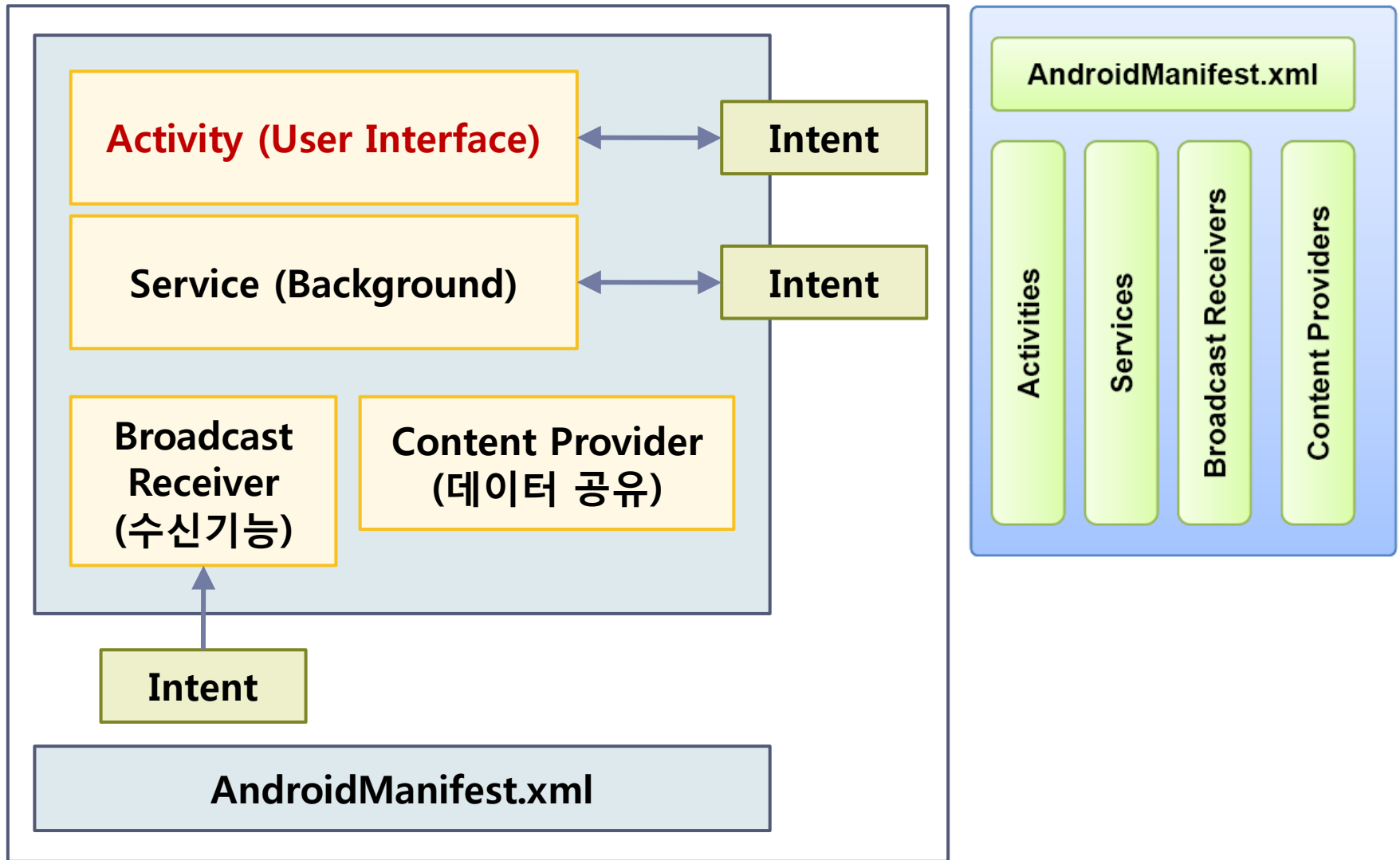


Android의 Application의 구성 요소

Android Application 구성 요소



Application Components (구성 요소)

- 안드로이드 어플리케이션의 기본적인 빌딩 블록(essential building block)
- 각각의 컴포넌트들은 시스템이 해당 어플리케이션에 진입할 수 있는 서로 다른 진입 점(Entry Point)
- 모든 컴포넌트들이 유저에 대해서 실제의 진입 포인트가 되는 것은 아님 (외부에서 Intent를 이용하여 호출 할 수 있어야 함)
- 어떠한 컴포넌트들은 서로에게 의지하기도 하지만, 각각의 컴포넌트들은 그 자체로써 존재하며 구체적인 역할을 감당함
- 어떤 어플리케이션도 다른 어플리케이션의 컴포넌트를 시작할 수 있음
- 컴포넌트를 시작하면, (이미 프로세스가 만들어져 있지 않은 경우) 해당 컴포넌트를 가지는 어플리케이션에 대한 프로세스를 시작하고, 해당 컴포넌트에 대한 클래스를 객체화(instantiate) 함



Activating Components (활성화 컴포넌트)

- 네 가지 컴포넌트 중 액티비티, 서비스, 브로드캐스트 리시버는 인텐트(intent) 라고 불리는 비동기 메시지에 의해서 활성화 됨
- 콘텐츠 프로바이더는 인텐트에 의해 활성화 되지 않고, ContentResolver 로 부터 발생한 요구(request) 에 의해 타겟이 됐을 때, 활성화 됨
- 인텐트(Intent)
 - 실행 시간 때 (runtime) 에 독립적 컴포넌트들을 서로 연결시킴
 - Intent 오브젝트로 만들어지며, 특정한 컴포넌트나 혹은 특정한 타입의 컴포넌트를 활성화 시키는 메시지를 정의함
 - 명시적(explicit)일 수도 있고 암시적(implicit)일 수 있음
 - 액티비티나 서비스에 대해서는 인텐트가 수행해야 하는 액션을 정의하고, 수행해야 하는 액션의 대상이 되는 데이터의 URI 를 명시할 수도 있음



Application, Activity, Process, Thread의 관계

- **Application** – Manifest.xml에 <application>으로 표기
- **Activity** – Manifest.xml에 <activity>로 표기
- **Process**
 - 기본적으로 Package명을 Process 이름으로 사용하며 하나의 Application에 있는 구성 요소들은 같은 Process
 - 구성 요소 중 Activity와 Service에는 android:process 속성을 부여할 수 있음
 - android:process에 Package명을 명시하면 그 Package명으로 새로운 process로 취급됨



1개 Application, 2개 Activity, 2개 Process

```
<manifest    package="com.android.second" ... >
<application android:icon="@drawable/icon"
    android:label="@string/app">
    <activity android:name=".MainActivity" android:label="@string/app"
        android:process="com.android.second">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".SubActivity" android:label="@string/app"
        android:process="com.android.third">
        <intent-filter>
            <action android:name="com.android.app.SOYOUNG" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>
```





Activity의 생명 주기

Activity

1. Android Application의 UI를 담당하는 구성요소
2. 기본 기능
 - UI 표시 (화면 표시) : view component 사용
 - 사용자와 상호작용 : 사용자 Event에 따른 처리
3. Activity 상태(State)
 - 1) 활성화 상태(Resumed) – 현재 화면에 표시된 상태로 사용자 Event를 수신할 수 있는 상태 (1개의 activity만 활성화 상태가 됨, 상태표시줄을 제외한 모든 영역 사용)
 - 2) 일시 정지 상태(Paused) – 화면에는 표시가 되지만 사용자 Event 수신 불가능 (앞에 있는 Activity 의 배경이 투명해서 뒤쪽의 Activity가 표시될 때, 뒤쪽의 Activity가 보이지만 Event 수신은 할 수 없음)
 - 3) 정지 상태(Stopped) – 화면 표시 불가 상태
< 2), 3) 상태에서는 강제 종료가 될 수 있음>

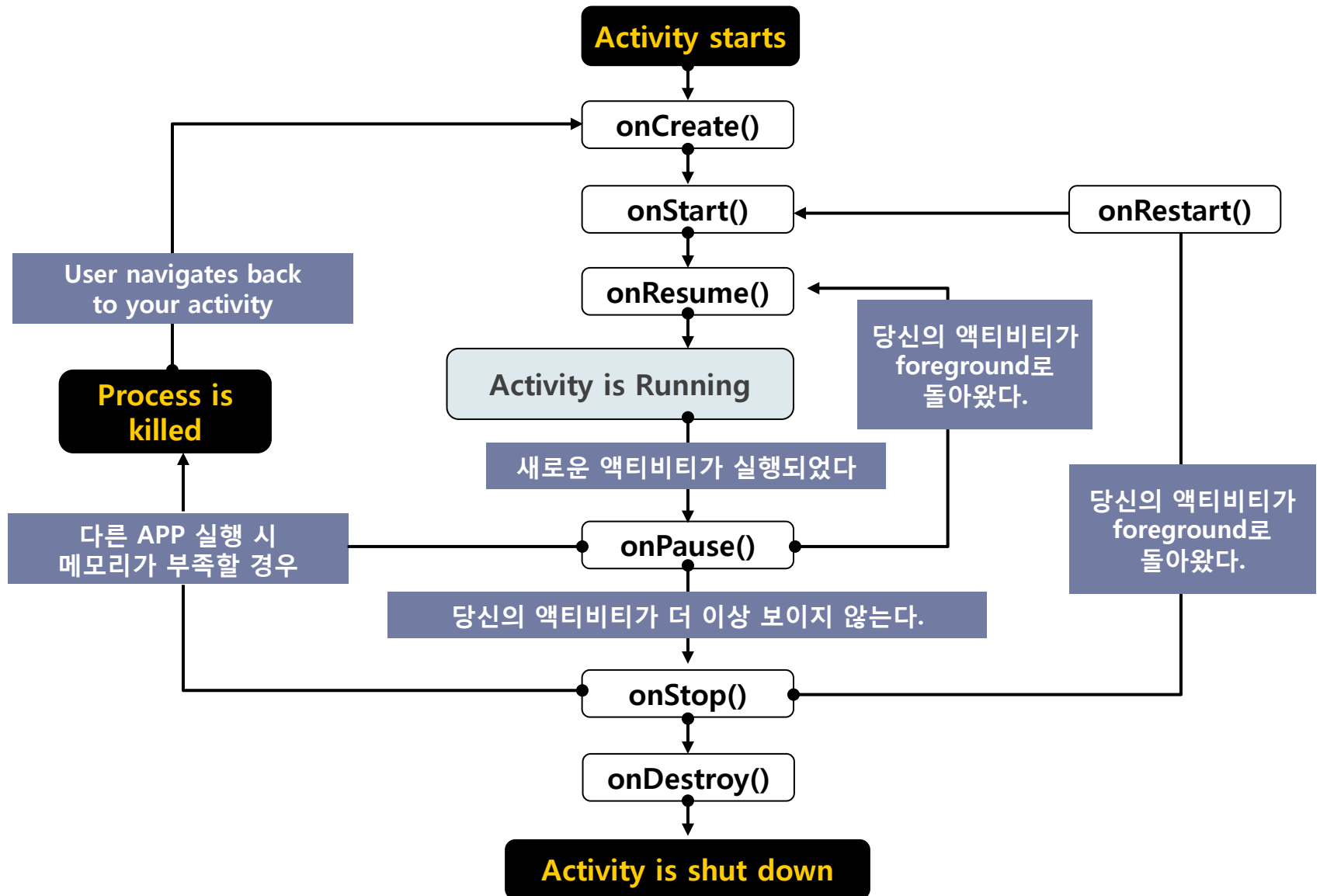


Activity

1. 레이아웃을 정의하는 xml 문서 하나와 동작을 정의하는 java 파일을 쌍으로 생성해야, 하나의 Activity가 정의됨 (코드만으로 레이아웃 구성하는 것은 권장 하지 않는 방법임)
2. Activity 클래스로부터 상속을 받으며 onCreate를 반드시 재정의 해야 함
3. 모든 Activity는 AndroidManifest.xml에 등록되어야 함
4. setContentView 메서드를 호출하여 Activity 안에 View나 View Group을 채움
5. Activity 추가 절차
 - ① XML 파일 작성 (/layout/main.xml, 이름은 지정하여 사용)
 - ② java 파일 작성 (/src/Package/xx.java)
 - ③ AndroidManifest.xml에 등록
 - ④ startActivity 메서드로 Activity 호출



