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：

年级	15	专业（方向）	移动互联网
学号	15352155	姓名	赖贤城
电话	13727024851	Email	754578682@qq.com
开始日期	2017/9/27	完成日期	2017/9/28

一、 实验题目

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

二、 实现内容

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、 课堂实验结果

（1） 实验截图

（2） 实验步骤以及关键代码

1. 商品列表界面及其逻辑

a. 主界面（列表以及浮动按钮）

相当于定义了两套 UI，一套是 ListView 加一个 FloatActionButton，另一个是 recyclerView 加一个 FloatActionButton。

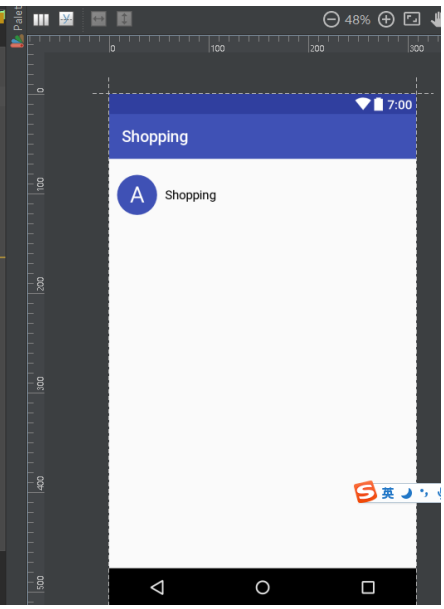
```
<ListView
    android:id="@+id/lv_shop_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:visibility="gone">
</ListView>
<android.support.v7.widget.RecyclerView
    android:id="@+id/rcv_goods"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</android.support.v7.widget.RecyclerView>

<android.support.design.widget.FloatingActionButton
    android:theme="@style/ButtonTheme"
    android:id="@+id/fb_shoplist"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:elevation="5dp"
    android:src="@drawable/shoplist"
    android:layout_marginBottom="20dp"
    android:layout_marginRight="20dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"/>
<android.support.design.widget.FloatingActionButton
    android:theme="@style/ButtonTheme"
    android:id="@+id/fb_goods"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:elevation="5dp"
    android:src="@drawable/mainpage"
    android:layout_marginBottom="20dp"
    android:layout_marginRight="20dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:visibility="gone"/>
```

注意到 FloatActionButton 里面我设定了 theme 为自定义的 ButtonTheme，这是因为文档中给出的颜色是白色，FloatActionButton 的颜色是根据主题里面的 colorAccent 来定的，但是直接改 activity 的 theme 会有影响其他组件的颜色，于是自定义了一个主题（见下图），专门给 FloatActionButton 使用

```
<style name="ButtonTheme" parent="AppTheme">
    <!-- Customize your theme here. -->
    <item name="colorAccent">@color/white</item>
</style>
```

b. 商品页的每一项的布局

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
             android:layout_marginTop="20dp"
             android:layout_marginBottom="20dp"
              android:layout_marginLeft="10dp"
              android:layout_marginRight="10dp">
    <TextView
        android:id="@+id/tv_first_letter"
        android:layout_gravity="center_vertical"
        android:gravity="center"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:text="A"
        android:maxLength="1"
        android:background="@drawable/circle_shape"
        android:textColor="@android:color/white"
        android:textSize="28sp"/>
    <TextView
        android:id="@+id/tv_name"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:text="Shopping"
        android:textSize="16sp"
        android:textColor="@android:color/black"
        android:gravity="center_vertical|start"/>
</LinearLayout>
```

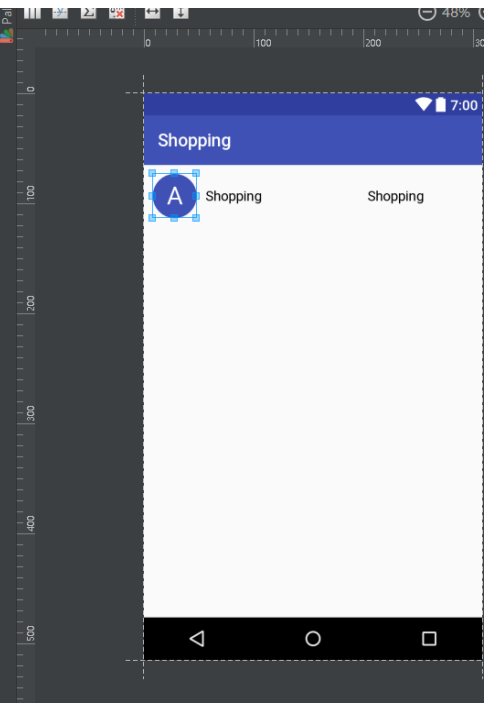


其中圆形是通过设置背景设置的，当然，需要用 xml 自定义圆形（下图）

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#3F51B5"/>
    <size android:height="20dp"
          android:width="20dp"/>
</shape>
```

