Android Intents

Mobile Software

2022 Fall

What to do next?

- **Activity Stack과 intent**
- Explicit intent
- Activity로부터 결과 돌려받기
- Implicit intent
- Intent filter

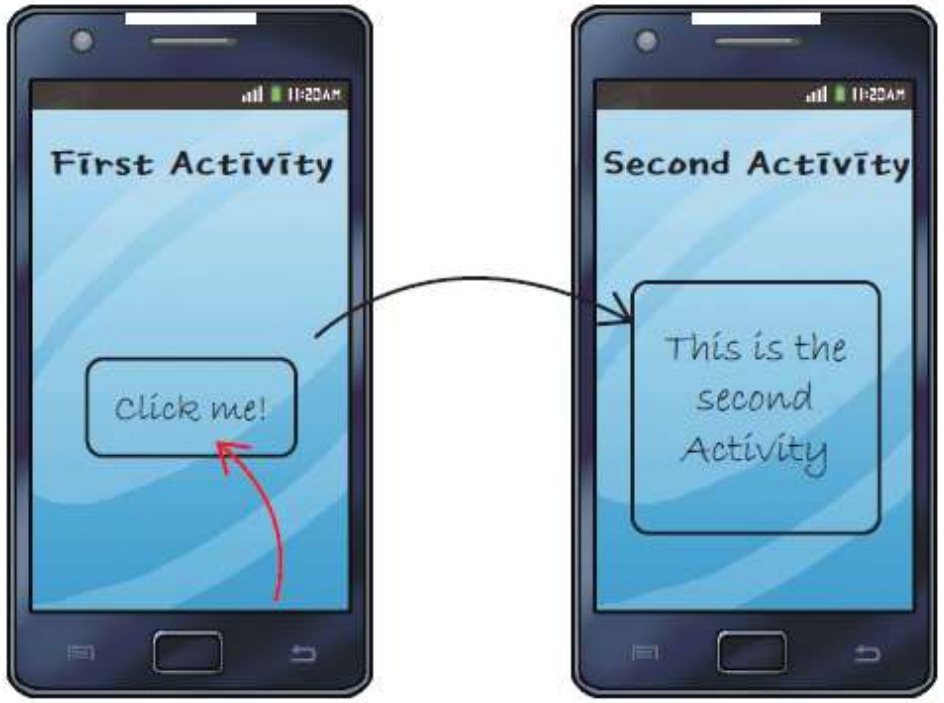
- **강의 노트에 포함된 코드: 소스코드(intent).hwp**
  - **Layout 관련 설명은 가급적 생략. 상세 내용은 소스 코드 참조.**

Application

- 한 개 이상의 activity들로 구성
- 애플리케이션 안에서 activity들은 **느슨하게** 연결

Intent (1/4)

- First Activity에서 Second Activity로 전환하려면?
  - **Intents** are the **messaging system** by which one activity is able to **launch** another activity.

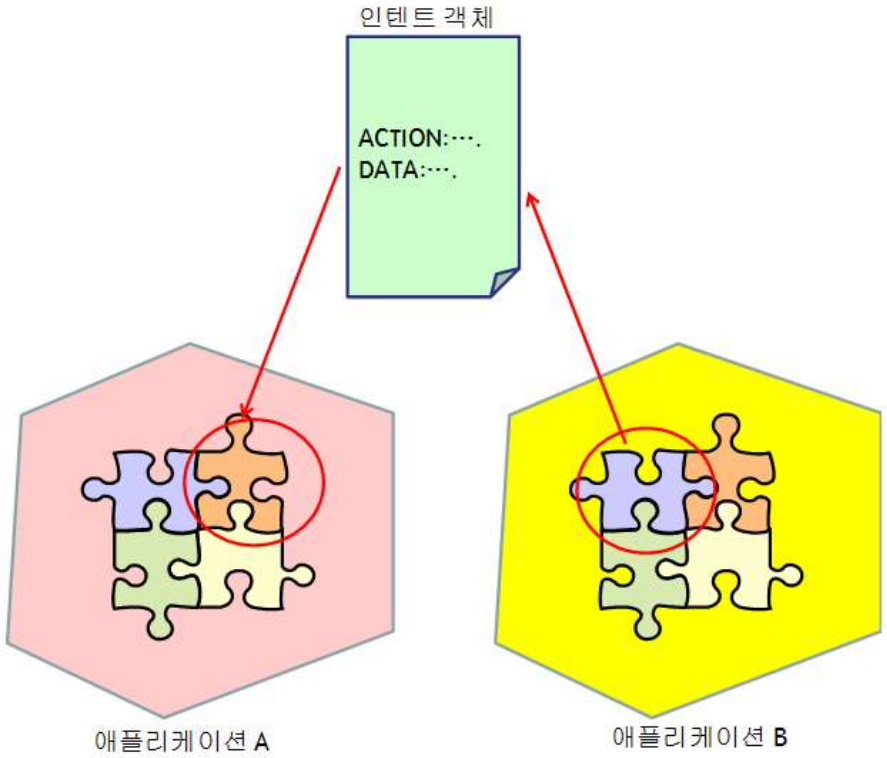


Intent (2/4)

- An intent is an ***abstract description*** of an **operation to be performed**.
- It can be used with
  - **startActivity** to launch an **Activity**
  - **sendBroadcast** to send it to any interested **BroadcastReceiver** components.
  - **startService**(Intent) or
  - **bindService**(Intent, ServiceConnection, flags) to communicate with a background **Service**.

Intent (3/4)

- 다른 activity를 시작하려면 activity 실행에 필요한 여러 가지 정보를 전달해야 한다.
  - 정보를 intent에 실어서 보낸다.



Intent (4/4)