Activity의 라이프 사이클



Process의 종류

1. foreground process : 활성화 상태의 activity를 가지고 있는 process
2. visible process : 일시 정지 상태의 activity를 가지고 있는 process
3. background : 정지 상태에 놓여 있는 activity를 가지고 있는 process
4. service process : Android Application 구성 요소 중 service 요소가 동작하고 있는 process (UI 없음), 강제 종료가 되었다가도 Kernel 이 다시 복구해 주어야 함
5. empty process : process 생성에 시간이 많이 걸림 그래서 Android Run Time이 없는 빈 process를 만들어 놓았다가 application이 필요로 하면 바로 배정하여 쓰도록하여 시간을 절약함 그러나 메모리 부족 등의 상황이 발생하면 제거 1순위가 됨

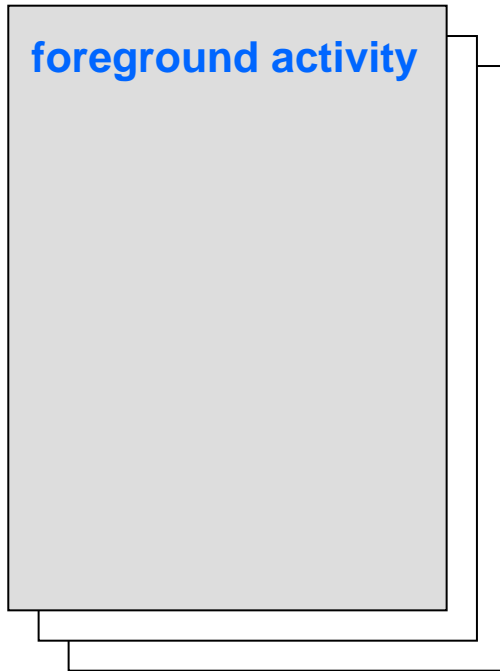


foreground process

→ foreground process

사용자가 현재 조작중인 최상위 화면의 Activity

가장 중요한 프로세스이므로 새로운 App가 실행될 때 메모리가 부족하다면 제일 마지막에 제거되는 프로세스



- onResume()가 호출된 프로세스
- 화면의 전면 부에 나타나며(in the foreground of the screen) 유저의 포커스(focus) 를 가지고 있음



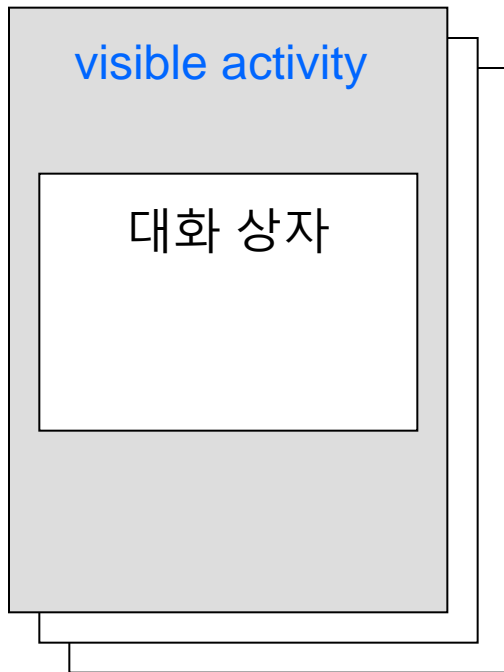
visible process

→ visible process

사용자 화면에 보여지지만 foreground가 아닌 것

예) foreground 대화상자의 뒤에 있는 activity

visible process는 foreground process 다음으로 중요함



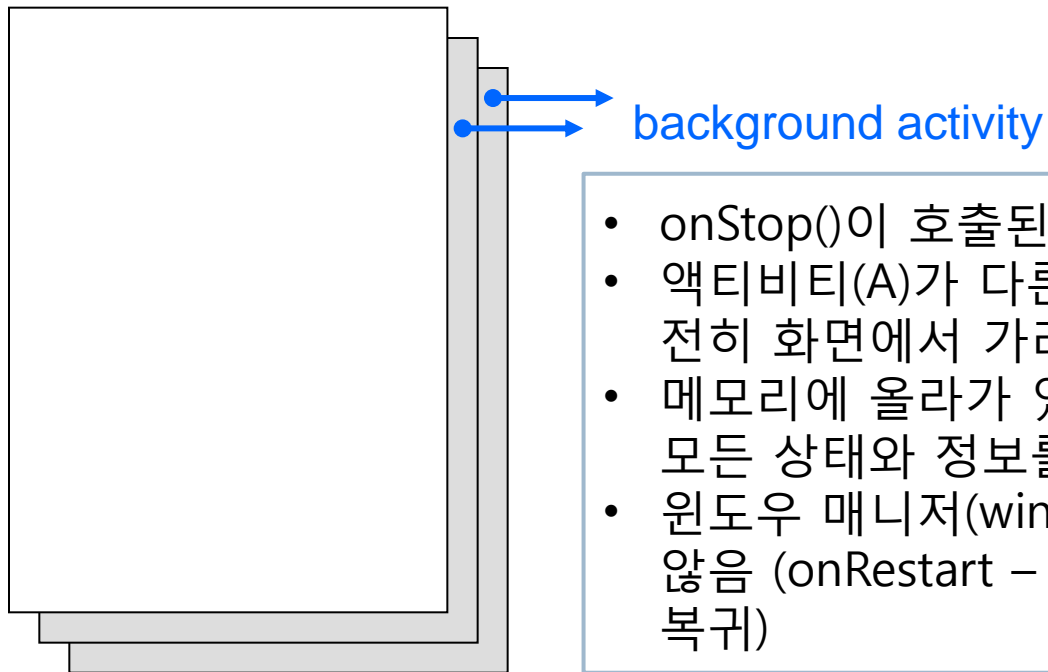
- onPause()가 호출된 프로세스
- 다른 activity가 화면의 전면부에 있어서 유저의 포커스를 가지고 있지만, 여전히 보여지는 상태(still visible)의 activity
- 메모리에 올라가 있는 상태, 이 액티비티는 모든 상태와 정보를 계속 유지하고 있음
- 윈도우 매니저(window manager)와 연결된 상태로 유지
- onResume()로 복귀

background process

→ background process

사용자에게 보여지지 않는 activity.

더 이상 중요하지 않으며 시스템은 foreground나 visible 프로세스가 메모리를 요구하면 background activity를 안전하게 제거할 수 있음

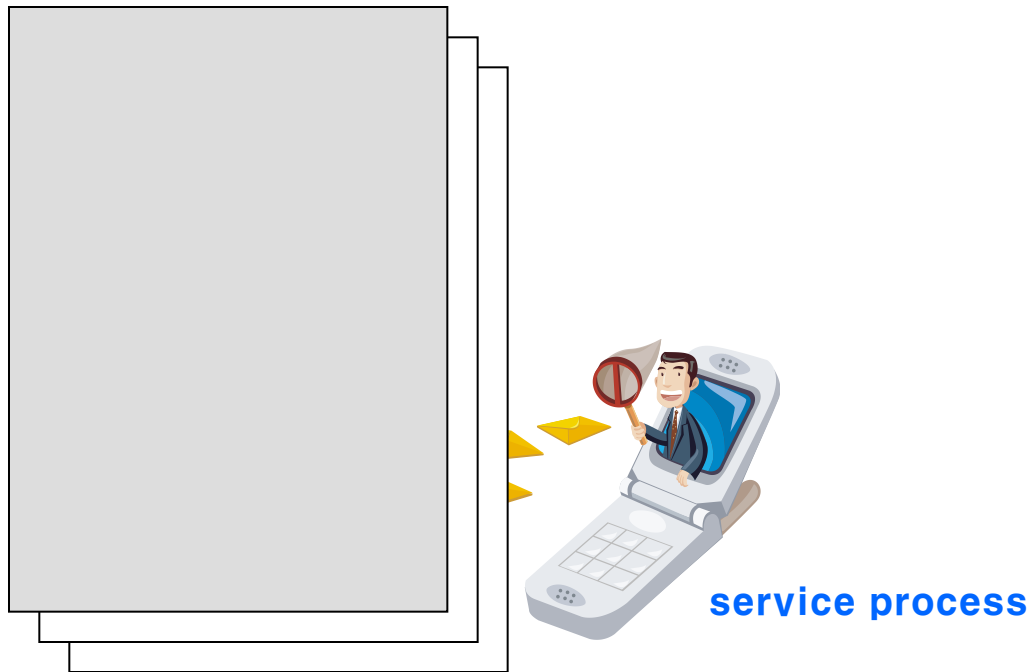


service process

→ service process

특정한 서비스를 위하여 존재하며 UI가 없음.

예) 백그라운드에서 SMS가 수신되면 사용자는 모르게 수신된 SMS를 데이터베이스에 저장한다거나 스팸 확인을 하는 등의 처리를 위해 존재하는 경우
이 프로세스는 foreground, visible process 를 보유하기에 충분한 메모리가 존재하는 한 제거되지 않고 유지 됨.

