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 级	专业（方向）	移动信息工程（移动互联网）
学号	15352083	姓名	冯灏帆
电话	13712309295	Email	ares_fung@qq.com
开始日期	2017.10.24	完成日期	2017.10.30

一、 实验题目

1、掌握 Broadcast 编程基础

2、掌握动态注册 Broadcast 和静态注册 Broadcast

3、掌握 Notification 编程基础

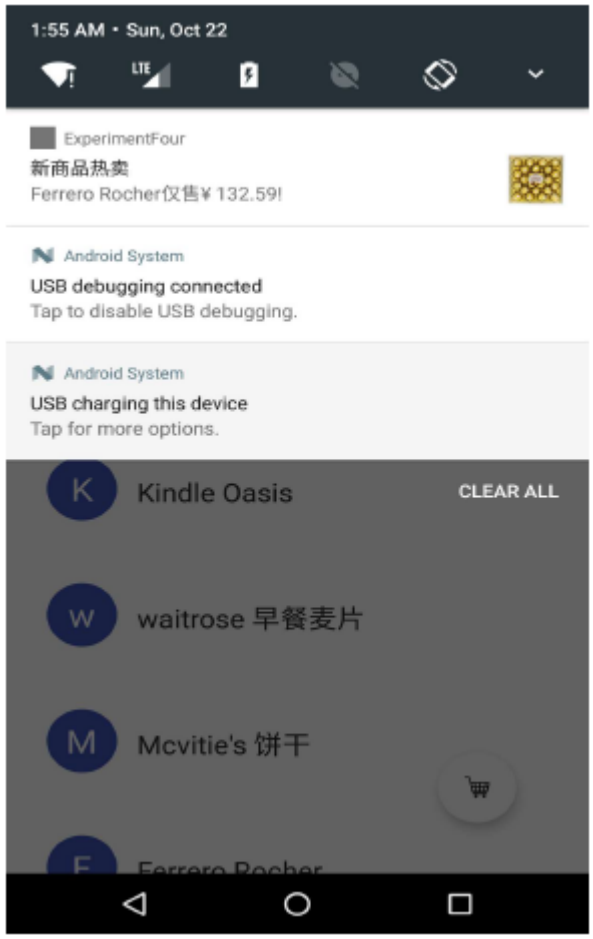
4、掌握 EventBus 编程基础

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。

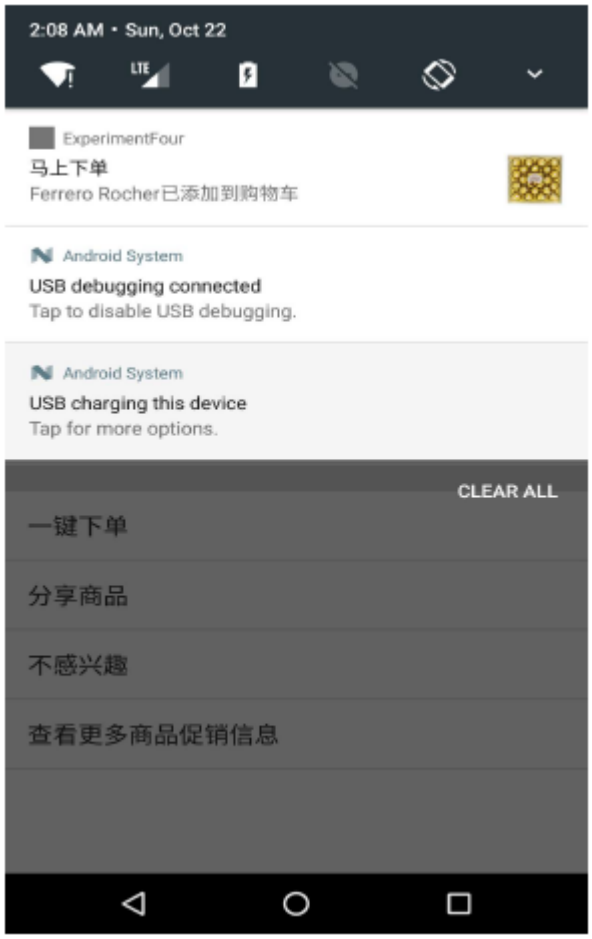
具体要求：

(1)在启动应用时，会有通知产生，随机推荐一个商品:

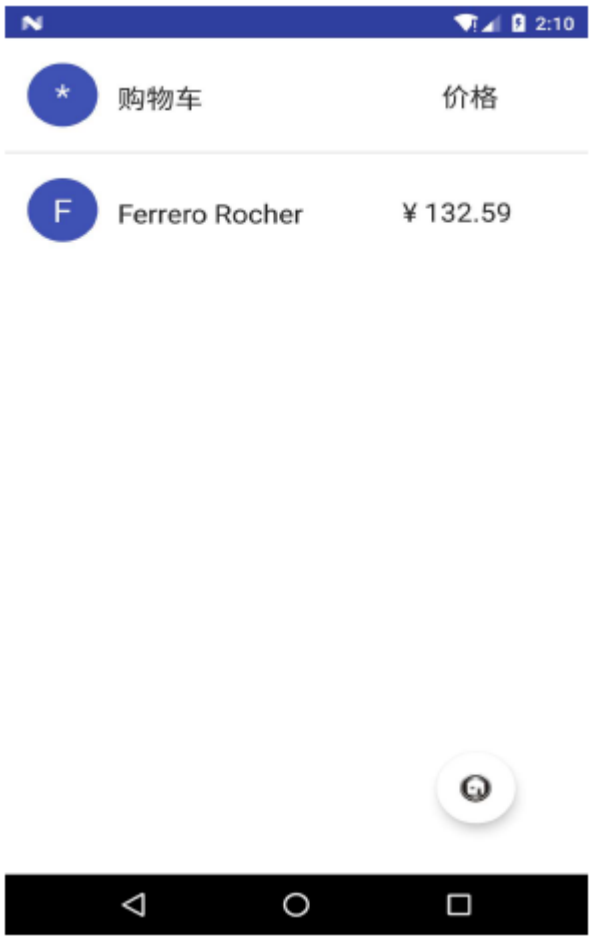


(2)点击通知跳转到该商品详情界面:

(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:



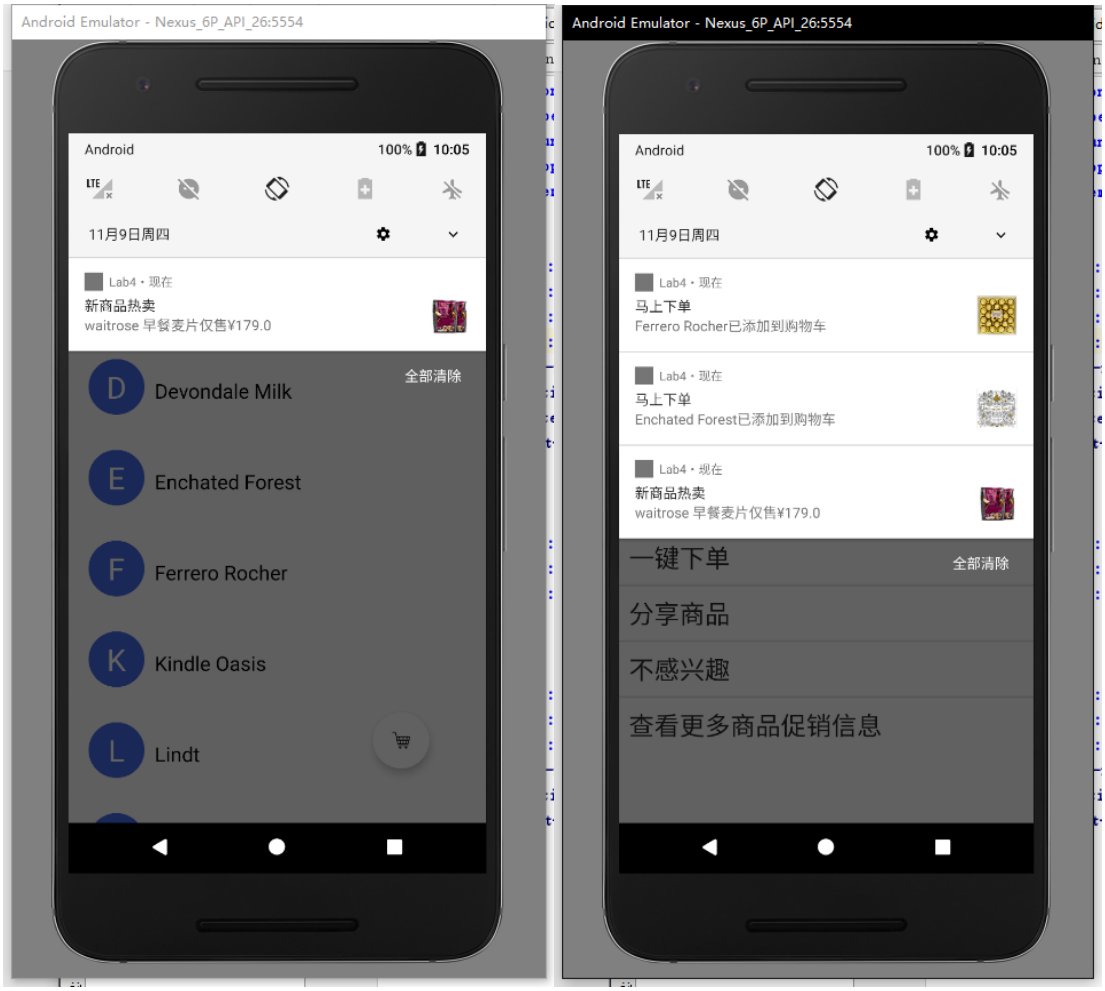
(4)点击通知返回购物车列表:



(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



（2） 实验步骤以及关键代码

首先需要对本次实验中广播接收器进行定义，在 AndroidManifest.xml 中加入静态广播接收器的定义及调用的 action 名并对包含购物车与商品列表的 activity 的 launchmode 属性进行改变。Activity 一共有以下四种 launchMode：1.standard；2.singleTop；3.singleTask；4.singleInstance。standard 启动模式，不管有没有已存在的实例，都生成新的实例。singleTop 启动模式，如果发现有对应的 Activity 实例正位于栈顶，则重复利用，不再生成新的实例。singleTask 模式，如果发现有对应的 Activity 实例，则使此 Activity 实例之上的其他 Activity 实例统统出栈，使此 Activity 实例成为栈顶对象，显示到幕前。singleInstance 模式非常接近于 singleTask，系统中只允许一个 Activity 的实例存在。区别在于持有这个 Activity 的任务中只能有一个 Activity：即这个单例本身。

```
<activity
    android:name="ares_android.lab4.MainActivity"
    android:enabled="true"
    android:exported="true"
    android:launchMode="standard">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity
    android:name="ares_android.lab4.details"
    android:enabled="true"
    android:exported="true">
</activity>

<receiver
    android:name=".MyStaticReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="com.lab4.MyStaticFilter" />
    </intent-filter>
</receiver>
```

对于启动应用后发出的静态广播，首先需要重载 BoardcastReceiver 的 onReceive 成员函数，使得接收到该静态广播后发出商品推荐通知。创建 notification 时需要首先创建新的 notification manager、通知的创建类 builder 以及发出该 notification 所需要的信道 channel，然后设置 builder 的各项属性，包括文字信息、通知标题、图标和点击事件。点击事件的设置中采用了 penddingintent 类，penddingintent 类不会随着 activity 的消失而被摧毁，只要该 boardcastReceiver 不被摧毁，该 penddingintent 就会一直存在，该通知点击事件依然有效。

```
public class MyStaticReceiver extends BroadcastReceiver {
    private Merchandise tempMerchandise;
    @RequiresApi(api = Build.VERSION_CODES.O)
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("com.lab4.MyStaticFilter")) {
            tempMerchandise = (Merchandise)intent.getSerializableExtra( name: "Merchandise");
            NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
            NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
            NotificationChannel mChannel = new NotificationChannel( id: "MyChannel", name: "channel",NotificationManager.IMPORTANCE_LOW);
            mChannel.enableLights(true);
            manager.createNotificationChannel(mChannel);
            Bitmap bm = BitmapFactory.decodeResource(context.getResources(), tempMerchandise.getPicname());
            Intent to_details = new Intent(context, details.class);
            to_details.putExtra( name: "Merchandise", tempMerchandise);
            builder.setContentTitle("新商品热卖")
                    .setContentText(tempMerchandise.getName()+"仅售¥"+Double.toString(tempMerchandise.getPrice()))
                    .setTicker("您有一条新消息")
                    .setSmallIcon(tempMerchandise.getPicname())
                    .setLargeIcon(bm)
                    .setAutoCancel(true)
                    .setChannelId("MyChannel")
                    .setContentIntent(PendingIntent.getActivity(context, requestCode: 0, to_details, PendingIntent.FLAG_UPDATE_CURRENT));
            Notification notify = builder.build();
            manager.notify( id: 0, notify);
        }
    }
}
```

