

# Intent Filter

# Implicit Intent

- 특징

- 특정 클래스를 지정해서 메시지를 호출하는 것이 아님
- Action의 유형이나, 데이터의 유형 등의 지정에 의해 수행됨
- 해당 Action 유형을 받고자 하는 Activity 등에 전달됨

- Actions 와 Categories <<Activity Actions>>

- <<Broadcast Actions>>

- ACTION\_TIME\_TICK
- ACTION\_TIME\_CHANGED
- ACTION\_TIMEZONE\_CHANGED
- ACTION\_BOOT\_COMPLETED
- ACTION\_PACKAGE\_ADDED
- ACTION\_PACKAGE\_CHANGED
- ACTION\_PACKAGE\_REMOVED
- ACTION\_PACKAGE\_RESTARTED
- ACTION\_PACKAGE\_DATA\_CLEARED
- ACTION\_UID\_REMOVED
- ACTION\_BATTERY\_CHANGED
- ACTION\_POWER\_CONNECTED
- ACTION\_POWER\_DISCONNECTED
- ACTION\_SHUTDOWN

- ACTION\_MAIN
- ACTION\_VIEW
- ACTION\_ATTACH\_DATA
- ACTION\_EDIT
- ACTION\_PICK
- ACTION\_CHOOSER
- ACTION\_GET\_CONTENT
- ACTION\_DIAL
- ACTION\_CALL
- ACTION\_SEND
- ACTION\_SENDTO
- ACTION\_ANSWER
- ACTION\_INSERT
- ACTION\_DELETE
- ACTION\_RUN
- ACTION\_SYNC
- ACTION\_PICK\_ACTIVITY
- ACTION\_SEARCH
- ACTION\_WEB\_SEARCH
- ACTION\_FACTORY\_TEST

- <<Category>>

- CATEGORY\_DEFAULT
- CATEGORY\_BROWSABLE
- CATEGORY\_TAB
- CATEGORY\_ALTERNATIVE
- CATEGORY\_SELECTED\_ALTERNATIVE
- CATEGORY\_LAUNCHER
- CATEGORY\_INFO
- CATEGORY\_HOME
- CATEGORY\_PREFERENCE
- CATEGORY\_TEST
- CATEGORY\_CAR\_DOCK
- CATEGORY\_DESK\_DOCK
- CATEGORY\_LE\_DESK\_DOCK
- CATEGORY\_HE\_DESK\_DOCK
- CATEGORY\_CAR\_MODE
- CATEGORY\_APP\_MARKET

# Intent Filter

- 인텐트 필터
  - 컴포넌트가 어떤 액션과 데이터를 처리할 수 있는지에 대한 정보를 기술
    - 컴포넌트가 자신이 처리할 수 있는 인텐트의 종류를 인텐트 필터에 기록
  - 응용 프로그램의 매니페스트에 기록



# Intent Filter

- Intent Filter의 구성요소
  - Action 필터
    - 인텐트 객체 내의 action을 검사하여 인텐트 필터에 정의된 액션과 일치하는지 검사
  - Category 필터
    - 인텐트 객체 내의 category을 검사하여 인텐트 필터에 정의된 category와 일치하는지 검사, 꼭 명시되어야 함.
      - 인텐트 객체의 디폴트 **category** : `android.intent.category.DEFAULT`
  - Data 필터
    - 인텐트 객체 내의 data 항목 및 type을 검사하여 인텐트 필터에 정의된 값과 비교하여 일치 여부 검사

# Intent Filter

< <AndroidManifest.xml> >

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <action android:name="android.intent.action.EDIT"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="com.example.myapp.PHONE"/>
        <data android:scheme="tel"/>
    </intent-filter>
</activity>
```

```
val i = Intent(Intent.ACTION_VIEW, Uri.parse(uriString: "tel:"))
context.startActivity(i)
```