- Manifest 파일에 컴포넌트에 관한 상세 정보를 등록
  - 안드로이드 시스템에서 manifest에 등록된 컴포넌트 목록과 intent의 정보를 비교하여 적절한 컴포넌트를 찾기 때문


```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.intentsample">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Intent Sample"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.IntentSample">
        <activity android:name=".SubActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

What to do next?

- Activity Stack과 intent
- **Explicit intent**
- Activity로부터 결과 돌려받기
- Implicit intent
- Intent filter

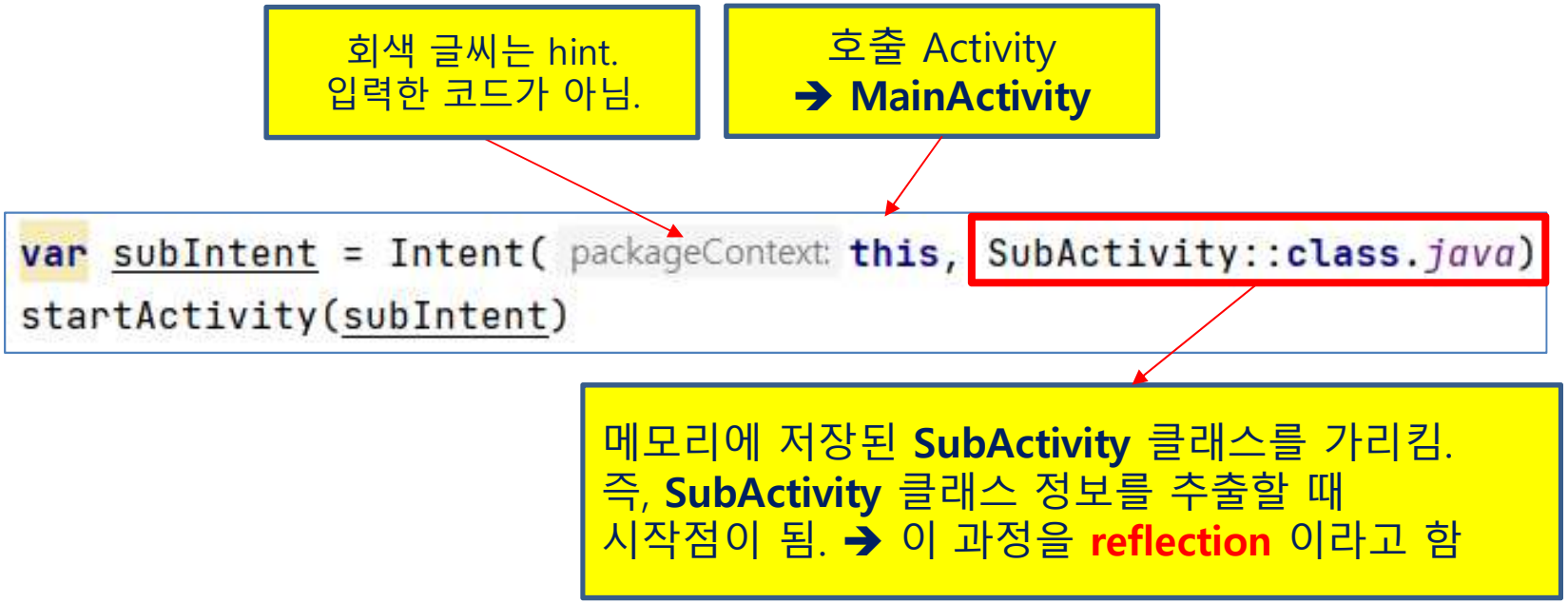
Intent 종류

- **Explicit intent (명시적 인텐트)**
  - 구체적으로 지정
    - "애플리케이션 A의 컴포넌트 B를 작동시켜라"

- **Implicit intent (암시적 인텐트)**
  - 기본 조건만 지정
    - "지도를 보여줄 수 있는 컴포넌트라면 어떤 거라도 괜찮아"

Explicit intent

- 실행하려고 하는 activity의 이름을 지정
  - This approach is most common when launching **an activity residing in the same application** as the sending activity.
  - 개발자는 이미 이 클래스 이름을 알고 있음

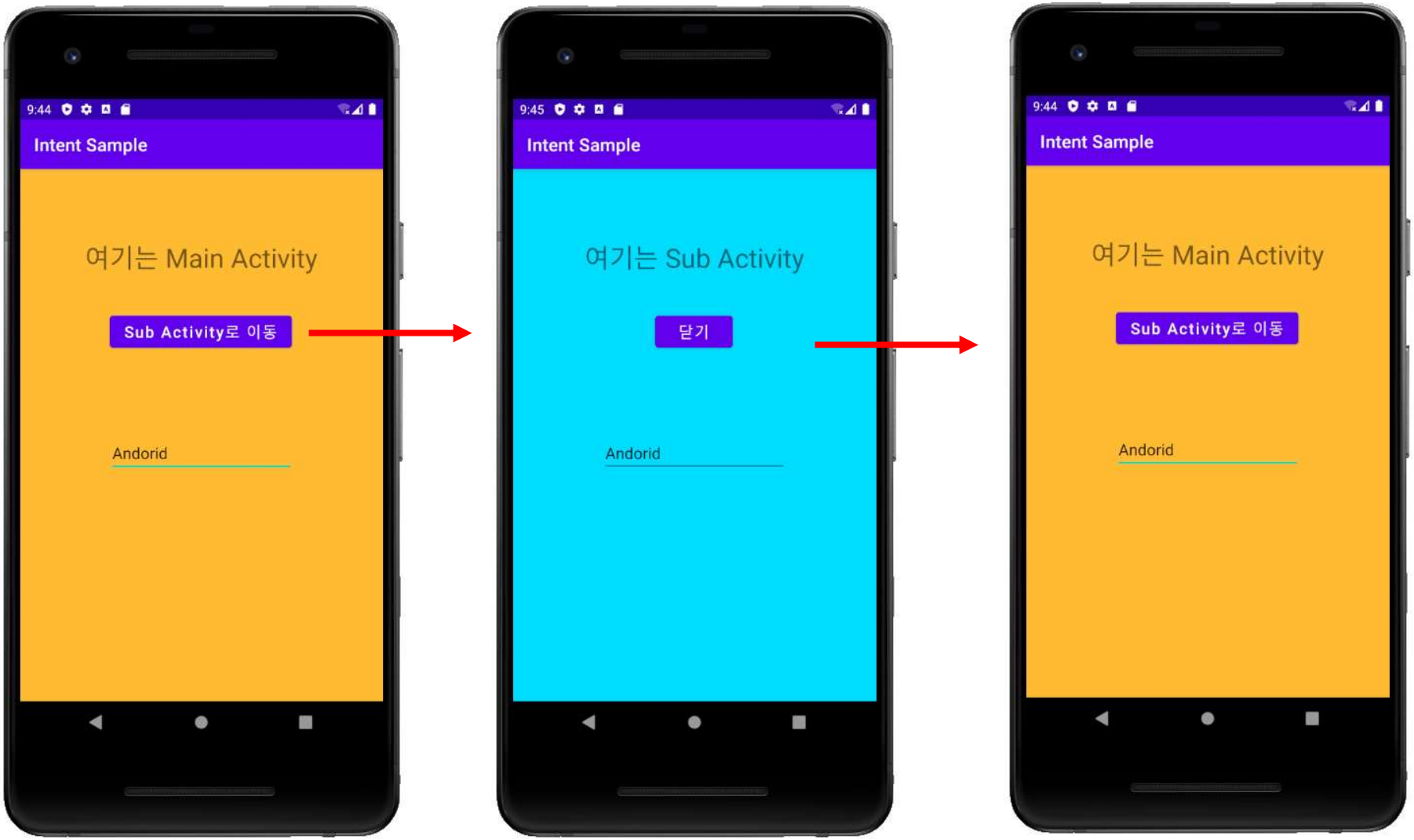


실습 프로젝트 생성

- 새 프로젝트 생성
  - **Project name** : **Intent Sample**
  - **Package name : com.example.intentsample**
  - **Activity** : **Empty Activity**
  - **Activity name** : **MainActivity.kt**
  - **Layout name** : **activity_main.xml**

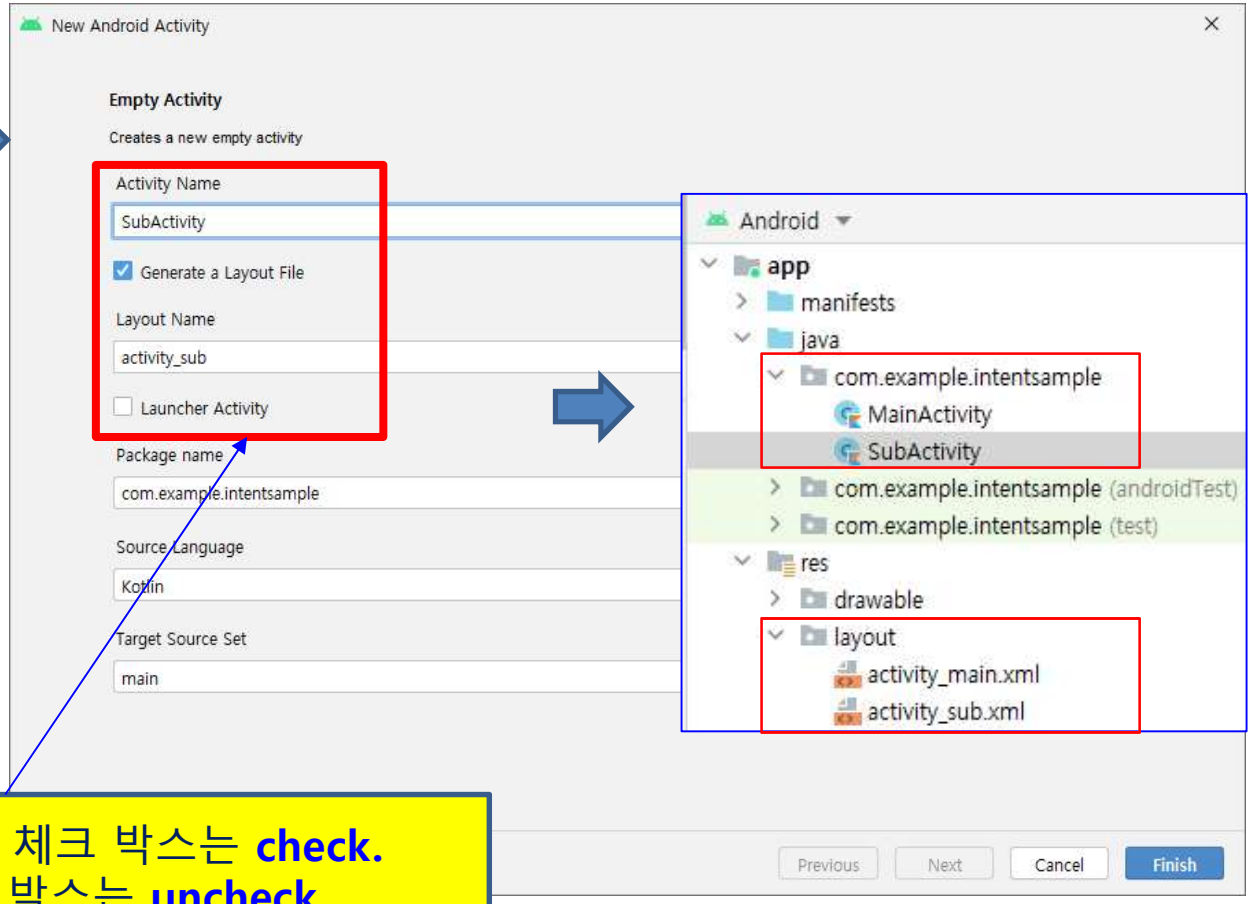
- **자동 생성된 XML 파일**의 root view는 **ConstraintLayout**

실습 1: 2개 Activity – Explicit Intent



실습 1: SubActivity 생성

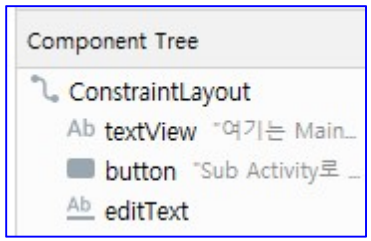
App 클릭
→ 마우스 오른쪽 버튼
→ New → **Activity**
→ **Empty Activity**

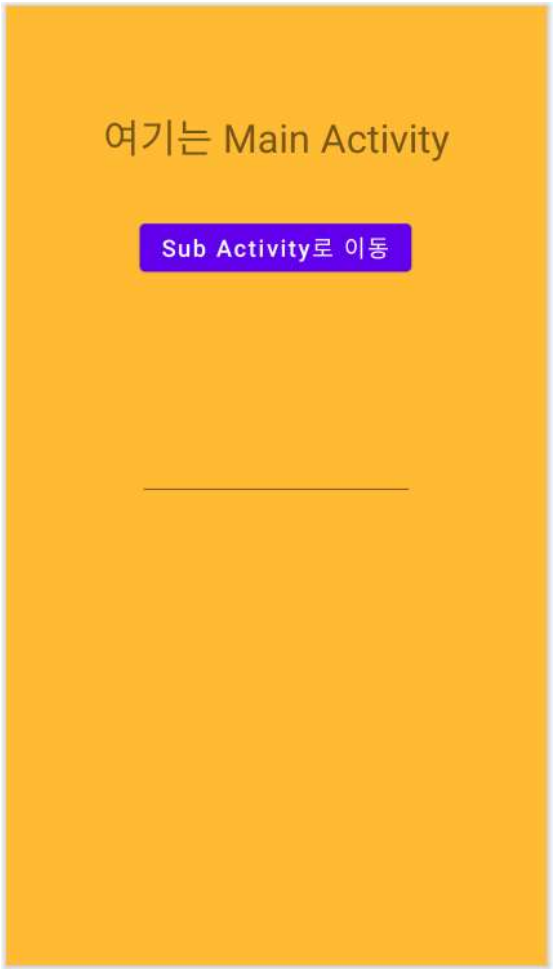


Generate a Layout File 체크 박스는 **check.**
Launcher Activity 체크 박스는 **uncheck.**

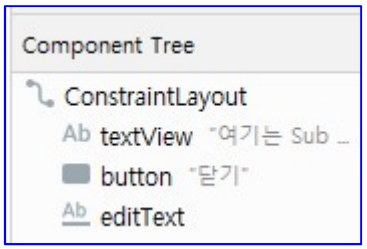
실습 1: Activity - 레이아웃

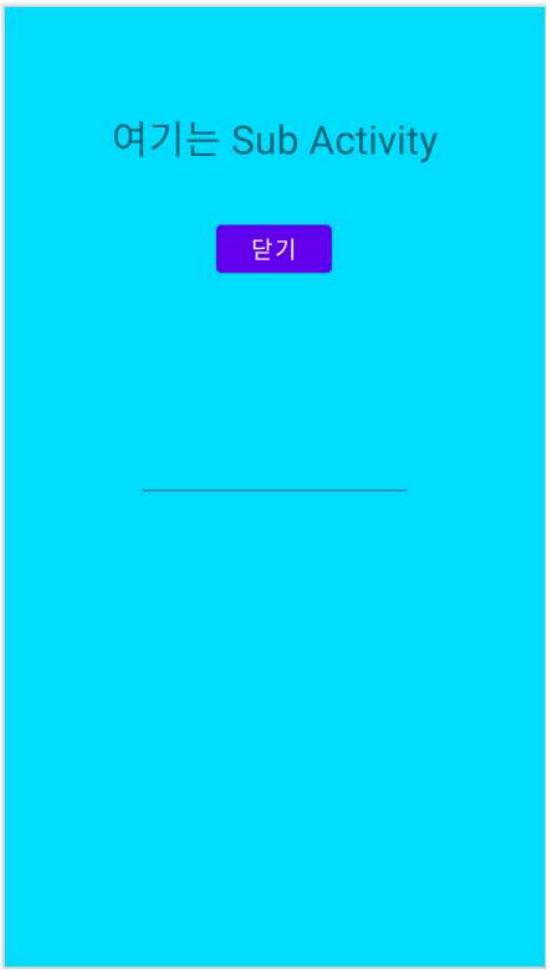
activity_main.xml





activity_sub.xml

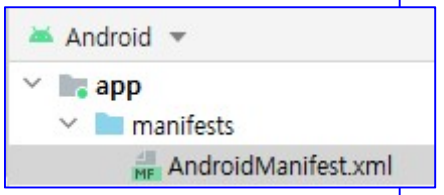




소스코드 – 1~3쪽

실습 1: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.intentsample">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Intent Sample"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.IntentSample">
        <activity
            android:name=".SubActivity"
            android:exported="true" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```



exported = "true"
외부에서 intent를 전달하여
해당 컴포넌트를 활성화할 수 있음.

Activity 이름 맨 앞의 dot(.)는
패키지 표현의 일부임을 나타냄.
즉 패키지(**com . Example. intentsample**)
이름에 이어진다는 뜻.

소스코드 – 4쪽

실습 1: MainActivity

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        findViewById<Button>(R.id.button).setOnClickListener {  it: View!
            var subIntent = Intent( packageContext: this@MainActivity,
                                    SubActivity::class.java)
            startActivity(subIntent)
        }
    }
}
```

