자바응용SW(앱)개발자양성

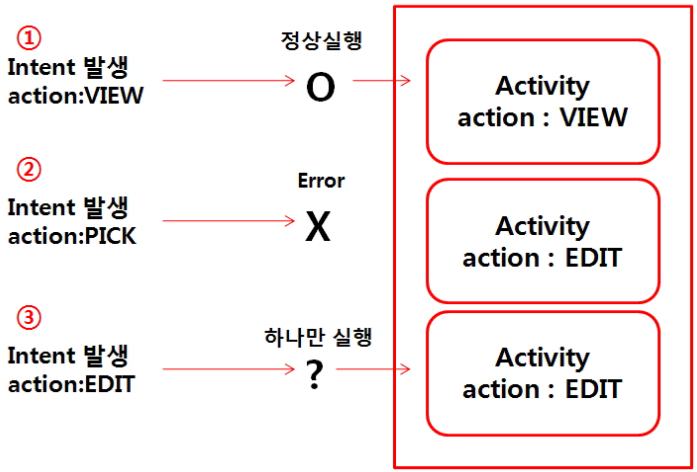
브로드캐스트 리시버와 여러가지 설정

백제직업전문학교

김영준 강사

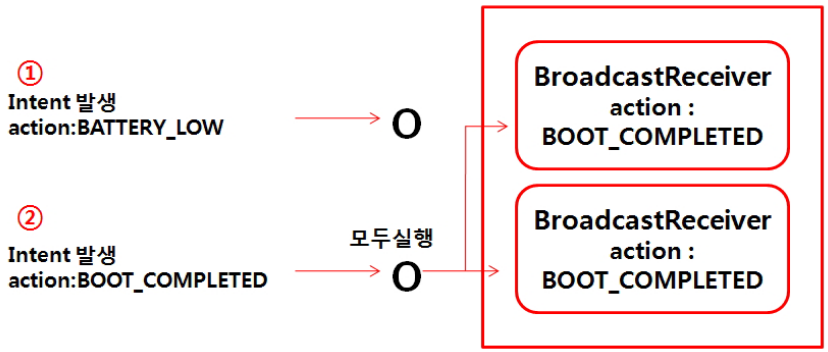
19.1.1. 브로드캐스트 리시버 이해

- 이벤트 모델로 수행되는 컴포넌트
- 액티비티 인텐트의 동 작 원리



- 브로드캐스트 리시버

: 인텐트가 발생하여 실행될
브로드캐스트 리시버가 없다고 하더라도
에러가 발생하지 않으며, 실행될 브로드
캐스트 리시버가 여러 개라면 모두 실행됨



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.LENGTH_SHORT);
        toast.show();
    }
}
```

- AndroidManifext.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);
```

19.1.2. 브로드캐스트 리시버 작성 방법

- 로컬 방송 수신

매니페스트가 아닌 리시버를 동적으로 등록하는 방식으로 해당 액티비티가 동작중일 때만 동작하는 리시버이다

특히 Oreo부터는 미리 정의된 방송은 로컬 방송 수신 방법을 사용해야만 수신된다

```
public class MainActivity extends AppCompatActivity {
    private BroadcastReceiver mReceiver;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mReceiver = new MyReceiver();
    }
    @Override
    protected void onResume() {
        super.onResume();
        IntentFilter filter = new IntentFilter();
        filter.addAction(Intent.ACTION_POWER_CONNECTED);
        registerReceiver(mReceiver, filter);
    }
    @Override
    protected void onPause() {
        super.onPause();
        unregisterReceiver(mReceiver);
    }
}
```

19.1.2. 브로드캐스트 리시버 작성 방법

- 나만의 방송 보내기

```
public static final String MY_ACTION =
        "com.example.broadcastreceiverexam.action.ACTION_MY_BROADCAST";
```

나만의 방송 이름을 정의하고

```
public void sendMyBroadcast(View view) {
    Intent intent = new Intent(MyReceiver.MY_ACTION);
    sendBroadcast(intent);
}
```

sendBroadcast()를 사용하여 방송

19.1.2. 브로드캐스트 리시버 작성 방법

- 나만의 방송 보내기

```
public class MyReceiver extends BroadcastReceiver {
    public static final String MY_ACTION =
            "com.example.broadcastreceiverexam.action.ACTION_MY_BROADCAST";

    @Override
    public void onReceive(Context context, Intent intent) {
        if (Intent.ACTION_POWER_CONNECTED.equals(intent.getAction())) {
            Toast.makeText(context, "전원 연결 됨", Toast.LENGTH_SHORT).show();
        } else if (MY_ACTION.equals(intent.getAction())) {
            Toast.makeText(context, "이 방송은 나만의 방송", Toast.LENGTH_SHORT).show();
        }
    }
}
```

