中山大學
SUN YAT-SEN UNIVERSITY

《手机平台应用开发（与 Google 共建）实验》

实验四：broadcast 使用
实验报告

学 院 名 称 ：	数据科学与计算机学院
专　　业	软件工程（计应）
学 生 姓 名 ：	张凯鑫
学　　号 ：	14331362
班　　级 ：	周三上午 4-5 节、周五下午 7-8 节

【实验目的】

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。

【实验内容】

实现一个 Android 应用，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）该界面为应用启动后看到的界面。

（2）点击静态注册按钮，跳转至如下界面（下图左），点击表单项目，如 orange，会有对应通知产生（下图右），点击通知返回主界面（上图）：

（3）点击动态注册按钮，跳转至如下界面。

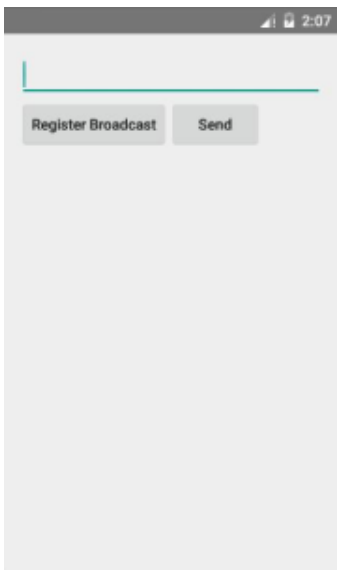
实现以下功能：

a) 可以编辑广播的信息，点击 Send 按钮发送广播。

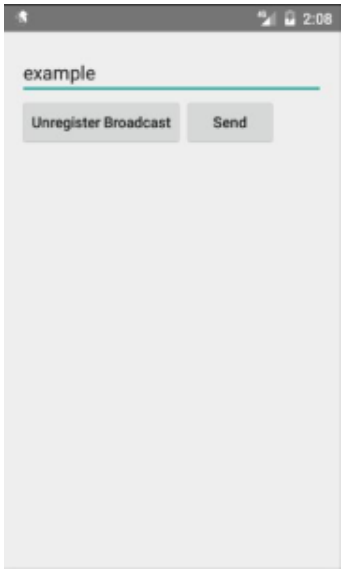
b) 设置一个按钮进行广播接收器的注册与注销。

c) 广播接收器若已被注册，发送出的广播信息会产生一个对应通知。

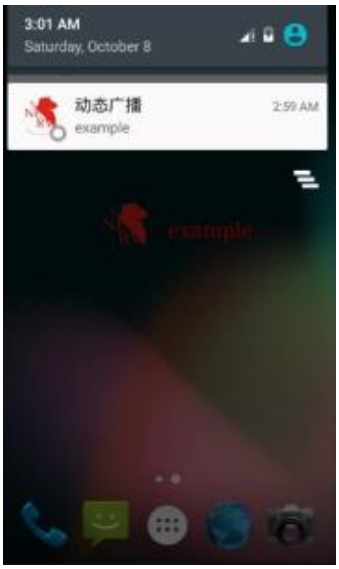
d) 点击 Notification 可以跳转回主界面。



（未注册广播）



（已注册广播）



（已注册后点击发送）

注：在设置按钮内容的时候注意大小写问题。

【实验过程】

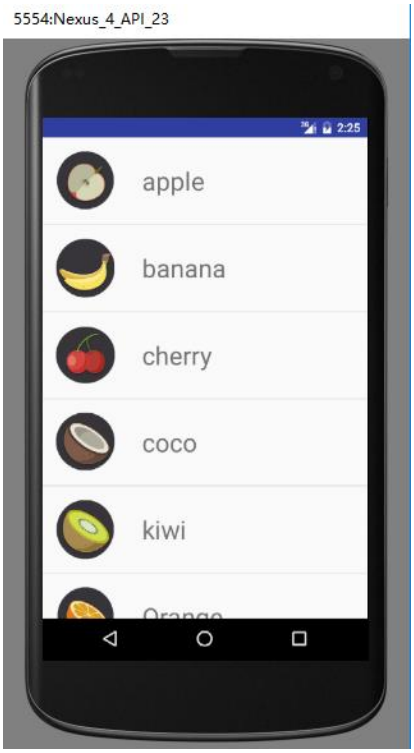
1. 创建新的 Android Studio 项目，命名为：ExperimentFour。
2. 主界面比较简单，为两个 Button，直接写出其 XML 和 java 文件（activity_main，MainActivity）。
3. 创建 java 文件(StaticActivity)和 XML 文件(static_layout)，并使用控件 ListView，创建自定义 Adapter（java 文件 Fruit 和 FruitAdapter），对所给的图片使用 Image assert 创建位图 BitMap，在 AndroidManifest 文件中注册静态广播，设置表单的点击事件为发送静态广播，创建 java 文件（StaticReceiver）用于实现广播接收器，接收所点击项的水果名和水果图，广播的事件处理可以暂时处理为 Toast。
4. 创建 java 文件(DynamicActivity)和 XML 文件(dynamic_layout)，使用 RelativeLayout 布局，RegisterBroadcast 和 Send 按键的点击事件为发送动态广播，创建 java 文件（DynamicRece 的）用于于实现广播接收器，接收所发送的信息，广播的事件处理先暂时处理为 Toast。
5. 学习 Notification，并将广播事件处理的 Toast 更改为 Notification。
6. 设置 Notification 的点击事件，将所有 Activity 的 launchMode 设置为“singleInstance”。
7. 实验代码调整。