소스코드 – 5쪽

실습 1: SubActivity

```
class SubActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sub)

        findViewById<Button>(R.id.button).setOnClickListener {  it: View!
            finish( )
        }
    }
}
```

소스코드 – 5쪽

잠깐! 컴포넌트 이름을 사용

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        findViewById<Button>(R.id.button).setOnClickListener {  it: View!
            val cname = ComponentName( pkg: "com.example.intentsample",
                cls: "com.example.intentsample.SubActivity")
            var subIntent = Intent( )
            subIntent.component = cname
            startActivity(subIntent)
        }
    }
}
```

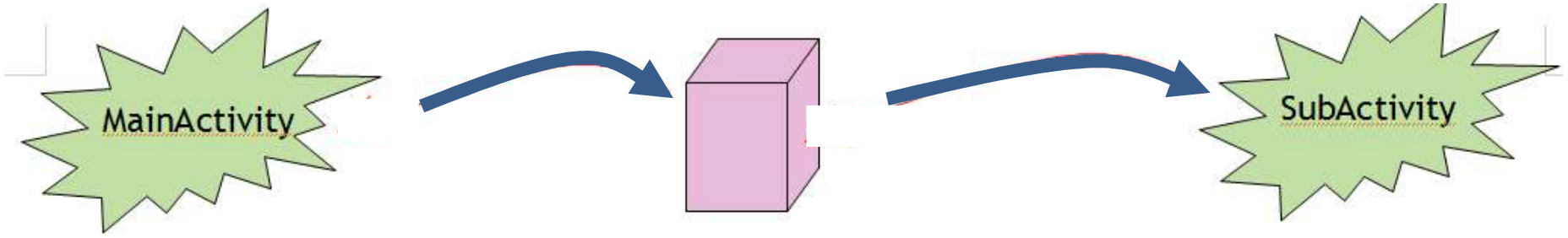
(Package, Activity)

Activity를 지정할 때
Package를 생략하면
Runtime error 발생!

소스코드 – 6쪽

실습 2: Extra 속성을 사용한 데이터 전달

Extra : intent에 포함되어 전달되는 데이터




```
var i = Intent(this, SubActivity::class.java)

val mainString = editText.text.toString()

i.putExtra("mainStr", mainString)
i.putExtra("myInt"    , 100)
startActivity(i)
```

```
if (intent == null || intent.extras == null) return

val extras = intent.extras

val qString = extras.getString("mainStr")
val qInt = extras.getInt("myInt")
editText.setText(qString + qInt.toString())
```

Key-value 쌍의 형태로 **Extra**에 추가

Key만을 사용해서 value 을 가져 옴.
단 **value의 타입**을 알아야 함.

실습 2: MainActivity

```
const val EXTRA_MESSAGE = "com.example.intentsample.MESSAGE"

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val editText: EditText = findViewById(R.id.editText)
        findViewById<Button>(R.id.button).setOnClickListener {  it: View!
            var str = editText.text.toString()

            var subIntent = Intent( packageContext: this,
                                    SubActivity::class.java)
            subIntent.putExtra(EXTRA_MESSAGE, str)
            startActivity(subIntent)
        }
    }
}
```

클래스 밖에서 선언한 문자열 . SubActitivity에서도 참조할 수 있음.

Intent 객체에 **메시지를 포함해서 전달**

소스코드 – 7쪽

실습 2: SubActivity

```
class SubActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sub)

        val message:String? = intent.getStringExtra(EXTRA_MESSAGE)
        val editText: EditText = findViewById(R.id.editText)
        editText.setText(message)

        findViewById<Button>(R.id.button).setOnClickListener {  it: View!
            finish()
        }
    }
}
```

Extra에서
String 값을 가져옴

Activity stack에서
SubActivity를 제거
➔ MainActivity가
다시 활성화 됨.
(화면에 보임)

소스코드 – 7쪽

잠깐! Kotlin – apply 함수

