세상의 속도를
따라잡고 싶다면

Do it!

깡샘의 안드로이드 앱 프로그래밍 with 코틀린

이지스퍼블리싱(주)

10

다이얼로그와 알림 이용하기

이지스퍼블리싱(주)

10-1 API 레벨 호환성 고려하기

- 앱은 31 버전의 API로 개발하지만 21 버전 기기에서도 오류가 발생하지 않고 동작해야 합니다.
- minSdk 설정값보다 상위 버전에서 제공하는 API를 사용한다면 호환성을 고려

Added in API level 31

Notification.CallStyle

Kotlin | **Java**

```
public static class Notification.CallStyle
extends Notification.Style
```

java.lang.Object
↳ android.app.Notification.Style
↳ android.app.Notification.CallStyle


```
val builder: Notification.Builder = Notificati
    .setStyle(
        Notification.CallStyle.forIncomingCall
```

Call requires API level S (current min is 21): `android.app.Notification.CallStyle#forIncomingCall`

Cast from `CallStyle` to `Style` requires API level 31 (current min is 21)

10-1 API 레벨 호환성 고려하기

- @RequiresApi 애너테이션이나 @TargetApi 애너테이션을 이용
- 안드로이드 스튜디오에서 오류를 무시하는 설정

• API 호환성 애너테이션 1

```
@RequiresApi(Build.VERSION_CODES.S)
fun noti() {
    (...생략...)
    val builder: Notification.Builder = Notification.Builder(this, "1")
        .setStyle(
            Notification.CallStyle.forIncomingCall(caller, declineIntent, answerIntent)
        )
    (...생략...)
}
```

10-1 API 레벨 호환성 고려하기

- API 레벨 호환성 문제를 막으려면 직접 코드로 처리

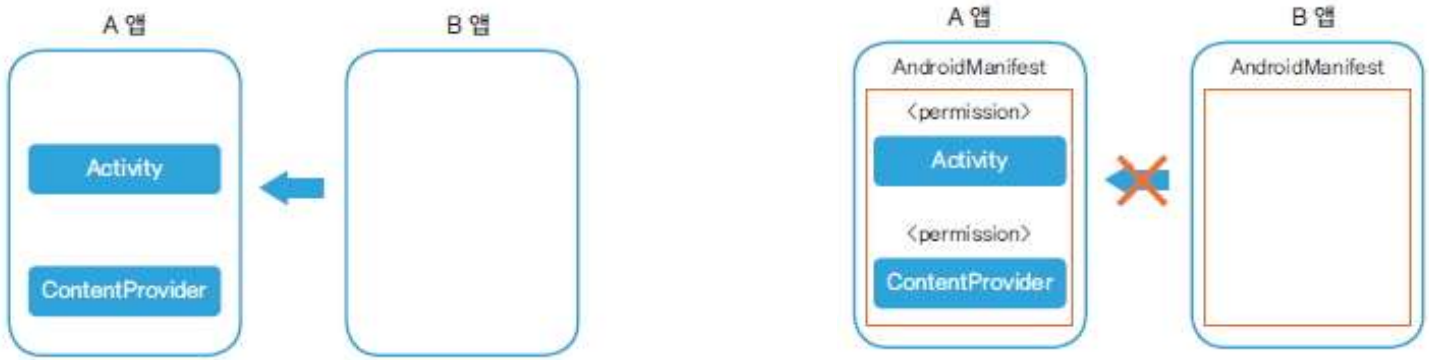
• API 레벨 30 이상에서만 addCallback() 함수 실행

```
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.S){
    val builder: Notification.Builder = Notification.Builder(this, "1")
        .setStyle(
            Notification.CallStyle.forIncomingCall(caller, declineIntent, answerIntent)
        )
}
```

10-2 퍼미션 설정하기

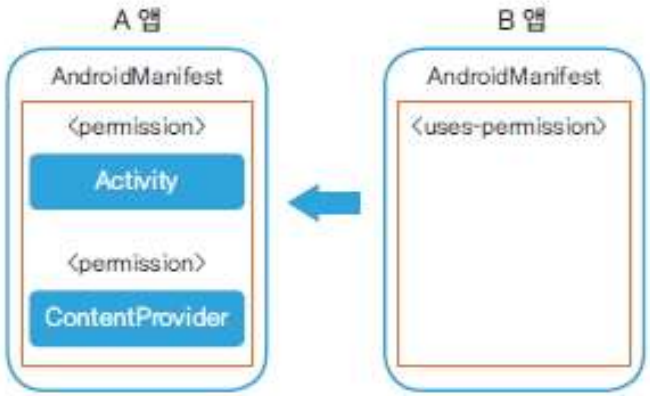
퍼미션 설정과 사용 설정

- A 앱의 컴포넌트를 B 앱에서 사용하는 상황
- 만약 A 앱의 컴포넌트에 퍼미션을 설정하면 B 앱에서 연동할 때 문제가 발생



10-2 퍼미션 설정하기

- 매니페스트 파일에 <permission> 태그로 퍼미션을 설정
- 매니페스트 파일에 <uses-permission> 태그로 해당 퍼미션을 이용하겠다고 설정
- <permission>: 기능을 보호하려는 앱의 매니페스트 파일에 설정합니다.
- <uses-permission>: 퍼미션으로 보호된 기능을 사용하려는 앱의 매니페스트 파일에 설정합니다.



