

# **Android Intents**

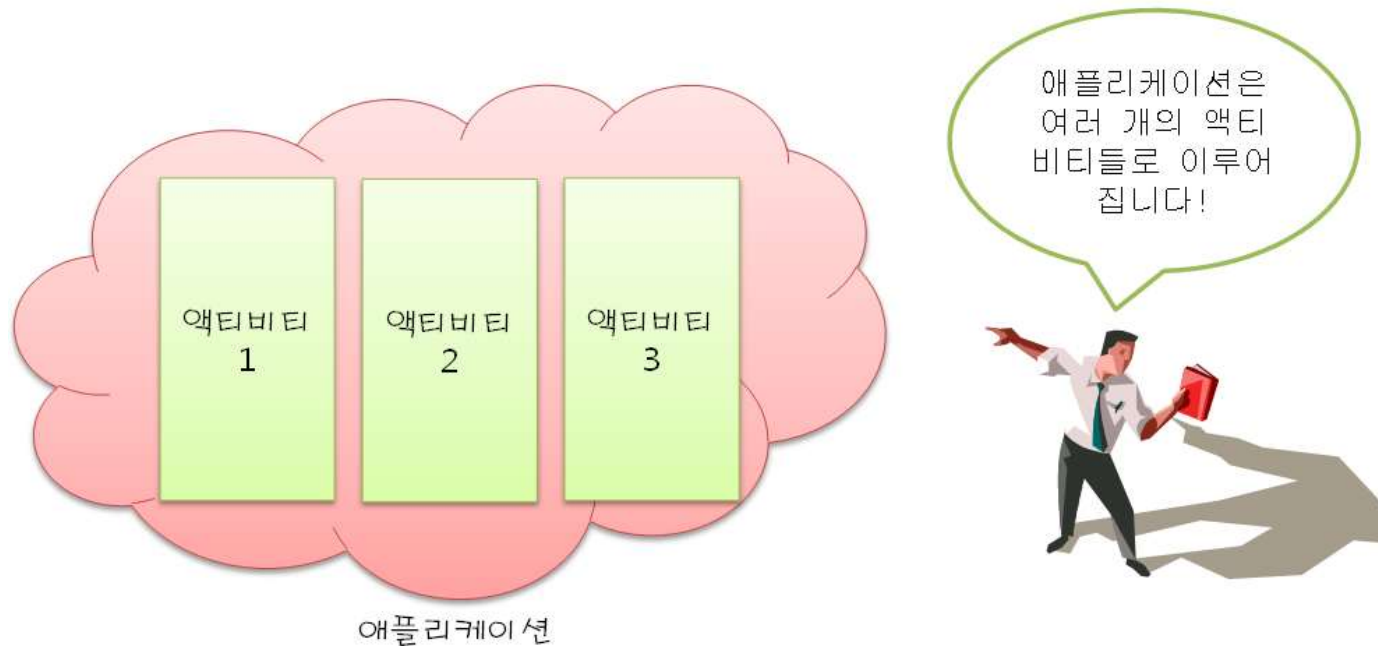
Mobile Software  
2019 Fall

# What to do next?

- **Intent 란?**
- Explicit intent
- Activity로부터 결과 돌려받기
- Implicit intent
- Intent filter