19.1.3. 시스템 브로드캐스트 인텐트 : 브로드캐스트리시버는 시스템에서 실행하는 인텐트에 반응하여 각종 시스템 상황을 감지하려고 자주 사용

- 부팅 완료 : 부팅 완료 시점에 앱이 동작하여 간단한 업무를 수행하거나 서비스를 구동해야 할 때 사용

```
<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 : 이 상황을 체크해서 특정 로직의 수행을 구동하거나 정지
- 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);
```

- 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" /> // 전화 발신하는 순간
<action android:name="android.intent.action.PHONE_STATE" />//전화 수신하는 순간을 감지(수신 상황)
    </intent-filter>
</receiver>
```

```
if (action.equals("android.intent.action.NEW_OUTGOING_CALL")) { //발신전화번호 얻음
    String phoneNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER);
    //...} else if (action.equals("android.intent.action.PHONE_STATE")){
    Bundle bundle = intent.getExtras();  //수신전화번호는 Bundle 객체를 통해 얻을 수 있음
    String state = bundle.getString(TelephonyManager.EXTRA_STATE);
    String phoneNumber = bundle.getString(TelephonyManager.EXTRA_INCOMING_NUMBER);
    //...}
```

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)){
            addListItem("ON CONNECTED.....");
        }else if(action.equals(Intent.ACTION_POWER_DISCONNECTED)){
            addListItem("ON DISCONNECTED.....");
        }
    }
};
```

BroadcastReceiver

- 배터리 상황을 파악(실제 브로드캐스트 리시버를 등록하는 구문이 아니라 시스템 배터리 상태의 정보값만 얻기 위한 구문. 이렇게 얻은 인텐트 객체에 대양한 배터리 정보가 담기게 됨

```
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

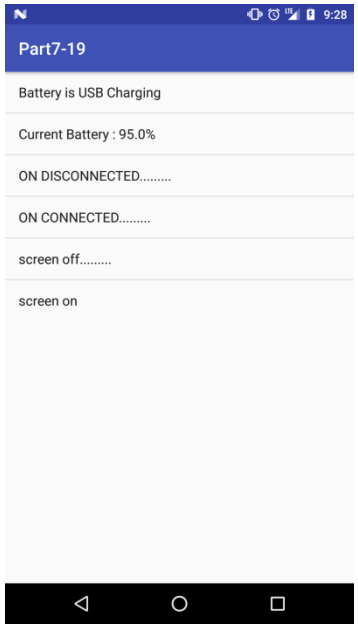
```
int level = batteryStatus.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
int scale = batteryStatus.getIntExtra(BatteryManager.EXTRA_SCALE, -1);

float batteryPct = (level / (float)scale) * 100;
```

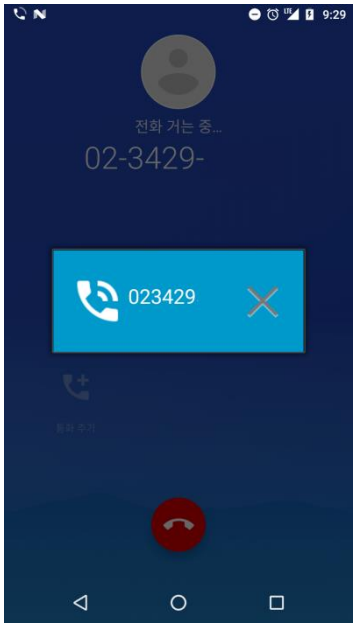
BroadcastReceiver

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





Notification

19.2.1. 알림의 기본 구성

- 알림(Notification)은 앱의 각종 상황을 사용자에게 알릴 목적으로 이용
  : 서비스 컴포넌트나 브로드캐스트 리시버에서는 화면으로 사용자에게 특정 상황이나 데이터를 보여줄 수 없으므로 알림을 이용하는 경우가 많다

support 라이브러리에서 NotificationCompat 클래스 이용 하위 호환성

- NotificationManager: 알림을 시스템에 발생시키는 SystemService
- Notification: 알림 구성 정보를 가지는 객체
- NotificationCompat.Builder: 알림을 다양한 정보로 생성

- NotificationManager : 각종 정보를 담고 시스템에 등록하려면 NotificationManager 필요

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

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_LIGHTS을 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 Massage");
builder.setDefaults(Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE);
builder.setAutoCancel(true);
```