10-2 퍼미션 설정하기

- <permission> 태그와 속성
  - name: 퍼미션의 이름입니다.
  - label, description: 퍼미션을 설명합니다.
  - protectionLevel : 보호 수준입니다.

• 퍼미션 설정

```
<permission android:name="com.example.permission.TEST_PERMISSION"
    android:label="Test Permission"
    android:description="@string/permission_desc"
    android:protectionLevel="dangerous" />
```

- protectionLevel 속성값은 보호 수준을 의미
  - normal: 낮은 수준의 보호입니다. 사용자에게 권한 허용을 요청하지 않아도 됩니다.
  - dangerous: 높은 수준의 보호입니다. 사용자에게 권한 허용을 요청해야 합니다.
  - signature: 같은 키로 인증한 앱만 실행합니다.
  - signatureOrSystem: 안드로이드 시스템 앱이거나 같은 키로 인증한 앱만 실행합니다.

10-2 퍼미션 설정하기

- <uses-permission>을 2개 설정했다고 가정

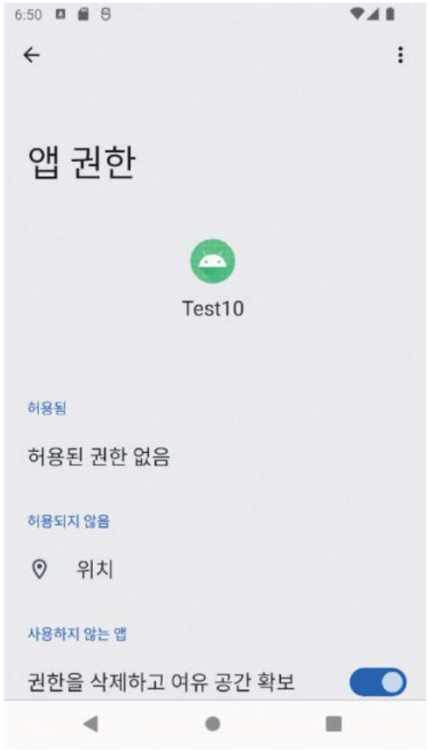
• 퍼미션 사용 설정

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

- <per mission>을 설정한 다음 이 퍼미션으로 보호하려는 컴포넌트에 적용

• 컴포넌트에 퍼미션 적용

```
<activity android:name=".OneActivity" android:permission="com.example.TEST_PERMISSION">
    <intent-filter>
        <action android:name="android.intent.action.PICK" />
    </intent-filter>
</activity>
```



10-2 퍼미션 설정하기

- 컴포넌트를 이용하는 곳에서는 매니페스트 파일에 <uses-permission>을 선언해 줘야 정상으로 실행

• 퍼미션 사용 설정

```
<uses-permission android:name="com.example.permission.TEST_PERMISSION" />
```

- 시스템에서 보호하는 기능을 사용할 때도 매니페스트 파일에 퍼미션 사용 설정
  - ACCESS_FINE_LOCATION: 위치 정보 접근
  - ACCESS_NETWORK_STATE: 네트워크 정보 접근
  - ACCESS_WIFI_STATE: 와이파이 네트워크 정보 접근
  - BATTERY_STATS: 배터리 정보 접근
  - BLUETOOTH: 블루투스 장치에 연결

