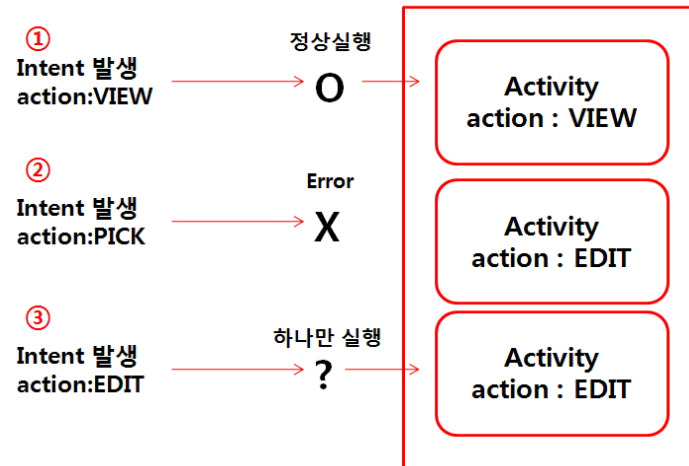


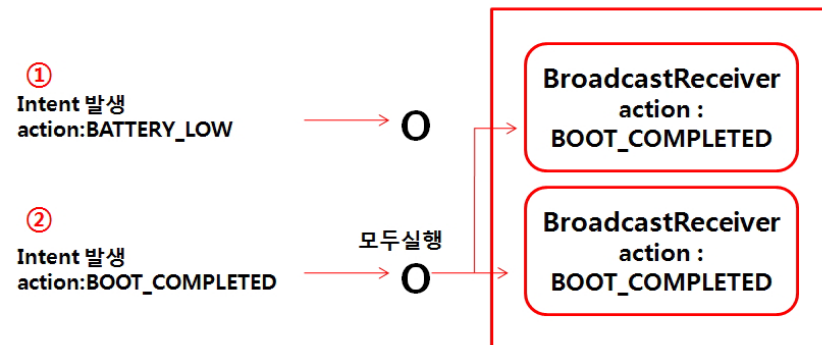
19.1 BroadcastReceiver

19.1.1. 브로드캐스트 리시버 이해

- 이벤트 모델로 수행되는 컴포넌트
- 액티비티 인텐트의 동작 원리



- 브로드캐스트 리시버



19.1 BroadcastReceiver

19.1.2. 브로드캐스트 리시버 작성 방법

- BroadcastReceiver를 상속받은 클래스

```
public class MyReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Toast toast=Toast.makeText(context, "I am BroadcastReceiver",  
        toast.show();  
    }  
}
```

Toast.LENGTH_SHORT);

- AndroidManifest.xml 파 일에 등록

```
<receiver  
    android:name=".MyReceiver"  
    android:enabled="true"  
    android:exported="true"></receiver>
```

- 인텐트를 발생

```
Intent intent=new Intent(this, MyReceiver.class);  
sendBroadcast(intent);
```



19.1 BroadcastReceiver

19.1.3. 시스템 브로드캐스트 인텐트

- 부팅 완료

```
<receiver android:name=".MyReceiver">  
  <intent-filter>  
    <action android:name="android.intent.action.BOOT_COMPLETED"/>  
  </intent-filter>  
</receiver>
```

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

- 화면 On/Off
- 액티비티, 서비스 등의 코드에서 동적으로 등록해야만 실행

```
BroadcastReceiver brOn=new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Log.d("kkang", "screen on.....");  
    }  
};
```

```
registerReceiver(brOn, new IntentFilter(Intent.ACTION_SCREEN_ON));
```

```
unregisterReceiver(brOn);
```



19.1 BroadcastReceiver

- 전화 수신/발신

```
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

```
<receiver android:name=".MyReceiver">
    <intent-filter>
        <action android:name="android.intent.action.NEW_OUTGOING_CALL" />
        <action android:name="android.intent.action.PHONE_STATE" />
    </intent-filter>
</receiver>
```

```
if (action.equals("android.intent.action.NEW_OUTGOING_CALL")) {
    String phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
    //...} else if (action.equals("android.intent.action.PHONE_STATE")){
    Bundle bundle = intent.getExtras();
    String state = bundle.getString(TelephonyManager.EXTRA_STATE);
    String phoneNumber = bundle.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
    //...}
```



19.1 BroadcastReceiver

배터리

- android.intent.action.BATTERY_LOW: 배터리가 낮은 상태가 되었을 때
- android.intent.action.BATTERY_OKAY: 배터리가 낮은 상태에서 벗어날 때
- android.intent.action.BATTERY_CHANGED: 충전 상태가 변경되었을 때
- android.intent.action.ACTION_POWER_CONNECTED: 외부 전원공급이 연결되었을 때
- android.intent.action.ACTION_POWER_DISCONNECTED: 외부 전원공급이 끊어질 때