# 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.

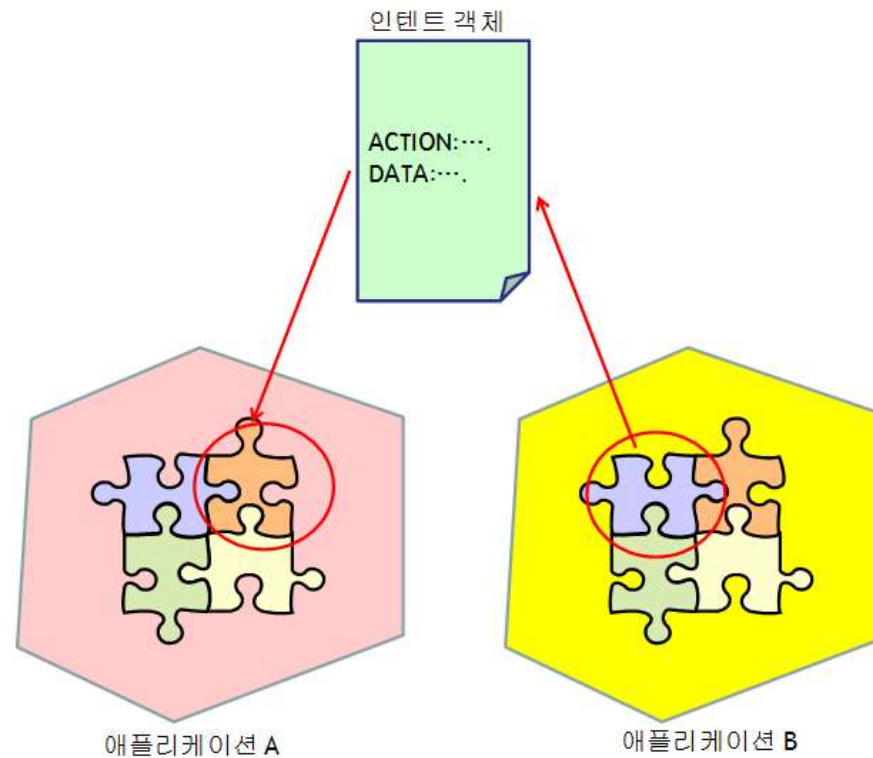


# Intent (2/4)

- 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

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Explicit Intent"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".ActivityB">
    </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>
```

AndroidManifest.xml

# What to do next?

- Intent 란?
- **Explicit intent**
- Activity로부터 결과 돌려받기
- Implicit intent
- Intent filter



# Intent 종류

- **Explicit intent (명시적 인텐트)**
  - 구체적으로 지정
    - “애플리케이션 A의 컴포넌트 B를 작동시켜라”
- **Implicit intent (암/묵시적 인텐트)**
  - 기본 조건만 지정
    - “지도를 보여줄 수 있는 컴포넌트이면 어떤 것이라도 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

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

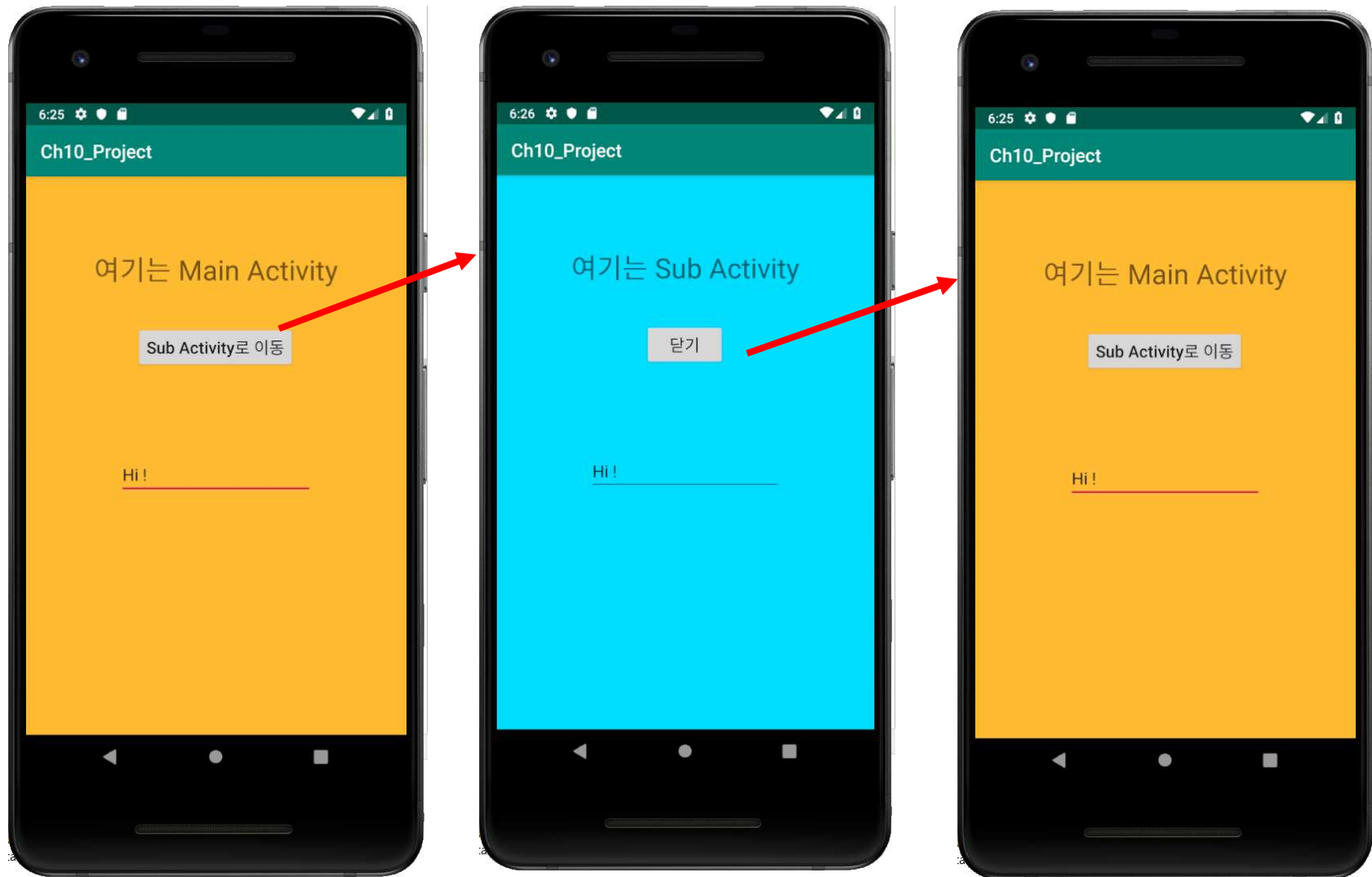
val mainString = editText.text.toString()
i.putExtra("mainStr", mainString)
startActivity(i)
```

메모리에 저장된 **NextActivity** 클래스를 가리킴.  
즉, **NextActivity** 클래스 정보를 추출할 때  
시작점이 됨. → 이 과정을 **reflection** 이라고 함

# 실습 준비

- 새 프로젝트 생성
  - Activity : **Empty Activity**
  - Application name : **Ch10\_Project**
  - Minimum API level : **API 26** (Oreo)
  - Activity name : **MainActivity.kt** (자동 생성)
  - Layout name : **activity\_main.xml** (자동 생성)
- 자동 생성된 **activity\_main.xml**은 기본 layout으로 **ConstraintLayout** 이 지정되어 있음.