```
val cname = ComponentName( pkg: "com.example.intentsample",
     cls: "com.example.intentsample.SubActivity")
var subIntent = Intent( )
subIntent.component = cname
subIntent.putExtra(EXTRA_MESSAGE, str)
startActivity(subIntent)
```

```
val cname = ComponentName( pkg: "com.example.intentsample",
     cls: "com.example.intentsample.SubActivity")
var subIntent = Intent().apply {  this: Intent
        component = cname
        putExtra(EXTRA_MESSAGE, str)
    }
startActivity(subIntent)
```

Apply 함수는
객체를 생성하면서
객체의 속성을
초기화할 때 사용

블록 { … } 에
객체를 전달하고
블록 실행이 끝나면
객체를 반환

소스코드 – 8쪽

잠깐! Elvis 연산을 사용한 Null safe 검사

```
val message:String? = intent.getStringExtra(EXTRA_MESSAGE)
val editText: EditText = findViewById(R.id.editText)
editText.setText(message)
```

Intent 객체인 경우
getStringExtra( ) 사용

```
var extras:Bundle? = intent.extras ?: null
if (extras?.isEmpty == false) {
    var message = extras?.getString(EXTRA_MESSAGE)
    var editText:EditText = findViewById(R.id.editText)
    editText.setText(message)
}
```

extras 의 타입은 Bundle이기 때문에
getString ( ) 사용

Elvis 연산 기호 사용

```
extras = intent.extras if intent.extras != null
extras = null if intent.extras == null
```

소스코드 – 8쪽

잠깐! 인텐트 객체가 부가 메시지를 포함하는지 확인

SubActivity.kt

```
val message:String? = intent.getStringExtra(EXTRA_MESSAGE)
val editText: EditText = findViewById(R.id.editText)
editText.setText(message)
```

```
if (intent.hasExtra(EXTRA_MESSAGE)) {
    var extras:Bundle? = intent.extras
    var message = extras?.getString(EXTRA_MESSAGE)
    var editText:EditText = findViewById(R.id.editText)
    editText.setText(message)
}
```

소스코드 – 8쪽

What to do next?

- Activity Stack과 intent
- Explicit intent
- **Activity로부터 결과 돌려받기**
- Implicit intent
- Intent filter

Starting Activities and Getting Results(1/3)

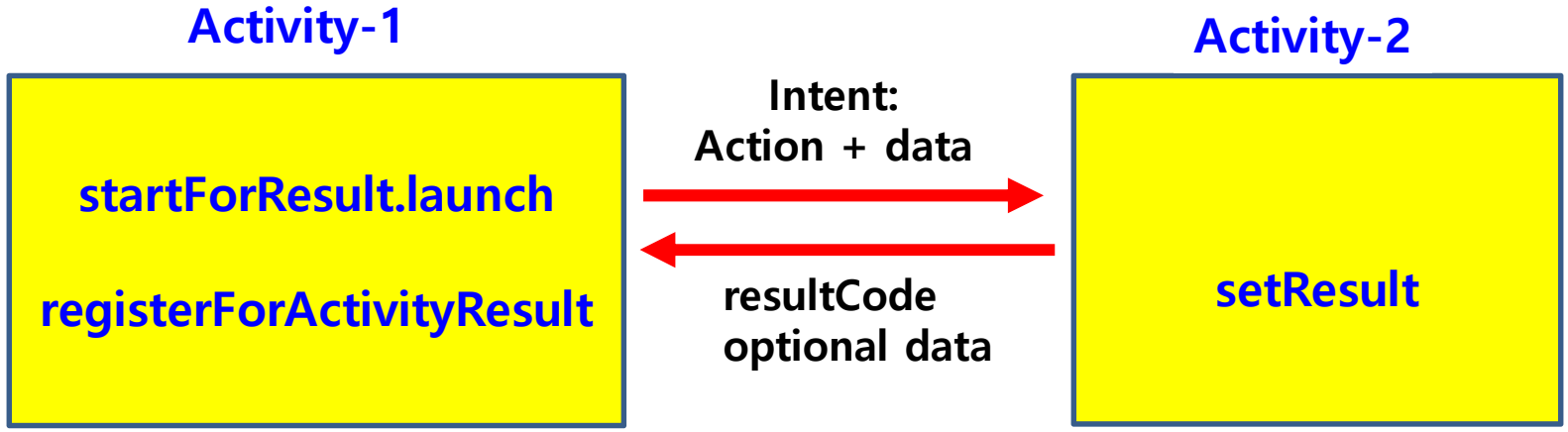
- In order to get results back from the called activity we use the

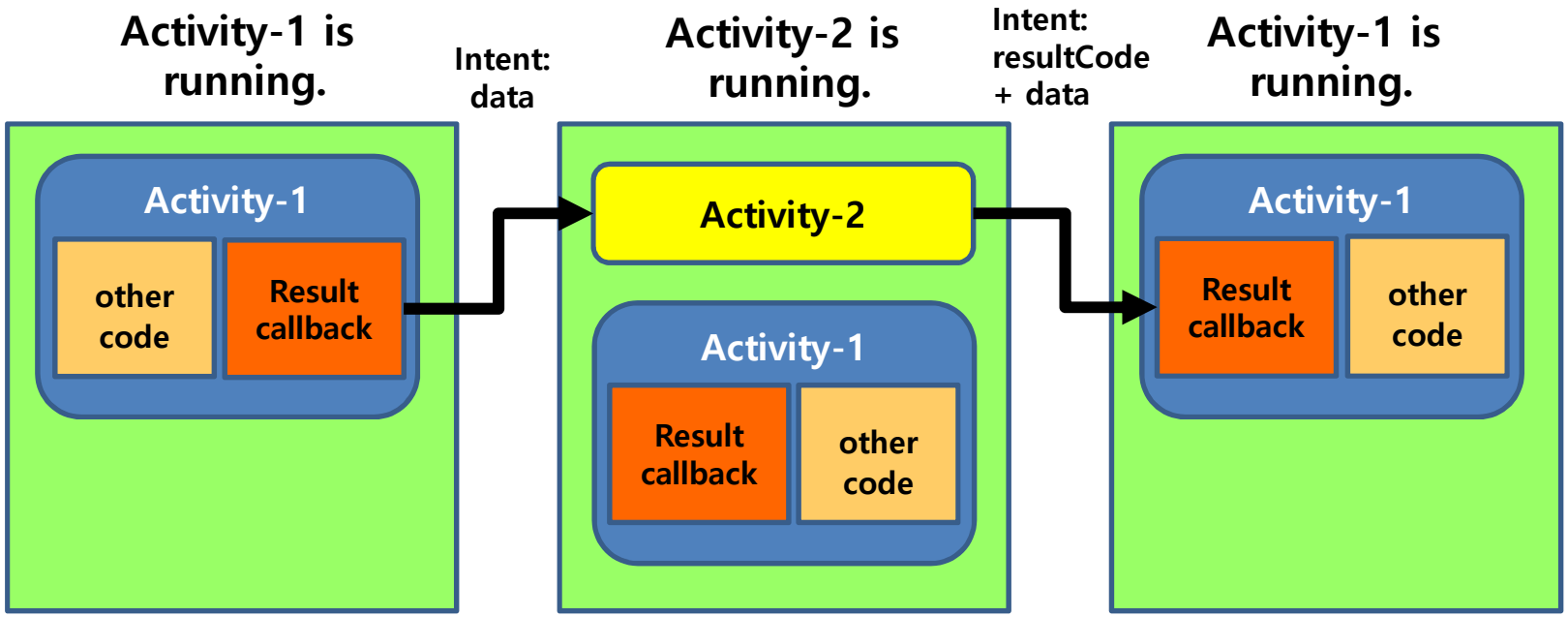
  startForResult . launch (Intent)

- The result sent by the sub-activity could be picked up through the method

  StartActivityForResult ( )

Starting Activities and Getting Results(2/3)





Starting Activities and Getting Results(3/3)

- Before an activity exits, it can call **setResult** (resultCode) to return a termination signal back to its parent.

- It must always supply a result code, which can be the standard results **Activity. RESULT_CANCELED, Activity. RESULT_OK**, or any custom values.

- All of this information can be capture back on the ***parent's*** **StartActivityForResult** .

- If *a child activity fails* for any reason (such as crashing), the parent activity will receive a result with the code **RESULT_CANCELED**.

Activity로부터 Result 가져오기(1)

- **Activity 사이의 작업은 양방향 작업**
  - 어떤 Activity가 다른 Activity로부터 결과를 전달 받음
    - 카메라 App 시작 → 사진 촬영 → 캡처한 사진을 전달 받음.

- **Activity Result API**
  - 메모리 부족으로 인해 프로세스와 Activity가 소멸될 수 있음.
    - 카메라 처럼 메모리를 많이 사용하는 작업의 경우 소멸 확률이 매우 높음.
  - **결과를 돌려줘야 할 Activity가 없어지면 어떻게 될까?**
    - 해결 방법 : **Result callback을 분리**.
      - Activity Result API는 결과를 처리하는 코드와 다른 작업을 하는 코드를 분리.
    - Result callback은 소멸된 프로세스와 Activity가 다시 생성될 때 사용할 수 있어야 하기 때문에 **반드시 사전에 등록**해야 함

참고 - https://developer.android.com/training/basics/intents/result?hl=ko (2021.4.29. 업데이트)

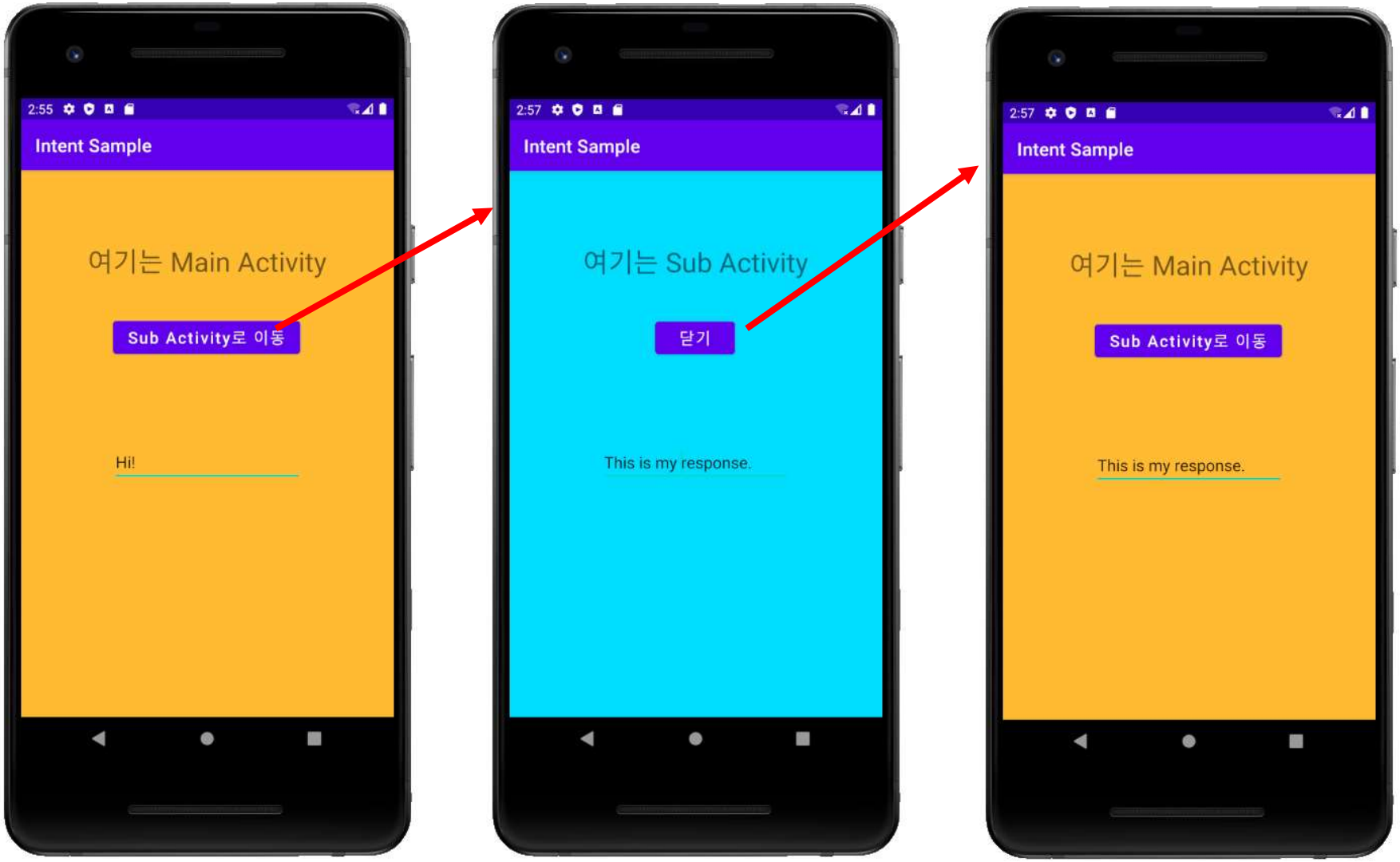
Activity로부터 Result 가져오기(2/2)

- **registerForActivityResult** ( )
  - **ActivityResultContract** 및 **ActivityResultCallback** 을 가져와서
    - 다른 활동을 실행하는 데 사용할 **ActivityResultLauncher**를 반환
  - **callback 등록만 담당**
    - 다른 활동을 실행하거나 결과 요청을 하지 않음
    - 이 작업은 반환된 ActivityResultLauncher 인스턴스가 담당