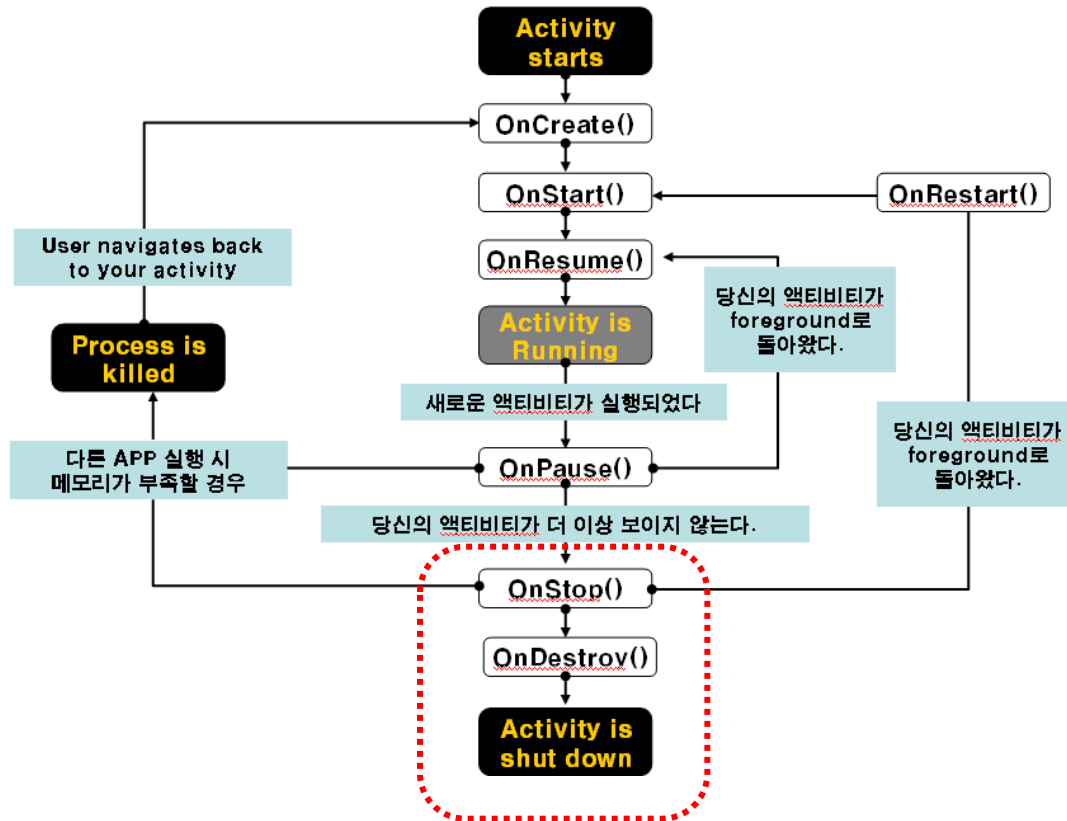


empty process

→ empty process

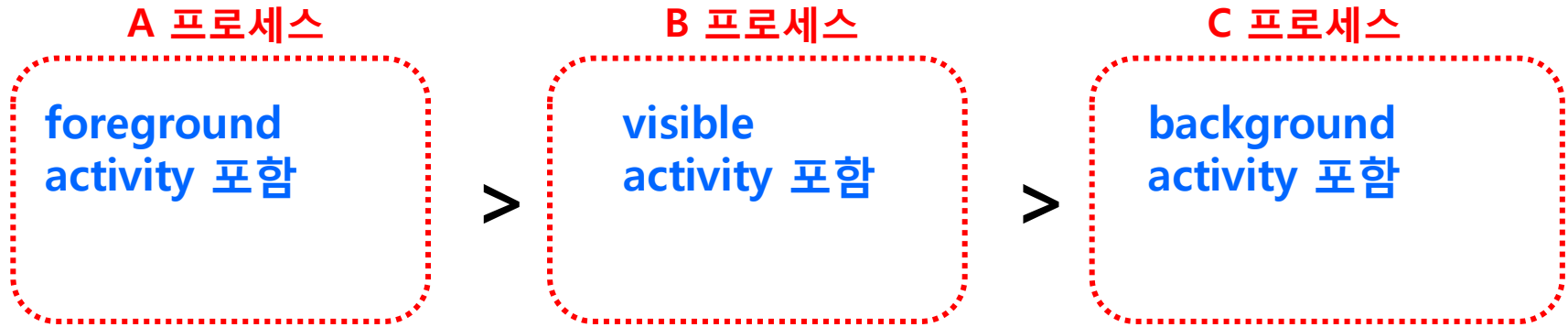
라이프사이클에서 빨강색 부분처럼 Activity shutdown된 Activity

프로세스는 어떤 활성화된 App 컴포넌트도 가지고 있지 않지만, 차후 다시 이 APP가 실행될 때를 대비해서 캐쉬의 용도로 메모리를 점유하고 있으며 (startup 시간 단축) 언제 사라질지 모름



프로세스의 우선순위

; 우선순위가 낮을 수록 메모리 부족 시 우선 제거 대상



- **'Killable after?'**
- 메소드가 리턴된 뒤에, 액티비티를 호스팅한 프로세스(process) 가 시스템에 의해 죽을 수 있는지 여부
- Yes 에 해당하는 메소드가 onPause(), onStop(), onDestroy() 로, onPause() 메소드가 리턴 된 뒤에는, 시스템이 메모리가 급하게 필요할 때 onStop() 이나 onDestroy() 호출이 없어도 프로세스를 강제로 종료(kill) 시킬 수 있음을 의미함
- 어플리케이션에서 지속적으로 저장해야 하는 필수적인 데이터들이 있다면 onPause() 에서 저장해 주어야 함
- No 라고 표시된 메소드 들은, 이 메소드들이 리턴 된 뒤에는 강제로 종료 되는 일이 없음

Killable After의 대처 –“saving activity state”

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
}
```

- 유저 입장에서선 액티비티가 죽어서 다시 만들어 졌다는 이런 뒷 이야기를 알 수 없기 때문에, 개발자는 사용자가 눈치 채지 못하도록, 액티비티가 다시 만들어질 때, 이전의 종료되기 전의 액티비티 상태로 똑같이 만들어줘야 함
- 적절한 콜백 메소드를 구현하여, 이전의 액티비티의 상태에 대한 중요한 정보를 저장해 놓고, 나중에 시스템에서 다시 액티비티를 만들 때, 그 정보를 활용하여 똑같이 복원될 수 있도록 해야 하는 것
- 이와 관련된 콜백 메소드가 바로 onSaveInstanceState()
- 시스템은 이 콜백메소드를 액티비티를 죽이기 전에 호출하여, 이 액티비티의 UI 상태를 번들 객체(Bundle object) 에 넘겨줌
- 시스템은 액티비티를 다시 만들어야 할 때, 번들(bundle) 객체를 onCreate() 의 전달인자로 넘겨주고, onCreate() 메소드에서 전달인자의 번들 객체를 이용해 다시 이전 액티비티 상태로 복원할 수 있는 것임

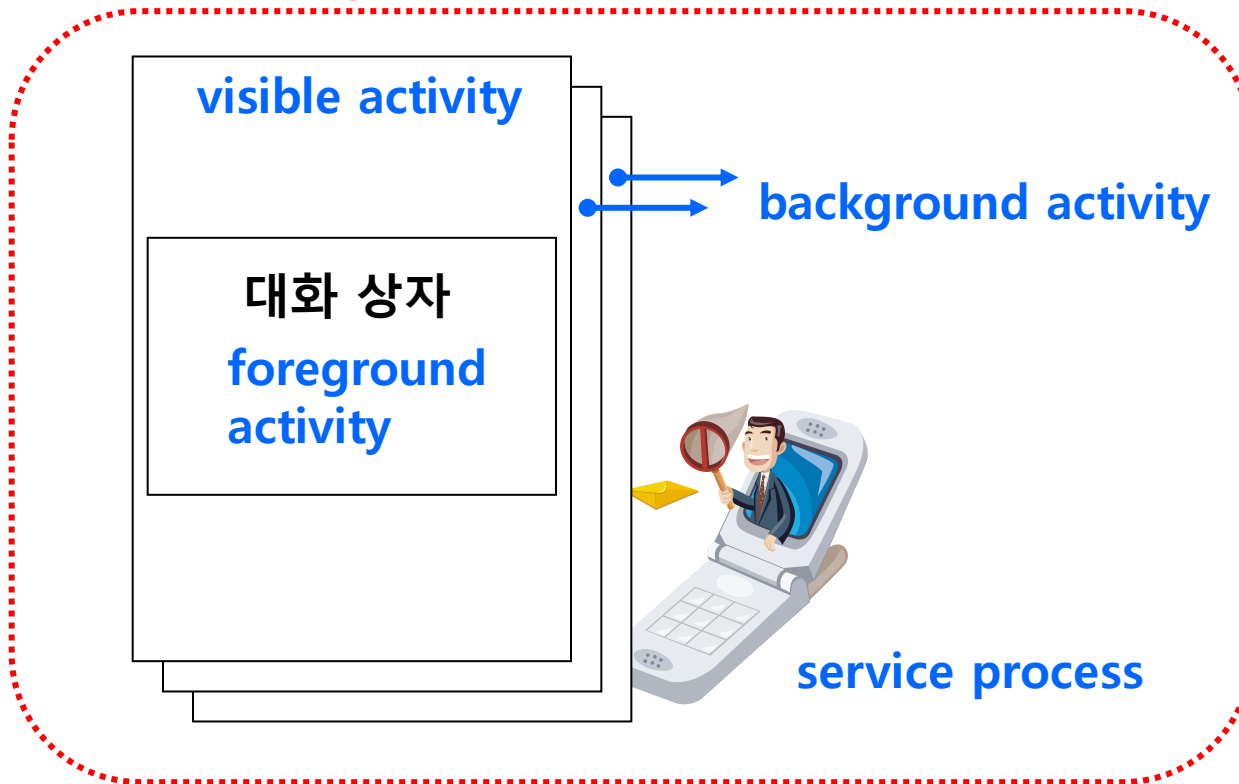


안드로이드의 기본 구조

; Activity와 프로세스가 동일한 것이 아님!

한 프로세스는 여러 개의 Activity를 가질 있고, 또한 한 프로세스에 foreground activity, visible activity, background activity 등을 모두 가질 수 있음

A 프로세스



Task

Task 1



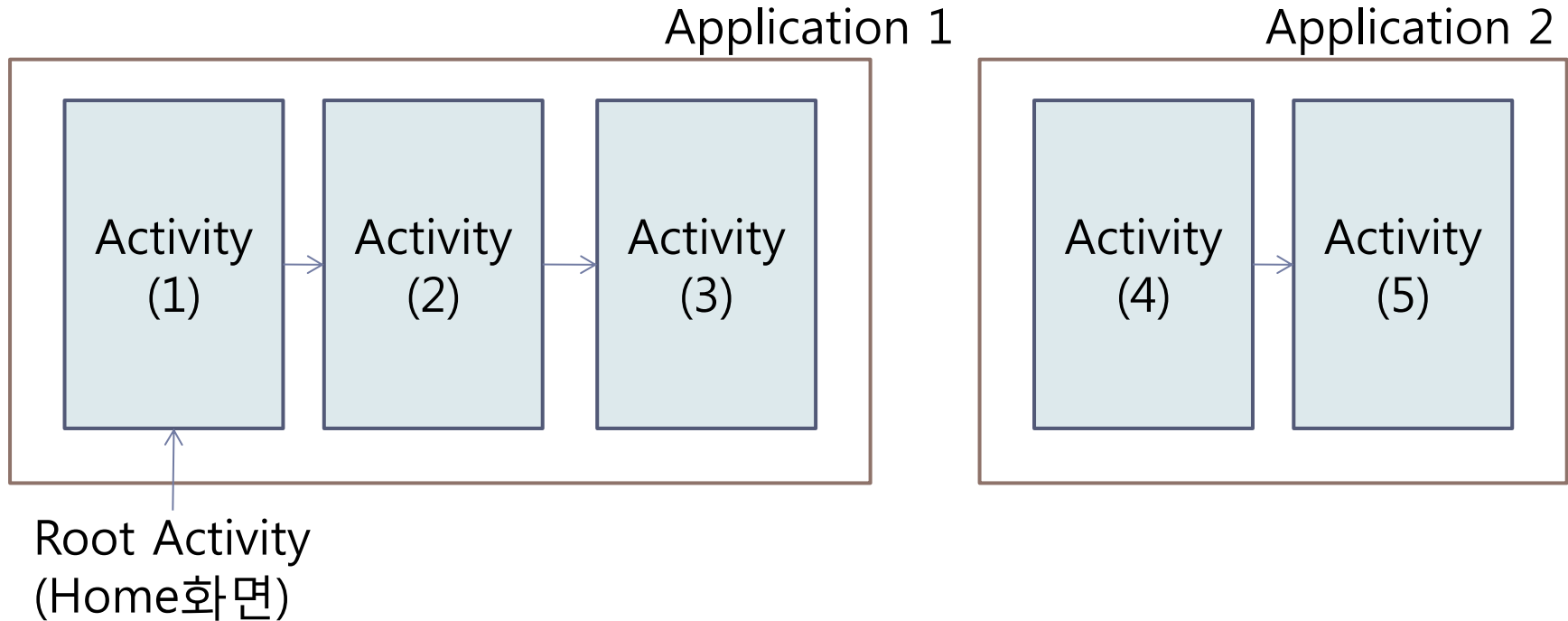
Back Key



Task 2

Task

- Home Key : Home 화면으로 이동
- 이전 Key : 이전 화면으로 이동



Task : Activity Stack



07_ActivityLifecycle Project

Project name: 07_ActivityLifecycle

Contents

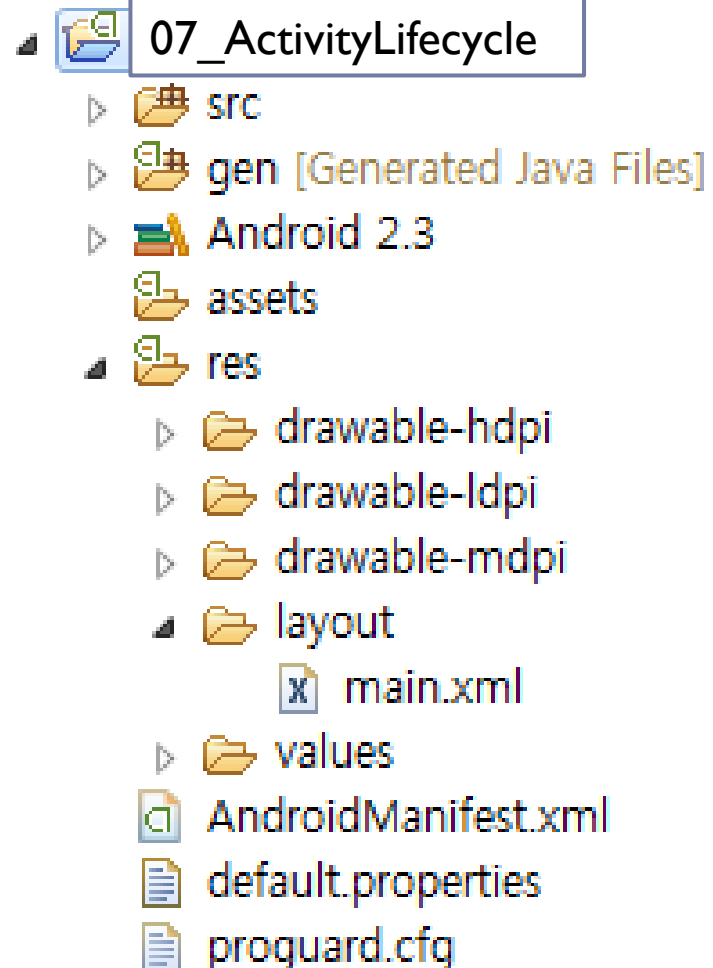
☒ Create new project in workspace

Properties

Application name: Activity LifeCycle

Package name: com.android.second

☐ Create Activity:



string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World!</string>
    <string name="app_name">Activity LifeCycle</string>
    <string name="title">Activity 생명주기</string>
    <string name="btn_txt_finish">Finish MainActivity</string>
</resources>
```