c. 购物车的每一项的布局

```
<TextView
    android:id="@+id/tv_first_letter"
    android:gravity="center"
    android:layout_gravity="center_vertical"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:text="A"
    android:maxLength="1"
    android:background="@drawable/circle_shape"
    android:textColor="@android:color/white"
    android:textSize="28sp"/>
<TextView
    android:id="@+id/tv_name"
    android:layout_gravity="center_vertical"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="3"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:text="Shopping"
    android:textSize="16sp"
    android:textColor="@android:color/black"
    android:gravity="center_vertical|start"/>
<TextView
    android:id="@+id/tv_price"
    android:layout_gravity="center_vertical"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="2"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
```



其中，用的是三个 TextView 来实现，借助 LinearLayout 的 weight 来控制它们宽度的比例

d. 自定义商品页 recyclerView 的 Adapter

首先是自定义 ViewHolder，我将其放在 Adapter 内部

```
class GoodsVH extends RecyclerView.ViewHolder{
    TextView tvName;
    TextView tvFirstLetter;
    GoodsVH(View view) {
        super(view);
        tvFirstLetter = view.findViewById(R.id.tv_first_letter);
        tvName = view.findViewById(R.id.tv_name);
    }
}
```

recyclerView 的适配器，构造时需要传入对应的数据列表，实现了的接口，这里没有用 pdf 里面说的设置 convert 虚函数的方法，而是直接在 onBingViewHolder 里面绑定数据，原因是我觉得这样比较简单

```
public class GoodsItemAdapter extends RecyclerView.Adapter<GoodsItemAdapter.GoodsVH> {

    private List<GoodsItemBean> goodsList;//列表对应的数据
    private Context context;
    private recyItemClickListener itemClickListener;//自定义的接口

    public GoodsItemAdapter(Context context, List<GoodsItemBean> goodsList, recyItemClickListener itemClickListener) {
        this.goodsList = goodsList;
        this.context = context;
        this.itemClickListener = itemClickListener;
    }

    @Override
    public GoodsVH onCreateViewHolder(ViewGroup parent, int viewType) {
        //直接在此处将布局文件inflate成View
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.goods_item, parent, false);
        return new GoodsVH(view);
    }

    @Override
    public void onBindViewHolder(final GoodsVH holder, int position) {
        //直接在这里设置数据
        String name = goodsList.get(position).name;
        holder.tvFirstLetter.setText(name.substring(0,1));
        holder.tvName.setText(name);
        holder.itemView.setOnClickListener((v) -> {
                itemClickListener.onClick(holder.getAdapterPosition());
        });
        holder.itemView.setOnLongClickListener((v) -> {
                itemClickListener.onLongClick(holder.getAdapterPosition());
                return true;//消费长按事件
        });
    }

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

其中的 GoodsItemBean 是一个商品属性的类，定义如下：

```
/**
 * 一个商品的全部属性
 * Created by LaiXiancheng on 2017/10/19.
 * Email: lxc.sysu@qq.com
 */

public class GoodsItemBean implements Serializable{
    public String name;
    public String price;
    public String type;
    public String furtherInformation;
    public int imageRes;

    public GoodsItemBean(String name, String price, String type, String furtherInformation, int imageRes) {
        this.name = name;
        this.price = price;
        this.type = type;
        this.furtherInformation = furtherInformation;
        this.imageRes = imageRes;
    }
}
```

接着在首页的 activity 里面给 recyclerView 设置适配器，由于适配器构造函数需要传入 onItemClickListener 接口，所以这里实例化一个适配器的时候也把点击和长按的接口给实现了