```
    button.setOnClickListener {  it: View!
        var subIntent = Intent(this, SubActivity::class.java)
        val str = editText.text.toString()
        subIntent.putExtra(EXTRA_MESSAGE, str)
        startForResult.launch((subIntent))
    }
}

private val startForResult = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{...}
```

참고 - https://developer.android.com/training/basics/intents/result?hl=ko (**2021.4.29. 업데이트**)

실습 3: Sub-Activity로부터 결과 돌려받기



실습 3: MainActivity

```
const val EXTRA_MESSAGE = "com.example.intentsample.EXTRA_MESSAGE"
const val RETURN_MESSAGE = "com.example.intentsample.RETURN_MESSAGE"

class MainActivity : AppCompatActivity() {
    lateinit var editText:EditText

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val button:Button = findViewById(R.id.button)
        editText = findViewById(R.id.editText)

        button.setOnClickListener { it: View!
            var subIntent = Intent(this, SubActivity::class.java)
            val str = editText.text.toString()
            subIntent.putExtra(EXTRA_MESSAGE, str)
            startForResult.launch((subIntent))
        }
    }

    private val startForResult = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult())
    {...}
}
```

단순한 Activity 전환이 아니라
결과를 돌려받기 위한 Activity 호출

결과를 돌려받았을 때
처리할 메소드 정의

소스코드 – 9쪽

실습 3: MainActivity

```
private val startForResult = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{ result: ActivityResult ->
    if (result.resultCode == Activity.RESULT_OK) {
        if (result.data != null) {
            val extras:Bundle? = result.data?.extras
            val returnString = extras?.getString(RETURN_MESSAGE) ?: ""
            editText.setText(returnString)
        }
    }
}
```

result. data는
intent 객체를 가리킴

소스코드 – 9쪽

실습 3: SubActivity

```
class SubActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sub)

        var editText:EditText = findViewById(R.id.editText)
        val extras = intent.extras ?: null
        if (intent.hasExtra(EXTRA_MESSAGE)) {
            val message = extras?.getString(EXTRA_MESSAGE)
            editText.setText(message)
        }

        val button:Button = findViewById(R.id.button)
        button.setOnClickListener{ it: View!
            val str = editText.text.toString()
            val data = Intent()
            data.putExtra(RETURN_MESSAGE, str)
            setResult(Activity.RESULT_OK, data)
            finish()
        }
    }
}
```

결과를 bundle 객체와 함께 전달

소스코드 – 10쪽

잠깐! Kotlin - let 함수

```
if (result.resultCode == Activity.RESULT_OK) {
    if (result.data != null) {
        val extras:Bundle? = result.data?.extras
        val returnString = extras?.getString(RETURN_MESSAGE) ?: ""
        editText.setText(returnString)
    }
}
```