```
val i = Intent(Intent.ACTION_EDIT, Uri.parse(uriString: "tel:"))
i.addCategory(category: "com.example.myapp.PHONE")
context.startActivity(i)
```

# Intent Filter

- Intent.resolveActivity(PackageManager) 함수
  - ResolveActivity : 설정에 맞는 Activity를 찾는 클래스
    - ResolveActivity가 조건에 해당하는 컴포넌트가 없으면 null 값을 반환 , 여러 개이면 선택 창을 띄움

```
val i = Intent(ACTION_EDIT, Uri.parse(uriString: "tel:"))
val info : ComponentName! = i.resolveActivity(context.packageManager)
if(info!=null)
    context.startActivity(i)
else
    Toast.makeText(context, text: "실행할 앱이 존재하지 않음",
                    Toast.LENGTH_SHORT).show()
```

# Intent Filter

- PackageManager.queryIntentActivities 함수
  - 인텐트에 설정된 조건에 맞게 실행가능한 Activity 찾는 함수
    - 인텐트
    - Flags : 결과 데이터에 적용될 추가 옵션 ( 0 또는 flag조합(MATCH\_DEFAULT\_ONLY))

```
val i = Intent(Intent.ACTION_CALL, Uri.parse(uriString: "tel:"))
i.addCategory(category: "com.example.myapp.PHONE")
val activities : (Mutable)List<ResolveInfo!> = context.packageManager.queryIntentActivities(i, 0)
if(activities.size!=0){
    for(info : ResolveInfo! in activities){
        Log.d(tag: "test", info.activityInfo.name.toString())
    }
}
```

# Intent Filter

- PackageManager.getLaunchIntentForPackage 함수
  - 패키지 이름으로 앱 실행

```
val intent = context.packageManager.getLaunchIntentForPackage(packageName)
if (intent != null) {
    context.startActivity(intent)
} else {
    Toast.makeText(context, "앱을 실행할 수 없습니다.", Toast.LENGTH_SHORT).show()
}
```



# 실습. 앱 리스트 작성

- 스마트 폰에 등록된 앱 정보를 가지고 와서 리스트를 작성하고, 클릭할 경우 해당 앱을 실행한다

\* android 11 이상