```
recycleAdapter = new GoodsItemAdapter(this, goodsItems, new recyItemClickListener() {
    @Override
    public void onClick(int position) {
        Intent i = new Intent(GoodsActivity.this, DetailActivity.class);
        i.putExtra(JUMP_INFO, goodsItems.get(position));
        startActivity(i);
    }

    @Override
    public void onLongClick(final int position) {
        goodsItems.remove(position);
        recycleAdapter.notifyItemRemoved(position);
        Toast.makeText(
                GoodsActivity.this,
                "移除第" + String.valueOf(position) + "个商品",
                Toast.LENGTH_SHORT).show();
    }
});
```

e. ***设置完 recyclerView 的适配器，还有 listView 的适配器需要定义***

由于购物车页面的布局较为复杂，所以这里不使用自带的 adapter，而是通过继承 BaseAdapter 来定义。

并且，使用了优化的方法，首先是判断 convertView 能否重用，其次是使用 ViewHolder 来保存每一项里面的各种 view（例如 TextView）

```
@Override
public View getView(final int position, View convertView, ViewGroup parent) {
    View itemView;
    final ViewHolder holder;
    //是否可以重用
    if (convertView == null) {
        LayoutInflater inflater = LayoutInflater.from(context);
        itemView = inflater.inflate(R.layout.shop_item, parent, false); //加载布局，创建View
        holder=new ViewHolder(itemView);
        itemView.setTag(holder);
    } else {
        itemView = convertView;
        holder= (ViewHolder) itemView.getTag();
    }

    //设置数据
    String name = goodsList.get(position).name;
    char ch = name.charAt(0);
    if (ch>='A' && ch<='Z'  ||  ch>='a' && ch<='z')
        holder.tvFirstLetter.setText(name.substring(0,1));
    else
        holder.tvFirstLetter.setText("*");
    holder.tvName.setText(name);
    String price = goodsList.get(position).price;
    holder.tvPrice.setText(price);

    return itemView;
}
```

其中的 ViewHolder 如下图

```
private static class ViewHolder {
    TextView tvName;
    TextView tvFirstLetter;
    TextView tvPrice;

    ViewHolder(View itemView) {
        tvName= itemView.findViewById(R.id.tv_name);
        tvPrice= itemView.findViewById(R.id.tv_price);
        tvFirstLetter= itemView.findViewById(R.id.tv_first_letter);
    }
}
```

实现好了自定义的适配器之后，由于 ListView 自身有 setOnItemClickListener（以及 LongClick）函数，因此可以直接设置监听器

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        //第一项是提示信息，不设置监听器
        if (position == 0)
            return;

        Intent i = new Intent(GoodsActivity.this, DetailActivity.class);
        i.putExtra(JUMP_INFO, shopItems.get(position));
        startActivity(i);
    }
});
listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        //第一项是提示信息，不设置监听器
        if (position == 0)
            return true;
        showDeleteDialog(position);
        return true;
    }
});
```

其中的 showDeleteDialog 函数如下所示，它构建了一个对话框，并实现了点击事件的处理

```
/**
 * 展示删除的警告对话框
 * @param position
 */
private void showDeleteDialog(final int position) {
    final AlertDialog.Builder builder = new AlertDialog.Builder(GoodsActivity.this);
    builder.setTitle("移除商品");
    String msg = "从购物车移除%s？";
    msg = String.format(msg, goodsItems.get(position).name);
    builder.setMessage(msg);

    builder.setPositiveButton("确定", (dialog, which) -> {
            shopItems.remove(position);
            listAdapter.notifyDataSetChanged();
    });
    builder.setNegativeButton("取消", (dialog, which) -> {
            dialog.cancel();
    });
    builder.show();
}
```