main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://~" ... >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="200dp"
        android:gravity="center"
        android:textSize="20sp"
        android:text="@string/hello"    />
    <Button
        android:id="@+id/btn_finish"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_txt_finish"    />
</LinearLayout>
```



src 만들기

src의 com.android.second에서 우클 – New – Class 선택

Source folder: 07_ActivityLifecycle/src

Package: com.android.second

☐ Enclosing type:

Name: MainActivity

Modifiers: ☒ public ☐ default
☐ abstract ☐ final ☐

Superclass: 4 android.app.Activity

Superclass Selection

Choose a type:

Activity

Matching items:

- 3 ☐ Activity - android.app
- ☐ ActivityGroup
- ☐ ActivityInfo

Superclass: 1 Activity 2 Browse...

Override

java 창에서

우클 – Source – Override/Implement Methods 선택

아래의 목록을 선택

onCreate()

onDestroy()

onPause()

onRestart()

onResume()

onStart()

onStop()



MainActivity.java

```
private static String TAG = "MainActivity";  
@Override  
public void onCreate(Bundle savedInstanceState) {  
  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    Log.i(TAG, TAG+" - onCreate()"); // Log.i(tag, message);  
}
```

void	<code>setContentView (int layoutResID)</code> Set the activity content from a layout resource.
void	<code>setContentView (View view)</code> Set the activity content to an explicit view.
void	<code>setContentView (View view, ViewGroup.LayoutParams params)</code> Set the activity content to an explicit view.



Log class

public final class

Log

extends [Object](#)

Summary: [Constants](#) | [Methods](#) | [Inherited Methods](#) | [\[Expand All\]](#)

Since: API Level 1

[java.lang.Object](#)
↳ [android.util.Log](#)

Class Overview

API for sending log output.

Generally, use the `Log.v()` `Log.d()` `Log.i()` `Log.w()` `Log.e()` methods.

The order in terms of verbosity, from least to most, is `Log.v()`, `Log.d()`, `Log.i()`, `Log.w()`, and `Log.e()`. These methods should never be compiled into a release build at runtime. Error, warning and info logs are not sent to the system log.

Constants		
int	ASSERT	Priority constant for the <code>println</code> method.
int	DEBUG	Priority constant for the <code>println</code> method; use <code>Log.d()</code> .
int	ERROR	Priority constant for the <code>println</code> method; use <code>Log.e()</code> .
int	INFO	Priority constant for the <code>println</code> method; use <code>Log.i()</code> .
int	VERBOSE	Priority constant for the <code>println</code> method; use <code>Log.v()</code> .
int	WARN	Priority constant for the <code>println</code> method; use <code>Log.w()</code> .



MainActivity.java

```
private static String TAG = "MainActivity";
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    // layout 표시
```

```
    setContentView(R.layout.main);
```

```
    Log.i(TAG, TAG+" - onCreate()");
```

```
    // view component
```

```
    Button finish = (Button)findViewById(R.id.finish);
```

```
    finish.setOnClickListener(new View.OnClickListener() {
```

```
        public void onClick(View v) { // 추상 메서드
```

```
            finish();
```

```
        }
```

```
    });
```

```
}
```

button을 click하면
프로그램 종료하도록
Code 작성

Override한 메서드에 모두 Log.i 붙이세요.

AndroidManifest.xml