```
if (result.resultCode == Activity.RESULT_OK) {
    result.data?.let { it: Intent
        if (it.hasExtra(RETURN_MESSAGE)) {
            val extras: Bundle? = it.extras
            val returnString = extras?.getString(RETURN_MESSAGE) ?: ""
            editText.setText(returnString)
        }
    }
}
```

T.let { ... } ➔ 객체 **T**를 블록 문의 인자로 전달하고,
블록 문의 실행 결과를 반환.
블록 문에서 **it**는 객체 **T**를 가리킴.

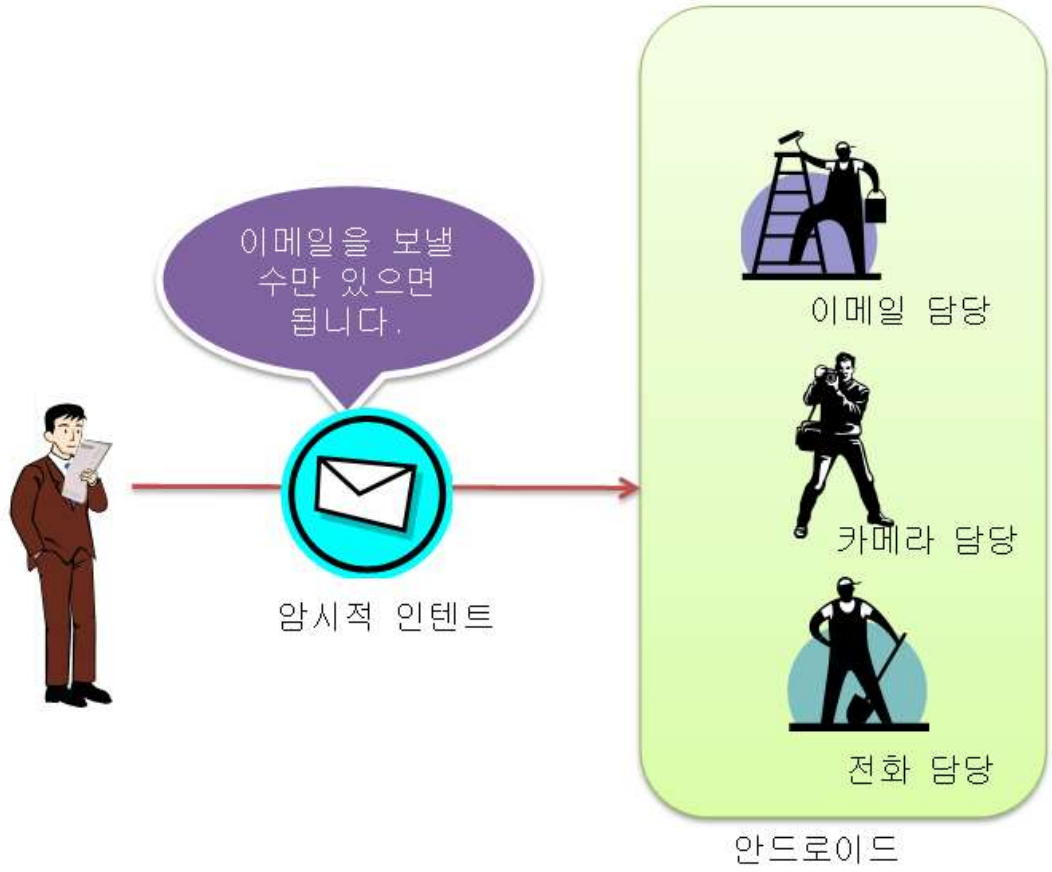
소스코드 – 11쪽

What to do next?

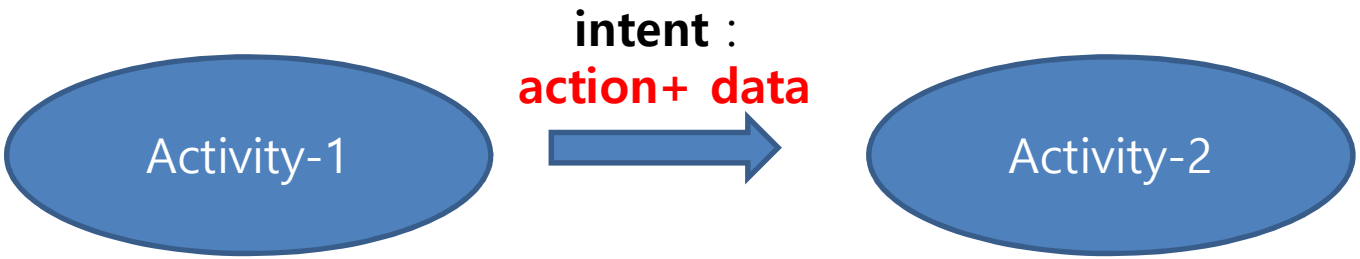
- Activity Stack과 intent
- Explicit intent
- Activity로부터 결과 돌려받기
- **Implicit intent**
- Intent filter

Implicit intent (암시 인텐트)

- 어떤 작업을 하고 싶은 데, 이 작업을 담당하는 컴포넌트의 이름을 정확히 모르는 경우에 사용



Using Standard Action



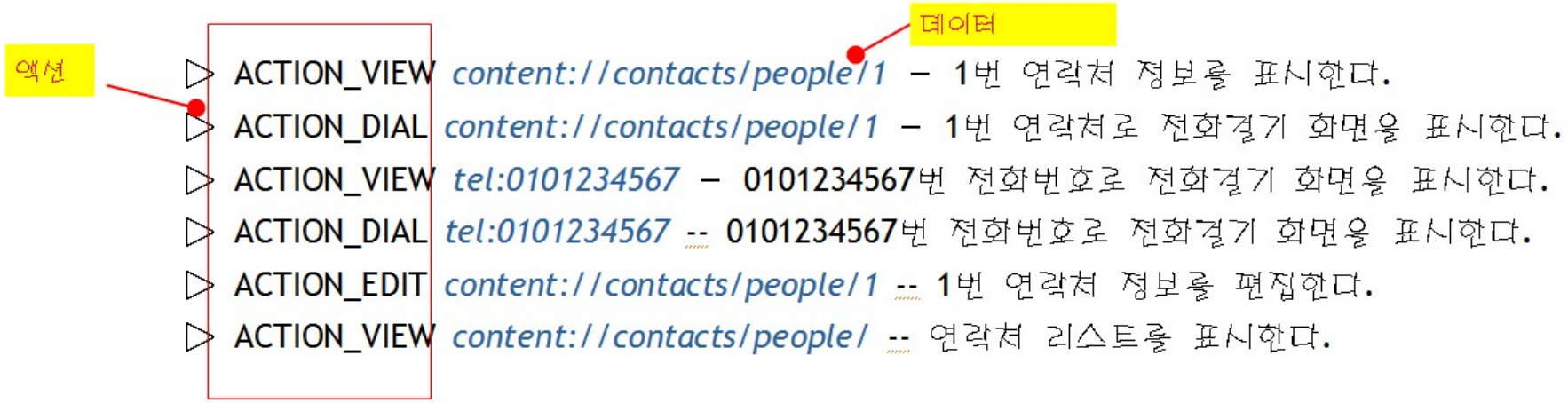

```
String myData = "http://www.youTube.com";

Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
                                Uri.parse(myData));