```
<queries>
```

```
    <package android:name="com.example.store" />
```

```
    <package android:name="com.example.services" />
```

```
</queries>
```

```
<queries>
```

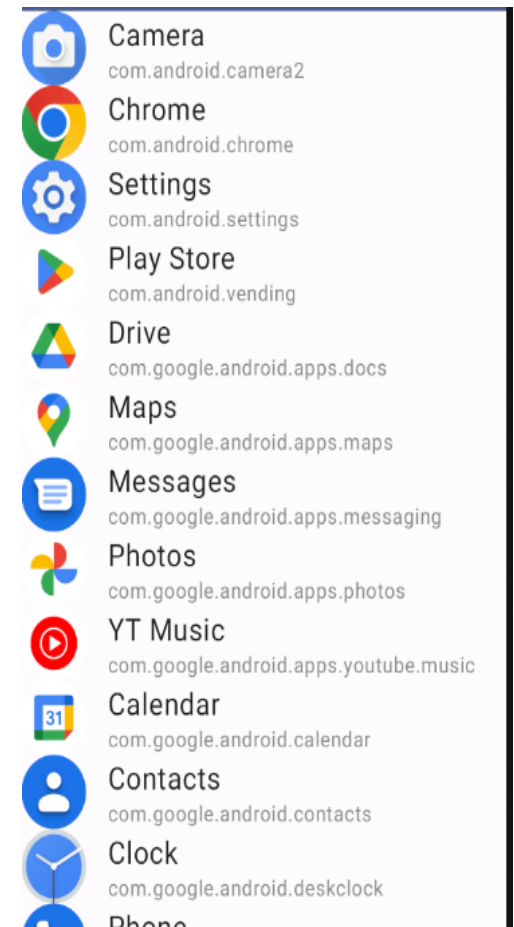
```
    <intent>
```

```
        <action android:name="android.intent.action.SEND" />
```

```
        <data android:mimeType="image/jpeg" />
```

```
    </intent>
```

```
</queries>
```



**Pending Intent**

# PendingIntent

- 인텐트 정보를 가지고 있다가 다른 어플리케이션에 인텐트를 전달함
  - 알림, 알람, AppWidget 등을 서비스 할 때 사용하는 Intent
    - . getActivity(Context context, int, Intent, int flags)
    - . getActivities, getBroadcast, getService(Context context, int, Intent, int flags)
  - int : request code
  - flags
    - FLAG\_CANCEL\_CURRENT
      - 이전에 생성한 PendingIntent는 취소하고, 새롭게 만들
    - FLAG\_UPDATE\_CURRENT
      - 이미 생성된 PendingIntent가 존재 한다면, 해당 Intent 의 내용을 업데이트
    - FLAG\_NO\_CREATE
      - 새로운 PendingIntent 객체가 만들어지지 않고, 이미 생성된 PendingIntent를 그대로 사용 함, 만약 만들어진게 없다면 null 반환
    - FLAG\_ONE\_SHOT
      - 단 한번만 PendingIntent를 만들기 위해 사용, 이미 만들어진 게 있다면 fail 발생

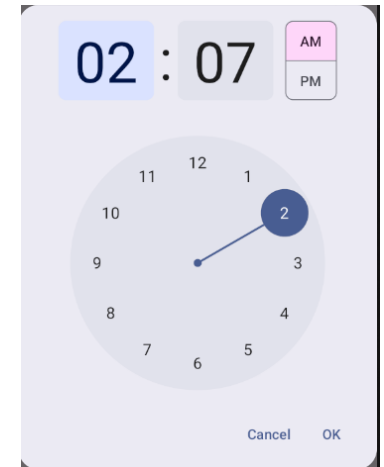
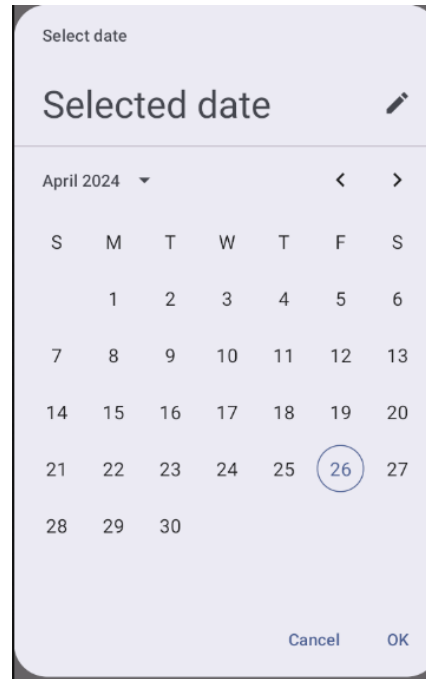
# PendingIntent

- Android 12 (S+ API 31) 이상을 타겟팅할 경우 변경여부 지정해야 함
  - flags
    - FLAG\_IMMUTABLE
      - 다른 앱이 인텐트를 수정하여 Intent 호출 결과 조정할 수 없음
    - FLAG\_MUTABLE
      - 다른 앱이 인텐트를 수정하여 Intent 호출 결과 조정할 수 있음

```
val pendingIntent: PendingIntent =  
    PendingIntent.getActivity(context, requestCode: 0,  
        intent,  
        flags: PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE)
```

# 예제. Notification (알림)

- 5초 후에 알림이 발생하도록 하며, 알림을 선택하면 해당 액티비티로 화면 전환
  - DatePickerDialog
  - TimePicker
  - Notification
    - NotificationChannel
    - Notification.Builder
    - NotificationManager
  - PendingIntent
  - navDeepLink



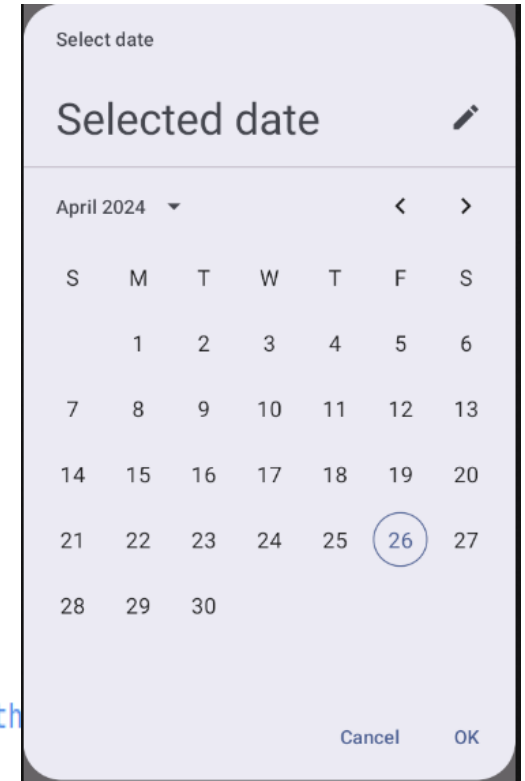
# DatePickerDialog

- DatePicker를 보여주는 다이얼로그

`@ExperimentalMaterial3Api`

`@Composable`