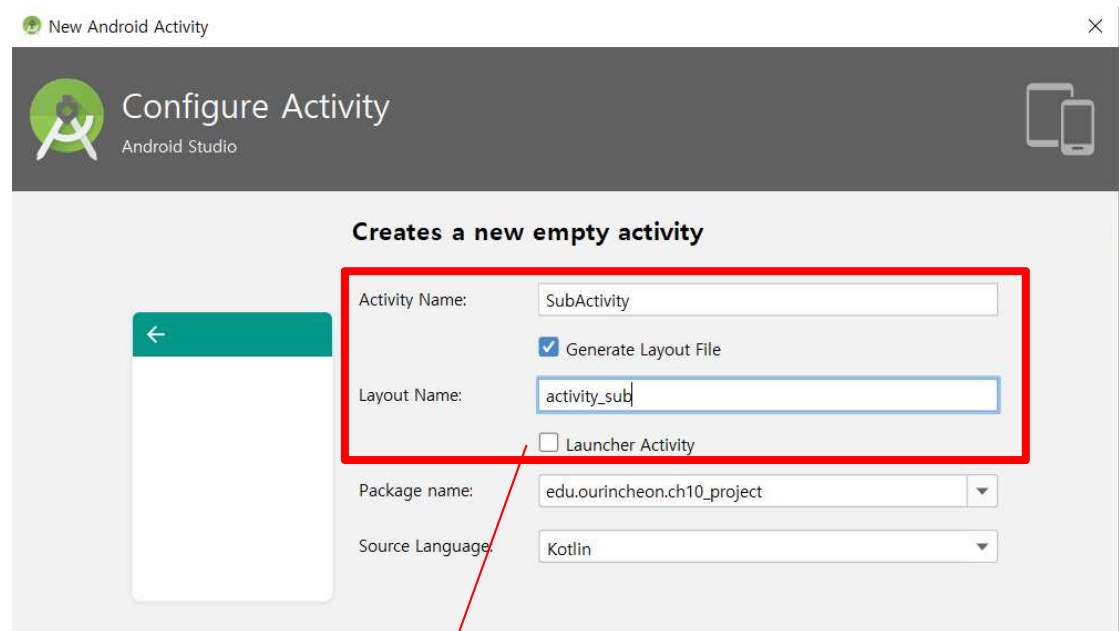
# 실습 1: 2개 activity – Explicit Intent



# 실습 1: SubActivity.kt 생성



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



**Generate Layout File**의 체크 박스는 **체크**  
**Launcher Activity**의 체크 박스는 **disable**.

# 실습 1: MainActivity 레이아웃

activity\_main.xml

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:text="여기는 Main Activity"
    android:textSize="30sp"

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:text="@string/sub_activity_str"
    android:textAllCaps="false"
    android:textSize="18sp"

<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:ems="10"
    android:inputType="text"
```

소스코드 - 1~2쪽



# 실습 1: SubActivity 레이아웃

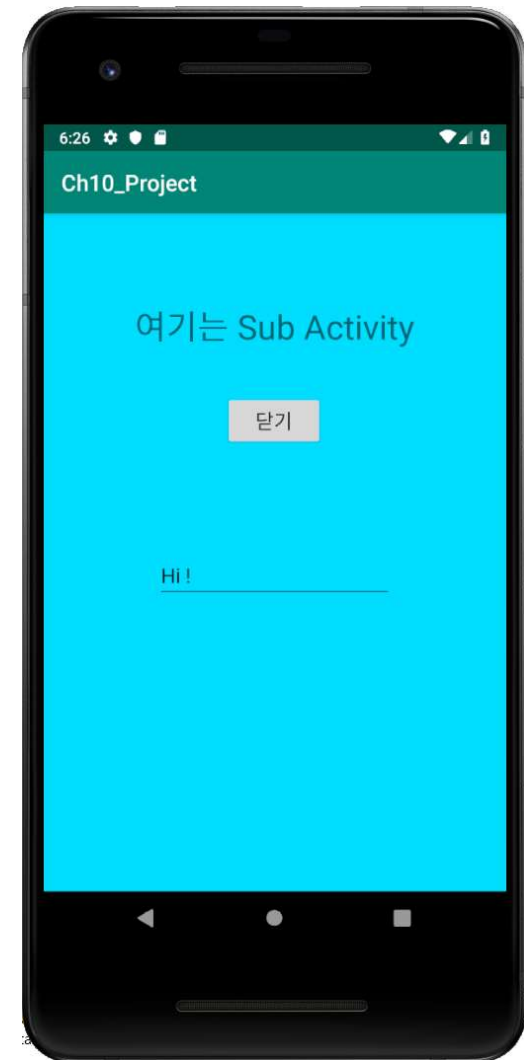
activity\_sub.xml

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="80dp"
    android:text="여기는 Sub Activity"
    android:textSize="30sp"

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:text="닫기"
    android:textAllCaps="false"
    android:textSize="18sp"

<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:ems="10"
    android:inputType="text"
```

소스코드 - 2~3쪽

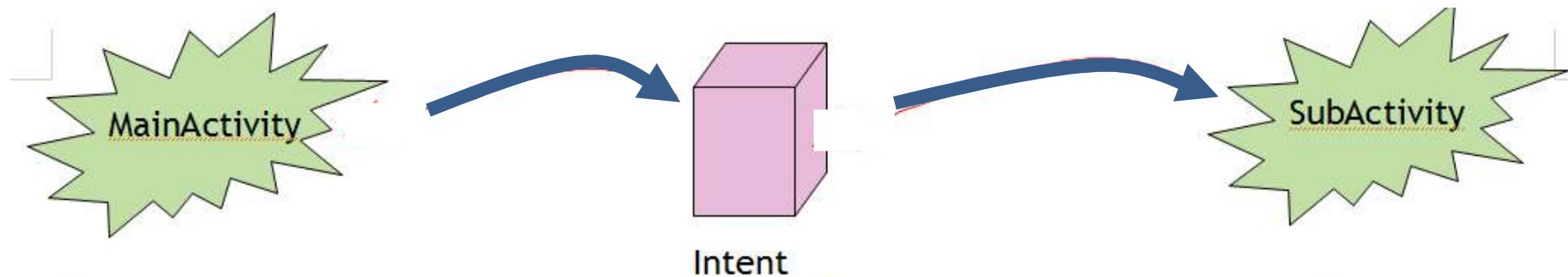


# 실습 1: MainActivity.kt

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        button.setOnClickListener {  
            var i = Intent(this, SubActivity::class.java)  
  
            val mainString = editText.text.toString()  
            i.putExtra("mainStr", mainString)  
            startActivity(i)  
        }  
    }  
}
```



# Extra : intent에 포함되어 전달되는 데이터



```
var i = Intent(this, SubActivity::class.java)
val mainString = editText.text.toString()
i.putExtra("mainStr", mainString)
i.putExtra("mainInt", 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 pair 형태로 extra에 추가

# 실습 1: SubActivity.kt

```
class SubActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_sub)  
  
        val extras = intent.extras ?: return  
  
        val qString = extras.getString("mainStr")  
        editText.setText(qString)  
  
        button.setOnClickListener{  
            finish()  
        }  
    }  
}
```