f. ***两个列表终于实现完了，那么首页就只剩下 FloatActionButton 需要进行点击事件定义了***

这里我们有两个 button，一个是跳转购物车界面的，一个是跳转商品列表界面的

由于本实验中，两个界面是通过设置可见不可见来实现切换的，那么，显然的，这两个 button 的点击事件要做的就是设置可见性

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.fb_shoplist:
            fab_shop.setVisibility(View.GONE);
            fab_main.setVisibility(View.VISIBLE);
            recyclerView.setVisibility(View.GONE);
            listView.setVisibility(View.VISIBLE);
            break;
        case R.id.fb_goods:
            fab_shop.setVisibility(View.VISIBLE);
            fab_main.setVisibility(View.GONE);
            recyclerView.setVisibility(View.VISIBLE);
            listView.setVisibility(View.GONE);
            break;
    }
}
```

至此，商品页界面（包含购物车）完工，下面开始详情页的实现

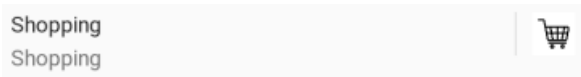
2. 详情页界面及其逻辑

a. *头部的 1/3 界面，使用 RelativeLayout 来实现这个布局*

主要是通过设置 align 属性以及 margin 属性来实现 view 的定位

```xml
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="@color/font_grey">
    <ImageView
        android:id="@+id/iv_goods"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/borggreve"/>
    <ImageView
        android:id="@+id/iv_back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/back"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_margin="10dp"/>
    <TextView
        android:id="@+id/tv_goods_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        android:text="NAME"
        android:gravity="center"
        android:textColor="@android:color/black"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="10dp"/>
    <ImageView
        android:id="@+id/iv_star"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:src="@drawable/empty_star"
        android:layout_marginBottom="20dp"
        android:layout_marginRight="10dp"/>
</RelativeLayout>
```

b. 商品信息栏与购物车图标



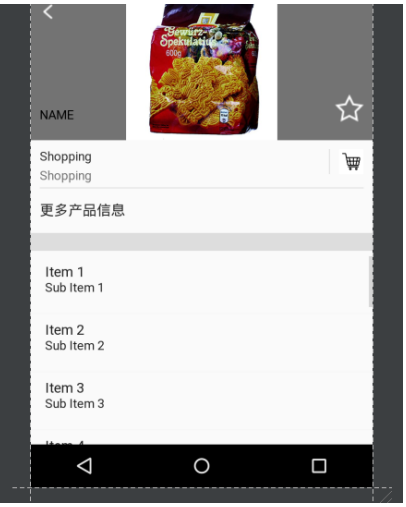
这里使用的是 constraintLayout 来实现，竖直的分割线使用宽度为 1dp 的 View 来实现

```xml
<android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/tv_price"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Shopping"
        android:textColor="@color/font_black"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_margin="10dp"/>
    <TextView
        android:id="@+id/tv_type"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Shopping"
        android:layout_marginTop="5dp"
        app:layout_constraintTop_toBottomOf="@+id/tv_price"
        app:layout_constraintLeft_toLeftOf="@+id/tv_price"/>
    <ImageView
        android:id="@+id/iv_shop"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@drawable/shoplist"
        app:layout_constraintDimensionRatio="1:1"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_margin="@dimen/common_margin"/>
    <View
        android:layout_width="1dp"
        android:layout_height="0dp"
        android:background="@color/line_color"
        app:layout_constraintTop_toTopOf="@id/iv_shop"
        app:layout_constraintBottom_toBottomOf="@id/iv_shop"
        app:layout_constraintRight_toLeftOf="@id/iv_shop"
        android:layout_marginRight="10dp"/>
</android.support.constraint.ConstraintLayout>
```

c. 下方四个操作项

下方操作项使用 ListView 实现，因此这里只是放了一个 ListView，需要在下一步中设置适配器

```
<include layout="@layout/line_gap"/>
<TextView
    android:id="@+id/tv_more_info"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="更多产品信息"
    android:textSize="16sp"
    android:textColor="@color/font_black"
    android:layout_margin="10dp"/>
<View
    android:layout_width="match_parent"
    android:layout_height="20dp"
    android:layout_marginTop="6dp"
    android:layout_marginBottom="6dp"
    android:background="@color/line_color"/>

<ListView
    android:id="@+id/lv_options"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</ListView>
```



d. 为 ListView 添加适配器

这里的布局很简单，就只是显示文字而已，因此只需要一个 item 的布局文件（内部只有一个 TextView），然后使用自带的 ArrayAdapter 就可以轻松做出这个列表

item 布局文件：

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
          android:text="Shopping"
          android:padding="10dp"
          android:textSize="16sp"
          android:textColor="@color/font_black"
          android:layout_width="match_parent"
          android:layout_height="wrap_content">

</TextView>
```

