Notification과 Alarm

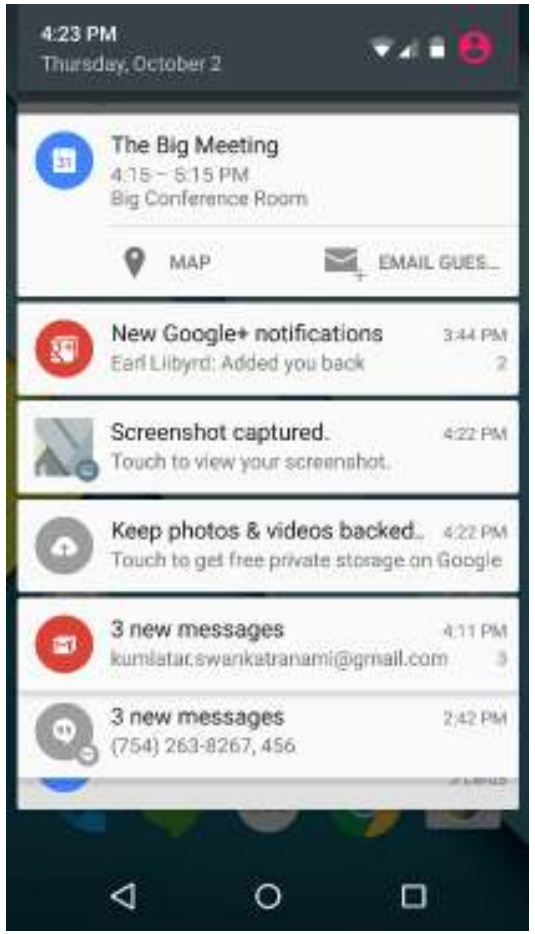
목차

Notification

개요

- 앱 내부가 아닌 외부(알림 영역)에 사용자에게 표시할 수 있는 메시지
- 아이콘 및 메시지를 사용하여 정보 표시
  - 다른 사용자로부터의 알림 확인 시
  - 작업 진행에 대한 적시 알림 시
- 세부 정보는 알림 창을 통해 확인
- 알림 영역과 알림 창은 시스템 제어 영역

•알림 창



•알림영역

- 헤드업 알림과 잠금화면 알림,
  앱 아이콘 배지 등도 사용

Notification 의 구성요소

- NotificationManager
  - 알림을 관리하는 시스템 제공클래스
  - getSystemService(NOTIFICATION_SERVICE);
  - 생성한 Notification을 전달 받아 알림 실행
- Notification
  - 알림을 나타내는 클래스
  - 알림에 대한 UI 정보와 작업을 표현
- NotificationCompat.Builder
  - 알림을 생성하는 내부 클래스
  - import android.support.v4.app.NotificationCompat 사용
  - API Level 26 부터 Notification.Builder 대신 사용
  - 알림에 대한 UI정보와 작업을 지정하여 알림 생성
- Notification Channel
  - Android 8.0 (API level 26) 이후부터 도입
  - 여러 개의 알림을 관리하기 위한 용도로 사용

알림 구현 및 사용 절차

- 권한 등록 (API 33 이상) – AndroidManifest

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

- NotificationChannel 생성
  - 알림의 우선순위 지정
- NotificationCompat.Builder 생성 및 항목 설정
  - 알림 형태 설정
  - 알림 탭 시 동작 설정
  - 기타 기능 설정
- 알림 표시
  - NotificationManagerCompat 생성 후
- 알림 수정 및 삭제

Notification 의 사용 1

- 알림 채널 별 관리를 통해 알림의 유형 별 관리 가능

알림 채널의 생성 및 등록

- Android 8.0 (API level 26) 이상에서 앱 사용 시 필요
- 앱을 실행하는 Android 버전을 구분한 후 채널 생성 수행
  - 등록 시 중요도 설정

- Android 8.0 이상일 경우 Channel 사용이 필요하므로 확인 수행

```
private fun createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        // Notification Channel 의 생성
        val name = "Test Channel"
        val descriptionText = "Test Channel Message"
        val importance = NotificationManager.IMPORTANCE_DEFAULT
        val mChannel = NotificationChannel("MY_CHANNEL_ID", name, importance)
        mChannel.description = descriptionText

        // Channel 을 시스템에 등록, 등록 후에는 중요도 변경 불가
        val notificationManager = getSystemService(NOTIFICATION_SERVICE) as
                                                        NotificationManager
        notificationManager.createNotificationChannel(mChannel)
    }
}
```

- 채널 아이디는 관리를 위해 보관 필요

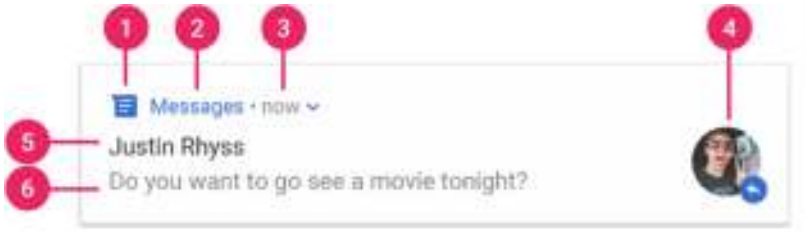
• 출처: https://developer.android.com/develop/ui/views/notifications/channels?hl=ko#importance

알림을 사용하기 전에 해당 함수 호출

- 액티비티의 onCreate() 등

Notification 주요 메소드

② 앱 이름 (시스템에서 제공)
③ 타임 스탬프 (시스템 제공 or 또는 setWhen(),setShowWhen(false) 사용



알림 표시 콘텐츠

- NotificationCompat.Builder 를 사용하여 설정

필수 표시 요소 설정

- ① setSmallIcon() : 알림을 나타내는 작은 아이콘 설정

참고: https://developer.android.com/studio/write/image-asset-studio?hl=ko#create-notification

선택적 표시 요소 설정

- ④ setLargeIcon()
- ⑤ setContentTitle(): 알림의 제목 설정
- ⑥ setContentText(): 알림의 세부 설명 설정
- setPriority(): 알림 표시의 우선 순위, 7.1 이하에서 사용 → 8.0부터는 알림 채널에서 지정 (채널의 importance)
- setStyle() 을 사용하여 다양한 유형(알림을 더 길게 설정 등)의 알림 생성 가능 → https://developer.android.com/training/notify-user/expanded.html 참조
- setSubtext(), setTicker(), setWhen(), setLights(), setNumber(), setOngoing(), setSound(), setVibrate() 등

알림 선택(탭) 시 동작 지정

- setContentIntent(intent : PendingIntent)

Notification 의 사용2

알림 생성

```
private fun showNotification() {
    val intent = Intent(this, AlertActivity::class.java).apply {
        flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    }
    val pendingIntent: PendingIntent
        = PendingIntent.getActivity(this, 0,  intent, PendingIntent.FLAG_IMMUTABLE)

    val channelID = resources.getString(R.string.channel_id)

    var builder = NotificationCompat.Builder(this, channelID)
        .setSmallIcon(R.drawable.ic_stat_name)
        .setContentTitle("알림 제목")
        .setContentText("짧은 알림 내용")
        .setStyle(NotificationCompat.BigTextStyle()
            .bigText("확장시 확인할 수 있는 긴 알림 내용"))
        // 8.0이상에서는 대신 Channel 중요도로 설정
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true)

    val notiManager = NotificationManagerCompat.from(this)
    notiManager.notify(100, builder.build())
}
```

- Intent 변경 불가 지정
- 알림 생성 Builder
- 알림 확장 시 긴 내용
- 알림 클릭 시 실행할 정보 (PendingIntent) 전달
- 알림 클릭 시 알림 닫기
- NotificationManager 를 사용하여 생성한 알림을 실행
- Notification 식별 값 : 취소 등을 위해 보관하여야 함

Notification 의 사용3

작업 버튼 추가

- 알림을 탭할 때와 다른 동작 지정(미리 알림 또는 문자 메시지 답장 지정)
- PendingIntent 를 addAction() 에 전달

```
private fun showNotificationWithAction() {
    val intent = Intent(this, AlertBroadcastReceiver::class.java).apply {
        action = "ACTION_SNOOZE"
        putExtra("NOTI_ID", 200)
    }
    val pendingIntent: PendingIntent
            = PendingIntent.getBroadcast(this, 0,  intent, PendingIntent.FLAG_IMMUTABLE)

    val channelID = resources.getString(R.string.channel_id)
    var builder = NotificationCompat.Builder(this, channelID)
        .setSmallIcon(R.drawable.ic_stat_name)
        .setContentTitle("알림 제목")
        .setContentText("짧은 알림 내용")
        .setStyle(NotificationCompat.BigTextStyle()
            .bigText("확장시 확인할 수 있는 긴 알림 내용"))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)   // 8.0이상에서는 대신 Channel 중요도로 설정
        .setContentIntent(pendingIntent)
        .setAutoCancel(true)
        .addAction(R.drawable.ic_stat_name, "쉬기", pendingIntent)

    val notiManager = NotificationManagerCompat.from(this)
    notiManager.notify(100, builder.build())
}
```

• BroadcastReceiver 호출

• 클릭 시 지정 Intent 전달

Broadcast

개요

- 안드로이드 앱은 시스템 또는 다른 안드로이드 앱을 대상으로 방송 메시지를 수신/송신할 수 있음 (게시-구독 과 유사)
- 관심사항에 해당하는 이벤트가 발생할 때 방송 수신
  - 시스템 부팅, 충전 시작, 네트워크 상태 변경 등

방송 수신 (BroadcastReceiver 사용)

- 특정 방송에 대해 수신 여부를 등록(구독 신청)
- 방송이 생성되었을 경우 시스템이 해당 방송 수신자(BR)를 찾아 방송 내용 전달

방송 송신

- 안드로이드 시스템은 다양한 종류의 방송을 송신함
- 방송 내용은 Intent 안에 포함
- 방송 Intent = Action(방송종류) + Extra(방송데이터)

BroadcastReceiver 1

방송수신 설정방법 1 – AndroidManifest 기록

- Manifest에 BR 정보 기록 – 앱 설치 이후부터 방송 수신

```
<application
    …
    <receiver android:name=".AlertBroadcastReceiver" android:exported="true">
        <intent-filter>
            <action android:name="ACTION_SNOOZE" />
        </intent-filter>
    </receiver>
</application>
```

• 수신할 방송을 action 으로 구분

- <action> 이 발생할 때 BR이 방송을 수신

방송수신 설정방법 2 – 코드에서 실행

```
val br: BroadcastReceiver = MyBroadcastReceiver()
val filter = IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION).apply {
    addAction(Intent.ACTION_AIRPLANE_MODE_CHANGED)
}
registerReceiver(br, filter);  // 방송 수신 등록
this.unregisterReceiver(br);    // 방송 수신 해제
```

- 앱 실행 도중 수신여부 결정
- 등록 실행 후 해제 전까지 방송 수신

BroadcastReceiver 2

BroadcastReceiver 구현

- BroadcastReceiver 상속
- onReceive() 재정의
  - context: receive가 실행 중인 context
  - intent: 방송 수신 메시지를 담고 있는 intent (PendingIntent 에 저장)

• Broadcast 요청 시 데이터를 전달하기 위해 생성한 Intent

```
class AlertBroadcastReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context?, intent: Intent?) {
        Toast.makeText(context, "휴식중...", Toast.LENGTH_SHORT).show()
        val notiId = intent?.getIntExtra("NOTI_ID", 0)
        Log.d("AlertBroadcastReceiver", "Notification ID: ${notiId}")
    }

}
```

- onReceive()는 빠른 시간 안에 수행을 마쳐야 함

방송 송신

- 방송을 직접 생성하여 송신 가능
  - sendOrderedBroadcast(Intent, String)
    - 방송 수신자에게 차례차례 방송 전달 (android:priority에 따라 순서가 결정됨)
    - Intent: 방송 내용
    - String: 방송 수신 권한 여부
  - sendBroadcast(Intent)
    - 모든 방송수신자에게 순서와 상관 없이 방송 전달

```
Intent ().also { intent ->
    intent.setAction("com.example.broadcast.MY_NOTIFICATION")
    intent.putExtra("data","Notice me!");
    sendBroadcast(intent);
}
```

알람(Alarm)

개요

- 지정한 시간과 간격으로 특정 이벤트를 발생시키는 서비스 (== 특정 인텐트를 실행하는 서비스)
  - 알람을 통해 Activity, BR, 또는 Service 실행
  - 보통 BR을 실행시킨 후 BR에서 Activity 또는 Service 실행

알람 서비스 관리

- 시스템(운영체제)에서 관리
  - 앱은 알람정보를 설정하여 시스템에 등록
  - 앱 종료 후에도 시스템에 등록한 알람 정보는 정해진 스케줄에 실행
- 시스템 절전 상태에서도 알람 실행 가능
- 장치를 재부팅하면 모든 알람은 취소
  - 재등록 필요
- 사용 시 시스템 리소스 영향을 파악할 필요가 있음

알람 서비스 사용 절차

1. AlarmManager 객체 준비
   - getSystemService(Context.ALARM_SERVICE)

2. PendingIntent 준비
   - 알람 수행 시 동작할 정보를 Intent 를 사용하여 지정
   - 알람 시간에 동작할 Intent 준비 후 PendingIntent에 설정
   - getActivity/getBroadcast/getService

3. AlarmManager 에 알람 설정
   - 알람 기준 시간 방식 및 sleep 모드 여부 설정
   - Calendar 객체로 시간 설정
   - PendingIntent 설정
   - 반복 알람인 경우 간격 설정

알람 관리자(AlarmManager)

- 알람 관리를 위해 시스템이 제공하는 서비스 클래스

- AlarmManager 생성 및 Alarm 취소

```
val requestId = 100
val alarmManager =
    getSystemService(Context.ALARM_SERVICE) as? AlarmManager

val pendingIntent =
    PendingIntent.getActivity(applicationContext, requestId, intent,
        PendingIntent.FLAG_NO_CREATE)

if (pendingIntent != null && alarmManager != null) {
    alarmManager.cancel(pendingIntent)
```

- 이전에 등록한 PI가 없을 경우 null 반환
- 알람 생성 시 사용
- requestId : 정수형 알람 식별값
- Intent : 알람 정보 저장
- 취소 실행

Alarm 유형 선택

기준 시계의 유형 설정 (Elapsed vs. Realtime)

- 알람 설정 시 AlarmManager에 실행 유형(상수) 전달

예약 시간의 기준 종류와 장비 기동 여부 지정

- 실시간 vs. 경과시간, Sleep vs. Wake Up

값	의미
RTC	System.currentTimeMillis()로 설정한 세계 표준시(UTC)를 기준으로 알람 시간 지정
RTC_WAKEUP	위와 동일, 절전모드 해제
ELAPSED_REALTIME	SystemClock.elapsedRealtime()으로 설정한 부팅 후 경과시간을 기준으로 알람 시간 지정
ELAPSED_REALTIME_WAKEUP	위와 동일, 절전모드 해제

※ 절전 모드 해제: 화면을 켜는 것이 아닌 알람을 알릴 수 있도록 장비를 활성화하는 것

알람 설정 메소드

set()

- 특정 시간에 한 번만 동작하는 알람 지정

```
set (type : Int, triggerAtMillis : Long, operation : PendingIntent)
```

setRepeating()

- 정확하게 지정한 시간 간격으로 알람 실행
- 배터리 소모가 클 수 있음 → 5.1부터 최소 60초 이상으로 반복 간격 제한 (그 이하가 필요할 경우 Handler 등 사용)

```
setRepeating (type : Int, triggerAtMillis : Long, intervalMillis : Long,
                                                  operation : PendingIntent)
```

setInexactRepeating()

- interval: 정확한 간격 대신 AlarmManager의 상수로 간격 지정
  - AlarmManager.*INTERVAL_FIFTEEN_MINUTES* 등
- 배터리 효율이 상대적으로 좋음

```
setInexactRepeating (type : Int, triggerAtMillis : Long,
                                  interval : Long, operation : PendingIntent)
```

알람 사용의 예 1

실제 경과시간 일회성 알람의 예

- 경과시간 기준 1분 후 알람

```
alarmManager?.set(
    AlarmManager.ELAPSED_REALTIME_WAKEUP,
    SystemClock.elapsedRealtime() + 60 * 1000,
    alarmIntent
)
```

• 절전모드 해제

• 밀리세컨드 단위

실제 경과시간 반복 알람의 예

- 경과시간 기준 30분 후부터 30분 간격 알람(부정확 할 수 있음)

```
alarmMgr?.setInexactRepeating(
        AlarmManager.ELAPSED_REALTIME_WAKEUP,
        SystemClock.elapsedRealtime() + AlarmManager.INTERVAL_HALF_HOUR,
        AlarmManager.INTERVAL_HALF_HOUR,
        alarmIntent
)
```

• INTERVAL_DAY
• INTERVAL_HALF_DAY
• INTERVAL_HOUR
• INTERVAL_HALF_HOUR
• INTERVAL_FIFTEEN_MINUTES

알람 사용의 예 2

실시간 시계 알람의 예

- 대략 오후 2시에 절전모드 해제 후 알람 실행, 하루 간격 반복

```
val calendar: Calendar = Calendar.getInstance().apply {
    timeInMillis = System.currentTimeMillis()
    set(Calendar.HOUR_OF_DAY, 14)    // 오후 2시로 시간 지정
//  set(Calendar.MINUTES, 30)  // setRepeating() 사용 시 해제
}

// 대략 오후 2시에 절전모드 해제 후 알람 실행, 하루 간격 반복
alarmManager?.setInexactRepeating(
    AlarmManager.RTC_WAKEUP,          // 실시간 시계 사용
    calendar.timeInMillis,
    AlarmManager.INTERVAL_DAY,         // 하루 간격
    alarmIntent
)

// 정확히 오후 2시 30분에 절전모드 해제 후 알람 실행, 20분 간격 반복
alarmManager?.setRepeating(
    AlarmManager.RTC_WAKEUP,          // 실시간 시계 사용
    calendar.timeInMillis,
    1000 * 60 * 20,                   // 상수가 아닌 시간 직접 지정
    alarmIntent
)
```

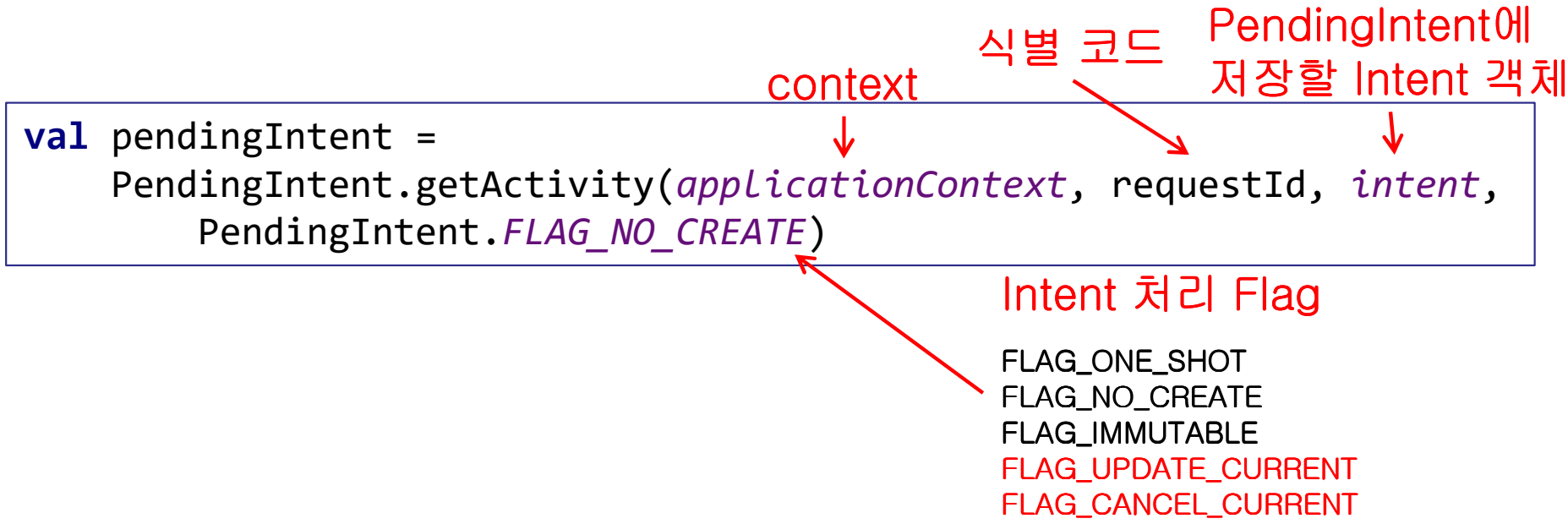
PendingIntent 설정 1

알람에 적용 시의 역할

- 알람을 구분할 수 있는 코드 지정
- 알람 동작 시 수행하여야 할 Intent 를 지정

PendingIntent 의 생성

- getActivity(), getBroadcast(), getService()
  - PendingIntent에 지정한 Intent 의 대상에 따라 사용 결정
- 알람의 경우 보통 getBroadcast 사용



PendingIntent 설정 2

- PendingIntent 구분
  - *REQ_CODE* 와 *Intent* 가 일치하여야 동일한 PendingIntent 취급
  - Intent의 경우 Extra 값은 달라도 동일한 Intent 취급

1. 새로운 알람 추가
   - *REQ_CODE* 를 변경하며 알람 설정

2. 기존 알람 수정
   - 동일한 *REQ_CODE* 와 *Intent*를 갖는 PendingIntent를 사용하여 알람 설정

3. 기존 알람 삭제
   - 동일한 *REQ_CODE* 와 *Intent*를 갖는 PendingIntent를 cancel에 적용
   - AlarmManager.FLAT_NO_CREATE 사용

반복 알람의 변경점

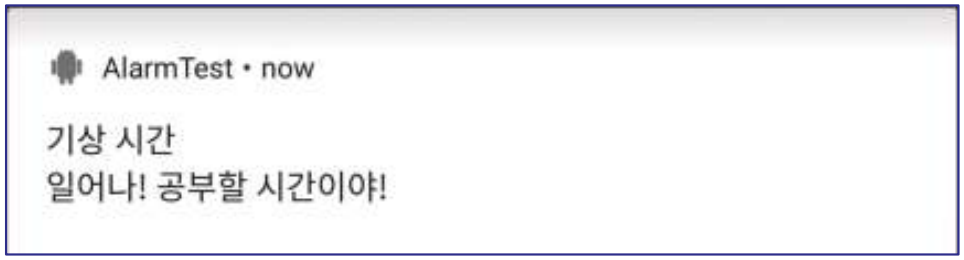
- 반복 알람의 사용
  - setRepeating(…)
  - setInexactRepeating(…)

- API Level 19의 변경점 (Kitkat 4.4)
  - setRepeating(…) → setExact(…): 반복이 되지 않으므로 알람을 울린 후 다시 재등록 필요

- API Level 23의 변경 (Marshmallow 6.0)
  - Doze 모드(잠자기 모드) 시 setExact(…) 는 연기됨
    → setExactAndAllowWhileIdle(…) 사용: 부정확할 수 있음
    → 정확한 알람이 필요하지 않을 경우 setExactAndAllowWhileIdle(…) 사용
  - Doze 모드에서도 작업이 필요할 경우: setAlarmClock(…) 사용

실습

- 예제를 수정하여 알람 시간이 되면 아래와 같은 Notification을 표시하도록 구성



- 예제를 수정하여 알람 시간이 되면 알람 앱 화면이 나타나도록 수정
  - 알람 설정 시 MyBroadcastReceiver 를 호출하도록 지정
  - MyBroadcastReceiver 의 onReceive() 에서 MainActivity 호출 intent 생성 및 호출

참고

- 알림 개요
  - https://developer.android.com/develop/ui/views/notifications?hl=kr
- Broadcast 개요
  - https://developer.android.com/guide/components/broadcasts?hl=ko
- 반복 알람 예약
  - https://developer.android.com/training/scheduling/alarms?hl=ko
- AlarmManager
  - https://developer.android.com/reference/android/app/AlarmManager.html
- 잠자기 모드
  - https://developer.android.com/training/monitoring-device-state/doze-standby.html