```
registerReceiver(batteryReceiver, new IntentFilter(Intent.ACTION_POWER_CONNECTED));  
registerReceiver(batteryReceiver, new IntentFilter(Intent.ACTION_POWER_DISCONNECTED));
```

```
BroadcastReceiver batteryReceiver=new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        String action=intent.getAction();  
        if(action.equals(Intent.ACTION_POWER_CONNECTED)){  
            addItem("ON CONNECTED.....");  
        }else if(action.equals(Intent.ACTION_POWER_DISCONNECTED)){  
            addItem("ON DISCONNECTED.....");  
        }  
    }  
};
```



19.1 BroadcastReceiver

- 배터리 상황을 파악

```
IntentFilter ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);  
Intent batteryStatus = registerReceiver(null, ifilter);
```

- 스마트폰에 전원이 공급되고 있는지를 파악

```
int status = batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);  
boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING ;
```

- 전원 공급의 유형을 파악

```
int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);  
boolean usbCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_USB;  
boolean acCharge = chargePlug == BatteryManager.BATTERY_PLUGGED_AC;
```

- 몇 퍼센트 충전된 상황인지를 파악

```
int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);  
int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);  
  
float batteryPct = (level / (float)scale) * 100;
```



19.1 BroadcastReceiver

19.1.4. 백그라운드 서비스 제한 (Android Oreo 변경사항)

- Android Oreo(API Level 26)부터는 백그라운드 실행에 제한
- 백그라운드 서비스 제한과 브로드캐스트 제한
- 브로드캐스트 리시버의 백그라운드 제약은 암시적 인텐트에 의한 리시버의 실행 제한
- 앱에서 암시적 방법으로 BroadcastReceiver를 실행하는 부분만 제한
- 리시버가 백그라운드에서 암시적인 방법으로 실행되는 것을 제한, 리시버가 백그라운드가 아니라면 암시적 인텐트로도 잘 실행

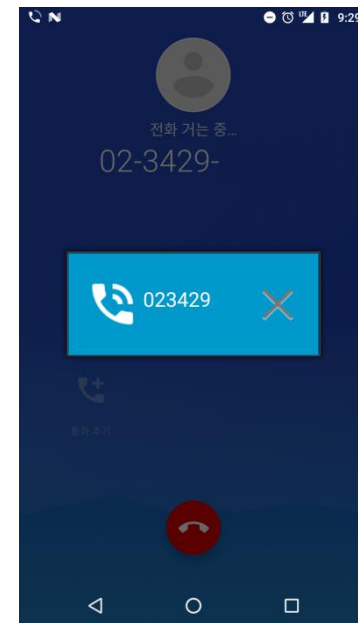
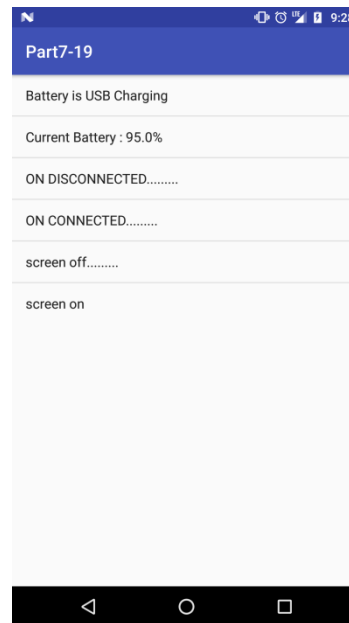


Step by Step 19-1 – 브로드캐스트 리시버

시스템에서 실행시키는 Intent에 반응할 BroadcastReceiver 테스트

•전화 수신/발신을 감지, 폰의 각종 상태 파악 하는 BroadcastReceiver

1. Module 생성
2. Component 생성
3. 파일 복사
4. AndroidManifest.xml 작업
5. MyReceiver.java 작성
6. MainActivity.java 작성
7. 실행



19.2 Notification

19.2.1. 알림의 기본 구성

- 알림(Notification)은 앱의 각종 상황을 사용자에게 알릴 목적으로 이용



support 라이브러리에서 NotificationCompat 클래스 이용 하위 호환성

- NotificationManager: 알림을 시스템에 발생시키는 SystemService
 - Notification: 알림 구성 정보를 가지는 객체
 - NotificationCompat.Builder: 알림을 다양한 정보로 생성
 - NotificationChannel: 알림의 관리 단위(Android Oreo에서 추가)
-
- NotificationManager