startActivity(myActivity2);
```

Caution. Add to the Manifest a request to use the Internet:
<uses-permission android:name="*android.permission.INTERNET*" />

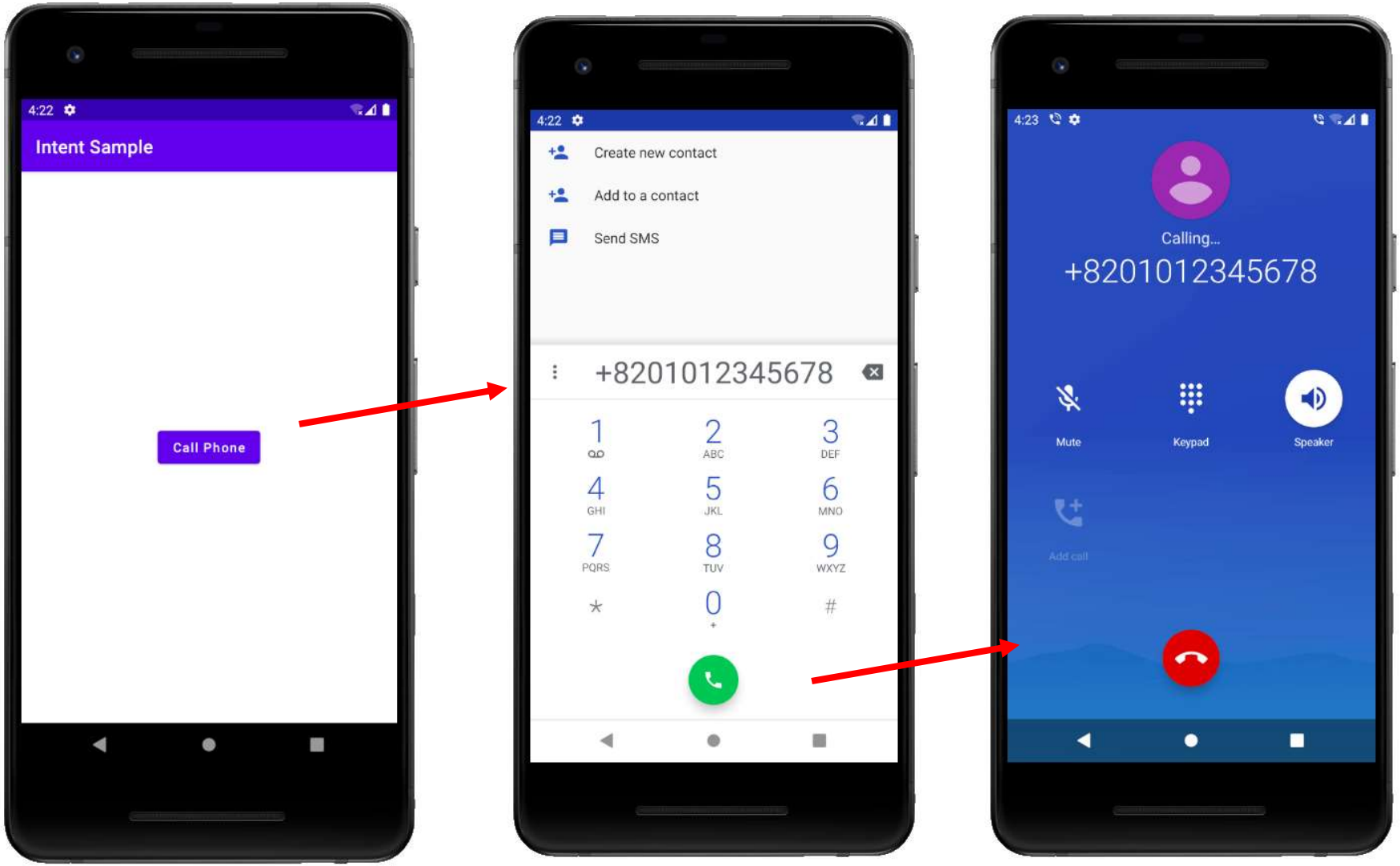
ACTION의 종류

상수	target 컴포넌트	Action
ACTIN_VIEW	activity	데이터를 사용자에게 표시한다.
ACTION_EDIT	activity	사용자가 편집할 수 있는 데이터를 표시한다.
ACTION_MAIN	activity	태스크의 초기 액티비티로 설정한다.
ACTION_CALL	activity	전화 통화를 시작한다.
ACTION_SYNC	activity	모바일 장치의 데이터를 서버 데이터와 일치시킨다.
ACTION_DIAL	activity	전화를 걸기 위해 전화번호를 누르는 화면을 나타낸다.

Implicit intent 예



실습 4: Implicit Intent 예 (1)



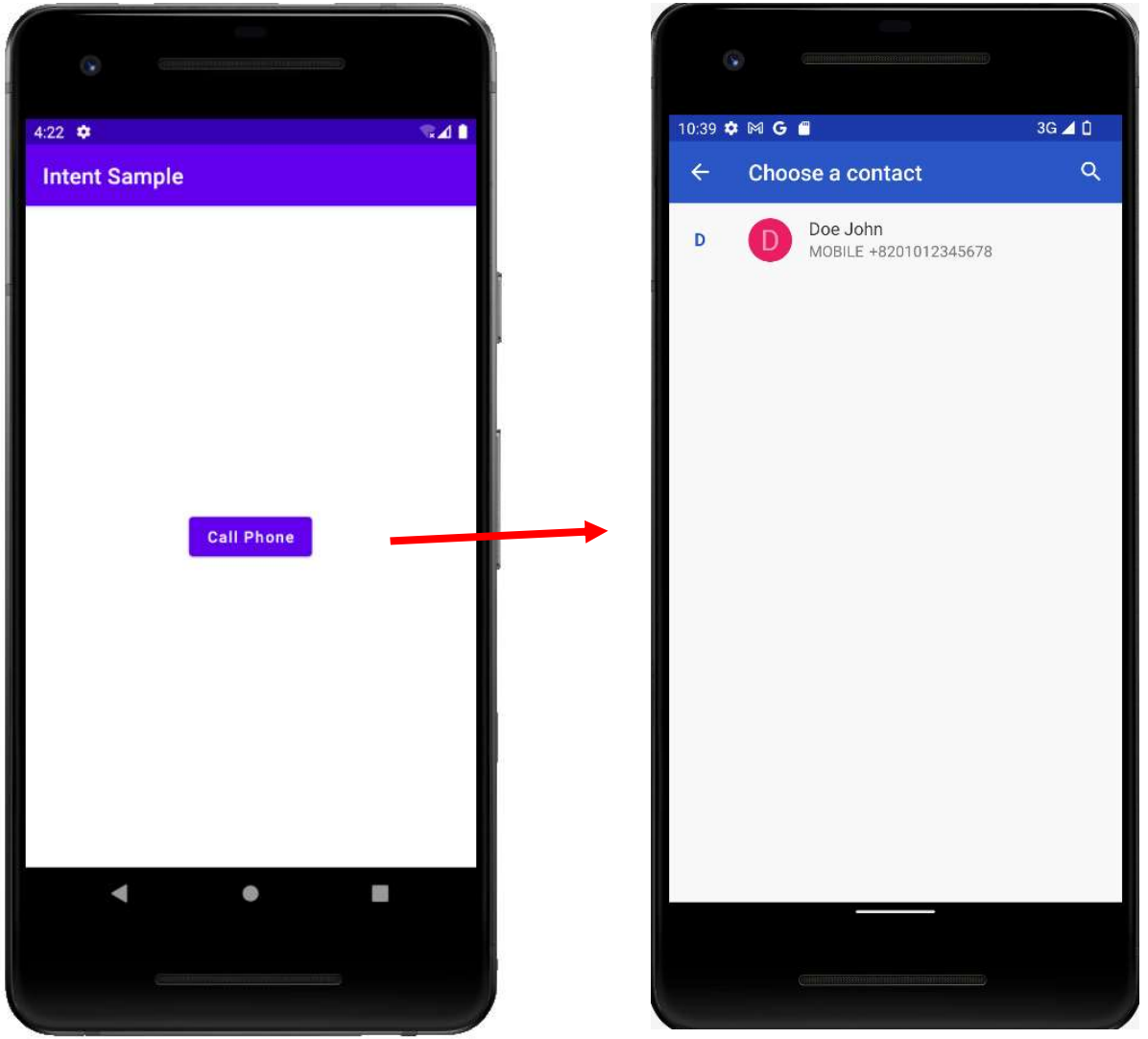
소스코드 – 12쪽(XML layout)

실습 4: MainActivity.kt

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        findViewById<Button>(R.id.callButton).setOnClickListener {  it: View!
            val intent = Intent(Intent.ACTION_DIAL,
                Uri.parse( uriString: "tel:(+82)01012345678"))
            startActivity(intent)
        }
    }
}
```