```
<application android:icon= "@drawable/icon" android:label= "@string/app_name">  
    <activity android:name= ".MainActivity"  
        android:label= "@string/app_name">  
        <intent-filter>  
            <action android:name= "android.intent.action.MAIN" />  
            <category android:name= "android.intent.category.LAUNCHER" />  
        </intent-filter>  
    </activity>  
</application>
```



LogCat – Log Filter 설정

	pid	tag	Message
I	310	MainActivity	MainActivity - onCreate()
I	310	MainActivity	MainActivity - onStart()
I	310	MainActivity	
I	65	ActivityManag	
I	65	ActivityManag	
I	65	ActivityManag	
D	248	dalvikvm	
D	265	Email	
W	65	ActivityManag	
D	265	Email	

Log Filter

Filter Name:

by Log Tag:

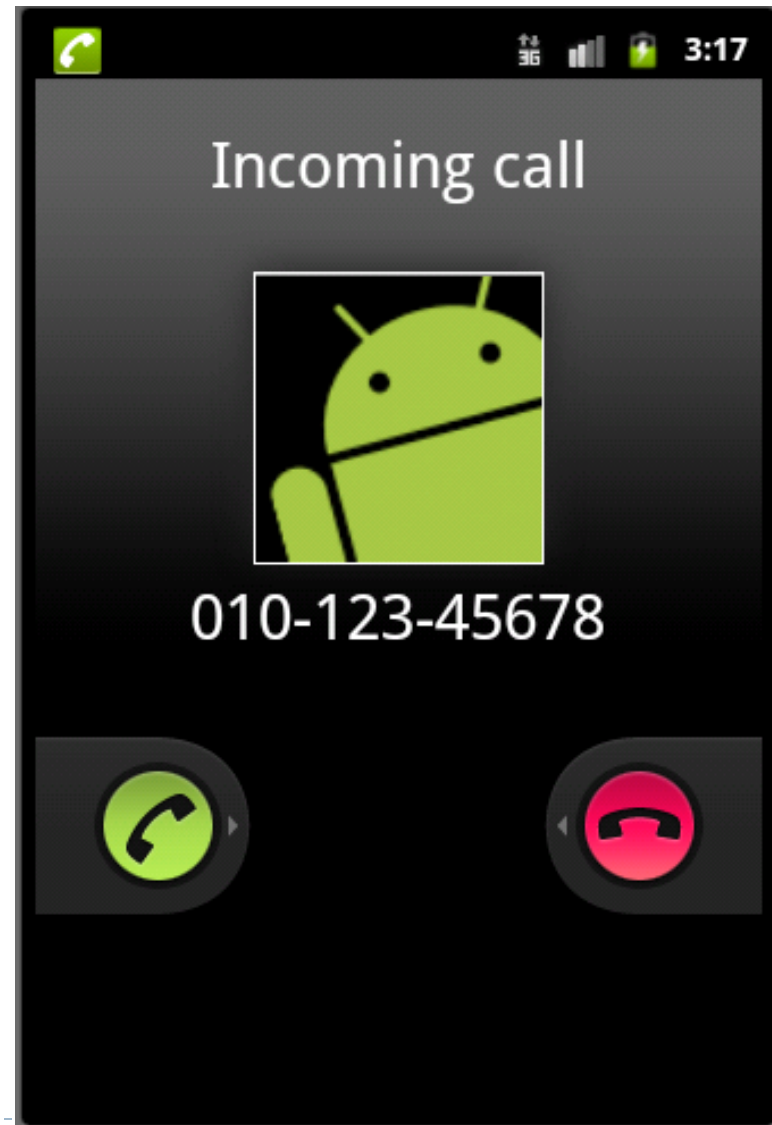
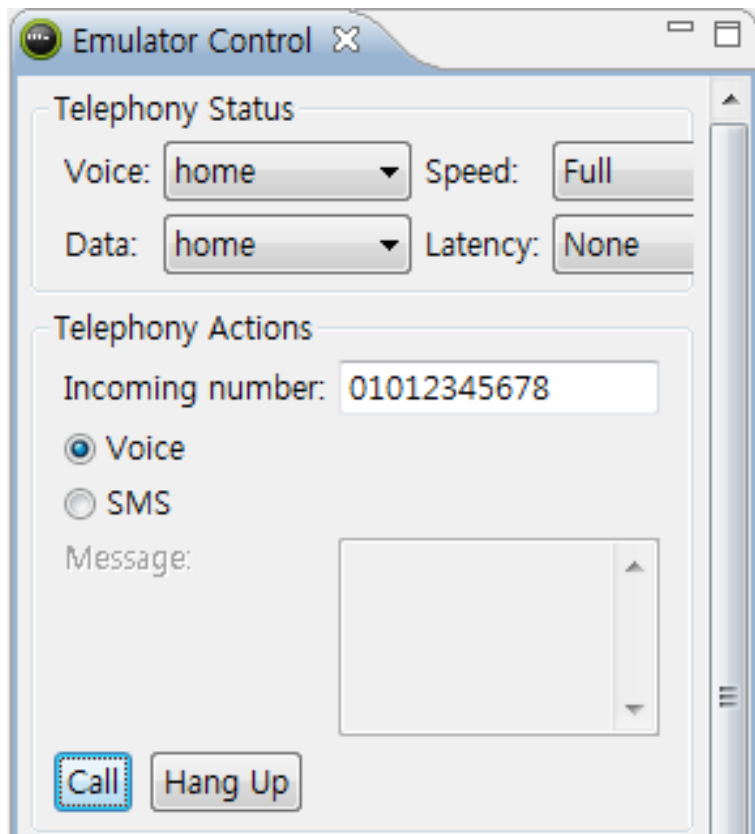
by pid:

by Log level:

일반적 수행 & 종료

Log (2)		Tag.i			
Time			pid	tag	Message
02-15	10:26...	I	290	MainActivity	MainActivity- onCreate()
02-15	10:26...	I	290	MainActivity	MainActivity- onStart()
02-15	10:26...	I	290	MainActivity	MainActivity- onResume()
02-15	10:26...	I	290	MainActivity	MainActivity- onPause()
02-15	10:26...	I	290	MainActivity	MainActivity- onStop()
02-15	10:26...	I	290	MainActivity	MainActivity- onDestroy()

수행 중 Incoming Call 발생



수행 중 Incoming Call 발생 & 종료

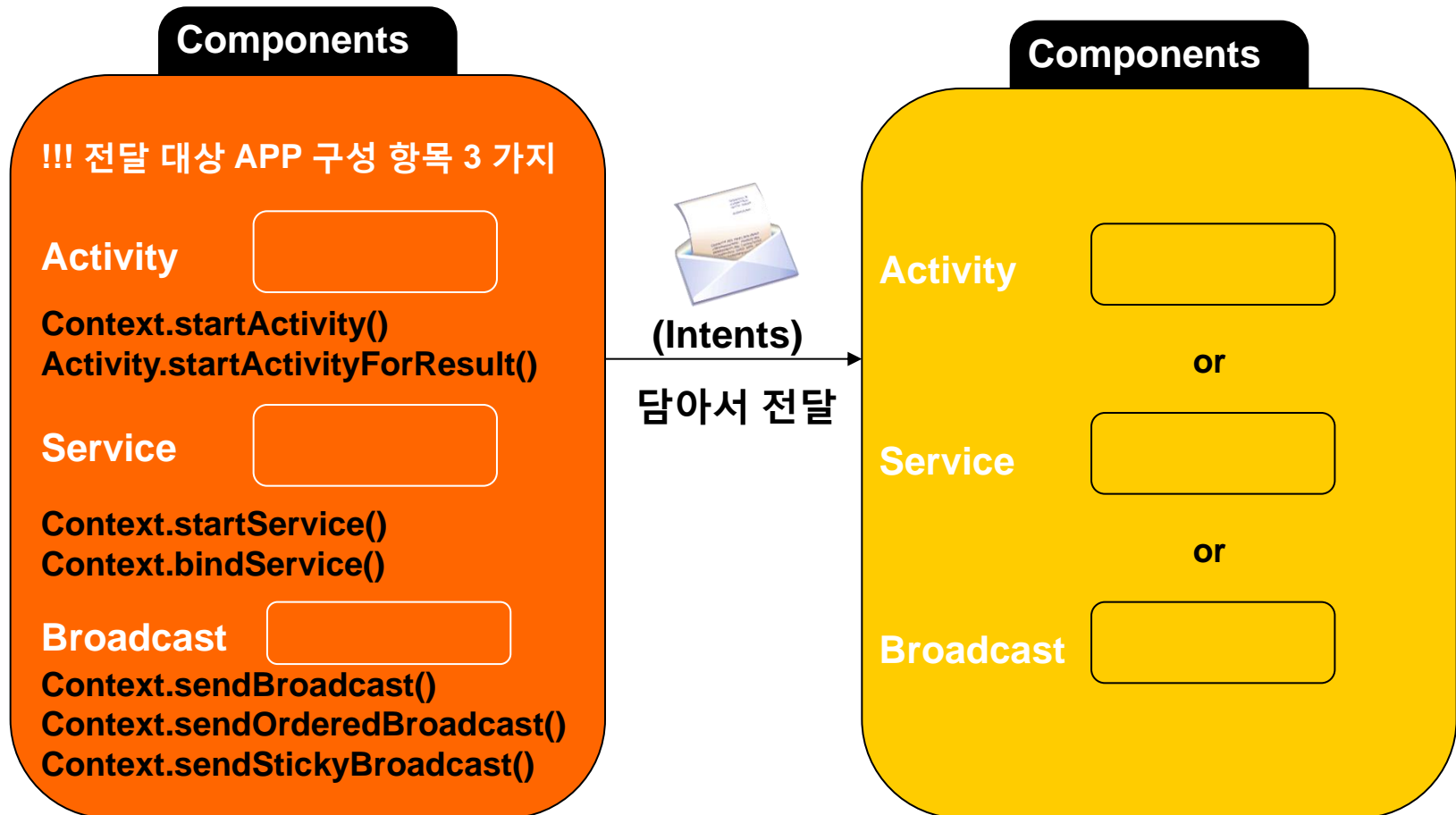
Log (18)		Tag.i			
Time			pid	tag	Message
02-15 10:36...		I	290	MainActivity	MainActivity- onCreate()
02-15 10:36...		I	290	MainActivity	MainActivity- onStart()
02-15 10:36...		I	290	MainActivity	MainActivity- onResume()
02-15 10:36...		I	290	MainActivity	MainActivity- onPause()
02-15 10:36...		I	290	MainActivity	MainActivity- onStop()
02-15 10:37...		I	290	MainActivity	MainActivity- onRestart()
02-15 10:37...		I	290	MainActivity	MainActivity- onStart()
02-15 10:37...		I	290	MainActivity	MainActivity- onResume()
02-15 10:37...		I	290	MainActivity	MainActivity- onPause()
02-15 10:37...		I	290	MainActivity	MainActivity- onStop()
02-15 10:37...		I	290	MainActivity	MainActivity- onDestroy()



Intent의 이해

Intents

A라는 컴포넌트가 B 컴포넌트를 실행 시키기 위해서 메시지를 전달하는 오브젝트
`startXXX`, `bindXXX`, `sendXXX` 등의 메서드가 인텐트를 통해서 활성화



다른 컴포넌트 호출

- 구성요소 중 Activity, Service, Broadcast를 호출 할 수 있음
- 다른 구성요소를 호출할 때 Intent에 내용을 담아 호출함
- Intent object(message object)를 매개인수로 하는 **호출메소드를 이용**하여 호출
- 구성요소 별 호출메소드가 다름
 - **Activity**: Context.startActivity(), Activity.startActivityForResult()
 - **Service**: Context.startService(), Context.bindService()