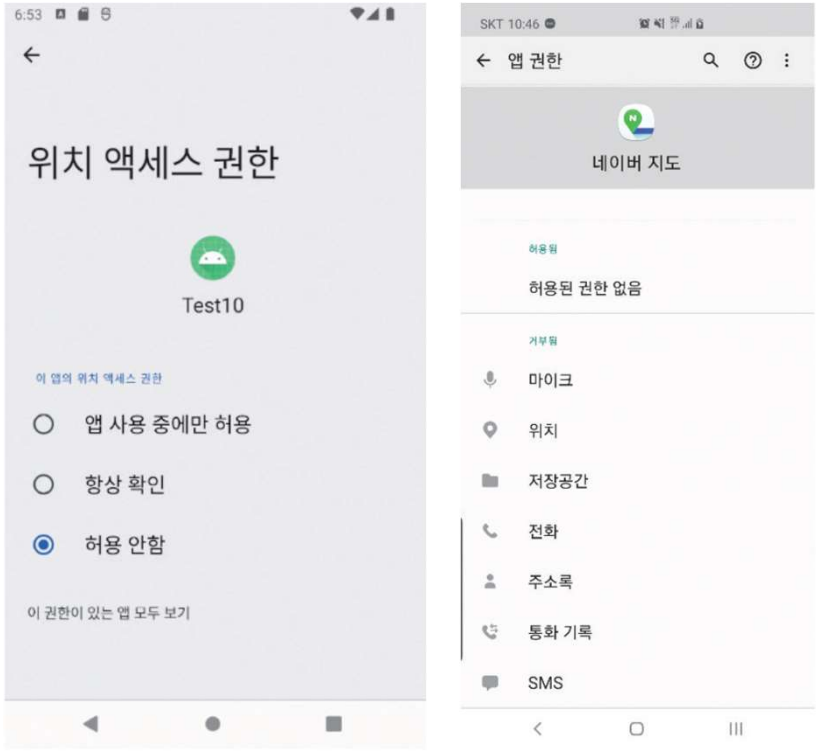
10-2 퍼미션 설정하기

- BLUETOOTH_ADMIN: 블루투스 장치를 검색하고 페어링
- CAMERA: 카메라 장치에 접근
- INTERNET: 네트워크 연결
- READ_EXTERNAL_STORAGE: 외부 저장소에서 파일 읽기
- WRITE_EXTERNAL_STORAGE: 외부 저장소에 파일 쓰기
- READ_PHONE_STATE: 전화기 정보 접근
- SEND_SMS: 문자 메시지 발신
- RECEIVE_SMS: 문자 메시지 수신
- RECEIVE_BOOT_COMPLETED: 부팅 완료 시 실행
- VIBRATE: 진동 울리기

10-2 퍼미션 설정하기

퍼미션 허용 확인

- API 레벨 23 버전부터 허가제로 변경
- <uses-permission>으로 선언했더라도 사용자가 권한 화면에서 이를 거부할 수 있게 되었습니다.
- 앱을 실행할 때 사용자가 퍼미션을 거부했는지 확인하고 만약 거부했으면 다시 퍼미션을 허용해 달라고 요청해야 합니다.



10-2 퍼미션 설정하기

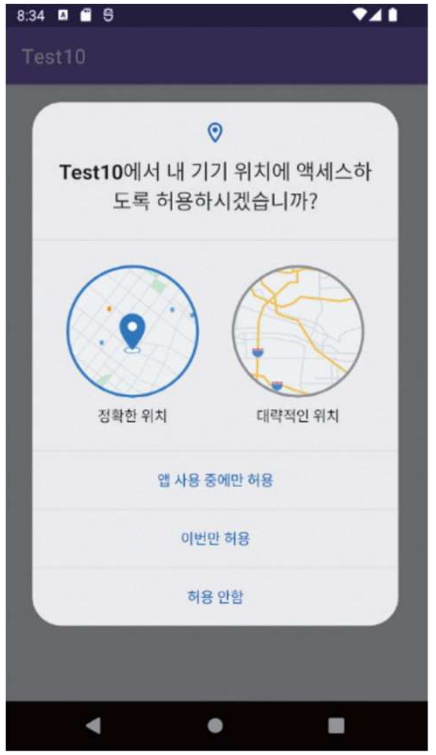
- 퍼미션을 허용했는지 확인하려면 checkSelfPer mission() 함수를 이용
- 결괏값은 다음 중 하나의 상수로 전달
  - PackageManager.PERMISSION_GRANTED: 권한을 허용한 경우
  - PackageManager.PERMISSION_DENIED: 권한을 거부한 경우

• 퍼미션 허용 확인 예

```
val status = ContextCompat.checkSelfPermission(this,
    "android.permission.ACCESS_FINE_LOCATION")
if (status == PackageManager.PERMISSION_GRANTED) {
    Log.d("kkang", "permission granted")
} else {
    Log.d("kkang", "permission denied")
}
```

10-2 퍼미션 설정하기

- 거부한 상태라면 사용자에게 해당 퍼미션을 허용해 달라고 요청
- ActivityResultLauncher를 이용
- ActivityResultLauncher 객체는 registerForActivity Result() 함수를 호출해서 만듭니다.
- 첫 번째 매개변수는 어떤 요청인지를 나타내는 ActivityResultContract 타입 객체
- 두 번째 매개변수는 결과를 받았을 때 호출되는 콜백
- ActivityResultLauncher 객체의 launch() 함수를 호출하여 요청을 실행



• 퍼미션 허용 요청 확인

```
val requestPermissionLauncher = registerForActivityResult(
    ActivityResultContracts.RequestPermission()
) { isGranted ->
    if (isGranted) {
        Log.d("kkang", "callback, granted..")
    } else {
        Log.d("kkang", "callback, denied..")
    }
}
```

• 퍼미션 허용 요청 실행

```
requestPermissionLauncher.launch("android.permission.ACCESS_FINE_LOCATION")
```

10-3 다양한 다이얼로그

토스트 메시지 띄우기

- 토스트는 화면 아래쪽에 잠깐 보였다가 사라지는 문자열
- 토스트는 Toast의 makeText() 함수로 만듭니다.
  - open static fun makeText(context: Context!, text: CharSequence!, duration: Int): Toast!
  - open static fun makeText(context: Context!, resId: Int, duration: Int): Toast!
- 세 번째 매개변수는 토스트가 화면에 출력되는 시간
  - val LENGTH_LONG: Int
  - val LENGTH_SHORT: Int

• 토스트 출력 예

```
val toast = Toast.makeText(this, "종료하려면 한 번 더 누르세요", Toast.LENGTH_SHORT)
toast.show()
```

10-3 다양한 다이얼로그

- 토스트가 화면에 보이거나 사라지는 순간을 콜백으로 감지해 특정 로직을 수행하게 할 수도 있습니다.
- 콜백 기능은 API 레벨 30 버전에서 추가

• 콜백 기능 이용하기

```
@RequiresApi(Build.VERSION_CODES.R)   API 레벨 호환성 애너테이션
fun showToast() {
    val toast = Toast.makeText(this, "종료하려면 한 번 더 누르세요.", Toast.LENGTH_SHORT)
    toast.addCallback(
        object : Toast.Callback() {
            override fun onToastHidden() {
                super.onToastHidden()
                Log.d("kkang", "toast hidden")
            }

            override fun onToastShown() {
                super.onToastShown()
                Log.d("kkang", "toast shown")
            }
        })
    toast.show()
}
```

10-3 다양한 다이얼로그

날짜 또는 시간 입력받기

- 날짜를 입력받을 때는 DatePickerDialog를, 시간을 입력받을 때는 TimePickerDialog를 사용
- 사용자가 설정한 날짜를 콜백 함수로 얻을 수 있습니다.