소스코드 – 12쪽

Implicit Intent 예 (2): 등록된 연락처 출력

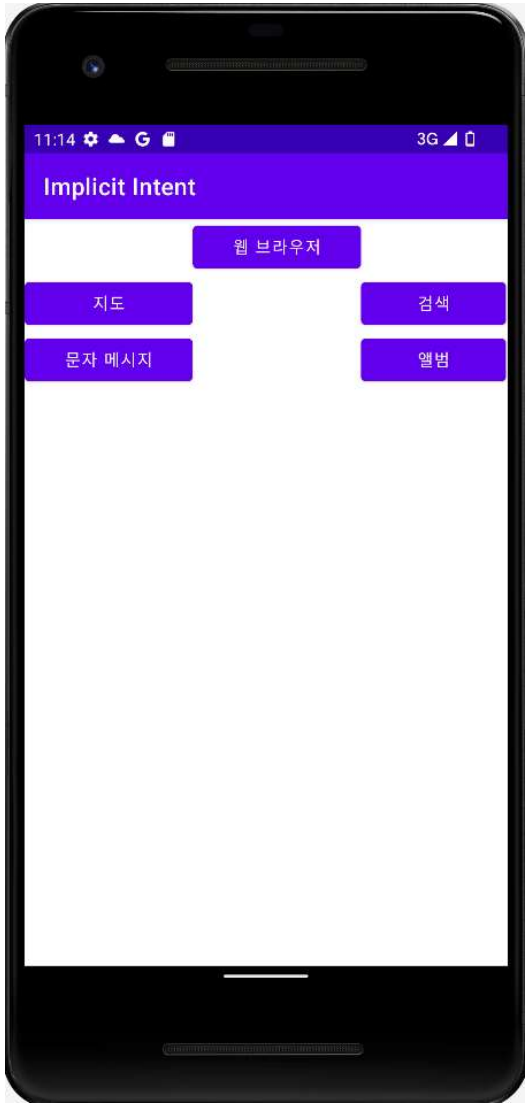


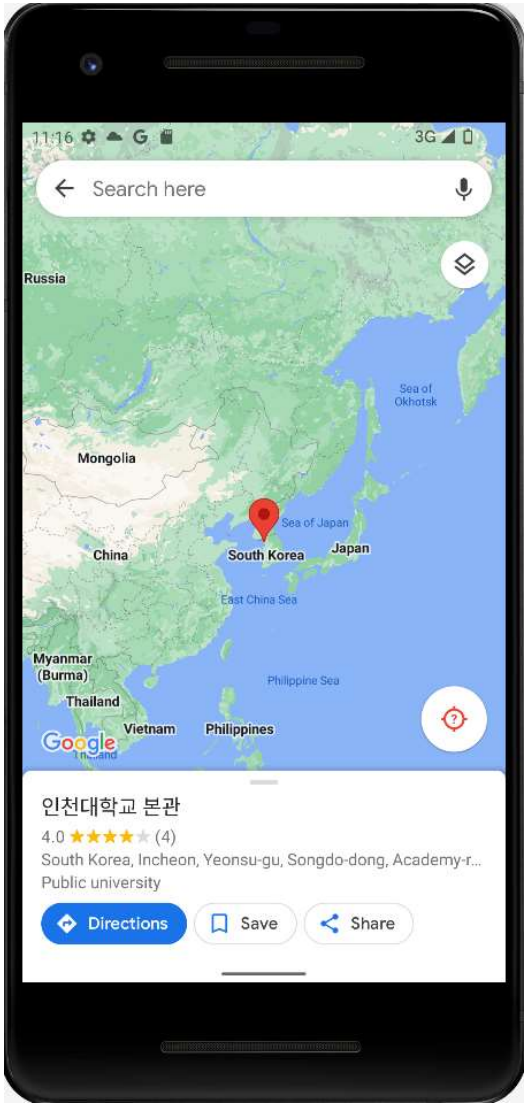
등록된 연락처 출력 : MainActivity.kt

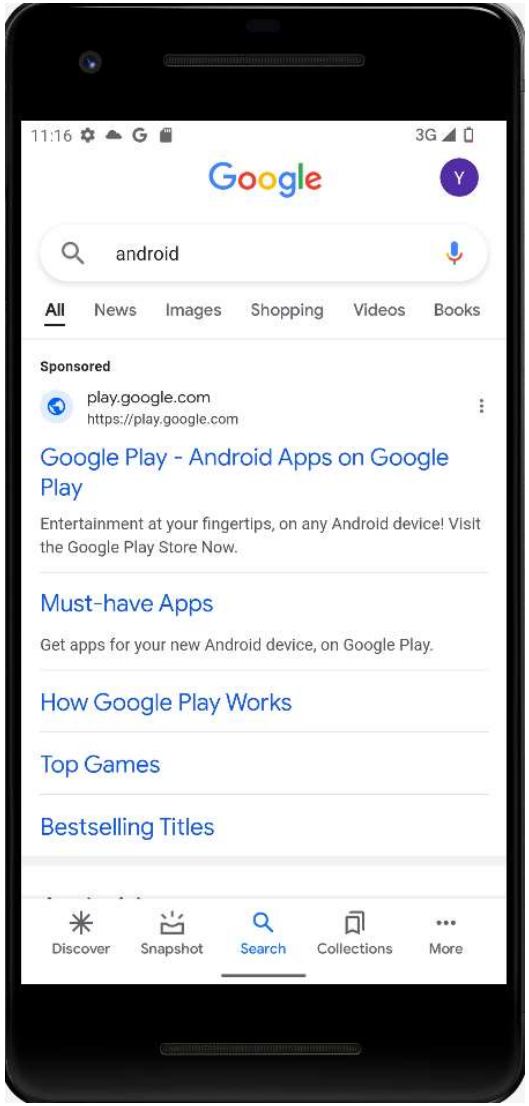
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        findViewById<Button>(R.id.callButton).setOnClickListener {  it: View!
            val intent = Intent(Intent.ACTION_PICK,
                ContactsContract.CommonDataKinds.Phone.CONTENT_URI)
            startActivity(intent)
        }
    }
}
```

소스코드 – 13쪽

Implicit Intent 예 (3)







레이아웃 – 14-15쪽

```
val browserButton = findViewById<Button>(R.id.browserButton)
browserButton.setOnClickListener {  it: View!
    val intent = Intent(Intent.ACTION_VIEW,
        Uri.parse( uriString: "http://m.google.com ") )
    startActivity(intent)
}
val mapButton = findViewById<Button>(R.id.mapButton)
mapButton.setOnClickListener {  it: View!
    val uri = Uri.parse( uriString: "http://maps.google.co.kr/maps?q=" +
            37.3766531 + "," + 126.6347248 + "&z=15")
    val intent = Intent(Intent.ACTION_VIEW, uri)
    startActivity(intent)
}
val googleButton = findViewById<Button>(R.id.googleButton)
googleButton.setOnClickListener {  it: View!
    val intent = Intent(Intent.ACTION_WEB_SEARCH)
    intent.putExtra(SearchManager.QUERY,  value: "android")
    startActivity(intent)
}
val smsButton = findViewById<Button>(R.id.smsButton)
smsButton.setOnClickListener {  it: View!
    val intent = Intent(Intent.ACTION_SENDTO)
    intent.putExtra( name: "sms_body",  value: "안녕?")
    intent.data = Uri.parse( uriString: "smsto:"+Uri.encode( s: "010-1234-5678"))
    startActivity(intent)
}
val photoButton = findViewById<Button>(R.id.photoButton)
photoButton.setOnClickListener {  it: View!
    val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
    startActivity(intent)
}
```

소스코드 – 15~16쪽

What to do next?

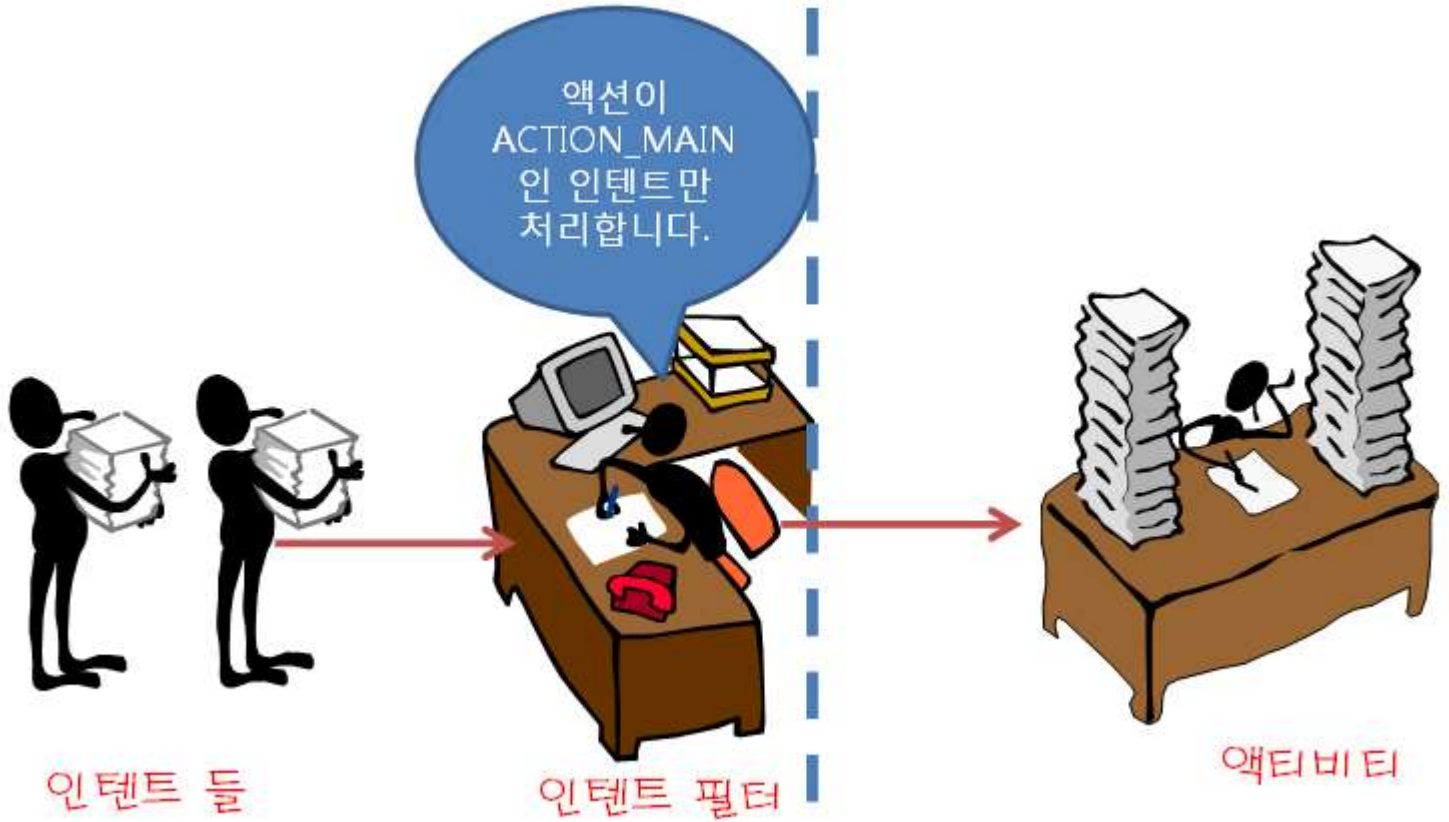
- Activity Stack과 intent
- Explicit intent
- Activity로부터 결과 돌려받기
- Implicit intent
- **Intent filter**

Intent filter (1/2)

- component는 자신이 처리할 수 있는 intent의 종류를 <intent-filter>에 기록한다.
  - Explicit intent는 <intent-filter>에 상관없이 항상 target component에 전달된다.
  - Implicit intent는 **<intent-filter>를 통과해야만** target component에 전달된다.
- **<intent-filter>는 암시적 인텐트에만 적용됨!**

Intent filter (2/2)

- component는 여러 개의 <intent-filter>를 정의할 수 있다.



<intent-filter> : category

```
<activity
  android:name=".AndDemo"
  android:label="@string/title_activity_and_demo" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

- Activity를 category로 분류할 수 있음
- **CATEGORY_HOME**
  - This is the **home activity**, that is the first activity that is displayed when the device boots.
- **CATEGORY_LAUNCHER**
  - Should be displayed in the **top-level launcher**.