【实验结果】

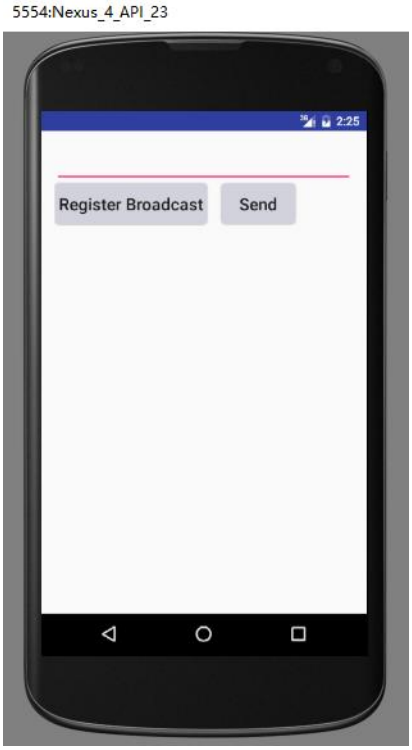
实验效果图如下（实验代码详见 lab4_code 文件夹）：

主界面

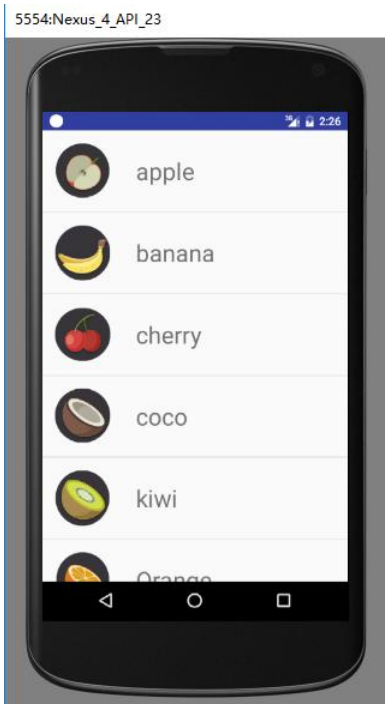
静态注册界面

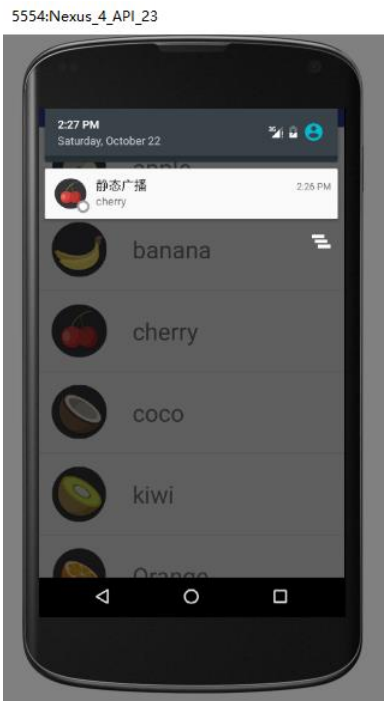


动态注册界面



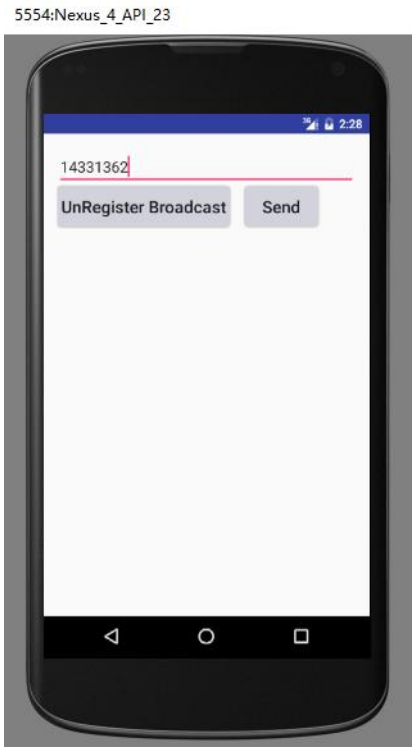
静态注册示例：点击表单中任一项（如 cherry）发送静态广播,可以看到屏幕左上角出现一个小图标（下图左），这个就是接收到广播的标志。