• 데이트 피커 다이얼로그 사용 예

```
DatePickerDialog(this, object: DatePickerDialog.OnDateSetListener {
    override fun onDateSet(p0: DatePicker?, p1: Int, p2: Int, p3: Int) {
        Log.d("kkang", "year : $p1, month : ${p2+1}, dayOfMonth : $p3")
    }
}, 2020, 8, 21).show()
```

10-3 다양한 다이얼로그

• 타임 피커 다이얼로그 사용 예

```
TimePickerDialog(this, object: TimePickerDialog.OnTimeSetListener {
    override fun onTimeSet(p0: TimePicker?, p1: Int, p2: Int) {
        Log.d("kkang", "time : $p1, minute : $p2")
    }
}, 15, 0, true).show()
```

10-3 다양한 다이얼로그

알림 창 띄우기

- 다이얼로그의 기본은 AlertDialog
- 제목, 내용, 버튼 영역으로 구분
- AlertDialog.Builder를 제공하므로 이 빌더를 이용해 알림 창을 만듭니다.
- open fun setIcon(iconId: Int): AlertDialog.Builder!
- open fun setTitle(title: CharSequence!): AlertDialog.Builder!
- open fun setMessage(message: CharSequence!): AlertDialog.Builder!

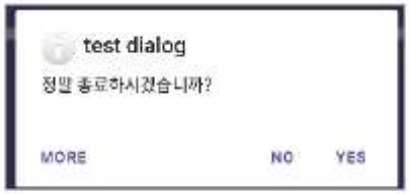
- 알림 창에 버튼을 지정하는 함수
- open fun setPositiveButton(text: CharSequence!, listener: DialogInterface.OnClick Listener!): AlertDialog.Builder!
- open fun setNegativeButton(text: CharSequence!, listener: DialogInterface.OnClick Listener!): AlertDialog.Builder!
- open fun setNeutralButton(text: CharSequence!, listener: DialogInterface.OnClick Listener!): AlertDialog.Builder!

10-3 다양한 다이얼로그

• 알림 창 띄우기

```
AlertDialog.Builder(this).run {
    setTitle("test dialog")
    setIcon(android.R.drawable.ic_dialog_info)
    setMessage("정말 종료하시겠습니까?")
    setPositiveButton("OK", null)
    setNegativeButton("Cancel", null)
    setNeutralButton("More", null)
    setPositiveButton("Yes", null)
    setNegativeButton("No", null)
    show()
}
```

▶실행 결과

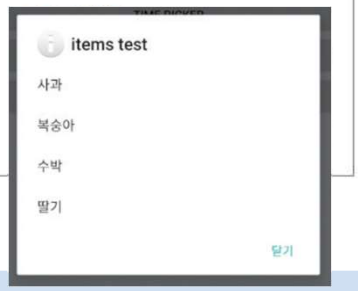


10-3 다양한 다이얼로그

- 목록을 제공하고 이 중 하나를 선택받는 알림 창을 만들고자 한다면 setItems(), setMultiChoiceItems(), setSingleChoiceItems() 함수를 이용
  - open fun setItems(items: Array<CharSequence!>!, listener: DialogInterface.On ClickListener!): AlertDialog.Builder!
  - • open fun setMultiChoiceItems(items: Array<CharSequence!>!, checkedItems: BooleanArray!, listener: DialogInterface.OnMultiChoiceClickListener!): AlertDialog.Builder!
  - • open fun setSingleChoiceItems(items: Array<CharSequence!>!, checkedItem: Int, listener: DialogInterface.OnClickListener!): AlertDialog.Builder!

• 목록을 출력하는 알림 창

```
val items = arrayOf<String>("사과","복숭아","수박","딸기")
AlertDialog.Builder(this).run {
    setTitle("items test")
    setIcon(android.R.drawable.ic_dialog_info)
    setItems(items, object: DialogInterface.OnClickListener {
        override fun onClick(p0: DialogInterface?, p1: Int) {
            Log.d("kkang", "선택한 과일 : ${items[p1]}")
        }
    })
    setPositiveButton("닫기", null)
    show()
}
```