Elvis 연산 기호 사용

Activity stack에서 제거

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

# 실습 1: AndroidManifest.xml

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="edu.ourincheon.ch10_project">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Ch10_Project"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <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

# Starting Activities and Getting Results(1/3)

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

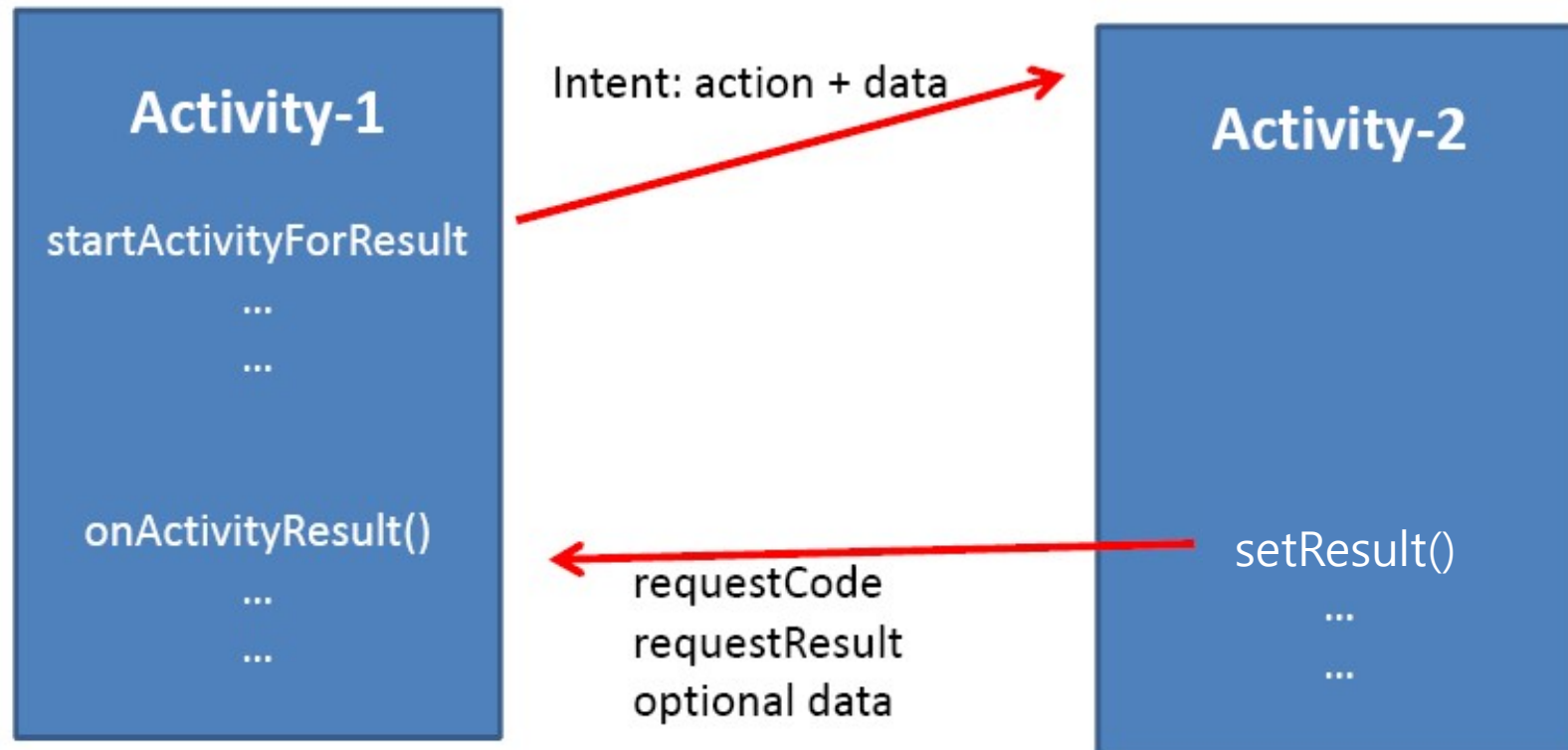
**startActivityForResult** (Intent, requestCode)

– with *a second integer parameter* **identifying the call**.