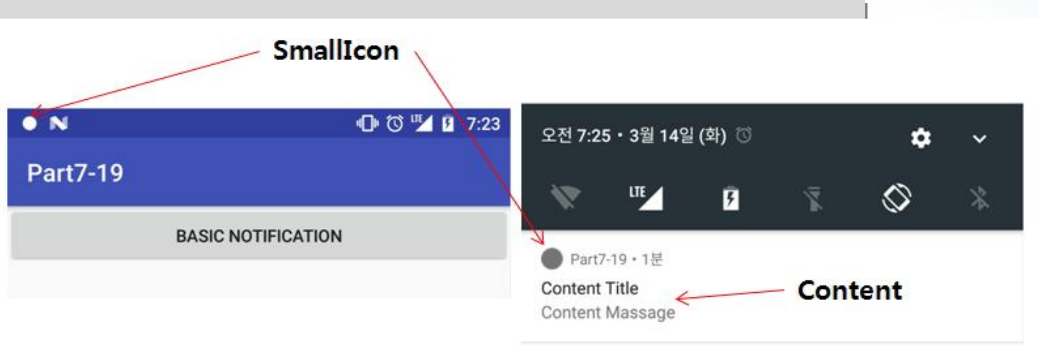
Notification

- 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);
```

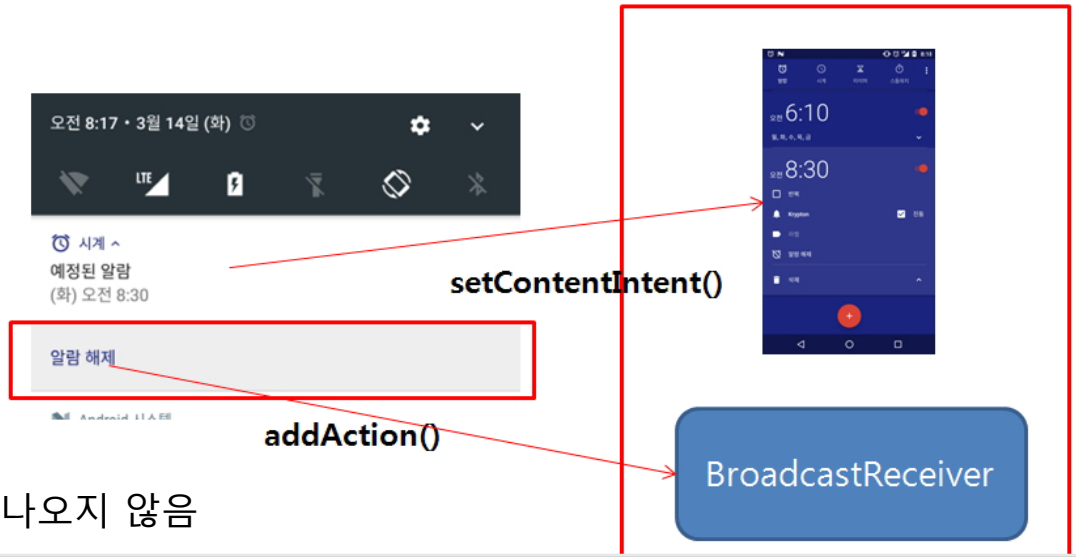
인텐트는 우리가 준비했지만 인텐트를 발생시키는 곳은 시스템이므로 준비된 인텐트 발생을 시스템에 의뢰

- FLAG_CANCEL_CURRENT: 이전에 생성한 PendingIntent는 취소하고 새로 만듦
- 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

Notification

19.2.4. 알림의 다양한 구성

- 간단한 이벤트 : Action

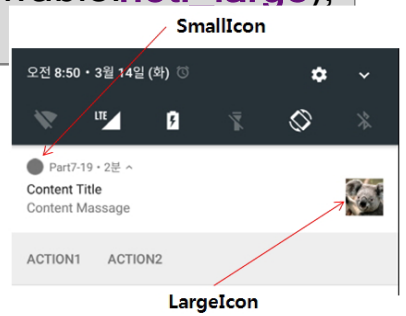


- 때로는 알림을 터치할때 화면 전환없이 알림 내에서 간단한 이벤트를 처리할 수 있음
- 누가버전 이후부터는 아이콘이 나오지 않음

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

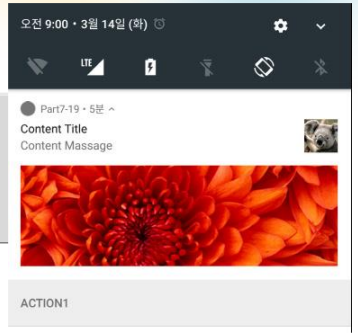
- 큰 아이콘 : setLargeIcon

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



Notification

- 큰 이미지 : BigPictureStyle-알림에 큰 크기의 이미지를 출력하기 위한 스타일



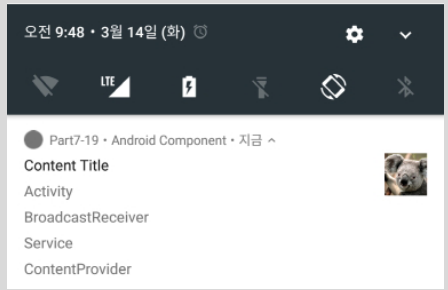