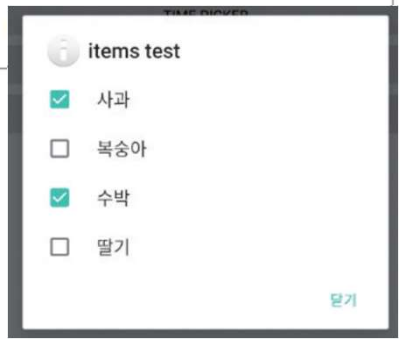


10-3 다양한 다이얼로그

- setMultiChoiceItems() 함수는 다중 선택을 위한 체크박스가 함께 출력되는 항목

• 체크박스를 포함하는 예

```
setMultiChoiceItems(items, booleanArrayOf(true, false, true, false), object: Di-
alogInterface.OnMultiChoiceClickListener {
    override fun onClick(p0: DialogInterface?, p1: Int, p2: Boolean) {
        Log.d("kkang", "${items[p1]} 이 ${if(p2) "선택되었습니다." else "선택 해제되었습니다."}")
    }
})
```

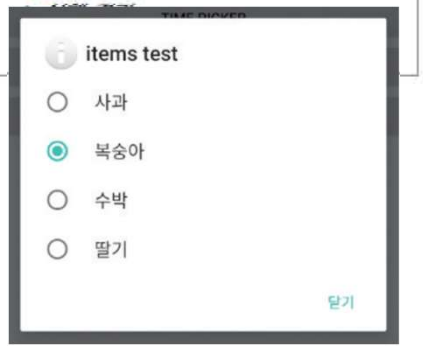


10-3 다양한 다이얼로그

- setSingleChoiceItems() 함수는 하나만 선택할 수 있는 라디오 버튼으로 구성된 항목을 만들어 줍니다.

• 라디오 버튼을 포함하는 예

```
setSingleChoiceItems(items, 1, object: DialogInterface.OnClickListener {
    override fun onClick(p0: DialogInterface?, p1: Int) {
        Log.d("kkang", "${items[p1]} 이 선택되었습니다.")
    }
})
```



10-3 다양한 다이얼로그

- setCancelable() 함수는 사용자가 기기의 뒤로가기 버튼을 눌렀을 때, setCanceledOnTouchOutside() 함수는 알림 창의 바깥 영역을 터치했을 때 매개변수가 true이면 닫고 false이면 닫지 않습니다.

• 알림 창을 닫는 설정(뒤로가기, 바깥 영역 터치)

```
AlertDialog.Builder(this).run {
    setTitle("items test")
    setIcon(android.R.drawable.ic_dialog_info)
    setItems(items, object: DialogInterface.OnClickListener {
        override fun onClick(p0: DialogInterface?, p1: Int) {
            Log.d("kkang", "선택한 과일 : ${items[p1]}")
        }
    })
    setCancelable(false)
    setPositiveButton("닫기", null)
    show()
}.setCanceledOnTouchOutside(false)
```

10-3 다양한 다이얼로그

커스텀 다이얼로그 만들기

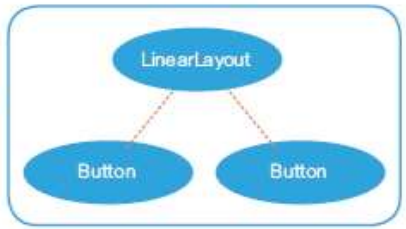
- 커스텀 다이얼로그도 AlertDialog를 이용
- LayoutInflater 클래스는 레이아웃 XML 파일을 코드에서 초기화(흔히 전개라고도 표현)하는 기능을 제공

레이아웃 XML 파일

```
<LinearLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button1" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button2" />
</LinearLayout>
```

초기화

뷰 객체 생성



10-3 다양한 다이얼로그

- get SystemService() 함수로 LayoutInflater를 얻습니다.
- inflate() 함수를 호출하면서 초기화할 레이아웃 XML 파일 정보를 매개변수로 전달

• XML 파일 초기화

```
val inflater = getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater
val rootView = inflater.inflate(R.layout.activity_one, null)
```

- 뷰 바인딩 기법을 이용한다면 XML 초기화 코드를 조금 더 쉽게 작성

• 뷰 바인딩을 적용한 XML 파일 초기화

```
val binding = ActivityOneBinding.inflate(layoutInflater)
val rootView = binding.root
```

10-3 다양한 다이얼로그

- 다이얼로그를 구성하는 레이아웃 XML 파일을 만들어야 합니다.
- 이 XML 파일을 LayoutInflater로 초기화해서 다이얼로그에 적용
- Alert Dialog의 setView() 함수에 매개변수로 뷰 객체를 전달하면 창의 내용 영역에 출력

• 커스텀 다이얼로그 출력

```
val dialogBinding = DialogInputBinding.inflate(layoutInflater)
AlertDialog.Builder(this).run {
    setTitle("Input")
    setView(dialogBinding.root)
    setPositiveButton("닫기", null)
    show()
}
```

▶ 실행 결과

10-4 소리와 진동 알림

소리 알림

- 안드로이드 시스템은 알림(NOTIFICATION), 알람(ALARM), 벨소리(RINGTONE) 등의 소리를 제공하며 이 소리는 RingtonManager로 얻을 수 있습니다.