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

**onActivityResult** (requestCode, resultCode, Intent)

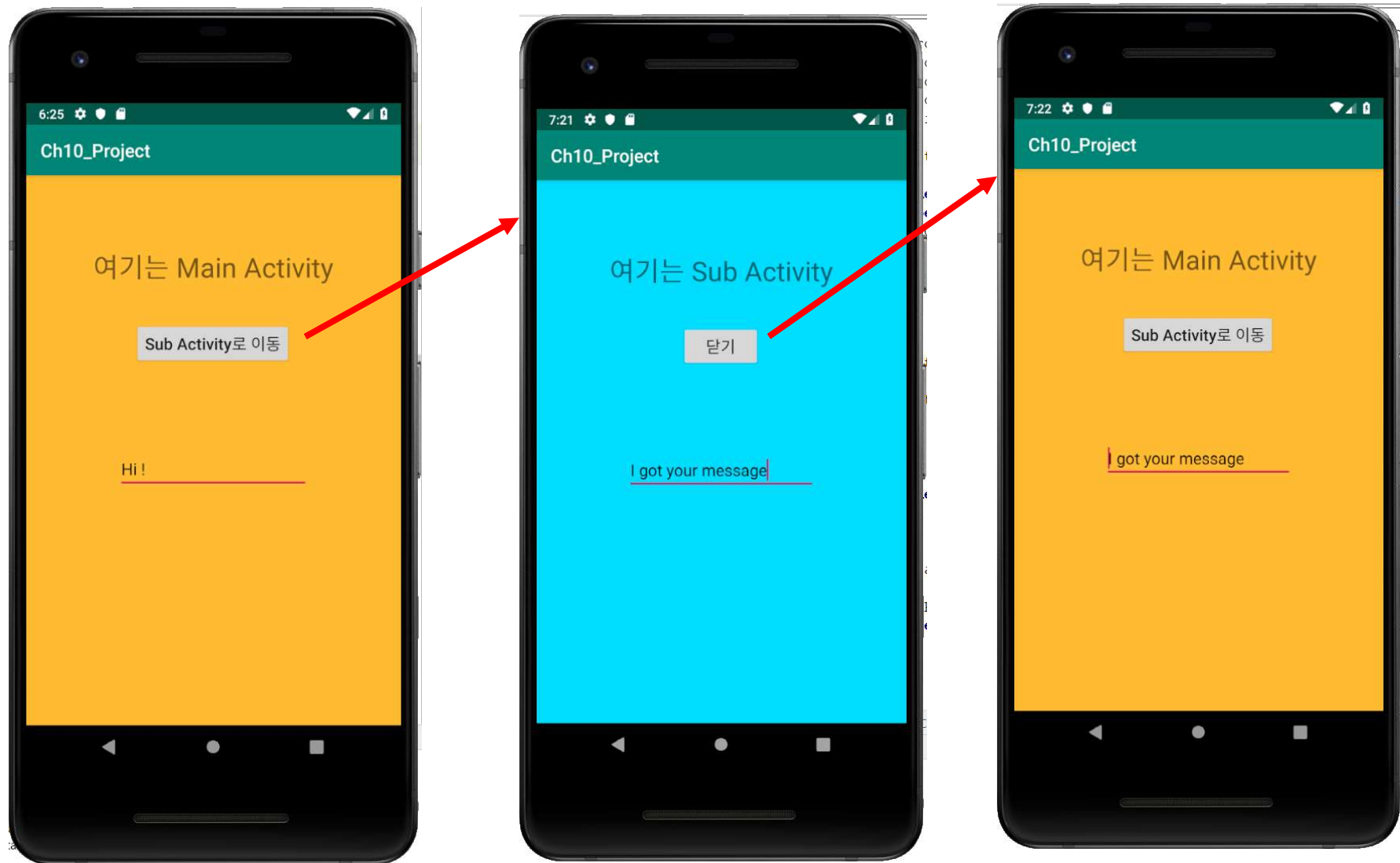
## Starting Activities and Getting Results(2/3)



## 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* **onActivityResult** (int requestCode, int resultCode, Intent data) along with the integer identifier it originally supplied.
- If *a child activity fails* for any reason (such as crashing), the parent activity will receive a result with the code **RESULT\_CANCELED**.

## 실습 2: Sub-Activity로부터 결과 돌려받기





# 실습 2: MainActivity.kt

```
class MainActivity : AppCompatActivity() {  
    private val request_code = 111  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        button.setOnClickListener {  
            var i = Intent(this, SubActivity::class.java)  
  
            val mainString = editText.text.toString()  
  
            i.putExtra("mainStr", mainString)  
            startActivityForResult(i, request_code)  
        }  
    }  
  
    override fun onActivityResult(requestCode: Int,  
                                   resultCode: Int, data: Intent?) {  
        if ((requestCode == request_code) &&  
            (resultCode == Activity.RESULT_OK)) {  
            if (data != null) {  
                if (data.hasExtra("returnData")) {  
                    val returnString = data.extras.getString("returnData")  
                    editText.setText(returnString)  
                }  
            }  
        }  
    }  
}
```

소스코드 - 5쪽

# Kotlin - let 함수

```
override fun onActivityResult(requestCode: Int,
                                resultCode: Int, data: Intent?) {
    if ((requestCode == request_code) &&
        (resultCode == Activity.RESULT_OK)) {
        if (data != null) {
            if (data.hasExtra("returnData")) {
                val returnString =
                    data.extras.getString("returnData")
                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.



```
data?.let {
    if (it.hasExtra("returnData")) {
        val returnString =
            it.extras.getString("returnData")
        editText.setText(returnString)
    }
}
```

## 실습 2: SubActivity.kt

```
class SubActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) { ... }  
  
    override fun finish() {  
        val data = Intent()  
  
        val returnString = editText.text.toString()  
        data.putExtra("returnData", returnString)  
  