```
fun DatePickerDialog(
    onDismissRequest: () -> Unit,
    confirmButton: @Composable () -> Unit,
    modifier: Modifier = Modifier,
    dismissButton: @Composable (() -> Unit)? = null,
    shape: Shape = DatePickerDefaults.shape,
    tonalElevation: Dp = DatePickerDefaults.TonalElevation,
    colors: DatePickerColors = DatePickerDefaults.colors(),
    properties: DialogProperties = DialogProperties(usePlatformDefaultWidth = false),
    content: @Composable ColumnScope.() -> Unit
) {
```



```
val datePickerState : DatePickerState = rememberDatePickerState()
```

\* content로 DatePicker 전달

```
DatePicker(state = datePickerState)
```

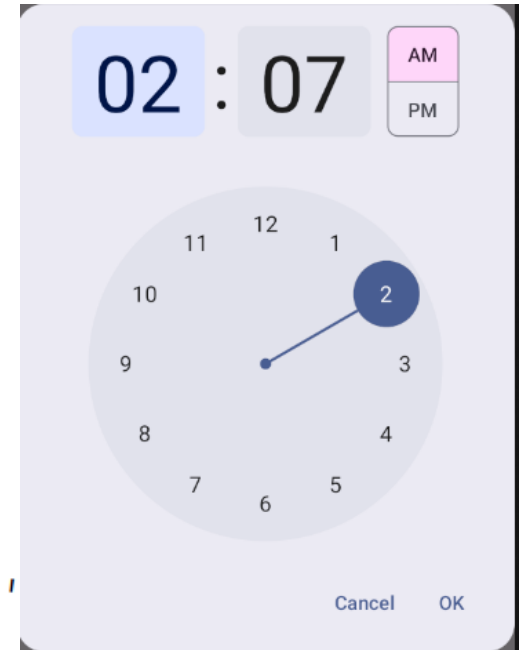
# TimePicker

- TimePicker

`@Composable`

`@ExperimentalMaterial3Api`

```
fun TimePicker(  
    state: TimePickerState,  
    modifier: Modifier = Modifier,  
    colors: TimePickerColors = TimePickerDefaults.colors(),  
    layoutType: TimePickerLayoutType = TimePickerDefaults.layoutType(),  
) {
```

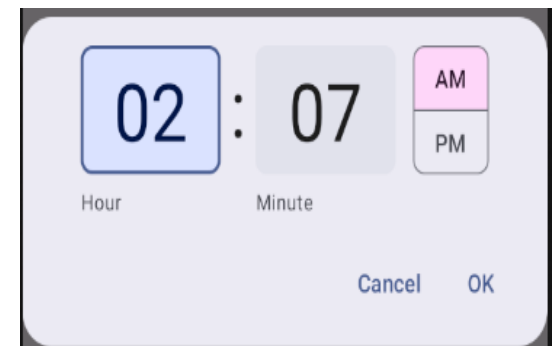


- TimeInput

`@Composable`

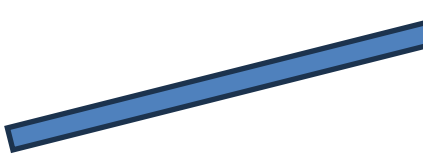
`@ExperimentalMaterial3Api`