Activity 中设置 ArrayAdapter：

```
optionsListView = (ListView) findViewById(R.id.lv_options);
String[] optionNames = {
        "一键下单",
        "分享商品",
        "不感兴趣",
        "查看更多商品促销信息"};
ArrayAdapter<String> optionAdapter = new ArrayAdapter<>(this, R.layout.options_item, optionNames);
optionsListView.setAdapter(optionAdapter);
```

e. 界面做好了，接着往里面填数据

首先，数据是哪里来的呢？ 当然是上一个 Activity 跳转的时候传递过来的，那么就先把数据传递的逻辑给实现了。 我们知道，activity 的跳转是通过 Intent 来完成的，而 Intent 对象又可以携带数据，那么，就可以把点击的那条商品的数据传递过来了，当然，由于 intent 不能直接传递对象，而是能被序列化的对象，所以我们需要先序列化商品类

```java
/**
 * 一个商品的全部属性
 * Created by LaiXiancheng on 2017/10/19.
 * Email: lxc.sysu@qq.com
 */

public class GoodsItemBean implements Serializable{
    public String name;
    public String price;
    public String type;
    public String furtherInformation;
    public int imageRes;

    public GoodsItemBean(String name, String price, String type, String furtherInfo
        this.name = name;
        this.price = price;
        this.type = type;
        this.furtherInformation = furtherInformation;
        this.imageRes = imageRes;
    }
}
```

序列化之后就可以愉快地用 intent 传数据了

```java
@Override
public void onClick(int position) {
    Intent i = new Intent(GoodsActivity.this, DetailActivity.class);
    i.putExtra(JUMP_INFO, shopItems.get(position));
    startActivity(i);
}
```

然后就可以在跳转过去的那个 Activity 里面将数据取出来了，取出来之后，就用得到的数据去填入 view 里面

```java
private void initData() {
    Intent i = getIntent();
    goodsItem = (GoodsItemBean) i.getSerializableExtra(GoodsActivity.JUMP_INFO);
    tvName.setText(goodsItem.name);
    tvPrice.setText(goodsItem.price);
    tvType.setText(goodsItem.type + " " + goodsItem.furtherInformation);
    ivGoods.setImageResource(goodsItem.imageRes);
}
```

f. *处理点击事件*

首先是“星型”，这里管它叫做“收藏”好了。由于我们要记录它当前的状态才能正确的改变它的状态，因此这里用了一个 Boolean 的变量 isStar 来标记当前状态。当点击时，如果当前状态是已收藏，那么将它设为“未收藏”（就是改变图片资源以及状态的记录），反之同理

```java
            break;
        case R.id.iv_star:
            changeStar();
            break;
        case R.id.iv_shop:
            addToShopList();
    }
}

/**
 * 改变“收藏”状态
 */
private void changeStar(){
    if (isStar){
        ivStar.setImageResource(R.drawable.empty_star);
        isStar = false;
    }
    else{
        ivStar.setImageResource(R.drawable.full_star);
        isStar = true;
    }
}
```

然后是“加入购物车”按钮。这里还是用一个变量来记录加入的状态，代码这么设置表明无论点击几次都只是一样的标记为“已加入”

```
/**
 * 物品加入购物车
 */
private void addToShopList(){
    isToShopList = true;
    Toast.makeText(this, "商品已添加到购物车", Toast.LENGTH_SHORT).show();
}
```

那么问题来了，加入购物车之后，在哪里更新购物车界面的数据呢？
我是在“返回”按钮这里设置的（其实还有 onBackPressed 函数），在返回上一个 activity 的时候检查是否加入购物车，如果加入的话，就使用 eventBus 来发出一个事件

```
@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.iv_back:
            if (isToShopList){
                AddToShopListEvent event = new AddToShopListEvent(goodsItem);
                EventBus.getDefault().post(event);
            }
            finish();
            break;
        case R.id.iv_star:
            changeStar();
            break;
        case R.id.iv_shop:
            addToShopList();
    }
}

@Override
public void onBackPressed() {
    super.onBackPressed();//这里面已经进行finish了
    if (isToShopList){
        AddToShopListEvent event = new AddToShopListEvent(goodsItem);
        EventBus.getDefault().post(event);
    }
}
```