        setResult(Activity.RESULT_OK, data)  
        super.finish()  
    }  
}
```

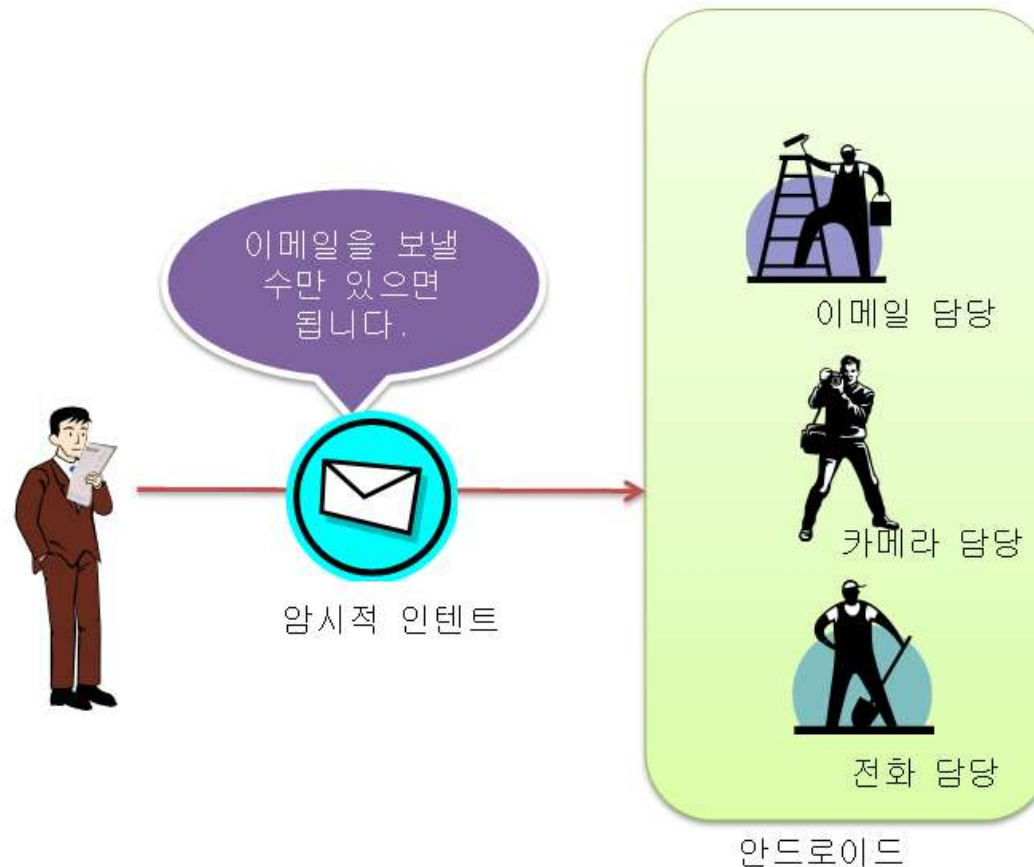


# What to do next?

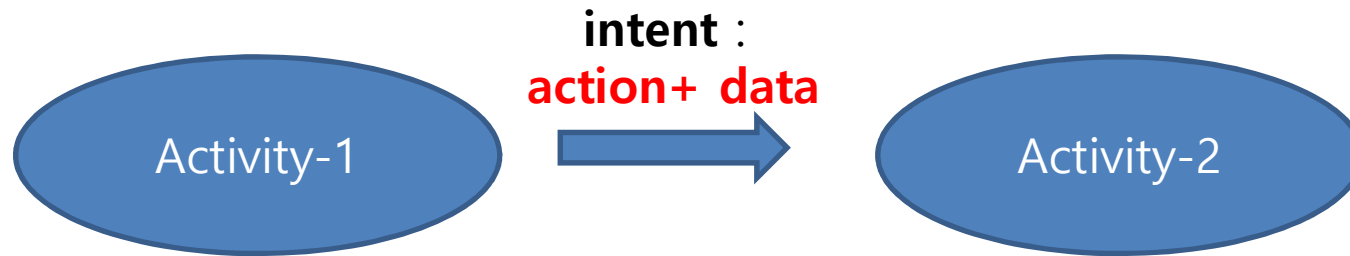
- Activity Stack과 intent
- Explicit intent
- Activity로부터 결과 돌려받기
- **Implicit intent**
- Intent filter

# 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

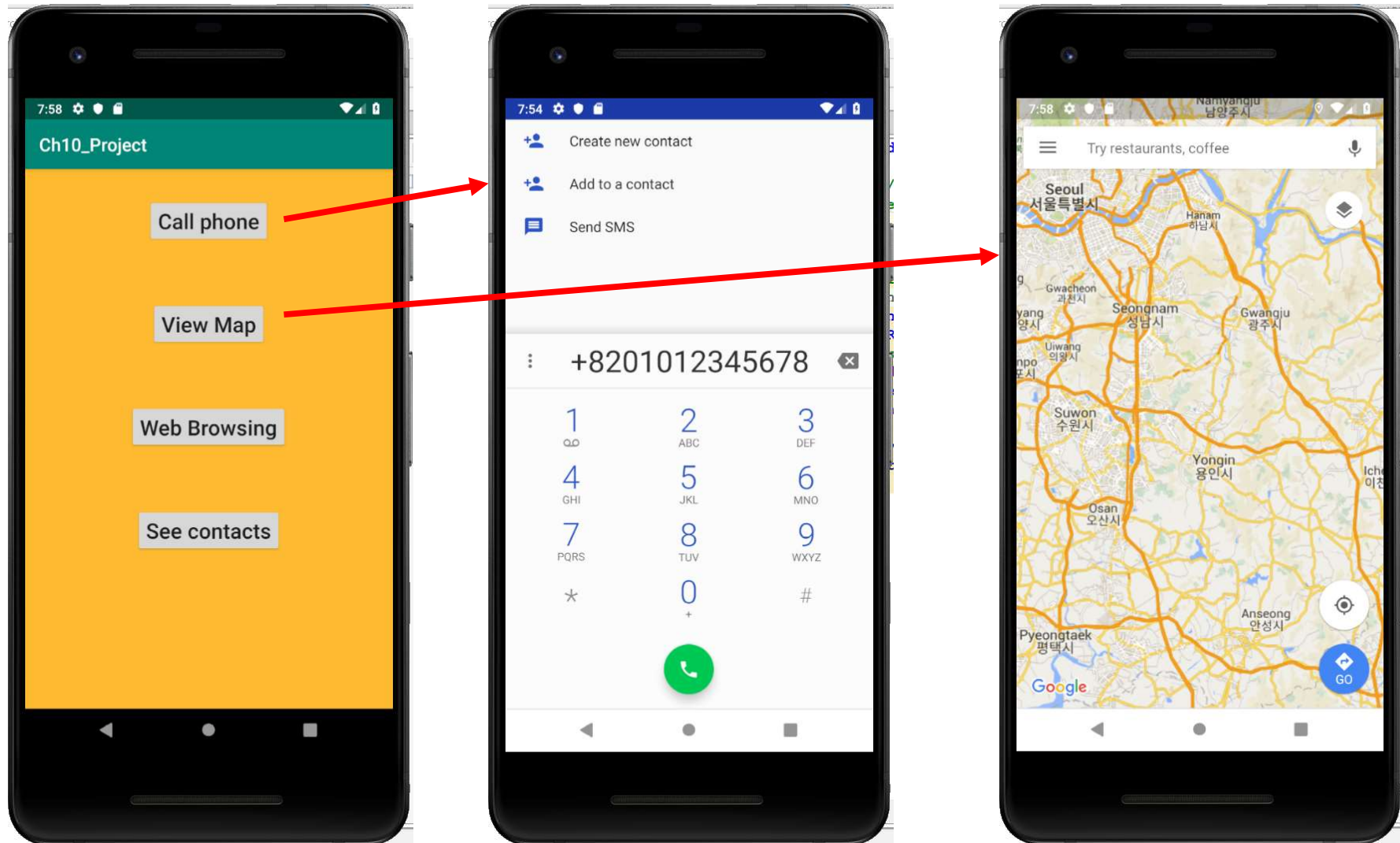
상수	target 컴포넌트	Action
<b>ACTIN_VIEW</b>	activity	데이터를 사용자에게 표시한다.
<b>ACTION_EDIT</b>	activity	사용자가 편집할 수 있는 데이터를 표시한다.
<b>ACTION_MAIN</b>	activity	태스크의 초기 액티비티로 설정한다.
<b>ACTION_CALL</b>	activity	전화 통화를 시작한다.
<b>ACTION_SYNC</b>	activity	모바일 장치의 데이터를 서버 데이터와 일치시킨다.
<b>ACTION_DIAL</b>	activity	전화를 걸기 위해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

# Implicit intent 예

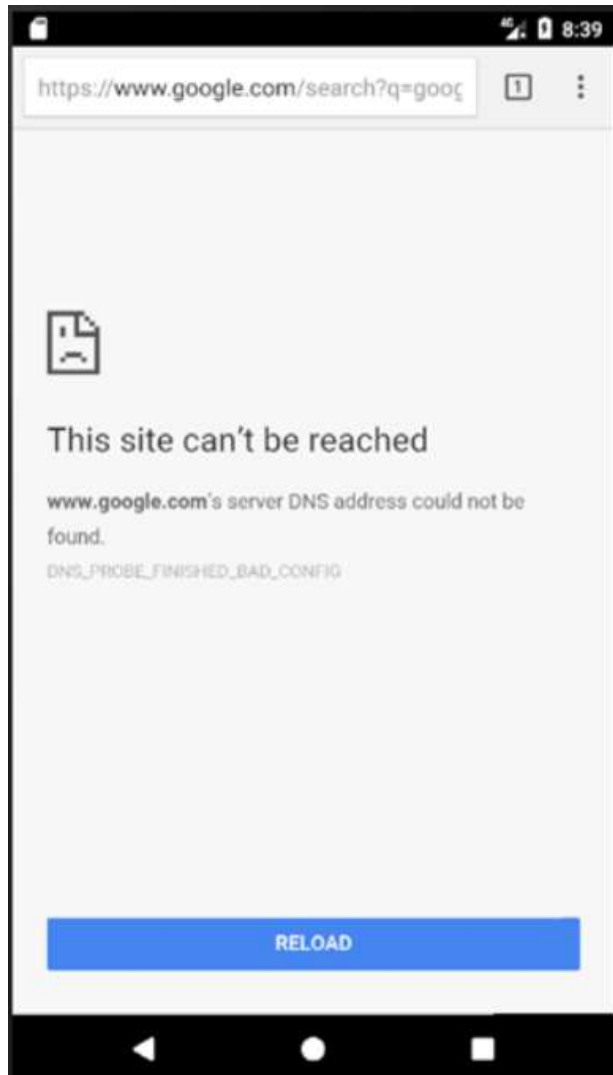
- 
- 액션
- 데이터
- ▷ ACTION\_VIEW `content://contacts/people/1` - 1번 연락처 정보를 표시한다.
  - ▷ ACTION\_DIAL `content://contacts/people/1` - 1번 연락처로 전화걸기 화면을 표시한다.
  - ▷ ACTION\_VIEW `tel:0101234567` - 0101234567번 전화번호로 전화걸기 화면을 표시한다.
  - ▷ ACTION\_DIAL `tel:0101234567` -- 0101234567번 전화번호로 전화걸기 화면을 표시한다.
  - ▷ ACTION\_EDIT `content://contacts/people/1` -- 1번 연락처 정보를 편집한다.
  - ▷ ACTION\_VIEW `content://contacts/people/` -- 연락처 리스트를 표시한다.



# 실습 3: Implicit Intent 예



# 잠깐! 왼쪽과 같은 에러 발생하면



Android Studio SDK가 설치된 경로를 찾음  
**jdoo** 는 사용자 이름