```
NotificationManager manager=(NotificationManager)getSystemService(NOTIFICATION_SERVICE);
```

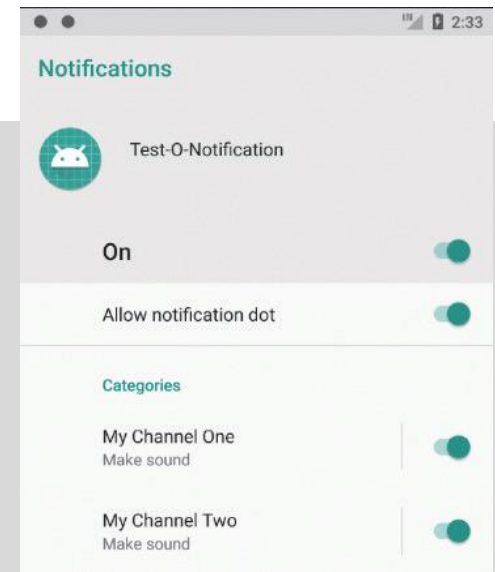


19.2 Notification

19.2.2. NotificationChannel (Android Oreo 추가사항)

- Notification 객체는 직접 생성되지 않으며 NotificationCompat.Builder로 생성 Android Oreo (API Level 26)부터는 Builder를 만드는 방법이 변경
- NotificationChannel이라는 개념이 추가되었으며 NotificationChannel에 의해서 Builder가 생성되게 변경
- NotificationChannel은 일종의 알림에 대한 관리 단위

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
    String channelId = "one-channel";  
    String channelName = "My Channel One";  
    String channelDescription = "My Channel One Description";  
    NotificationChannel channel = new NotificationChannel(channelId, channelName,  
        NotificationManager.IMPORTANCE_DEFAULT);  
    channel.setDescription(channelDescription);  
    //각종 채널에 대한 설정  
    channel.enableLights(true);  
    channel.setLightColor(Color.RED);  
    channel.enableVibration(true);  
    channel.setVibrationPattern(new long[]{100, 200, 300});  
    manager.createNotificationChannel(channel);  
    //channel이 등록된 builder  
    builder = new NotificationCompat.Builder(this, channelId);  
} else {  
    builder = new NotificationCompat.Builder(this);  
}
```



19.2 Notification

19.2.3. 기본적인 알림 구성

- `setSmallIcon`: 작은 아이콘 이미지 지정
- `setWhen`: 시간
- `setContentTitle`: 확장 내용의 타이틀 문자열
- `setContentText`: 확장 내용의 본문 문자열
- `setDefaults`: `DEFAULT_SOUND`, `DEFAULT_VIBRATE`, `DEFAULT_LIGHTS`을 함께 지정 가능
- `setAutoCancel`: 터치 시 자동 삭제 여부, `true` 값이 지정되면 터치 시 삭제됨
- `setOngoing`: 진행표시 여부, `true` 값이 설정되면 사용자가 손가락으로 밀어서 삭제 불가

```
builder.setSmallIcon(android.R.drawable.ic_notification_overlay);
builder.setWhen(System.currentTimeMillis());
builder.setContentTitle("Content Title");
builder.setContentText("Content Message");
builder.setDefaults(Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE);
builder.setAutoCancel(true);
```



19.2 Notification

- notify () 함수로 알림을 등록

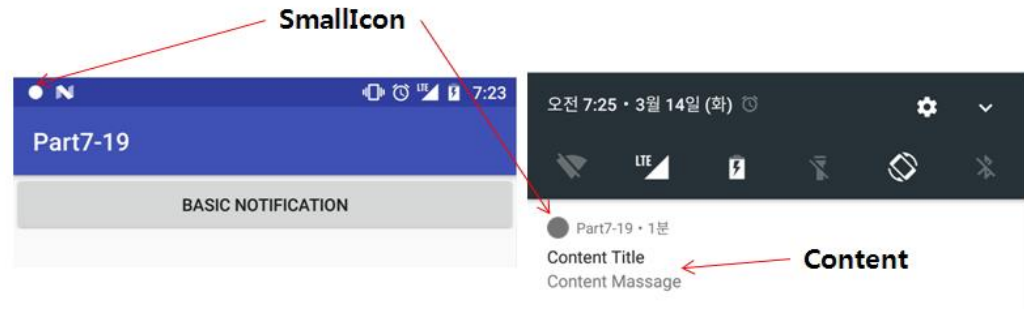
```
manager.notify(222, builder.build());
```