```
fun TimeInput(  
    state: TimePickerState,  
    modifier: Modifier = Modifier,  
    colors: TimePickerColors = TimePickerDefaults.colors(),  
) {
```



# TimePickerDialog 생성

@Composable

```
fun TimePickerDialog(
    onDismissRequest: () -> Unit,
    confirmButton: @Composable () -> Unit,
    dismissButton: @Composable () -> Unit)? = null,
    content: @Composable () -> Unit,
) {
    Dialog(
        onDismissRequest = onDismissRequest,
        properties = DialogProperties(
            usePlatformDefaultWidth = false
        ),
    ) {
        Surface(
            shape = MaterialTheme.shapes.extraLarge,
            tonalElevation = 10.dp,
            modifier = Modifier
                .fillMaxSize()
        ) {
            

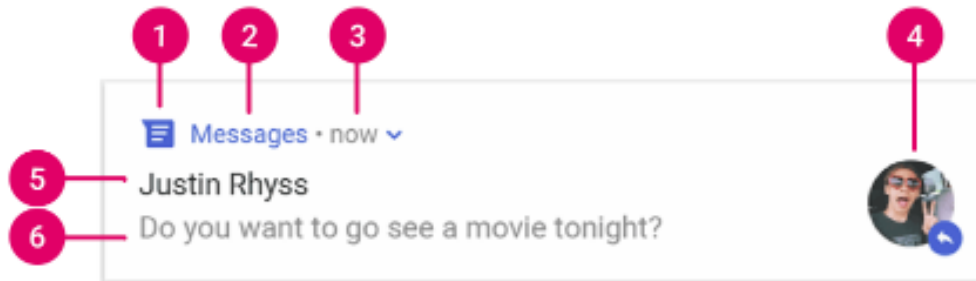
```

```
Column(
    modifier = Modifier
        .padding(20.dp)
        .fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally
) {
    content()
    Row(
        modifier = Modifier
            .height(40.dp)
            .fillMaxWidth()
    ) {
        Spacer(modifier = Modifier.weight(1f))
        dismissButton?.invoke()
        confirmButton()
    }
}
```



# Notification (알림)

- 안드로이드가 앱의 UI외부에 표시하는 메시지
  - 사용자에게 알림, 다른사람과의 커뮤니케이션 또는 앱의 기타 정보를 제공하는 메시지



- 1 Small icon: This is required and set with `setSmallIcon()`.
- 2 App name: This is provided by the system.
- 3 Time stamp: This is provided by the system but you can override with `setWhen()` or hide it with `setShowWhen(false)`.
- 4 Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with `setLargeIcon()`.
- 5 Title: This is optional and set with `setContentTitle()`.
- 6 Text: This is optional and set with `setContentText()`.

# Notification

- 알림 생성 및 관리 ( Android 8.0(API level 26), Oreo 버전 이후)
  - NotificationChannel 생성
  - Notification.Builder 생성
    - NotificationChannel을 기반으로 Notification.Builder 생성
    - Notification 관련 정보(아이콘, 텍스트, **PendingIntent** 등) 셋팅
    - Notification생성
  - NotificationManager
    - Builder가 생성한 Notification을 notify

<https://developer.android.com/reference/android/app/Notification>

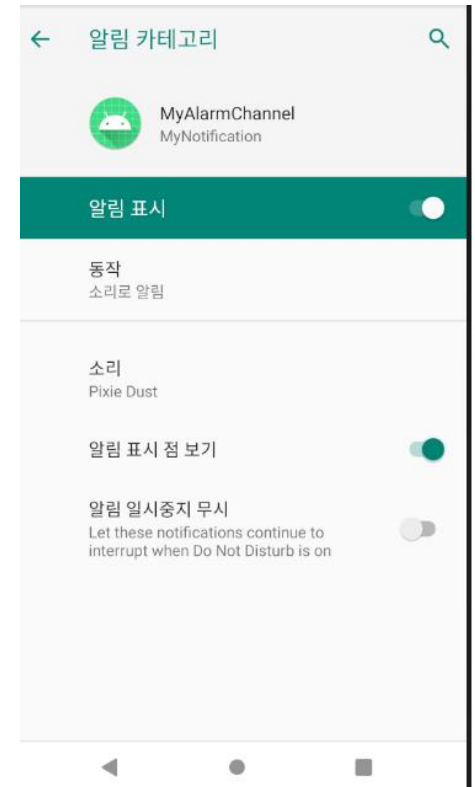
<https://developer.android.com/reference/android/app/NotificationChannel.html>

<https://developer.android.com/reference/android/app/Notification.Builder>