```
C:\Users\jdoo\AppData\Local\Android\sdk
```



해당 경로로 이동 → emulator.exe  
C:\W...Wemulator **-list-avds** → AVD를 찾음

```
New_Device_API_28  
Pixel_2_API_28
```

아래 명령 실행

```
emulator.exe -avd Nexus_5X_API_25 -dns-server 8.8.8.8
```

```
emulator -avd Pixel_2_API_28 -dns-server 8.8.8.8
```

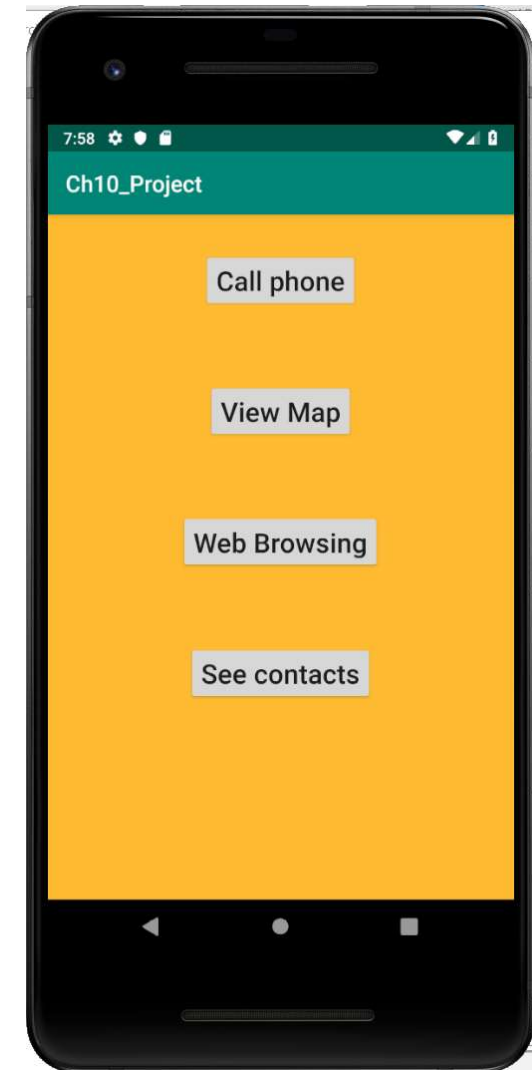
8.8.8.8 → Google public domain name server

# 실습 3: MainActivity 레이아웃

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="Call phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/call"
        android:textAllCaps="false"
        android:onClick="onClick"/>
    <Button
        android:text="View Map"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/map"
        android:textAllCaps="false"
        android:onClick="onClick"/>
    <Button
        android:text="Web Browsing"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/web"
        android:textAllCaps="false"
        android:onClick="onClick"/>
    <Button
        android:text="See Contacts"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/contact"
        android:textAllCaps="false"
        android:onClick="onClick"/>
</LinearLayout>
```