- 취소

```
manager.cancel(222);
```

- 이벤트를 처리

```
Intent intent=new Intent(this, MainActivity.class);
```



```
PendingIntent pIntent = PendingIntent.getActivity(this, 10, intent , PendingIntent.FLAG_UPDATE_CURRENT);
```

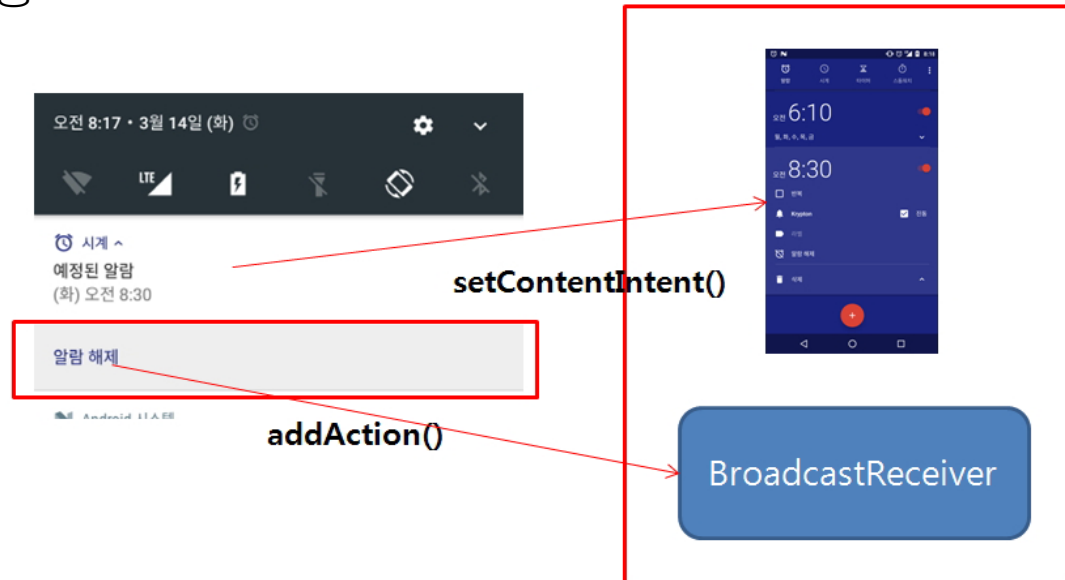
- FLAG_CANCEL_CURRENT: 이전에 생성한 PendingIntent는 취소하고 새로 만들
- FLAG_UPDATE_CURRENT: 현재의 내용으로 이번 객체를 업데이트
- FLAG_NO_CREATE: 새로운 PendingIntent 객체가 만들어지지 않고 이미 생성된 PendingIntent를 그대로 획득해서 사용 하기 위한 목적. 만약 만들어진 게 없다면 null 반환
- FLAG_ONE_SHOT: 한 번만 PendingIntent를 만들기 위해 사용. 이미 만들어진 게 있다면 fail 발생



19.2 Notification

19.2.4. 알림의 다양한 구성

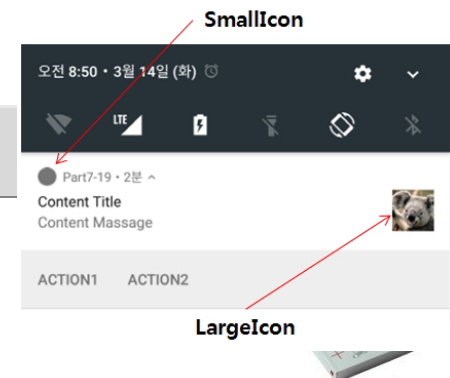
- 간단한 이벤트 : Action



```
builder.addAction(new NotificationCompat.Action.Builder(android.R.drawable.ic_menu_share, "ACTION1",  
pIntent1).build());
```