点击商品推荐通知时需要直接跳转到对应的商品页面，即跳转到另一个 activity，利用 bundle 绑定商品数据传入新的商品详情 activity 中即可。

同理对于商品详细页面点击购物车时发出动态广播并发出马上下单通知，也要先重载 BoardcastReceiver 后再发出对应的 notification。

```
public class MyDynamicReceiver extends BroadcastReceiver {
    private Merchandise tempMerchandise;
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("com.lab4.MyDynamicFilter")) {
            tempMerchandise = (Merchandise)intent.getSerializableExtra( name: "Merchandise");
            NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
            NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
            NotificationChannel mChannel = new NotificationChannel( id: "MyChannel", name: "channel",NotificationManager.IMPORTANCE_LOW);
            mChannel.enableLights(true);
            manager.createNotificationChannel(mChannel);
            Bitmap bm = BitmapFactory.decodeResource(context.getResources(), tempMerchandise.getPicname());
            Intent back_to_carts = new Intent(context, MainActivity.class);
            Bundle senddata = new Bundle();
            senddata.putSerializable("AddItem",tempMerchandise);
            senddata.putString("Backto","ListView");
            back_to_carts.putExtras(senddata);
            builder.setContentTitle("马上下单")
                    .setContentText(tempMerchandise.getName()+"已添加到购物车")
                    .setTicker("您有一条新消息")
                    .setSmallIcon(tempMerchandise.getPicname())
                    .setLargeIcon(bm)
                    .setAutoCancel(true)
                    .setChannelId("MyChannel")
                    .setContentIntent(PendingIntent.getActivity(context, requestCode: 0, back_to_carts,PendingIntent.FLAG_CANCEL_CURRENT));
            Notification notify = builder.build();
            manager.notify(rc_id, notify);
        }
    }
}
```

对于动态广播的使用还需要在 java 文件中创建对应的动态广播接收器，创建新的 Intent 类并绑定相关商品数据后发出动态广播。

```
IntentFilter dynamic_filter = new IntentFilter();
dynamic_filter.addAction("com.lab4.MyDynamicFilter");
myDynamicReceiver = new MyDynamicReceiver();
registerReceiver(myDynamicReceiver, dynamic_filter);
intent.putExtra( name: "AddItem",tempMerchandise);
Intent intentBroadcast = new Intent( action: "com.lab4.MyDynamicFilter");
intentBroadcast.putExtra( name: "Merchandise", tempMerchandise);
sendBroadcast(intentBroadcast);
```

两种广播有如下区别：

生存期，静态广播的生存期可以比动态广播的长很多，而动态广播会随着 context 的终止而终止；优先级动态广播的优先级比静态广播高；

动态广播无需在 AndroidManifest.xml 中声明即可直接使用，也即动态；而静态广播则需要，有时候还要在 AndroidManifest.xml 中加上一些权限的声明。

（3） 实验遇到困难以及解决思路

主要困难在于如何恰当地组织实现需要的功能，从功能到具体架构实现，采用什么样的数据结构或者框架去使得最终的代码能够在期望的流程中正常的执行 activity 的切换与广播发出与接收、以及 notification 的发出。尤其是对于马上下单的通知点击事件，如何正确的保证无论在哪个 activity、无论是否在应用内，只要点击该通知就跳转到目的 activity 并且设置商品列表的 RecycleView 不可见、仅购物车的 ListView 可见。之前存在一个误区，尝试往 Intent 绑定多种数据，忽略了 bundle 的使用，使得在对该点击事件的响应中每次都无法跳转至购物车页面。使用了 bundle 后能够将所有需要的数据传回目的 activity 中，对获得的信息响应设置两个 View 的可见性，完成实现。

四、 课后实验结果

为了实现每一次按下购物车按钮添加商品到购物车时，都能够发出一条是新的 notification，通过修改 notification manager 的 id，这个 id 就是系统对于每一条 notification 的独立标识，若不更改 id 则每次发出的 notification 都会被覆盖。通过修改后便能实现点击不同商品加入购物车时都会发出一条不被覆盖的新的 notification。

五、 实验思考及感想

本次实验看似相对简单，但为了实现最佳效果所花费的时间很多，主要是在对 android 设计机制的理解与消化上。对于 activity 多个运行阶段的理解与执行顺序，如 onCreate/onDestroy/onStart/onResume/onStop 等成员函数在什么情况下会被调用执行，都需要将我们的代码写入到合适的重载函数中，以实现预先期望的功能。