其中的 AddToShopListEvent 定义如下：

```
public class AddToShopListEvent {
    GoodsItemBean itemAdded;

    public AddToShopListEvent(GoodsItemBean itemAdded) {
        this.itemAdded = itemAdded;
    }

    public GoodsItemBean getItemAdded() { return itemAdded; }

    public void setItemAdded(GoodsItemBean itemAdded) { this.itemAdded = itemAdded; }
}
```

当然，需要在商品页注册 eventBus 并对该事件进行监听才可以

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

    EventBus.getDefault().register(this);
```

```java
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(AddToShopListEvent event) {
    shopItems.add(event.getItemAdded());
    listAdapter.notifyDataSetChanged();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().unregister(this);
}
```

3. 给 RecyclerView 添加自定义的动画

a. ***自定义 ItemAnimator（通过修改 DefaultItemAnimator）***

大部分代码 copy 自 DefaultItemAnimator（自带的安卓默认的 ItemAnimator），原本想直接继承 DefaultItemAnimator，但是发现要修改的地方是属于 private 方法，因此复制过来成新的动画类，

由于本次实验只涉及删除，所以只修改了 animateRemoveImpl 方法，也就是修改了移除 item 的动画效果。

DefaultItemAnimator 是 Android 中一个默认的 RecyclerView 动画实现类，由于我只想改动 item 移除时的动画效果，而保持其他不变，所以我就直接暴力的复制了 DefaultItemAnimator，然后对 animateRemoveImpl 方法进行修改

这里面最重要的是使用了自己定义的 MyRemoveAnimation（瞎起的名字），其中的 setListener 部分几乎没有修改原来的 DefaultItemAnimator，不过也说说我的一点点不全面的理解吧，onAnimationEnd 函数里面对 view 的操作应该是将变换过的 view 恢复以便 recyclerView 对其进行重用

```java
private void animateRemoveImpl(final ViewHolder holder) {
    final View view = holder.itemView;
    MyRemoveAnimation myRemoveAnimation = new MyRemoveAnimation();
    myRemoveAnimation.setDuration(getRemoveDuration());//使用itemAnimator的Remove时间
    myRemoveAnimation.setAnimationListener(new Animation.AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) { dispatchRemoveStarting(holder); }

        @Override
        public void onAnimationEnd(Animation animation) {
            animation.setAnimationListener(null);
            view.setAlpha(1);
            dispatchRemoveFinished(holder);
            mRemoveAnimations.remove(holder);
            dispatchFinishedWhenDone();
        }

        @Override
        public void onAnimationRepeat(Animation animation) {

        }
    });
    view.startAnimation(myRemoveAnimation);

    mRemoveAnimations.add(holder);
}
```

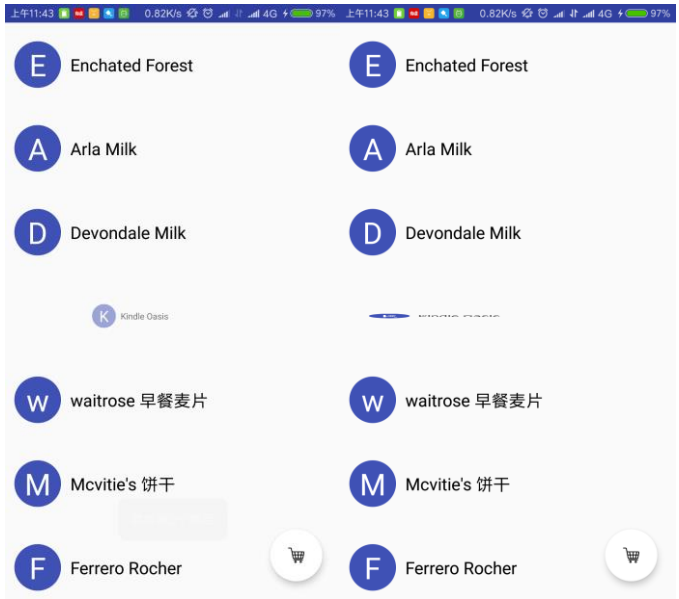
这里面的 MyRemoveAnimation 才是真正的变换的核心，首先在初始化的时候保存下 vie 的中心点，接着最重要是 applyTransformation 这个方法，先是将 view 收缩成一个点，然后将其展开成一条线