- 큰 아이콘 : setLargeIcon

```
Bitmap largeIcon= BitmapFactory.decodeResource(getResources(), R.drawable.noti_large);  
builder.setLargeIcon(largeIcon);
```

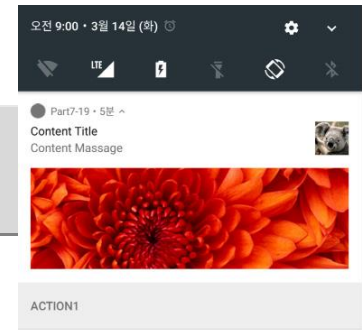


19.2 Notification

- 큰 이미지 : BigPictureStyle

```
NotificationCompat.BigPictureStyle bigStyle = new  
bigStyle.bigPicture(bigPicture);  
builder.setStyle(bigStyle);
```

```
NotificationCompat.BigPictureStyle(builder);
```

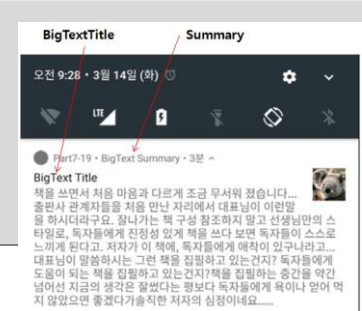


- 긴 문자열 : BigTextStyle

```
NotificationCompat.BigTextStyle bigTextStyle = new  
bigTextStyle.setSummaryText("BigText Summary");  
bigTextStyle.setBigContentTitle("BigText Title");  
bigTextStyle.bigText("책을 쓰면서.....");
```

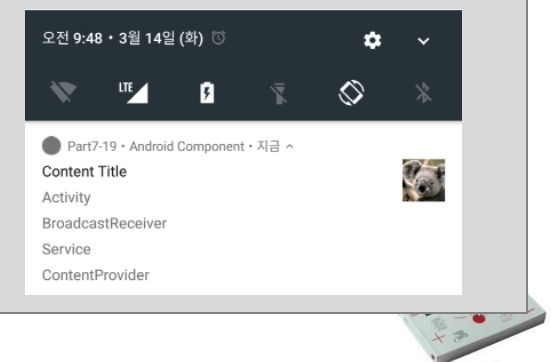
```
NotificationCompat.BigTextStyle(builder);
```

```
builder.setStyle(bigTextStyle);
```



- 목록 : InboxStyle

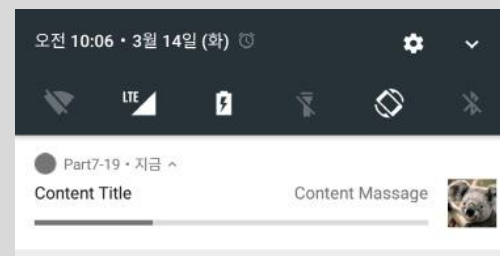
```
NotificationCompat.InboxStyle style = new NotificationCompat.InboxStyle(builder);  
style.addLine("Activity");  
style.addLine("BroadcastReceiver");  
style.addLine("Service");  
style.addLine("ContentProvider");  
style.setSummaryText("Android Component");  
  
builder.setStyle(style);
```



19.2 Notification

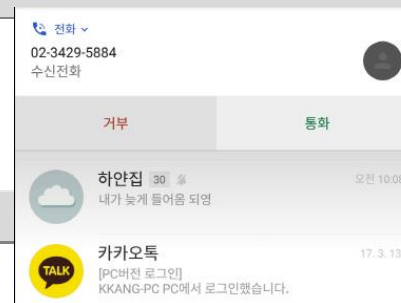
- 프로그레스 : Progress

```
Runnable runnable=new Runnable() {  
    @Override  
    public void run() {  
        for(int i=1 ;i<=10 ;i++){  
            builder.setAutoCancel(false);  
            builder.setOngoing(true);  
            builder.setProgress(10, i, false);  
            manager.notify(222, builder.build());  
            if(i>=10){  
                manager.cancel(222);  
            }  
            SystemClock.sleep(1000);  
        }  
    }  
};  
Thread t=new Thread(runnable);  
t.start();
```



- 상단에 띄우기 : Heads Up

```
builder.setFullScreenIntent(pIntent, true);
```



Step by Step 19-2 – Notification

다양한 Notification 테스트

1. Activity 생성
2. NotiReceiver 생성
3. 파일 복사
4. NotiReceiver 작성
5. Lab19_2Activity 작성
6. Lab19_2Activity.java 실행

