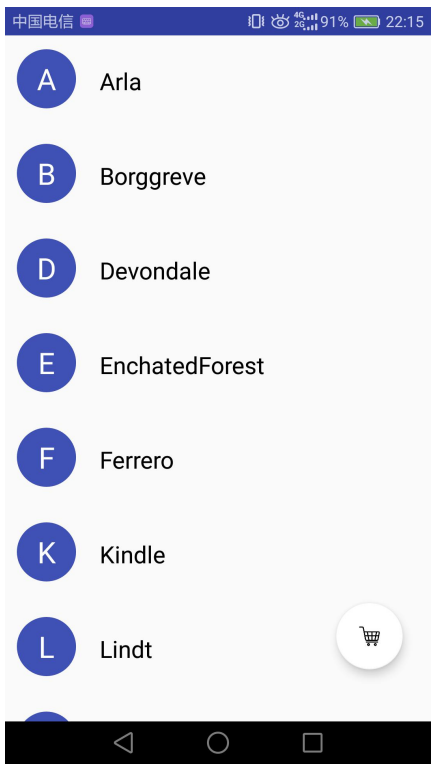
iv. 点击收藏按钮，将商品收藏状态赋值

```
        Star = (ImageView) findViewById(R.id.collect);
        Star.setOnClickListener(new FloatingActionButton.OnClickListener(){
            @Override
            public void onClick(View v){
                Data.setTag(data_id);
                if(Data.getTag(data_id) == false){
                    Star.setImageResource(R.mipmap.empty_star);
                }
                else{
```

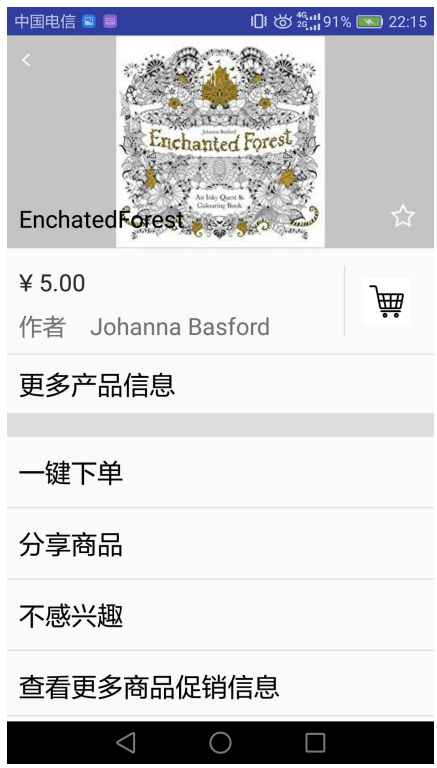
```
                Star.setImageResource(R.mipmap.full_star);
            }
        }
    });
```