下拉通知栏察看广播消息，可以看到广播的大图标为所点击项的图标，小图标由于 API 不同，无法看清小图标，广播内容为所点击项的内容（下图右）。点击通知，跳转到主界面。



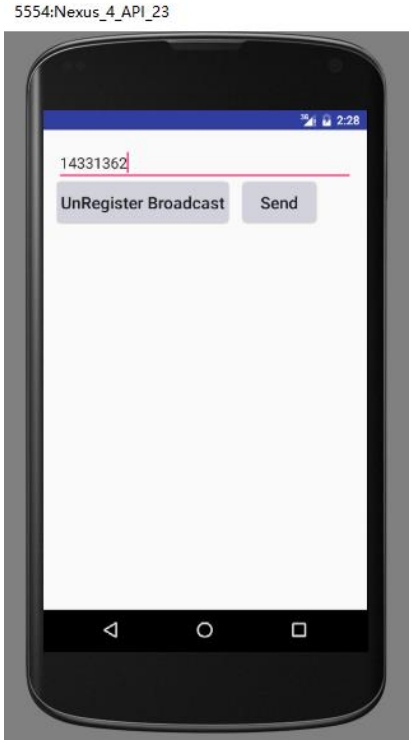


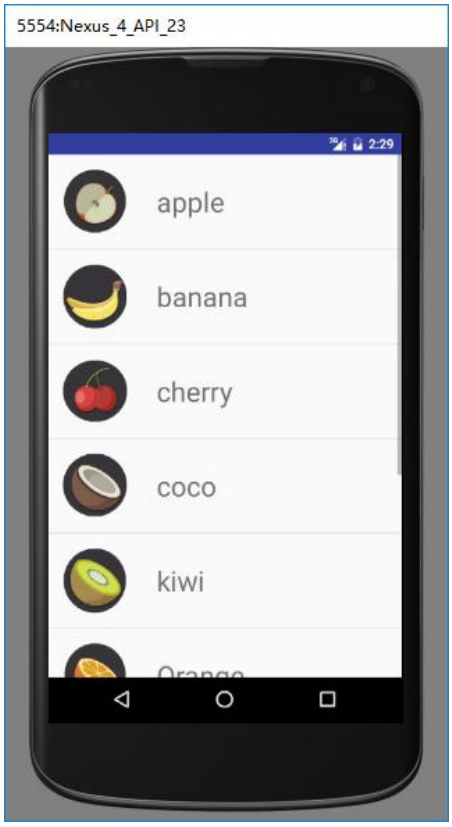
动态注册示例：点击按钮“Register Broadcast”注册动态广播，按钮文字变换为“UnRegister Broadcast”，输入框中输入要广播的信息,（如学号 14331362），点击按钮”Send”发送动态广播。此时可以在屏幕的左上角看到一个小图标，这就是接收到广播信息的反应（下图左），下拉手机通知栏，查看动态广播信息（下图右）。点击动态广播通知，跳转到主界面。