<https://developer.android.com/reference/android/app/NotificationManager>

# NotificationChannel

- Notification의 효과를 설정
  - Channel ID
  - Channel Name : 사용자에게 보여줄 채널이름
  - Description(옵션) : 사용자에게 표시되는 설명
  - Importance Level
    - IMPORTANCE\_HIGH (Urgent)
      - 사운드 출력, Head-up 알림으로 나타남
    - IMPORTANCE\_DEFAULT (HIGH)
      - 사운드 출력
    - IMPORTANCE\_LOW (Medium)
      - 사운드 출력 안됨
    - IMPORTANCE\_MIN (LOW)
      - 상태바에 표시되지 않고, 상태바를 내려야 보여짐



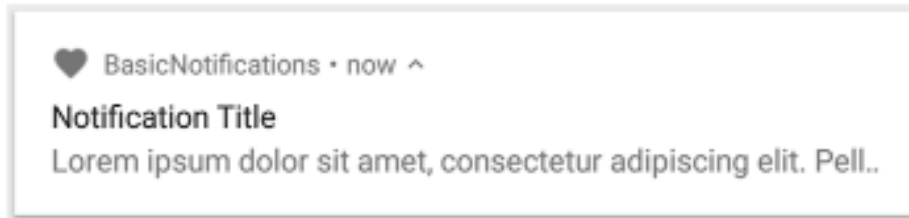
# NotificationChannel

- NotificationChannel 생성

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  
    val name = "MyTestChannel"  
    val descriptionText = "My important test channel"  
    val importance : Int = NotificationManager.IMPORTANCE_HIGH  
    val channel : NotificationChannel = NotificationChannel(channelId, name, importance).apply { this: NotificationChannel  
        description = descriptionText  
    }  
  
    with(NotificationManagerCompat.from(context)) { this: NotificationManagerCompat  
        createNotificationChannel(channel)  
    }  
}
```

# NotificationBuilder

- NotificationBuilder 생성



```
val mymessage = "일정확인하세요"
```

```
val builder : NotificationCompat.Builder! =  
    NotificationCompat.Builder(context: this, channelId)  
        .setSmallIcon(R.mipmap.ic_launcher)  
        .setContentTitle("할일확인")  
        .setContentText(mymessage)  
        .setColor(ContextCompat.getColor(context: this, R.color.colorPrimary))  
        .setAutoCancel(true)
```

- SmallIcon : 24x24dp 크기로 투명 배경에 흰색 아이콘
- Title : 한 줄에 표시되는 제목
- Text : 세부 정보

# PendingIntent 추가

- 알림 메시지 확인 후 특정 위치 이동

```
val intent = Intent(packageContext: this, MainActivity::class.java)
intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TOP
intent.putExtra(name: "time", alarmString)
```

```
val pIntent: PendingIntent! = PendingIntent.getActivity(context: this, requestCode: 1,
                                                        intent,
                                                        PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE)
builder.setContentIntent(pIntent)
```

\*android 12 이상

- 인텐트 플래그 : 인텐트에 의해 액티비티 생성 방법 설정

- FLAG\_ACTIVITY\_NEW\_TASK

- 액티비티를 새 Task에서 시작.

