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 级	专业（方向）	软件工程（移动信息工程）
学号	15352223	姓名	刘朝开
电话	15626489095	Email	1209354383@qq.com
开始日期	2017.9.30	完成日期	2017.10.1

一、 实验题目

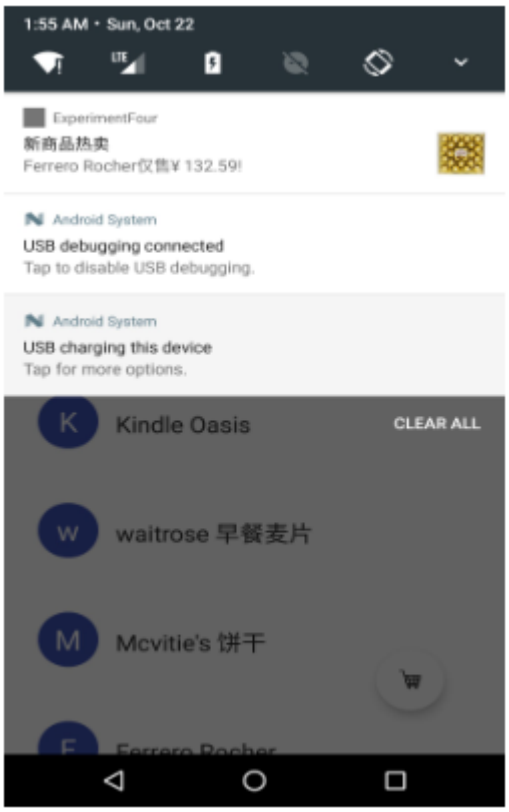
Broadcast 使用

二、 实现内容

在实验三的基础上，实现静态广播、动态广播两种改变 Notification 内容的方法。

具体要求：

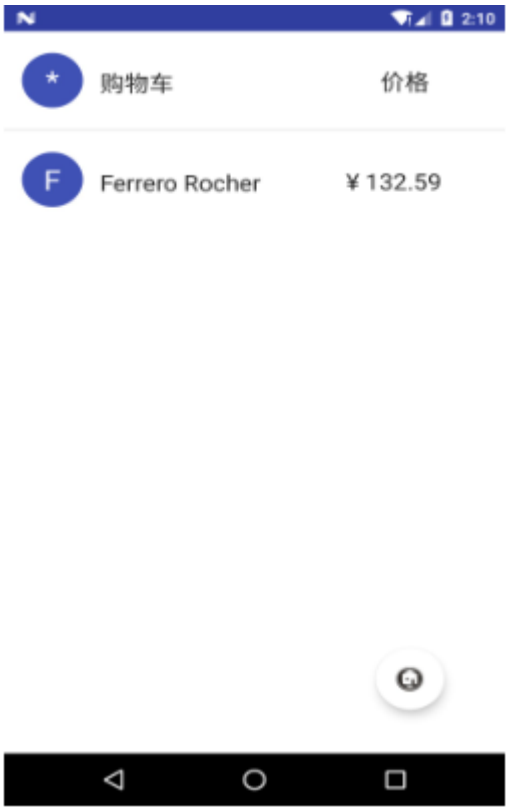
(1)在启动应用时，会有通知产生，随机推荐一个商品:



(2)点击通知跳转到该商品详情界面:

(3)点击购物车图标，会有对应通知产生，并通过 Eventbus 在购物车列表更新数据:

(4)点击通知返回购物车列表:

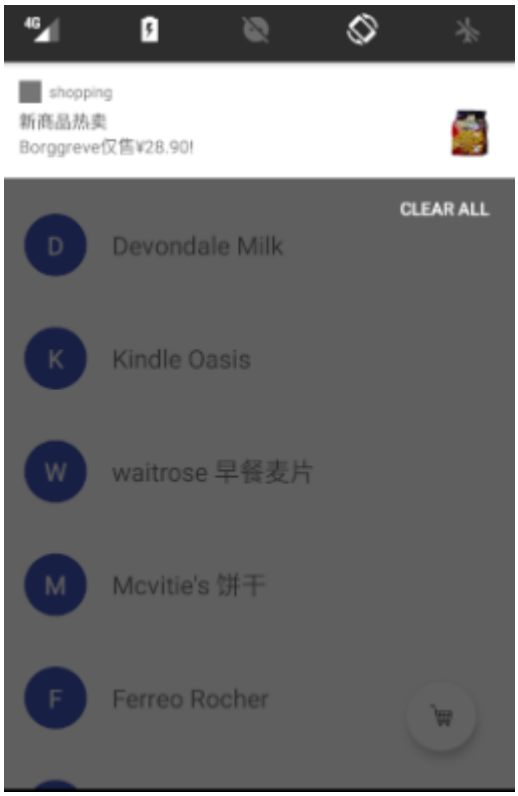


(5)实现方式要求:启动页面的通知由静态广播产生，点击购物车图标的通知由动态广播产生。

三、 课堂实验结果

（1） 实验截图

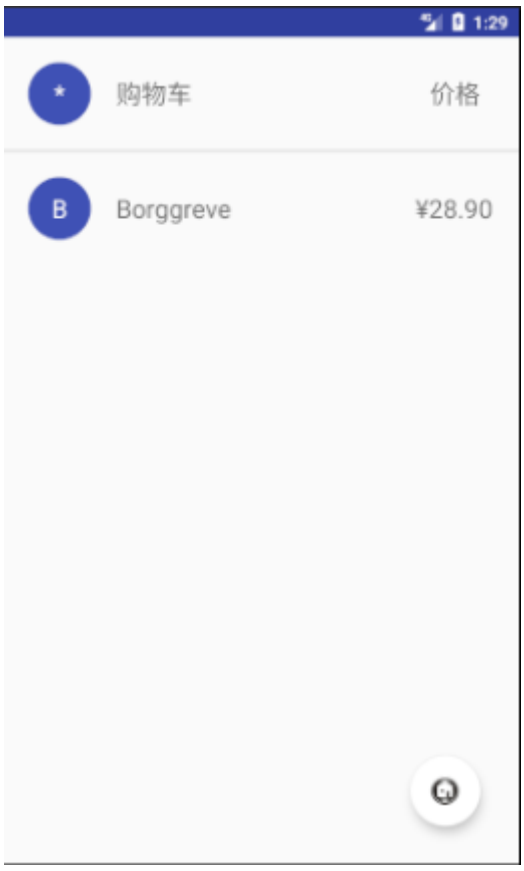
1)启动应用时产生通知，随机推荐一个商品:



2)点击通知跳转到该商品详情界面:

3)点击购物车图标，产生对应通知，并通过 Eventbus 在购物车列表更新数据:

4)点击通知返回购物车列表:



（2） 实验步骤以及关键代码

实验步骤如下：

1)静态广播

新建 Broadcast Receiver,命名为 StartApp。

```
public class StartApp extends BroadcastReceiver {
```

静态注册事件

```
<intent-filter>
    <action android:name="com.example.liuchaokai.lab3_listview_recyclerview.STARTAPP" />
</intent-filter>
```

生成 1-10 之间的随机数：

```
//生成随机数
Random pNumber = new Random();
randompNum = pNumber.nextInt( n: 10)%10+1;
```

给广播传递数据：

```
//给广播传递数据
Bundle smsBundle = new Bundle();
smsBundle.putInt("pIcon",pImages[randompNum-1]);//传递图标
smsBundle.putString("pTile","新商品热卖");//传递标题
smsBundle.putString("pText",Names[randompNum]+"仅售"+Prices[randompNum]+'!');//传递内容
smsBundle.putString("pName",Names[randompNum]);//传递商品名字
```

启动广播：

```
//应用启动广播
Intent smsintent = new Intent( action: "com.example.liuchaokai.lab3_listview_recyclerview.STARTAPP");
smsintent.putExtras(smsBundle);
sendBroadcast(smsintent);
```

广播接收外部数据：

```
//gbundle接收外部数据
Bundle gBundle = intent.getExtras();
int pIcon = gBundle.getInt( key: "pIcon");
String pTile = gBundle.getString( key: "pTile");
String pText = gBundle.getString( key: "pText");
String pName = gBundle.getString( key: "pName");
```

设置通知属性，发出通知：

```
//获取 NotificationManager 对象
NotificationManager mNotifyManager = (NotificationManager) context
        .getSystemService(Context.NOTIFICATION_SERVICE);
//实例化NotificationCompat.Builde并设置相关属性
Notification.Builder showProduct = new Notification.Builder(context);
showProduct.setSmallIcon(pIcon)//设置小图标
        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(),pIcon))//设置大图标
        .setContentTitle(pTile)//设置通知标题
        .setContentText(pText)//设置通知内容
        .setAutoCancel(true)//点击通知后自动清除
        .setDefaults(Notification.DEFAULT_ALL);//设置震动，声音，led等效果
```