```
NotificationCompat.BigPictureStyle bigStyle = new NotificationCompat.BigPictureStyle(builder);
bigStyle.bigPicture(bigPicture); //Bitmap 객체 설정
builder.setStyle(bigStyle); //준비한 스타일을 적용하는 구조
```

- 긴 문자열 : BigTextStyle-화면을 전환하지 않고도 긴 문자열을 출력

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

builder.setStyle(bigTextStyle);
```

- 목록 : InboxStyle -알림에 여러 데이터를 목록 형태로 제공해야 할때 지원

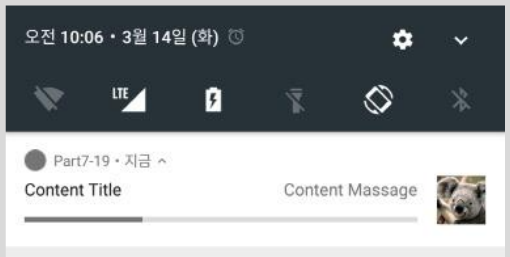
```
NotificationCompat.InboxStyle style = new NotificationCompat.InboxStyle(builder);
style.addLine("Activity");
style.addLine("BroadcastReceiver");
style.addLine("Service");
style.addLine("ContentProvider");
style.setSummaryText("Android Component");

builder.setStyle(style);
```



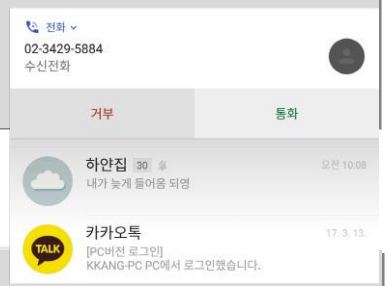
Notification

- 프로그레스 : Progress-작업의 진행 사항을 표시

```
Runnable runnable=new Runnable() {
    @Override
    public void run() {
        for(int i=1 ;i<=10 ;i++){
            builder.setAutoCancel(false); //프로그레스바를 알림에 표현하려면 false
            builder.setOngoing(true);//사용자가 알림을 없애지 못하게 함
            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-화면에 확장 내용이 떠 있는 듯하게 바로 나오게 하고 싶은 경우 화면을 그대로 유지한 상태에서 사용자의 행위를 최대한 방해X




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

다양한 Notification 테스트

키보드 제어

- 키보드 보이기와 숨김

```
InputMethodManager manager=(InputMethodManager)getSystemService(INPUT_METHOD_SERVICE);
```

- showSoftInput(View view, int flags): 키보드 보임
- hideSoftInputFromWindow(IBinder windowToken, int flags): 키보드 숨김

```
if(v==hideBtn){
   manager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
InputMethodManager.HIDE_NOT_ALWAYS);
}else if(v==showBtn) {
   manager.showSoftInput(editText, InputMethodManager.SHOW_IMPLICIT);
}
```

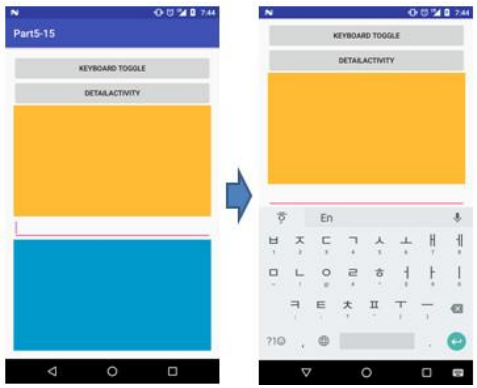
- toggleSoftInput(int showFlags, int hideFlags):

키보드 제어

키보드로 액티비티 화면 조정

- 기본(설정 안 된 경우): adjustUnspecified와 stateUnspecified가 적용
- adjustPan: 키보드가 올라올 때 입력 EditText에 맞춰 화면을 위로 올림
- adjustResize: 키보드가 올라올 때 액티비티의 크기 조정
- adjustUnspecified: 시스템이 알아서 상황에 맞는 옵션 설정
- stateHidden: Acitivty 실행 시 키보드가 자동으로 올라오는 것 방지
- stateVisible: Acitivty 실행 시 키보드가 자동으로 올라옴
- stateUnspecified: 시스템이 적절한 키보드 상태를 설정하거나 테마에 따라 설정
- adjustPan

```
<activity android:name=".Lab2Activity" android:windowSoftInputMode="adjustPan">
```

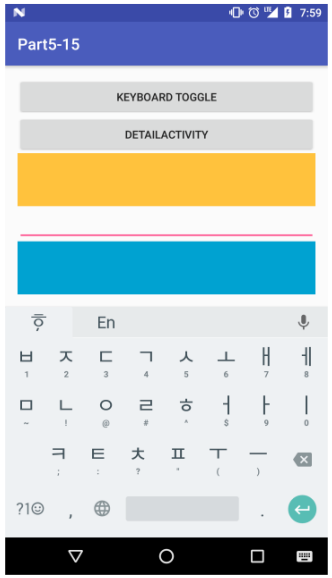


키보드 제어

- 초기에 키보드가 자동으로 올라오게 설정

```
<activity android:name=".Lab2Activity"        android:windowSoftInputMode="adjustPan|stateVisible">
```

- adjustResize

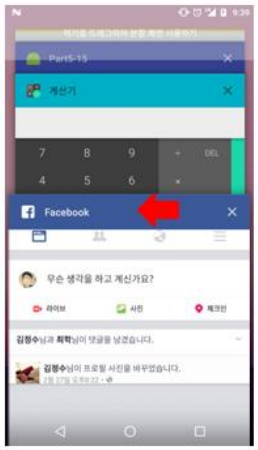



```
<activity android:name=".Lab2Activity"        android:windowSoftInputMode="adjustResize|stateHidden">
```

다중 창 지원

- 다중 창(Multi Window)은 API Level 24(Android 7.0)에 추가된 기능
- 스크린을 분할하여 한 화면에 두 앱을 동시에 띄울 수 있는 기능

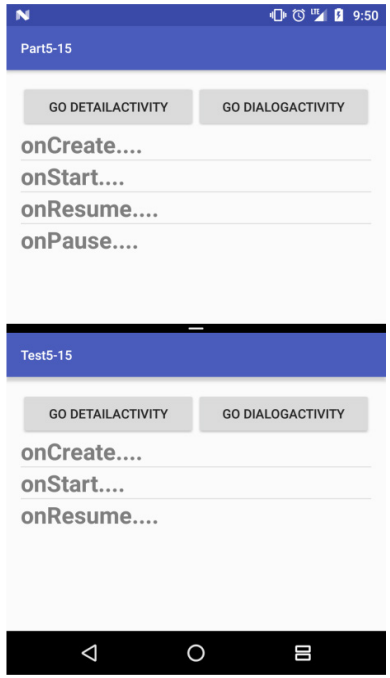
Overview Button Long Click



Overview 화면에서 Activity Title Long Click

다중 창 지원

- 다중 창 모드가 되었을 때의 생명주기



- 앱에서 다중 창을 지원하고 싶지 않으면 액티비티의 resizeableActivity 속성 값을 false로 지정

```
<activity android:name=".Lab2Activity"          android:windowSoftInputMode="adjustResize|stateHidden"
        android:resizeableActivity="false">
```

다중 창 지원

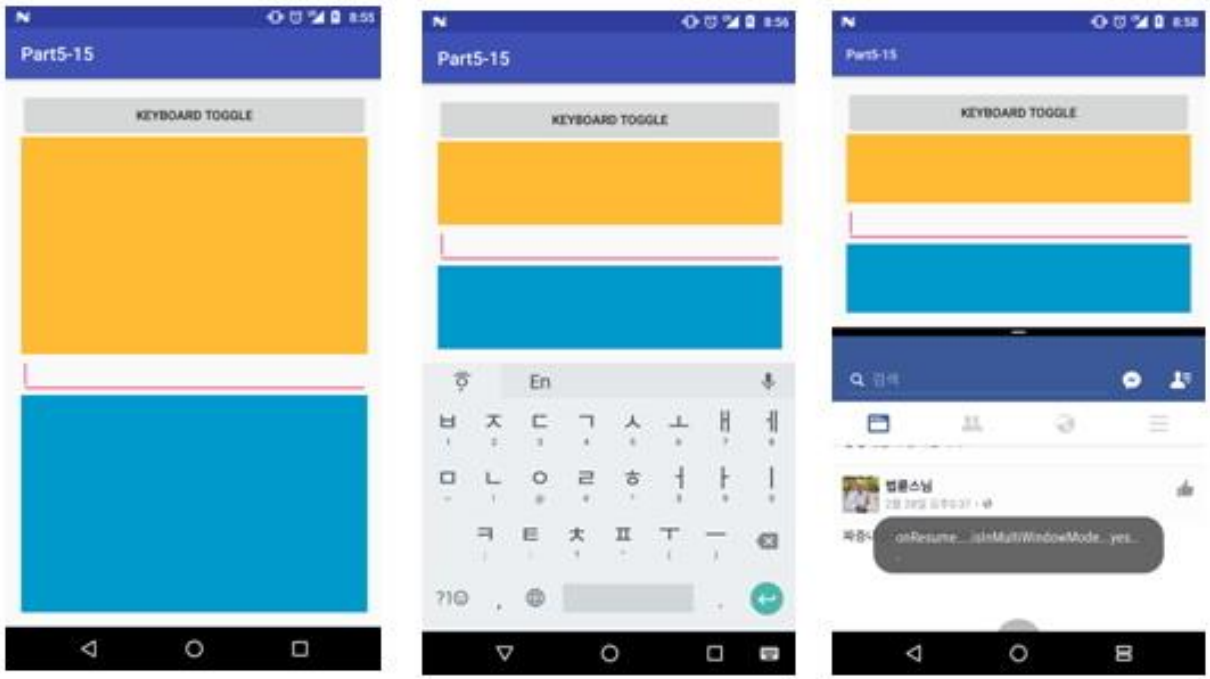
다중 창과 관련된 상황을 파악

- isInMultiWindowMode(): 액티비티가 다중 창 모드에 있는지 확인
- onMultiWindowModeChanged(): 액티비티가 다중 창 모드로 들어가거나 나올 때 콜백 함수

```
@Override
protected void onResume() {
    super.onResume();
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        if(isInMultiWindowMode()){
            showToast("onResume....isInMultiWindowMode...yes...");
        }
    }
}

@Override
public void onMultiWindowModeChanged(boolean isInMultiWindowMode) {
    super.onMultiWindowModeChanged(isInMultiWindowMode);
    showToast("onMultiWindowModeChanged....."+isInMultiWindowMode);
}
```

Step by Step – Activity 설정

Activity의 설정을 테스트



Step by Step – 서비스 및 브로드캐스트 설정

서비스 및 브로드캐스트 테스트

음악 Player이며 테스트 용이성을 위해 지정된 음악을 play, stop하고 ProgressBar에 currentTime 출력

Activity 가 종료되어 다른 앱을 수행하고 있더라도 음악이 계속 플레이

Activity와 Service 구조이며 두 Component간의 데이터를 BroadcastReceiver 방법으로 주고 받는것에 초점

플레이 하는 mp3 파일은 SDCard의 Music 폴더에 music.mp3 라는 이름의 파일이 있다는 가정하에 진행