activity\_main.xml

소스코드 - 7~8쪽



# 실습 3: MainActivity.kt

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
  
    fun onClick(view: View) {  
        when (view.id) {  
            R.id.web ->  
                intent = Intent(Intent.ACTION_VIEW,  
                                Uri.parse("http://www.google.com"))  
            R.id.call ->  
                intent = Intent(Intent.ACTION_DIAL,  
                                Uri.parse("tel: (+82) 01012345678"))  
            R.id.map ->  
                intent = Intent(Intent.ACTION_VIEW,  
                                Uri.parse("geo:37.30,127.2?z=10"))  
            R.id.contact ->  
                intent = Intent(Intent.ACTION_VIEW,  
                                Uri.parse("content://contacts/people/"))  
            else ->  
                return  
        }  
        startActivity(intent)  
    }  
}
```

# What to do next?

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

# Intent filter (1/2)

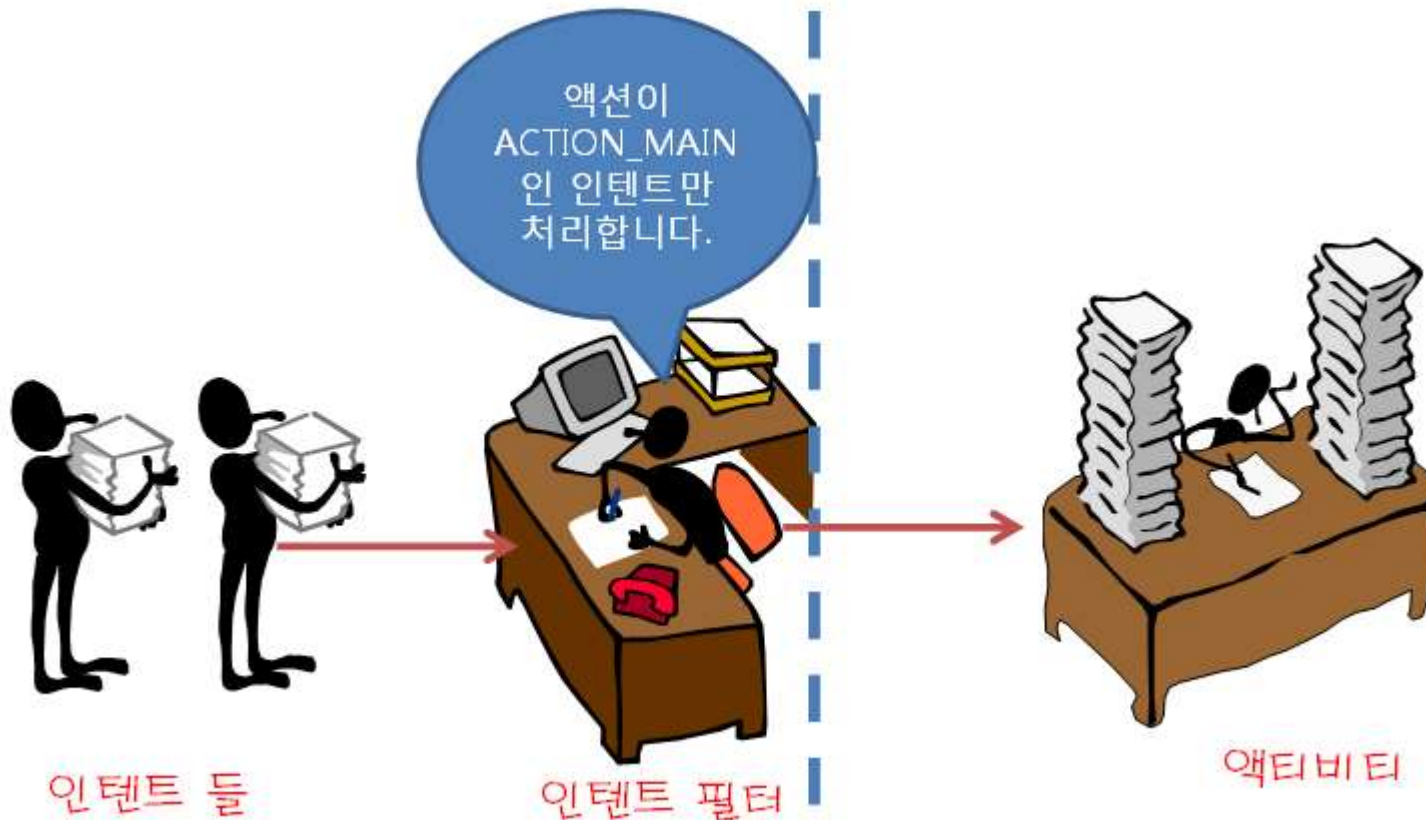
- component는 자신이 처리할 수 있는 intent의 종류를 <intent-filter>에 기록한다.
  - Explicit intent는 <intent-filter>에 상관없이 항상 target component에 전달된다.
  - Implicit intent는 **<intent-filter>를 통과해야만** target component에 전달된다.
- **<intent-filter>는 암시 인텐트에만 적용됨!**





# 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**.