• 소리 얻기

```
val notification: Uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION)
val ringtone = RingtoneManager.getRingtone(applicationContext, notification)
ringtone.play()
```

- 앱에서 자체 음원을 준비해서 재생하는 방법
- 음원 리소스 디렉터리는 res/raw

• 음원 재생하기

```
val player: MediaPlayer = MediaPlayer.create(this, R.raw.fallbackring)
player.start()
```

10-4 소리와 진동 알림

진동 알림

- 매니페스트 파일에 <uses-permission>으로 퍼미션 선언
- 진동은 Vibrator 클래스를 이용
- 31 이전 버전에서는 VIBRATOR_SERVICE로 식별되는 시스템 서비스를 이용
- 31 버전부터는 VIBRATOR_MANAGER_SERVICE로 식별되는 VibratorManager라는 시스템 서비스를 얻고 이 서비스에서 Vibrator를 이용

• 진동 객체 얻기

```
val vibrator  = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
    val vibratorManager = this.getSystemService(Context.VIBRATOR_MANAGER_SERVICE)
                          as VibratorManager
    vibratorManager.defaultVibrator;
} else {
    getSystemService(VIBRATOR_SERVICE) as Vibrator
}
```

10-4 소리와 진동 알림

진동 알림

- 시간과 패턴을 지정해 진동 울리기(API 레벨 1부터 제공하는 함수)
  - open fun vibrate(milliseconds: Long): Unit
  - open fun vibrate(pattern: LongArray!, repeat: Int): Unit
- 진동의 세기까지 지정해 진동 울리기(API 레벨 26부터 제공하는 함수)
  - open fun vibrate(vibe: VibrationEffect!): Unit
  - VibrationEffect 객체로는 진동이 울리는 시간 이외에 진동의 세기까지 제어
  - open static fun createOneShot(milliseconds: Long, amplitude: Int): VibrationEffect!

10-4 소리와 진동 알림

• 기본 세기로 진동 울리기

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    vibrator.vibrate(
        VibrationEffect.createOneShot(500,
        VibrationEffect.DEFAULT_AMPLITUDE))
} else {
    vibrator.vibrate(500)
}
```

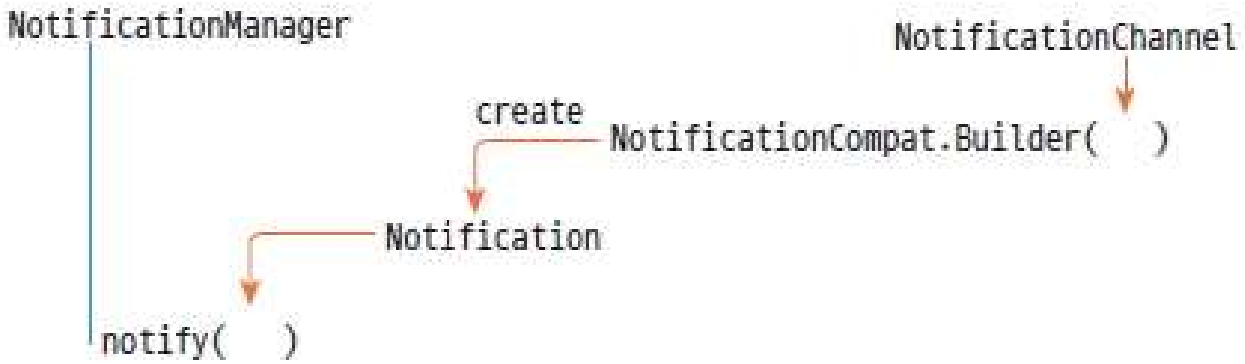
• 패턴대로 반복해서 울리기

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    vibrator.vibrate(VibrationEffect.createWaveform(longArrayOf(500, 1000, 500, 2000),
    intArrayOf(0, 50, 0, 200), -1))
} else {
    vibrator.vibrate(longArrayOf(500, 1000, 500, 2000), -1)
}
```

10-5 알림 띄우기

알림 채널

- 상태 바에 앱의 정보를 출력하는 것을 알림notification이라고 합니다.



10-5 알림 띄우기

- Notification을 만들려면 NotificationCompat.Builder가 필요
- 26 버전 이전까지는 빌더를 만들 때 NotificationChannel 정보가 필요 없었습니다.
- 26 버전부터는 빌더를 만들 때 NotificationChannel을 만들고 이 채널의 식별값을 빌더의 생성자 매개변수에 지정해 줘야 합니다.
- 앱의 알림을 채널로 구분하겠다는 의도

10-5 알림 띄우기

중요도 상수	설명
NotificationManager.IMPORTANCE_HIGH	긴급 상황으로 알림음이 울리며 헤드업으로 표시
NotificationManager.IMPORTANCE_DEFAULT	높은 중요도이며 알림음이 울림
NotificationManager.IMPORTANCE_LOW	중간 중요도이며 알림음이 울리지 않음
NotificationManager.IMPORTANCE_MIN	낮은 중요도이며 알림음도 없고 상태 바에도 표시되지 않음

- fun setDescription(description: String!): Unit: 채널의 설명 문자열
- fun setShowBadge(showBadge: Boolean): Unit: 홈 화면의 아이콘에 배지 아이콘 출력 여부
- fun setSound(sound: Uri!, audioAttributes: AudioAttributes!): Unit: 알림음 재생
- fun enableLights(lights: Boolean): Unit: 불빛 표시 여부
- fun setLightColor(argb: Int): Unit: 불빛이 표시된다면 불빛의 색상
- fun enableVibration(vibration: Boolean): Unit: 진동을 울릴지 여부
- fun setVibrationPattern(vibrationPattern: LongArray!): Unit: 진동을 울린다면 진동의 패턴