从主界面（下图左）再次进入动态注册界面，所看到的界面与原先一致，如下图右所示，因为界面的 launchMode 为 singleInstance，再次进入时不会另外新建一个 Activity，而是将栈中的所选的 Activity 移动到栈顶。点击按钮“UnRegister Broadcast”注销广播，此时再点击按钮”Send“，广播无法成功发送，左上角没有出现小图标，表明接收不到广播，广播发送失败。因为广播被注销，没有注册广播无法发送广播。



点击模拟器上界面的虚拟返回按钮，界面活动 Activity 依次被销毁，返回到的的界面依次如下：





即证明 Activity 栈中只有三个 Activity，分别为主界面、静态注册界面、动态注册界面对应的 Activity，在点击 Notification 通知栏消息和其他按钮跳转时，程序不是新建 Activity，而是因为这些 activity 返回栈是单独的，程序调用某 activity 是把栈中对应的 activity 移动到栈顶，其 launchMode 的值为 singleInstance，符合实验要求。

【遇到的问题与解决方法】

1. 实验过程发现静态注册的界面标题栏忘了去掉。使用如下代码去掉静态注册界面标题栏，

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
```

但发现一直不起作用，后来发现是代码问题,将代码

```
public class StaticActivity extends AppCompatActivity
```

改为：

```
public class StaticActivity extends Activity
```

问题得以解决。

2. Notification 使用过程中，不清楚课件中 setLargeIcon(bm)中 bm 是指什么，查了文档后知道是 BitMap 类型，问题在于如何将图片的 int 型转化为 BitMap 类型，最后在网上查到函数:

```
Bitmap bm = BitmapFactory.decodeResource(context.getResources(),imageId);
```

问题解决。

3. Notification 的点击跳转处理，由于课件所给资料特别少，只是提及到 PendingIntent 和 setContentIntent，于是搜索了相关信息，得到方法如下：

```
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent,
PendingIntent.FLAG_CANCEL_CURRENT);
Notification.Builder builder = new Notification.Builder(context);
```

```
builder.setContentIntent(pendingIntent);
```

然而由于 Notification 点击事件是在 onReceive(Context context, Intent intent)中处理的，传入的值有一个为 intent，与 PendingIntent 中的 intent 本来不同，但被我错误地以为是相同的，之后一直无法正确跳转，后来想清楚了，更改并增加了代码：

```
Intent intent1 = new Intent(context, MainActivity.class);
```

点击后可以成功跳转到主界面。

【实验心得与体会】

1. 广播的静态注册是在配置文件 AndroidManifest.xml 中，使用标签<receiver></receiver>，需要写的内容是<action android:name=”” ></action>,只有注册了广播，才可以发送广播。与静态注册不同的是，动态注册的关键代码如下：

```
dynamicReceiver = new DynamicReceiver();
IntentFilter dynamic_filter = new IntentFilter();
dynamic_filter.addAction("com.example.kaixin.experimentfour.dynamicreceiver
");
registerReceiver(dynamicReceiver, dynamic_filter);
```

2. 静态注册的广播在 app 运行后就注册，一经注册，无法注销；动态注册的广播在运行到相关代码时才注册，而且可以注销，代码如下：

```
unregisterReceiver(dynamicReceiver);
```

3. 活动的启动模式有四种：分别为 standard、singleTop、singleTask、singleInstance。一般默认为 standard，这种模式下每个出现的的 activity 都是新建的，栈中可能有多个相同的 activity。而 singleTop 模式下，在启动 activity 时如果发现返回栈的栈顶已经是该活动，则认为可以直接使用它，不会再创建新的 activity 实例。singleTask 模式会使得在整个应用程序的上下文中，某个活动只存在一个实例，当要用到这个 activity 时，在这个 activity 之上的所有 activity 会统统出栈，如果没有则新建该 activity。本次实验使用的是 singleInstance 模式，感觉比较复杂，有一个单独的返回栈来管理某 activity，不管是哪个应用程序来访问它，都共用同一个返回栈。个人觉得可以理解为，每次使用时，直接从栈中把该 activity 调到了栈顶，其他 activity 不出栈。
4. 本次实验要实现的界面不多，但要掌握的内容比较多，静态注册广播、动态注册广播、Notification 的使用，又同时结合了之前三次实验的内容：UI 界面设计、事件处理、ListView 等，成功做完实验还是很有成就感的。