 - **Broadcast**: Context.sendBroadcast(), Context.sendOrderedBroadcast(), Context.sendStickyBroadcast()
- 객체가 객체를 호출하는 것으로 class instance가 class instance를 호출하는 것임

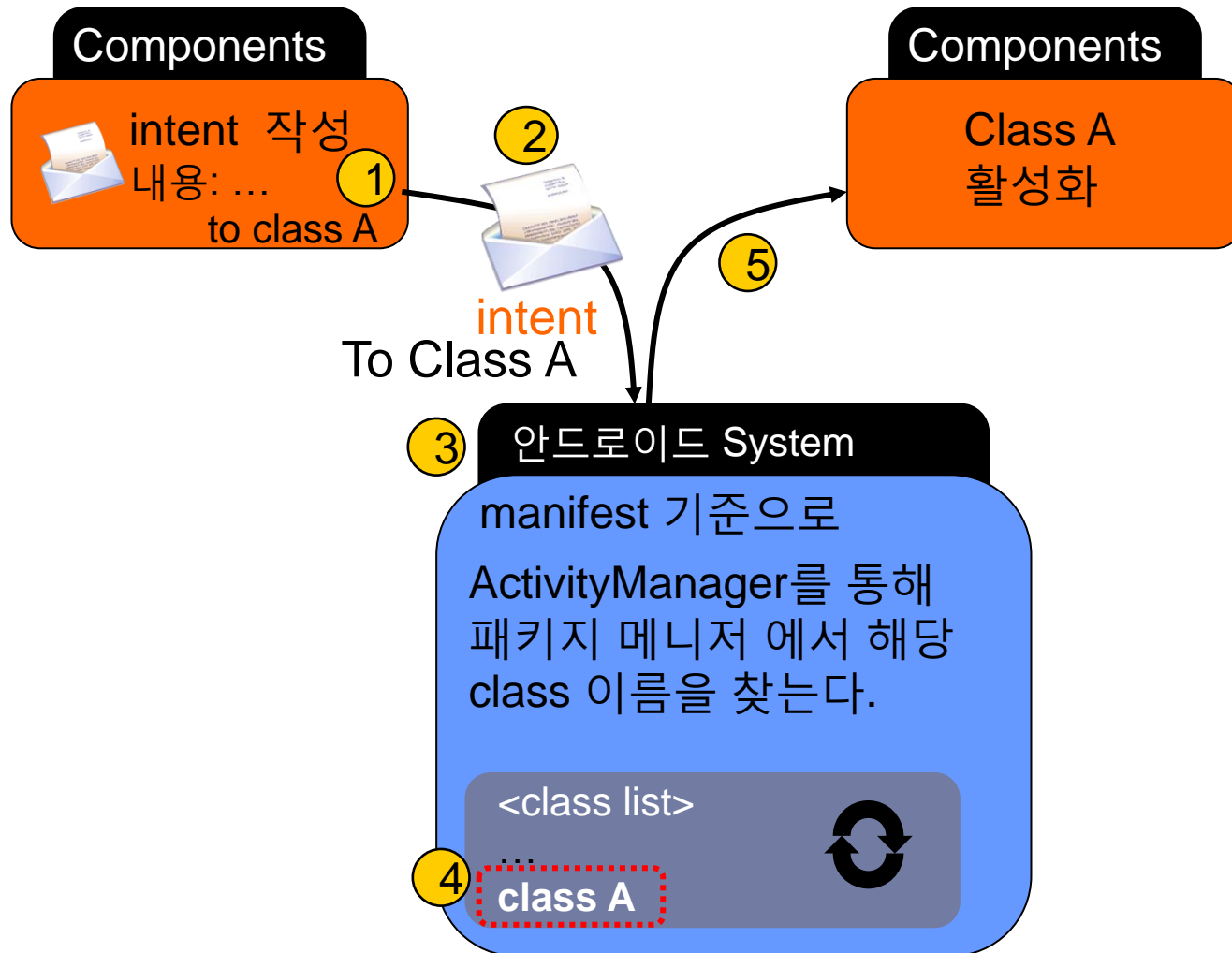


Intent의 구성요소

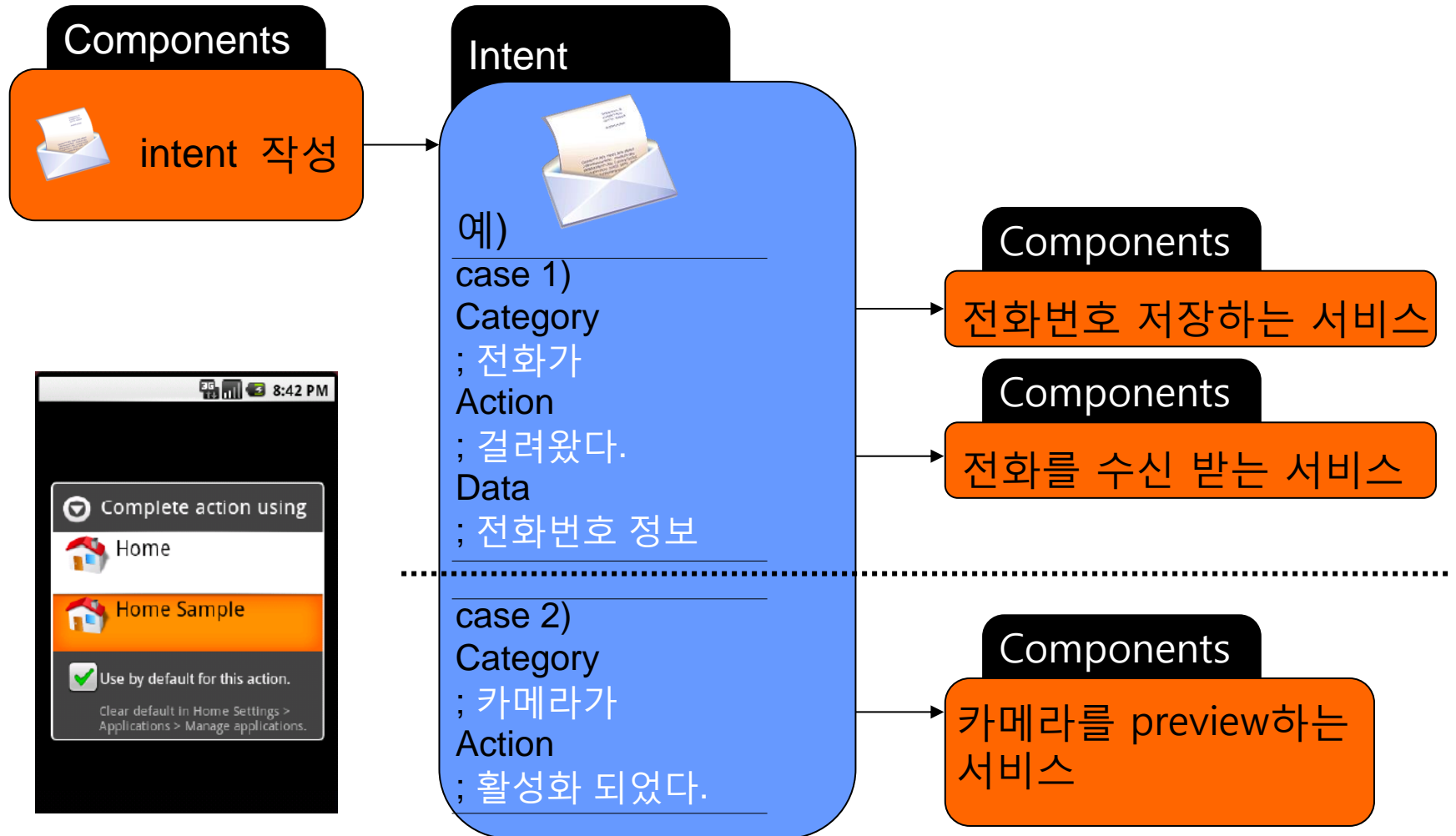
구성요소	설명
Component Name	인텐트 전달 대상의 컴포넌트 명
Action	동작 관련 (예 : SMS를 받았다.)
Category	대상의 종류 (예 : SMS)
Data	데이터 위치와 타입을 설정 (예 : 전화번호)
Extra	추가 정보 전달을 위함 (예 : 전화번호 관련 정보)
Flag	다양한 플래그, activity 상태 값 변화



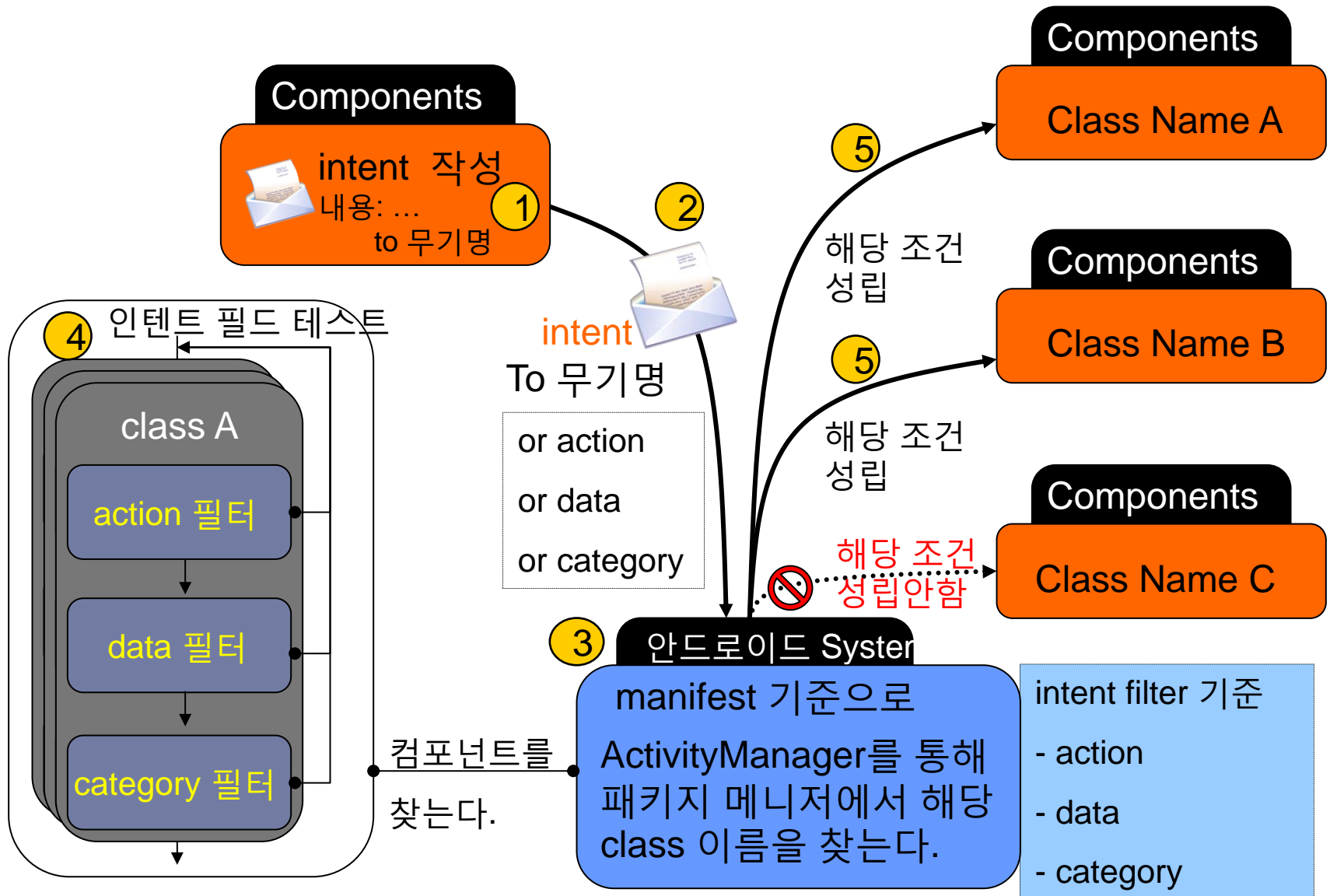
명시적 방법 (Explicit Method - Call Class)



암시적 방법 (Implicit Method – Intent Filter)



암시적 방법 (Implicit Method – Intent Filter)



명시적 방법의 Intent 사용

Activity의 호출

- 새로운 Activity 호출(Launch)
`Content.startActivity(Intent intent)` 사용
- 새로운 Activity 호출 후, 종료(finish)시 결과 값을 받는 호출
`Activity.startActivityForResult(Intent intent, int requestCode)` 사용
→ 반드시 **onActivityResult() override** 해야 함
→ 호출 받은 쪽에서 **setResult()**를 통해 호출 한 Activity로 Data 전달

void	<code>startActivityForResult(Intent intent, int requestCode)</code> Launch an activity for which you would like a result when it finished.
------	---



명시적 intent의 예

1. Intent Object에 호출대상 Component Name 포함
2. Intent.putExtra()를 이용해 호출대상 Component에 전달할 값 포함
3. startActivity() : Activity호출
startActivityForResult() : Activity 호출 및 되돌림 값 수신
→ 반드시 onActivityResult() override 해야 함
→ setResult() – 호출 Activity로 Data 전달



Intent

Public Constructors

`Intent()`

Create an empty intent.

`Intent(Intent o)`

Copy constructor.

`Intent(String action)`

Create an intent with a given action.

암시적 사용

`Intent(String action, Uri uri)`

Create an intent with a given action and for a given data url.

`Intent(Context packageContext, Class<?> cls)`

Create an intent for a specific component.

명시적 사용

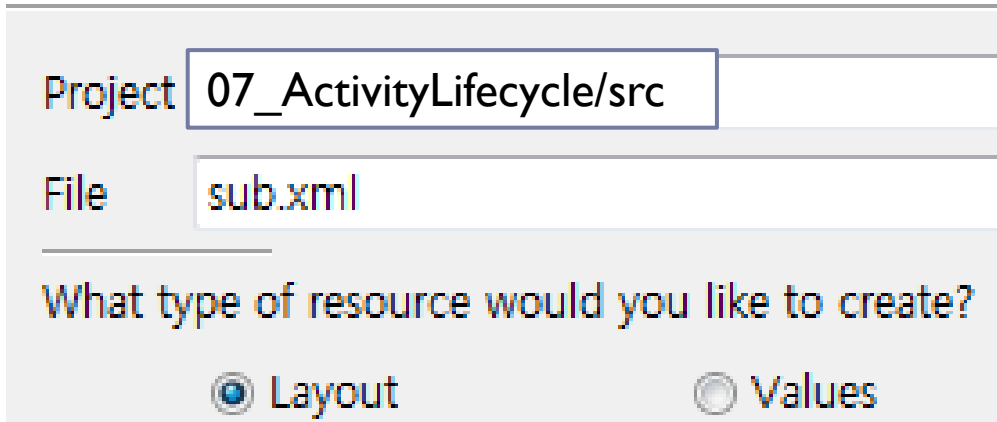
`Intent(String action, Uri uri, Context packageContext, Class<?> cls)`

Create an intent for a specific component with a specified action and data.



sub.xml

layout 폴더 위에서 우클 – Other – Android XML File - Next



Project 07_ActivityLifecycle/src

File sub.xml

What type of resource would you like to create?

☒ Layout ☐ Values



sub.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://~"
    android:orientation="vertical" ... >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="300dp"
        android:gravity="center"
        android:textSize="20sp"
        android:text="@string/subhello"    />
    <Button
        android:id = "@+id/back"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_txt_back"    />
</LinearLayout>
```



strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World!</string>
    <string name="app_name">Activity LifeCycle</string>
    <string name="title">Activity 생명주기</string>
    <string name="btn_txt_finish">Finish MainActivity</string>
    <string name="btn_txt_back">Back to MainActivity</string>
    <string name="btn_txt_callsub">Call SubActivity</string>
</resources>
```



SubActivity.java

```
private static String TAG = "SubActivity";
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.sub);

    Log.i(TAG, TAG+" - onCreate()"); // Log.i(tag, message);

    Button finish = (Button)findViewById(R.id.back);
    finish.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) { // 메시지
            Intent i = new Intent(SubActivity.this, MainActivity.class);
            finish();
        }
    });
}
```



AndroidManifest.xml

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
  <activity android:name=".MainActivity"
    android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
  <activity android:name=".SubActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Translucent">
  </activity>
</application>
```

모든 Activity는 AndroidManifest.xml에 등록되어 있어야 함



intent-filter

Application 외부에서 전달되는 Intent를 걸러내는 역할

filtering 기준 : action, category, data(선택적으로 사용 AND 조건이 됨)

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

```
<!-- DEFAULT category : category 상관 없이 action만 맞으면 통과 시키  
라는 뜻 -->
```



외부접근을 가능하도록 할 때의 intent-filter

```
<intent-filter>
  <action android:name= " com.android.app.SOYOUNG " />
  <category android:name= "android.intent.category.DEFAULT" />
</intent-filter>
```

- 이것은 외부에서 접근할 때 필요하므로 내부에서만 동작하는 activity는 intent-filter 없어도 되며,
- 만일 암시적 Intent를 위해 **사용자 정의 action**을 사용한다면 **category**를 **DEFAULT**로 주어야 함

```
<intent-filter>
  <action android:name= "android.intent.action.MAIN" />
  <category android:name= "android.intent.category.LAUNCHER" />
</intent-filter>
```

- **처음에 보여지는 activity**는 위의 intent-filter를 가지고 있어야 실행됨

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://~" ... >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="200dp"
        android:gravity="center"
        android:textSize="20sp"
        android:text="@string/hello"    />
    <Button
        android:id="@+id/callsub"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_txt_callsub"    />
    <Button ... />
</LinearLayout>
```



```
public void onCreate(Bundle savedInstanceState) {
```

... 위에 finish Button 처리 있음 ...

```
    Button callsub = (Button)findViewById(R.id.callsub);  
    callsub.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) { // 메서드  
            Intent i = new Intent(MainActivity.this, SubActivity.class);  
            i.putExtra("arg0","I Love You.");  
            startActivity(i);  
        }  
    });  
}
```

명시적 Intent의 예



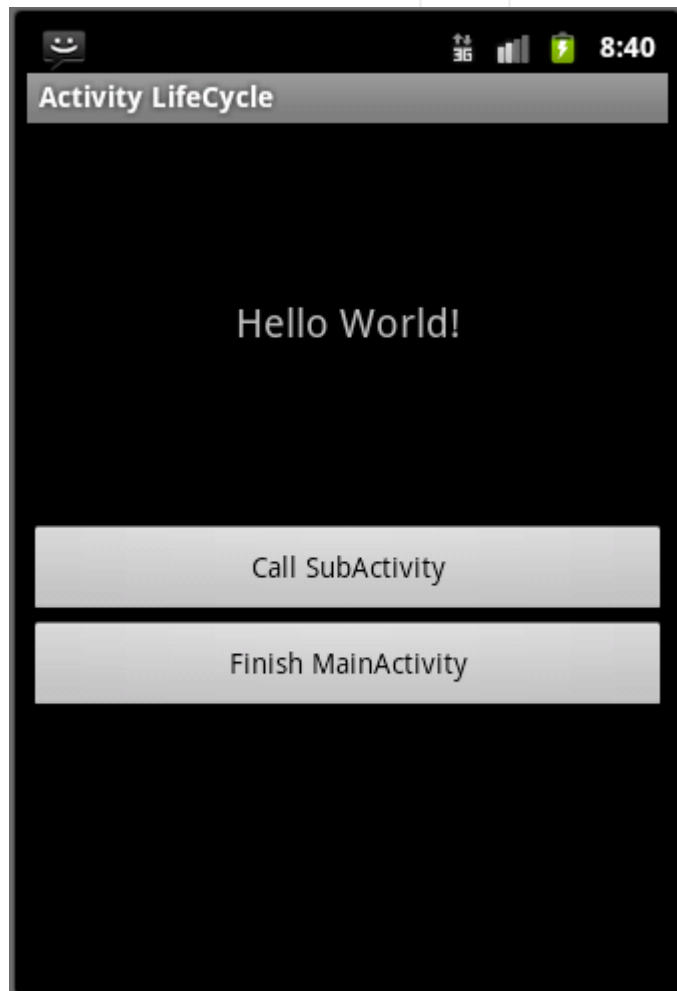
SubActivity.java Intent 정보 받기

```
private static String TAG = "SubActivity";
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.sub);

    Log.i(TAG, TAG+" - onCreate()"); // Log.i(tag, message);
    Intent i = getIntent();
    TextView arg = (TextView)findViewById(R.id.subtext);
    arg.setText(i.getStringExtra("arg0"));
    Button finish = (Button)findViewById(R.id.back);
    finish.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) { // 메서드
            finish();
        }
    });
}
```



Run (Main에서만 Intent로 Sub 호출시)

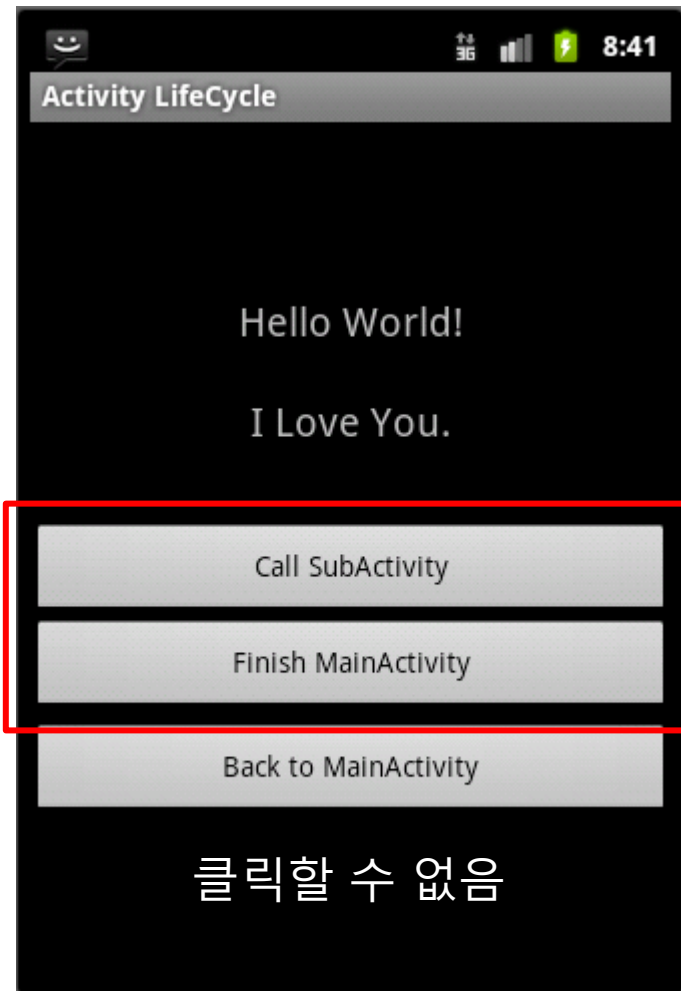