10-5 알림 띄우기

• 알림 빌더 작성

```
val manager = getSystemService(NOTIFICATION_SERVICE) as NotificationManager
val builder: NotificationCompat.Builder

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val channelId = "one-channel"
    val channelName = "My Channel One"
    val channel = NotificationChannel(
        channelId,
        channelName,
        NotificationManager.IMPORTANCE_HIGH
    )

    // 채널에 다양한 정보 설정
    channel.description = "My Channel One Description"
    channel.setShowBadge(true)
    val uri: Uri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION)
    val audioAttributes = AudioAttributes.Builder()
        .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
        .setUsage(AudioAttributes.USAGE_ALARM)
        .build()
    channel.setSound(uri, audioAttributes)
    channel.enableLights(true)
    channel.lightColor = Color.RED
    channel.enableVibration(true)
    channel.vibrationPattern = longArrayOf(100, 200, 100, 200)

    // 채널을 NotificationManager에 등록
    manager.createNotificationChannel(channel)

    // 채널을 이용해 빌더 생성
    builder = NotificationCompat.Builder(this, channelId)
} else {
    builder = NotificationCompat.Builder(this)
}
```

10-5 알림 띄우기

알림 객체

• 알림 객체 설정

```
builder.setSmallIcon(android.R.drawable.ic_notification_overlay)
builder.setWhen(System.currentTimeMillis())
builder.setContentTitle("Content Title")
builder.setContentText("Content Massage")
```

10-5 알림 띄우기

• 알림 발생

```
manager.notify(11, builder.build())
```

• 알림 취소

```
manager.cancel(11)
```

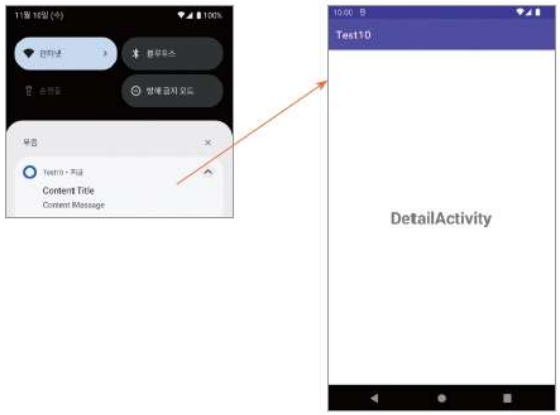
• 알림 취소 막기

```
builder.setAutoCancel(false)
builder.setOngoing(true)
```

10-5 알림 띄우기

알림 구성

- 알림 터치 이벤트
  - 인텐트를 준비한 후 Notification 객체에 담아서 이벤트가 발생할 때 인텐트를 실행해 달라고 시스템에 의뢰
  - static fun getActivity(context: Context!, requestCode: Int, intent: Intent!, flags: Int): PendingIntent!
  - static fun getBroadcast(context: Context!, requestCode: Int, intent: Intent!, flags: Int): PendingIntent!
  - static fun getService(context: Context!, requestCode: Int, intent: Intent, flags: Int): PendingIntent!



• 알림 객체에 액티비티 실행 정보 등록

```
val intent = Intent(this, DetailActivity::class.java)
val pendingIntent =
    PendingIntent.getActivity(this, 10, intent,
        PendingIntent.FLAG_IMMUTABLE)
builder.setContentIntent(pendingIntent)  // 터치 이벤트 등록
```

10-5 알림 띄우기

- 액션
  - 액션을 최대 3개까지 추가

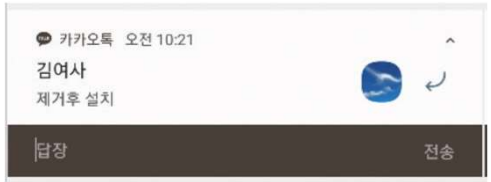
• 액션 등록하기

```
val actionIntent = Intent(this, OneReceiver::class.java)
val actionPendingIntent = PendingIntent.getBroadcast(this, 20, actionIntent,
    PendingIntent.FLAG_IMMUTABLE)
builder.addAction(
    NotificationCompat.Action.Builder(
        android.R.drawable.stat_notify_more,
        "Action",
        actionPendingIntent
    ).build()
)
```

▶ 실행 결과

10-5 알림 띄우기

- 원격 입력
  - 알림에서 사용자 입력을 직접 받는 기법



• 원격 입력

```
val KEY_TEXT_REPLY = "key_text_reply"
var replyLabel: String = "답장"
var remoteInput: RemoteInput = RemoteInput.Builder(KEY_TEXT_REPLY).run {
    setLabel(replyLabel)
    build()
}
```

  - PendingIntent를 준비

• 인텐트 준비

```
val replyIntent = Intent(this, ReplyReceiver::class.java)
val replyPendingIntent = PendingIntent.getBroadcast(this, 30, replyIntent,
    PendingIntent.FLAG_MUTABLE)
```