根据广播的 action 执行点击通知时跳转到商品详情界面：

```
if(intent.getAction().equals("com.example.liuchaokai.lab3_listview_recyclerview.STARTAPP"))
{
    //通知栏
    Intent intentT = new Intent(context,Shopping.class);
    intentT.putExtra( name: "name",pName);
    PendingIntent pi = PendingIntent.getActivity(context, requestCode: 0,intentT,
            PendingIntent.FLAG_UPDATE_CURRENT);
    showProduct.setContentIntent(pi);//跳转到ShoppingActivity
    mNotifyManager.notify( id: 0, showProduct.build());//通过builder.build()方法生成Notificatio
}
```

2）动态广播

注册广播

```
// 注册广播
intentFilter = new IntentFilter();
intentFilter.addAction("Order.Product");
pOrder = new StartApp();
registerReceiver(pOrder,intentFilter);
```

设置要传递的信息

```
//给广播传递数据
Bundle smsBundle = new Bundle();
smsBundle.putInt("pIcon",pImages[snumber]);//传递图标
smsBundle.putString("pTile","马上下单");//传递标题
smsBundle.putString("pText",pNames[snumber]+"已加入购物车");//传递内容
smsBundle.putString("pName",pNames[snumber]);//传递商品名字
smsBundle.putInt("sNumber",return_data);//传递商品序号
```

启动广播

```
// 发送广播通知
Intent smsintent = new Intent();
smsintent.setAction("Order.Product");
smsintent.putExtras(smsBundle);
sendBroadcast(smsintent);
```

注销广播

```
@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(pOrder);
}
```

接收广播，与上面为同一个类，接受的信息基本一致。主要有以下代码不同：

```
else if(intent.getAction().equals("Order.Product"))
{
    int pNumber = gBundle.getInt( key: "sNumber");
    //通知栏
    Intent intentT = new Intent(context,MainActivity.class);
    intentT.putExtra( name: "flag", value: 1);
    PendingIntent pi = PendingIntent.getActivity(context, requestCode: 0,intentT,
            PendingIntent.FLAG_UPDATE_CURRENT);
    showProduct.setContentIntent(pi);
    mNotifyManager.notify( id: pNumber+1, showProduct.build());//发送通知,id=pNumber+1
    //发信息
    EventBus.getDefault().post(
                new SmsEventbus(pName,pNumber));
}
```

3）通过 EventBus 传递数据

新建一个类 SmsEventbus，用于传递数据

```
public class SmsEventbus {
    private String msg;
    private int mNumber;
    public SmsEventbus(String msg, int number)
    {
        this.msg=msg;
        this.mNumber = number;
    }
    public String getMsg() { return msg; }
    public int getmNumber() { return mNumber; }
}
```

在需要接收信息的 activity 即 MainActivity 中注册 EventBus:

```
//注册eventbus
EventBus.getDefault().register( subscriber: this);
```

注销 EventBus:

```
@Override
protected void onDestroy() {
    super.onDestroy();
    EventBus.getDefault().unregister( subscriber: this);//反注册EventBus
}
```

传递信息，包括商品名字，商品序号：

```
//发信息
EventBus.getDefault().post(
            new SmsEventbus(pName,pNumber));
```

接收并处理信息：

```
//接受商品详情界面传回的消息，用于更新购物车列表
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(SmsEventbus event)
{
    //...
    int pNumber = event.getmNumber();
    Products product = new Products(Keywords[pNumber],
            Names[pNumber],Prices[pNumber]);
    shopList.add(product);//加入购物车数组
    shopAdapter.notifyDataSetChanged();//更新数据
}
```

修复按返回键返回时也会返回购物车列表界面的 bug。重载 onNewIntent 函数，如下：

```
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    flag = intent.getIntExtra( name: "flag", defaultValue: 0);
    //Toast.makeText(MainActivity.this,"flag="+flag,Toast.LENGTH_SHORT).show();
    setUI();
}
```

其中传入的数据标记是否按下通知栏。若按下，flag=1,跳转购物车列表界面，否则跳转到原来进入商品详情界面的界面。SetUI 为自定义函数，表示界面的显示为购物车界面还是商品列表界面。

```
private void setUI()
{
    if(flag == 1)//切换至购物车界面
    {
        fab.setImageResource(R.drawable.mainpage);
        listView.setVisibility(View.VISIBLE);
        recyclerView.setVisibility(View.INVISIBLE);
    }
    else//切换回商品列表
    {
        fab.setImageResource(R.drawable.shoplist);
        listView.setVisibility(View.INVISIBLE);
        recyclerView.setVisibility(View.VISIBLE);
    }
}
```

（3） 实验遇到困难以及解决思路

问题：

1 在商品详情界面中点击加入购物车后，不管以何种方式返回 Mainactivity 界面，都会返回到购物车列表。

解决方法：1 原因分析：之前出现错误的原因是直接在 EventBus 中更新显示信息，从而导致未点击通知就设置了 MainActivity 的界面。解决方案：重写 onNewIntent 函数，在里面实现更新列表的效果。（上面已有说明，这里就不重复了）

2.每次弹出通知都会覆盖先前的通知，无法显示多条通知。

解决方法：发送通知设置不同的 id 使得通知不会被覆盖（相同商品加入购物车设置为可以覆盖）。

3.点击购物车后再点击通知进入商品列表界面时，按退出会返回多次商品详情界面。
解决方法：Android 用栈来管理 Activity，Activity 的启动方式默认为 standard，即会启动多次相同的 activity，所以退出时会出现多次返回原界面的效果。方法：把 launchMode 改为 singleTask,使得每个 Activity 只能启动一次。

4.动态广播的注册放在了加入购物车按钮事件中，导致若没有点击加入购物车按钮直接返回会出现程序崩溃的现象。
解决方案：分析：由于未点击加入购物车按钮，导致广播没有注册，直接退出时注销注册会出现错误，导致程序崩溃。解决：动态广播的注册放在 onCreat 中实现，即每次打开 Activity 就会注册广播。

四、 课后实验结果

1.增加了一些简单的通知特效，包括设置震动，声音，led 灯闪烁等效果，并且弹出状态栏：（模拟器上无法看到效果，故没有截图）

```
.setDefaults(Notification.DEFAULT_ALL)//设置震动，声音，led等效果
.setTicker("新品热卖");
```

2.使用 SharedPreferences 实现了星星功能的保存功能。代码如下：

定义私有变量分别记录星星的状态和星星状态的键值：

```
private int Starstate [] = {0,0,0,0,0,0,0,0,0,0};//记录星星状态
private static String Sstate [] = {"zero","one","two","three","four","five","six","seven","eight","nine"};//星星状态键值
```

每次启动 activity 都要获得当前星星的状态：

```
//获取星星状态数据
SharedPreferences pref = getSharedPreferences( name: "data",MODE_PRIVATE);
for(int i=0;i<10;i++) Starstate[i]=pref.getInt(Sstate[i], defValue: 0);
if(Starstate[snumber] == 0) Star.setImageResource(R.drawable.empty_star);
else Star.setImageResource(R.drawable.full_star);
```

每次点击星星时，都要更新提交星星状态：

```
//存储星星状态的数据
SharedPreferences.Editor editor = getSharedPreferences( name: "data",MODE_PRIVATE).edit();
editor.putInt(Sstate[snumber],Starstate[snumber]);
editor.apply();//提交保存
```

五、 实验思考及感想

这次实验内容不是很多，主要是要掌握广播和通知的结合使用，还有 EventBus 传输数据的实现。由于是初次接触，对于广播刚上手还是比较缓慢，上网查阅了很多的样例才区分开静态广播和动态广播的使用和区别。我的理解是：静态广播和动态广播可以通过同一个类接收，而静态广播的类必须注册，动态的可以注册也可以不注册。关键的区别在于是否有在 androidManifest 对应广播类中的<intent-filter>内设置 action,若设置了，则为静态注册，否则需要在 java 文件中 addAction 的则为动态注册。

关于增加新功能，我上网查阅了有关数据存储方案的一些知识，总结一下，总共有五种方式进行数据的存储：1 文件存储；2SharedPreferences；3SQLite 数据存储；4LitePal;5 网络存储数据。这次实验重点看了 SharedPreferences：适用于保存少量的数据，且这些数据的格式非常简单：字符串型、基本类型的值。我使用了 putInt 的方法将星星的状态保存起来，每次打开 Activity 都会通过 getInt 将之前保存的状态调用一遍，从而实现保存记录功能。

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。