四、 课后实验结果

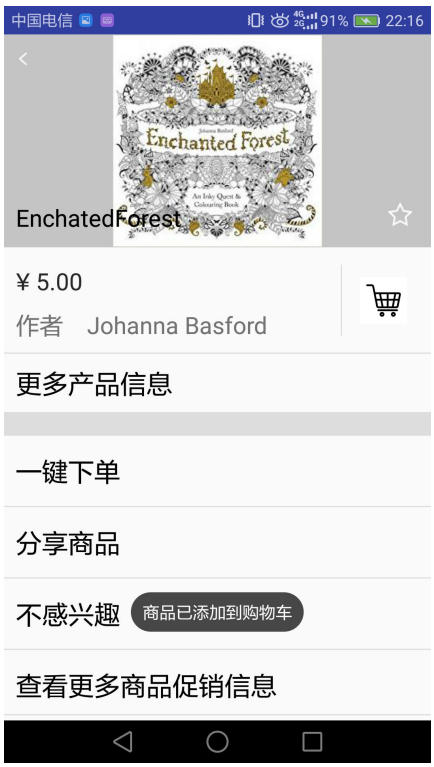
1. 进入 APP 后的商品列表



2. 点击商品列表进入商品详细信息

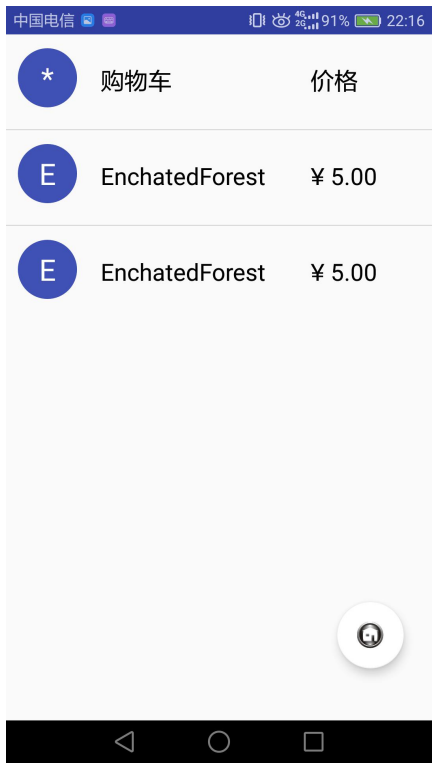


3. 点击购物车按钮，弹出 Toast 提示

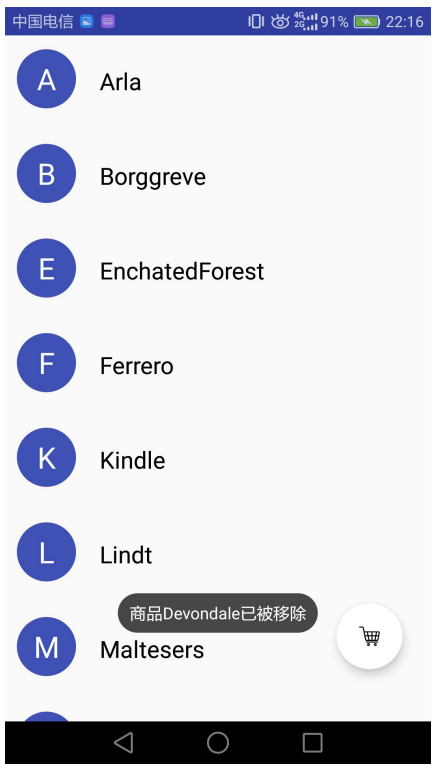


4. 返回，点击购物车按钮，切换至购物车列表，可以看见商品已被添加至购物车

（商品可被多次添加至购物车）



5. 点击右下角 Home 键，切换至商品列表，长按商品从列表中删除



6. 切换至购物车列表，长按商品，弹出对话框确认是否删除商品

五、 实验思考及感想

本次实验的遇到的比较大的困难就是 ListAdapter 和 RecycleAdapter 的重载了，因为的确不熟悉 Java 语言，所以在编写重载类时，不太清除具体要做些什么。通过在查阅资料后，按照他人的模板编写分析，大致有了初步的印象。对于 ListAdapter，大致思想就是，首先获取商品实例，然后获取布局文件中的控件实例，然后将商品的信息传入控件。对于 RecycleAdapter，看起来代码更加复杂，因为还需要修改 ViewHolder，但是其思路更加清晰好理解，就是先在布局中获取 Recycle 实例，然后创建一个 LinearLayoutManager 对象，将其设置到 RecyclerView 中，之后再创建 Good_recyclerAdapter 实例，将商品列表传入，最后通过 setAdapter 完成适配器设置。

其次的困难就是实现页面的跳转，本次实验，从商品列表/购物车列表跳转至商品详细信息界面。我使用了隐式的跳转，然后通过 intent.putExtra 将商品的信息数据传到商品详细信息界面。但是从商品详细信息界面点击购物车时，并不发生界面跳转，所以不能通过这种方法将商品信息从商品详细信息界面传递至购物车列表。后来，几经权衡后，我选择了一种比较简单的方法，即把购物车列表申明为全局静态变量，这样，在商品详细信息界面就能操作购物车列表的添加了。同样，利用这样的想法，实现商品收藏状态的标记。

但是，我认为这种方法有一定的弊端，在其他界面直接操作另外的界面的数据，emmmm，怎么想都有点怪怪的。所以，我希望在后续的学习中，能加深对 java 的理解，能更加熟悉地掌握 Android 的编写。