```java
class MyRemoveAnimation extends Animation {
    private int halfWidth;
    private int halfHeight;

    @Override
    public void initialize(int width, int height, int parentWidth, int parentHeight) {
        super.initialize(width, height, parentWidth, parentHeight);
        setDuration(getDuration());
        setFillAfter(true);
        //View的中心点
        halfWidth=width/2;
        halfHeight=height/2;
        //设置动画先加速后减速
        setInterpolator(new AccelerateDecelerateInterpolator());
    }

    @Override
    protected void applyTransformation(float interpolatedTime, Transformation t) {
        Matrix matrix = t.getMatrix();
        //interpolatedTime是从0~1的一个变化
        // 前85%让动画缩小成一个点（宽高均为原来的0.1倍），后15%保持线的高度将这个点展开成一条线，变换的中心为View的中心
        if (interpolatedTime<0.85){
            float scale = (1-interpolatedTime/0.85f) < 0.1f ? 0.1f : (1-interpolatedTime/0.85f);
            matrix.preScale(scale, scale, halfWidth, halfHeight);
            t.setAlpha(1-interpolatedTime/0.85f);
        }else{
            float scale = (interpolatedTime-0.85f)/0.15f < 0.1f ? 0.1f : (interpolatedTime-0.85f)/0.15f;
            matrix.setScale(scale , 0.1f,halfWidth,halfHeight);
        }

    }
```

b. 使用 ItemAnimator

```java
FadeItemAnimator itemAnimator = new FadeItemAnimator();
itemAnimator.setRemoveDuration(1000);//时间长一点，比较能看出效果
recyclerView.setItemAnimator(itemAnimator);
```

效果展示：



（3） 实验遇到困难以及解决思路

1. ListView 的 item 布局使用 margin 无效

解决：RecyclerView 的 item 设置 margin 是有效的，但是 ListView 居然无效，而且我在 inflate 的时候是有传入父布局的（itemView = inflater.inflate(R.layout.shop_item, parent, false)），按理说不会这样。

百思不得其解，直到看到了这篇博客，原因是 ListView 的父类 AbsListView 中 layoutparams 并没有实现 marginLayoutParams，所以之类获取不到相关值，也就无效了。了解原理之后，就还是乖乖地用 padding 吧

2. FloatingActionButton 的背景颜色问题

解决：我本来是想用 android:backgroundTint 属性的，结果被告知需要 api21 以上的才可以使用，无奈只好改了 colorAccent 的颜色值为白色，但是这样又引发新的问题，就是弹出的对话框的字也变成白色的了。这时候想起来可以单独给 FloatingActionButton 设置一个 theme 啊，所以就继承 activity 的主题，改了其中的 colorAccent

```
<style name="ButtonTheme" parent="AppTheme">
    <item name="colorAccent">@color/white</item>
</style>
```

四、 课后实验结果

添加“回到顶部”按钮

1. 效果展示

2. 代码

xml 布局文件中添加回顶部按钮

```
<android.support.design.widget.FloatingActionButton
    android:theme="@style/ButtonTheme"
    android:id="@+id/fb_top"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:elevation="5dp"
    android:src="@mipmap/ic_top"
    android:layout_marginBottom="80dp"
    android:layout_marginRight="20dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"/>
```

Java 中设置点击事件，调用 smoothScrollToPosition 即可

```
case R.id.fb_top:
    recyclerView.smoothScrollToPosition(0);
    break;
```

五、 实验思考及感想

1. 遇到问题的时候尝试看看源码。例如这次的自定义 Adapter，很多人出错的原因只要点进入 RecyclerView.Adapter 里面就能很容易知道自己是什么地方错了
2. 补充 Java 知识。这次实验下来我感觉自己有挺多 Java 的知识已经忘了，甚至是之前误解了，例如权限的问题，泛型的问题。还是要对 Java 有更多的了解之后再进行开发能更有效率一些
3. 开源库学习。这次文档给出的动画库就很好（虽然我没用它），github 上有很多像这个动画库一样的开源库，应该多去了解学习，不仅仅是需要的时候你能快速的找到合适的库使用，而且开源库一般来说设计模式，代码风格之类的都很好，很有学习价值

作业要求：

1. 命名要求: 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。