```
05-25... I 24382
05-25... I 24382
05-25... I 24382
```

tag	Message
MainActivity	MainActivity - onCreate()
MainActivity	MainActivity - onStart()
MainActivity	MainActivity - onResume()
MainActivity	MainActivity - onPause()
SubActivity	SubActivity - onCreate()
SubActivity	SubActivity - onStart()
SubActivity	SubActivity - onResume()
SubActivity	SubActivity - onPause()
MainActivity	MainActivity - onResume()
SubActivity	SubActivity - onStop()
SubActivity	SubActivity - onDestroy()
MainActivity	MainActivity - onPause()
SubActivity	SubActivity - onCreate()
SubActivity	SubActivity - onStart()
SubActivity	SubActivity - onResume()
SubActivity	SubActivity - onPause()
MainActivity	MainActivity - onResume()
SubActivity	SubActivity - onStop()
SubActivity	SubActivity - onDestroy()



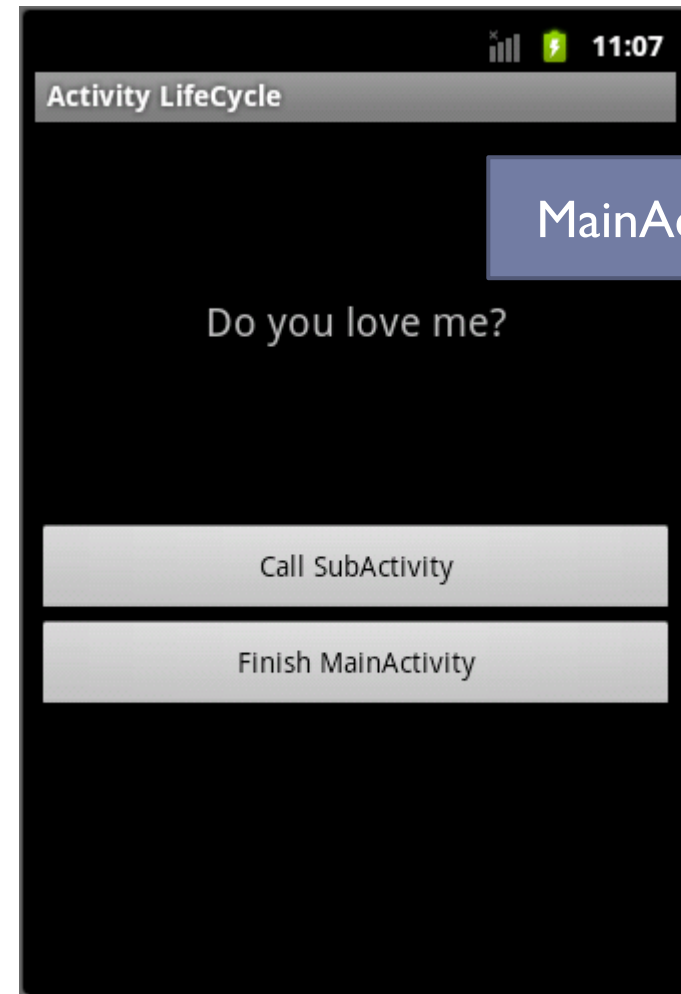
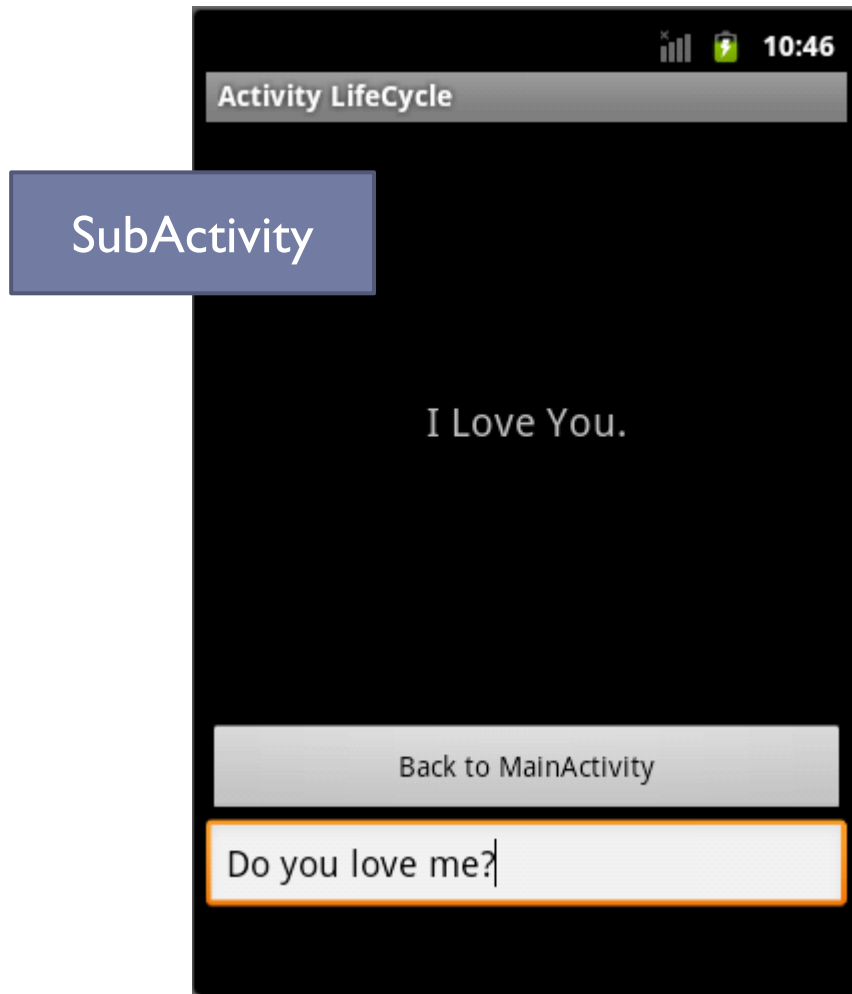
Run (양쪽에서 Intent 사용시)



05-25... I 24652

tag	Message
MainActivity	MainActivity - onCreate()
MainActivity	MainActivity - onStart()
MainActivity	MainActivity - onResume()
MainActivity	MainActivity - onPause()
SubActivity	SubActivity - onCreate()
SubActivity	SubActivity - onStart()
SubActivity	SubActivity - onResume()
SubActivity	SubActivity - onPause()
MainActivity	MainActivity - onStop()
MainActivity	MainActivity - onCreate()
MainActivity	MainActivity - onStart()
MainActivity	MainActivity - onResume()
SubActivity	SubActivity - onStop()
MainActivity	MainActivity - onPause()
SubActivity	SubActivity - onCreate()
SubActivity	SubActivity - onStart()
SubActivity	SubActivity - onResume()

SubActivity.java로 부터 데이터 돌려 받기



sub.xml 에 추가 사항

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://~"  
    android:orientation="vertical" ... >
```

```
    <TextView />
```

```
    <Button />
```

```
    <EditText
```

```
        android:id="@+id/subedit"
```

```
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="" />
```

```
</LinearLayout>
```

텍스트를 입력 받기 추가



AndroidManifest.xml

```
<application android:icon="@drawable/icon" android:label="@string/app_name">
  <activity android:name=".MainActivity"
    android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>
  <activity android:name=".SubActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Translucent">
  </activity>
</application>
```

투명 스타일 제거



SubActivity.java

```
private static String TAG = "SubActivity";
private EditText editText;
protected void onCreate(Bundle savedInstanceState) {
    ... 생략

    editText = (EditText)findViewById(R.id.subedit),
    Button finish = (Button)findViewById(R.id.back);
    finish.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) { // 메서드
            Intent i = new Intent(SubActivity.this, MainActivity.class);
            i.putExtra("arg1", editText.getText().toString());
            setResult(RESULT_OK, i);
            finish();
        }
    });
}
```

Intent 생성 및 Extra 넣어
setResult 호출



MainActivity.java Intent 생성 및 Activity 호출

```
private TextView textView;  
final static int SUB=0;  
public void onCreate(Bundle savedInstanceState) {
```

... 생략 ...

```
textView = (TextView)findViewById(R.id.textview);
```

```
Button callsub = (Button)findViewById(R.id.callsub);  
callsub.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) { // 메서드  
        Intent i = new Intent(MainActivity.this, SubActivity.class);  
        i.putExtra("arg0","I Love You.");  
        startActivityForResult(i, SUB);  
    }  
});  
}
```



MainActivity.java Intent 생성 및 Activity 호출

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    // TODO Auto-generated method stub
    //super.onActivityResult(requestCode, resultCode, data);
    switch(requestCode){
        case SUB:
            if (resultCode == RESULT_OK){
                textView.setText(data.getStringExtra("arg1"));
            }
            break;
    }
}
```



암시적 방법의 Intent 사용

암시적 intent

- 암시적 intent : 호출 대상 Component Name을 정확하게 알 수 없으나 해당 Component의 동작/data 위치/data type을 알고 있을 때 사용하는 방법
- action, category, data/MIME type 을 담을 수 있음

<대상이 1개 찾아지면 바로 실행>

```
Intent i = new Intent(Intent.ACTION_DIAL);  
i.setData(Uri.parse("tel:011-1234-5678"));  
startActivity(i);
```

xml에 Permission 필요 :

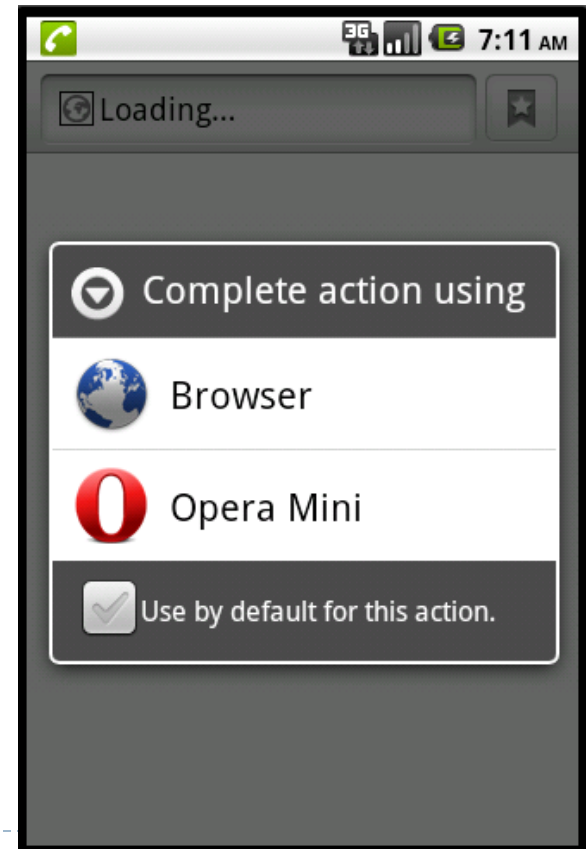
"android.permission.CALL_PHONE"

<대상이 여럿이면 선택화면 보임>

```
Intent i = new Intent(Intent.ACTION_VIEW);  
i.setData(Uri.parse("http://www.naver.com"));  
startActivity(i);
```

xml에 Permission 필요 :

"android.permission.INTERNET"



Intent

Public Constructors

`Intent()`

Create an empty intent.

`Intent(Intent o)`

Copy constructor.

`Intent(String action)`

Create an intent with a given action.

암시적 사용

`Intent(String action, Uri uri)`

Create an intent with a given action and for a given data url.

`Intent(Context packageContext, Class<?> cls)`

Create an intent for a specific component.

명시적 사용

`Intent(String action, Uri uri, Context packageContext, Class<?> cls)`

Create an intent for a specific component with a specified action and data.



암시적 Intent의 예) MainActivity.java

```
public void onCreate(Bundle savedInstanceState) {
```

... 위에 finish Button 처리 있음 ...

```
Button callsub = (Button)findViewById(R.id.callsub);  
callsub.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) { // 메서드  
        Intent i = new Intent(Intent.ACTION_CALL);  
        i.setData(Uri.parse("tel:011-1234-5678"));  
        startActivity(i);  
    }  
});  
}
```



AndroidManifest.xml - Permission

```
<application android:icon= "@drawable/icon" android:label= "@string/app_name">
    <activity android:name= ".MainActivity"
        android:label= "@string/app_name">
        <intent-filter>
            <action android:name= "android.intent.action.MAIN" />
            <category android:name= "android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name= ".SubActivity"
        android:label= "@string/app_name" >
    </activity>
</application>
<uses-permission android:name= "android.permission.CALL_PHONE" />
<uses-permission android:name= "android.permission.INTERNET" />
```