- FLAG\_ACTIVITY\_SINGLE\_TOP

- 시작 중인 액티비티가 현재 액티비티(백 스택의 맨 위에 있는)이면 액티비티의 새 인스턴스가 생성되는 대신 기존 인스턴스가 실행

- FLAG\_ACTIVITY\_CLEAR\_TOP

- 시작 중인 액티비티가 현재 Task에서 이미 실행중이면, 새 인스턴스가 실행되는 대신 Task의 위에 있는 모든 액티비티가 제거시키고, 실행

# NotificationManager

- 알림 메시지를 notify

```
val manager =
```

```
context.getSystemService(Context.NOTIFICATION_SERVICE)
```

```
as NotificationManager
```

```
manager.createNotificationChannel(notificationChannel)
```

```
val notification = builder.build()
```

```
manager.notify(10, notification)
```

\* Android 13 이상

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

# 예제. Notification 예제 실행하기

- Notification 예제 실행



# DeepLink

- 외부에 앱의 목적지 노출할 때 딥 링크 사용
  - 인텐트 필터 추가
    - Data 태그 : scheme, host 명시 ( 예, myapp://greenjoahome.com )

```
<intent-filter android:autoVerify="true">  
    <action android:name="android.intent.action.VIEW" />  
  
    <category android:name="android.intent.category.DEFAULT" />  
    <category android:name="android.intent.category.BROWSABLE" />  
  
    <data android:scheme="myapp"  
        android:host="greenjoahome.com" />  
</intent-filter>
```

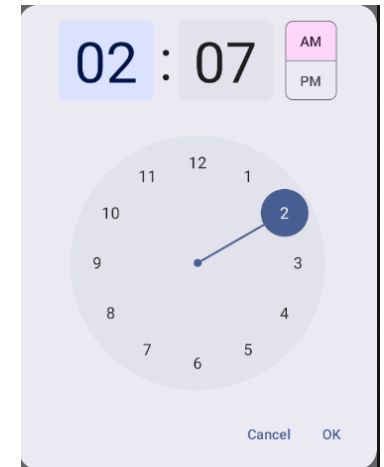
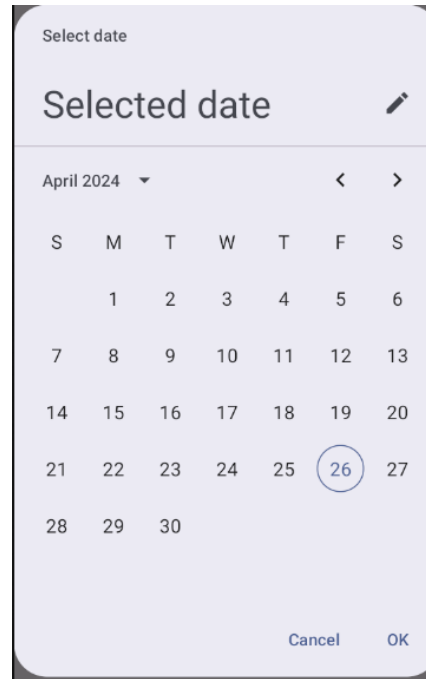
```
composable(  
    route = "Msg?msg={msg}",  
    deepLinks = listOf(  
        navDeepLink {  
            uriPattern = "myapp://greenjoahome.com/{msg}"  
            action = Intent.ACTION_VIEW  
        }  
    ),  
    arguments = listOf(  
        navArgument("msg") {  
            type = NavType.StringType  
        }  
    )  
)
```

# DeepLink 이용시 PendingIntent

```
val pendingIntent = TaskStackBuilder.create(context).run {  
    addNextIntentWithParentStack(  
        Intent(Intent.ACTION_VIEW, "myapp://greenjoahome.com/$msg".toUri())  
    )  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S)  
        getPendingIntent(1234, PendingIntent.FLAG_UPDATE_CURRENT or  
PendingIntent.FLAG_MUTABLE)  
    else  
        getPendingIntent(1234, PendingIntent.FLAG_UPDATE_CURRENT)  
}
```

# 팀 실습. Notification (알림)

- 5초 후에 알림이 발생하도록 하며, 알림을 선택하면 해당 액티비티로 화면 전환 (딥링크)
  - DatePickerDialog
  - TimePicker
  - Notification
    - NotificationChannel
    - Notification.Builder
    - NotificationManager
  - PendingIntent
  - navDeepLink



**수고하셨습니다.**