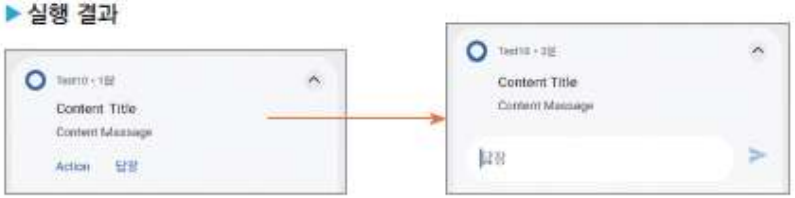
10-5 알림 띄우기

- 알림에 액션을 등록

• 원격 입력 액션 등록하기

```
builder.addAction(
    NotificationCompat.Action.Builder(
        R.drawable.send,
        "답장",
        replyPendingIntent
    ).addRemoteInput(remoteInput).build()
)
```

▶실행 결과



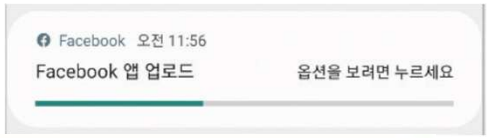
- 사용자가 입력한 글을 받을 때 코드

• 브로드캐스트 리시버에서 사용자가 입력한 글을 받는 코드

```
val replyTxt = RemoteInput.getResultsFromIntent(intent)
    ?.getCharSequence("key_text_reply")
```

10-5 알림 띄우기

- 프로그레스
  - 진행 상황을 프로그레스에 바로 알려 줍니다.



• 프로그레스 바의 진행값을 증가시키는 스레드

```
builder.setProgress(100, 0, false)
manager.notify(11, builder.build())

thread {
    for (i in 1..100) {
        builder.setProgress(100, i, false)
        manager.notify(11, builder.build())
        SystemClock.sleep(100)
    }
}
```

▶ 실행 결과

10-5 알림 띄우기

알림 스타일

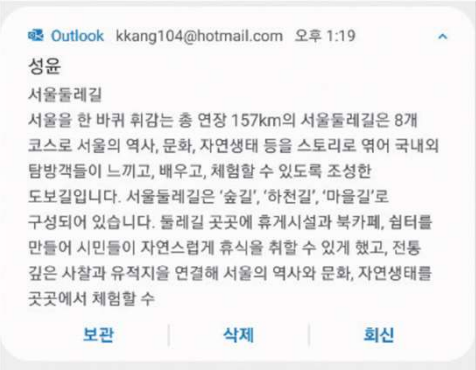
- 큰 이미지 스타일

• 큰 이미지 스타일

```
val bigPicture = BitmapFactory.decodeResource(resources, R.drawable.test)
val bigStyle = NotificationCompat.BigPictureStyle()
bigStyle.bigPicture(bigPicture)
builder.setStyle(bigStyle)
```

10-5 알림 띄우기

- 긴 텍스트 스타일



• 긴 텍스트 스타일

```
val bigTextStyle = NotificationCompat.BigTextStyle()
bigTextStyle.bigText(resources.getString(R.string.long_text))
builder.setStyle(bigTextStyle)
```

- 상자 스타일

• 상자 스타일

```
val style = NotificationCompat.InboxStyle()
style.addLine("1코스 - 수락.불암산코스")
style.addLine("2코스 - 용마.아차산코스")
style.addLine("3코스 - 고덕.일자산코스")
style.addLine("4코스 - 대모.우면산코스")
builder.setStyle(style)
```

▶ 실행 결과

10-5 알림 띄우기

- 메시지 스타일
  - Person은 알림에 출력될 한 사람의 정보를 담는 클래스

• Person 객체 생성

```
val sender1: Person = Person.Builder()
    .setName("kkang")
    .setIcon(IconCompat.createWithResource(this, R.drawable.person1))
    .build()

val sender2: Person = Person.Builder()
    .setName("kim")
    .setIcon(IconCompat.createWithResource(this, R.drawable.person2))
    .build()
```

• 메시지 객체 생성

```
val message1 = NotificationCompat.MessagingStyle.Message(
    "hello",
    System.currentTimeMillis(),
    sender1
)
val message2 = NotificationCompat.MessagingStyle.Message(
    "world",
    System.currentTimeMillis(),
    sender2
)
```

• 메시지 스타일 만들기

```
val messageStyle = NotificationCompat.MessagingStyle(sender1)
    .addMessage(message1)
    .addMessage(message2)
builder.setStyle(messageStyle)
```

▶ 실행 결과

Do it! 실습

카카오톡 알림 만들기

1단계. 새 모듈 생성하기

- Ch10_Notification이라는 이름으로 새로운 모듈을 만듭니다.
- 뷰 바인딩 기법 추가

2단계. 파일 복사하기

- drawable 디렉터리의 big.jpg, small.png, send.png 파일을 res/drawable 디렉터리에 복사
- layout/activity_main.xml 파일을 res/layout 디렉터리에 복사해 이전 파일을 대체

Do it! 실습 시계 앱의 스톱워치 기능 만들기

3단계. 브로드캐스트 리시버 작성하기

- [New → Other → Broadcast Receiver]
- ReplyReceiver 라는 브로드캐스트 리시버 작성

4단계. 메인 액티비티 작성하기

- MainActivity.kt 파일

Do it! 실습

시계 앱의 스톱워치 기능 만들기

5단계. 앱 실행하기

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare