자바응용SW(앱)개발자양성

태스크(Task)

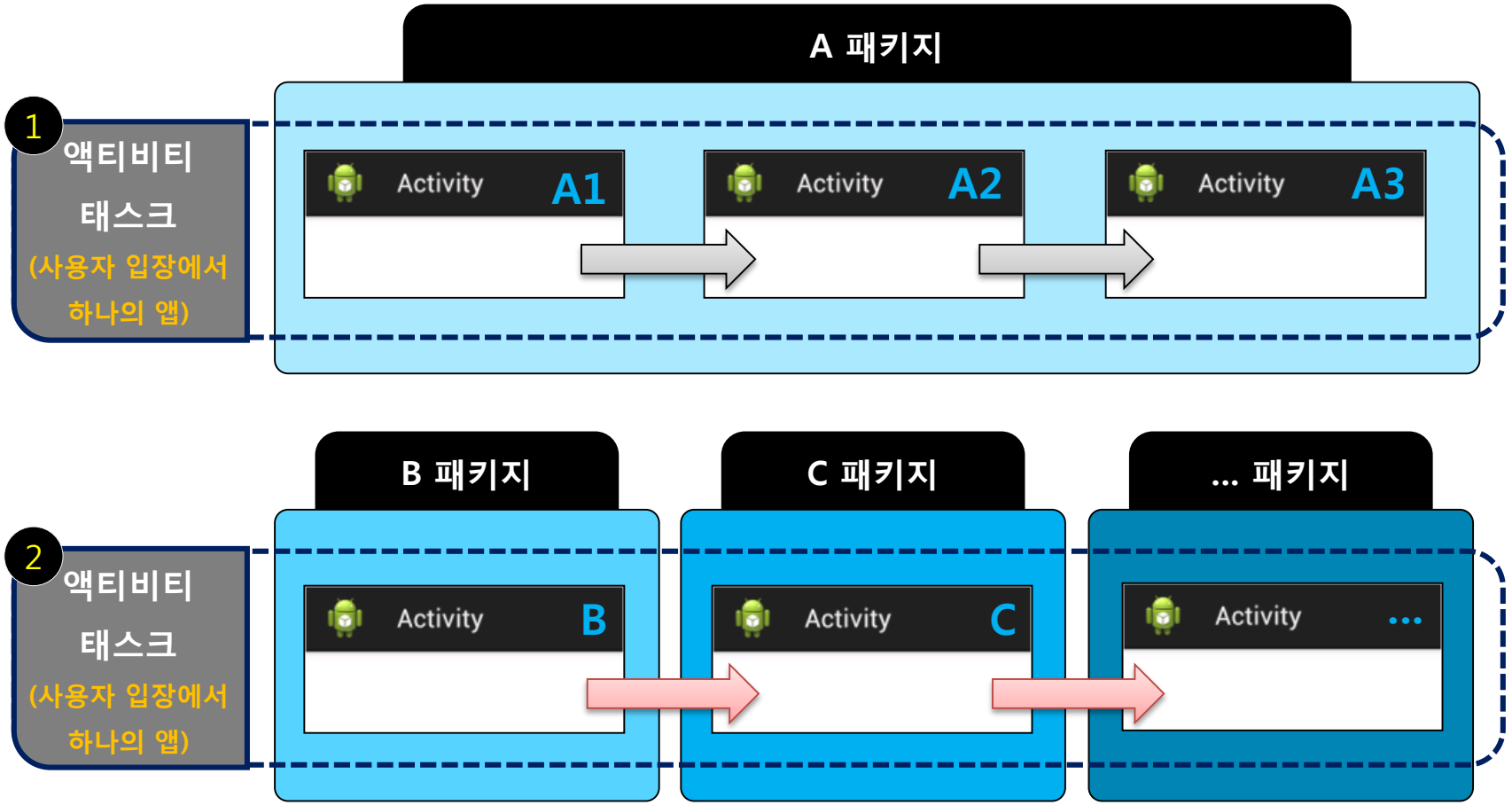
백제직업전문학교

김영준 강사

1. 태스크에 대하여

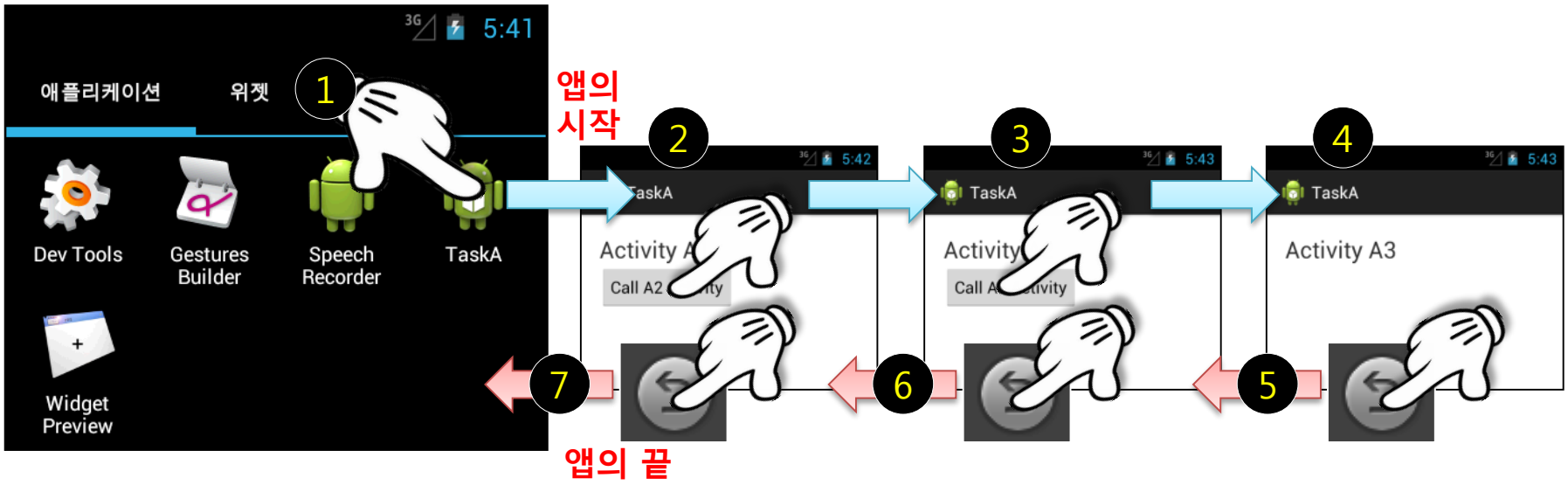
■ 안드로이드는 앱의 경계가 없다.

"안드로이드는 앱의 경계가 없다." 바로 이 것만 이해하면 태스크를 이해했다고 볼 수 있다.

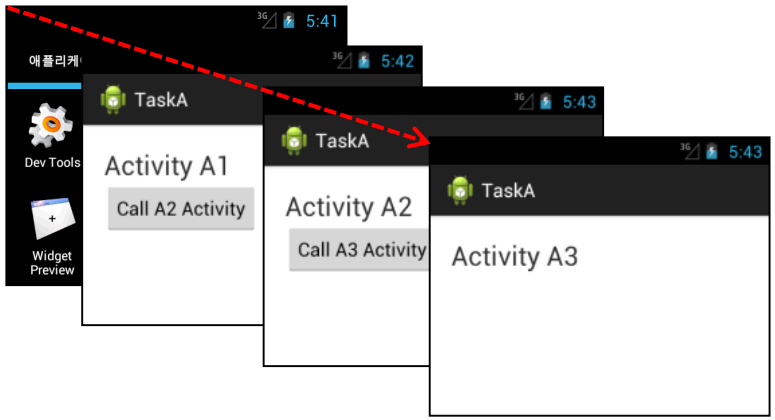


즉 태스크는 사용자의 입장에서 하나의 앱이고, 패키지는 개발자 입장에서 하나의 앱이다.

■ 사용자 입장에서 하나의 앱이란?

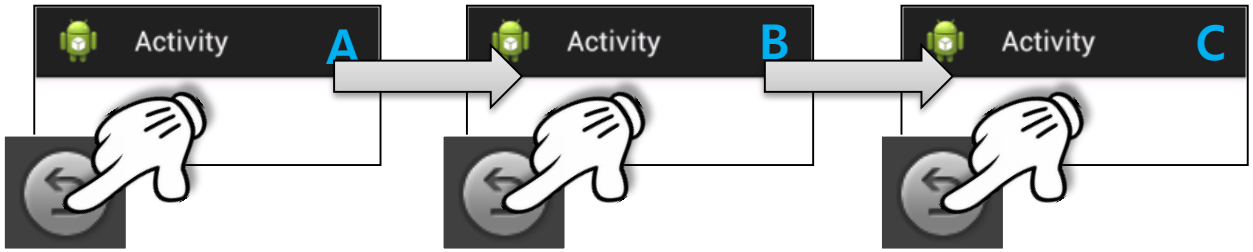


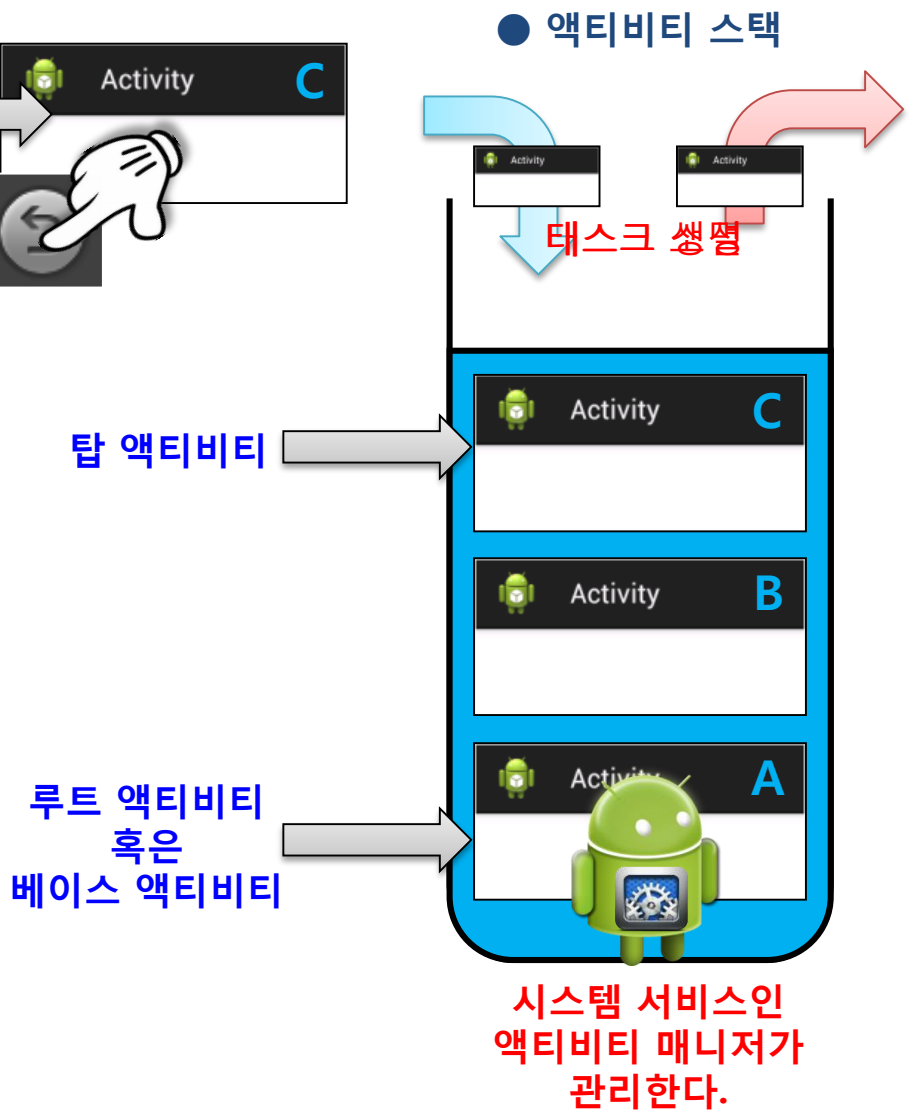
■ 이를 위해 액티비티의 실행 순서를 쌓아둘 스택이 필요한 것이다.



이 것이 바로 태스크고
이를 액티비티 태스크라고도 부른다.

■ 액티비티 태스크는 스택과 유사한 구조다. 따라서 액티비티 스택이라고 부르기도 한다.





■ 태스크는 왜 배워야 하나?

여러 가지 이유가 있겠지만 그 중 대표적인 한가지를 설명한다.

■ 먼저 A→B→C 액티비티 순으로 호출되는 예제 앱을 만든다.

AndroidManifest.xml

```
<?xml vers
<manifest
    packag
    androi
    androi
    <uses-
        an
    </a
</manifest>
```

res/layout/activity_a1.xml

```
<Li
```

res/layout/activity_a2.xml

```
<Li
```

res/layout/activity_a3.xml

src/ActivityA1.java

```
pub
{
}
```

src/ActivityA2.java

```
pub
{
    }
}
```

src/ActivityA3.java

```
public class ActivityA3 extends Activity
{
    @Override
    protected void onCreate( Bundle savedInstanceState )
    {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_a3 );
    }
}
```

■ 먼저 A→B→C 순으로 액티비티를 실행한다.

이 상태에서 액티비티 스택은 A→B→C 일 것이다. 하지만 직접 눈으로 확인할 방법은 없을까?

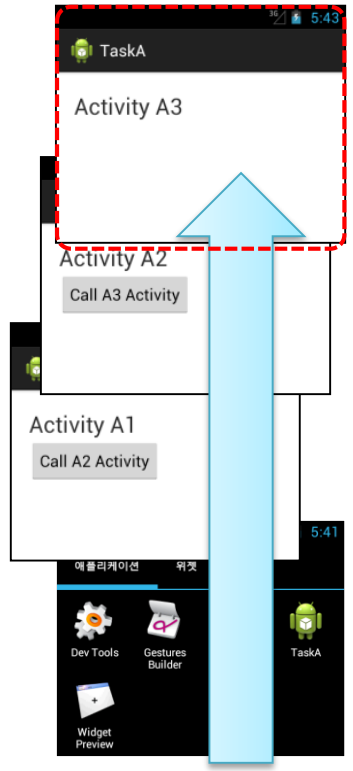
■ 액티비티 매니저가 가진 액티비티 스택 정보 보기 adb 명령어

- **adb shell dumpsys activity activities > activity_stack_1_1.txt**
  - adb shell : Linux shell 명령어를 사용한다.
  - dumpsys : 시스템 덤프를 뜬다.
  - activity : 액티비티 매니저 서비스를 의미한다.
  - activities : 액티비티 컴포넌트를 말한다.
  - > activity_stack_1_1.txt : 결과를 저장할 파일명이다.

activity_stack_1_1.txt

```
...
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

Main stack:
  * TaskRecord{4141c940 #6 A com.superdroid.test.TaskA U 0}
    numActivities=3 rootWasReset=true userId=0
    ...
    * Hist #3: ActivityRecord{415fd2d0 com.superdroid.test.TaskA/.ActivityA3}
        ...
        state=RESUMED
        ...
    * Hist #2: ActivityRecord{415d7f10 com.superdroid.tes    skA/.ActivityA2}
        ...
        state=STOPPED ...
        ...
    * Hist #1: ActivityRecord{4163fbb0 com.superdroid.tes    skA/.ActivityA1}
        ...
        frontOfTask=true ...
        state=STOPPED ...
        ...

  * TaskRecord{415c3d28 #2 A com.android.launcher U 0}
     numActivities=1 ...
    ...
    * Hist #0: ActivityRecord{415c1f68
                        com.android.launcher/com.androi    auncher2.Launcher}
    ...
```

현재 단말기에 보여지는 액티비티

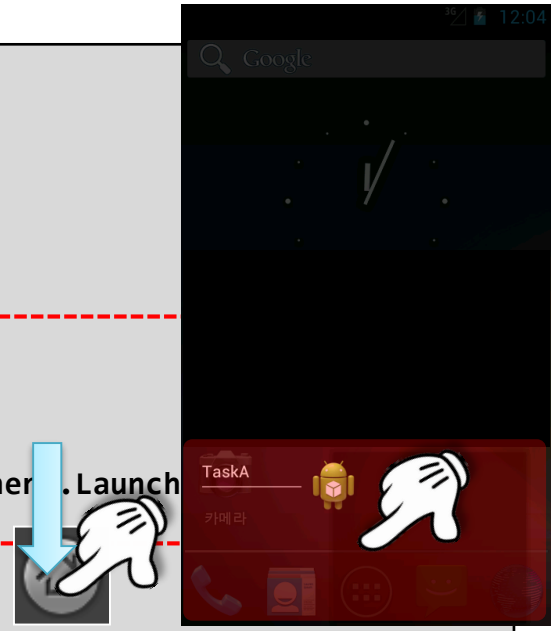



TaskRecord
History #1
History #2
History #3
5:42
TaskA
Activity A1
Call A2 Activity
Stop 상태
5:43
TaskA
Activity A2
Call A3 Activity
Stop 상태
현재 단말기에 보여지는 액티비티
5:43
TaskA
Activity A3
Resume 상태
루트 액티비티
탑 액티비티
TaskRecord
History #0
5:41
애플리케이션
위젯
Dev Tools
Gestures Builder
Speech Recorder
TaskA
Widget Preview
Stop 상태
루트/탑 액티비티

■ 태스크의 순서는 어떻게 변경할까?

activity_stack_1_1.txt

```
...
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

Main stack:
  * TaskRecord{415c3d28 #2 A com.android.launcher U 0}
    numActivities=1 ...
   ...
    * Hist #3: ActivityRecord{415c1f68
                              com.android.launcher/com.android.launcher.Launch
   ...

  * TaskRecord{4141c940 #6 A com.superdroid.test.TaskA U 0}
    numActivities=3 rootWasReset=true userId=0
    ...
    * Hist #2: ActivityRecord{415fd2d0 com.superdroid.test.TaskA/.ActivityA3}
       ...
       state=RESUMED
       ...
    * Hist #1: ActivityRecord{415d7f10 com.superdroid.test.TaskA/.ActivityA2}
       ...
       state=STOPPED ...
       ...
    * Hist #0: ActivityRecord{4163fbb0 com.superdroid.test.TaskA/.ActivityA1}
       ...
       frontOfTask=true ...
       state=STOPPED ...
       ...
```

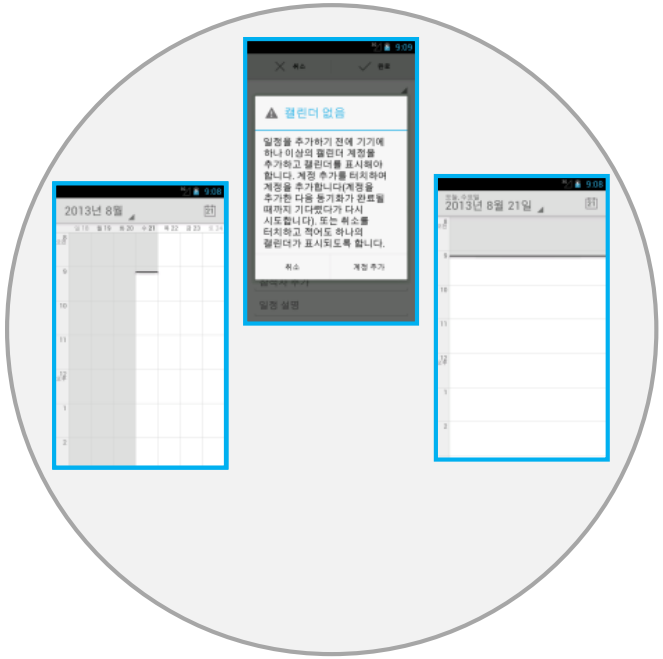



설정 앱
태스크
루트
탑
휴대전화
앱
태스크
루트
탑
캘린더
앱
태스크
루트
탑
최근 실행 앱
Google
설정
휴대전화
캘린더
카메라

■ 안드로이드는 태스크마다 이름이 존재한다. 만일 홈에서 앱을 실행하면 해당 액티비티의 패키지 명으로 태스크가 생성되고, 액티비티는 생성된 태스크에 배치된다.
또한 실행된 액티비티에서 다른 앱의 액티비티를 실행하더라도 호출한 액티비티에 포함된다.

휴대전화 앱

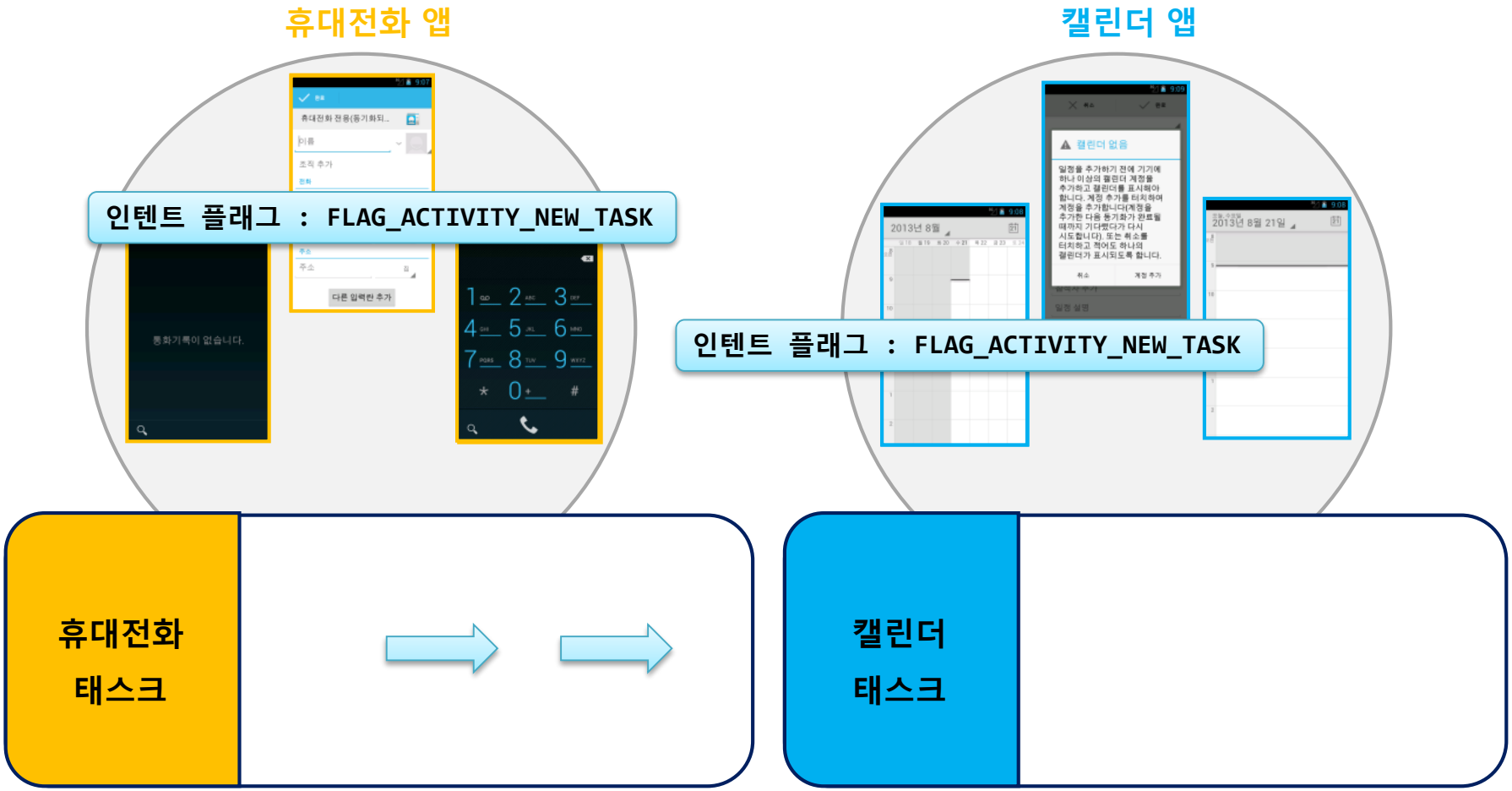
캘린더 앱



휴대전화
태스크

이것이 안드로이드 태스크 기본 정책이며, 사용자 측면에서 직관적이다.

■ 하지만 특정앱은 특성이 너무 강해 반드시 자신의 태스크에 모여 있어야 하는 경우 가 있을 것이다.



■ FLAG_ACTIVITY_NEW_TASK 인텐트 플래그를 추가하고 액티비티를 실행하면, 실행되는 액티비티는 자신의 패키지 명의 태스크에 배치된다. 그런데 만일 패키지 명의 태스크가 존재하지 않으면 패키지 명으로 **새로운 태스크를 생성하고 배치된다.**

그리고 안드로이드는 **태스크 명을 친화력(Affinity) 명이라 부른다.**

그 이유는 태스크 명에 의해 동일한 앱의 액티비티가 모여 있을 수 있기 때문이다.

바로 이전에 사용된 A→B →C 액티비티가 순서데로 실행되는 예제를 통해 확인해보자.

adb shell dumpsys activity activities

```
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

  Main stack:

* TaskRecord{41a81180 #6 A com.superdroid.test.TaskA U 0}
  affinity=com.superdroid.test.TaskA
  intent={ act=android.intent.action.MAIN
         cat=[android.intent.category.LAUNCHER]
         flg=0x10200000 cmp=com.superdroid.test.TaskA/.ActivityA1 …}
    * Hist #1: ActivityRecord{41a80f08 u0 com.superdroid.test.TaskA/.ActivityA1}
      taskAffinity=com.superdroid.test.TaskA

* TaskRecord{42099b90 #4 A com.android.launcher U 0}
  ffinity=com.android.launcher
    Hist #0: ActivityRecord{41aac390 u0 com.android.launcher/com.android.launcher2.Launcher}
      taskAffinity=com.android.launcher
```

■ 액티비티의 태스크 친화력(TaskAffinity) 이름은 패키지 명이라고 했다. 하지만 이는 변경이 가능하다.

androidManifest.xml

```
<activity android:name=".ActivityA1" android:taskAffinity="com.A">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".ActivityA2" android:taskAffinity="com.B"/>
<activity android:name=".ActivityA3" android:taskAffinity="com.A"/>
```

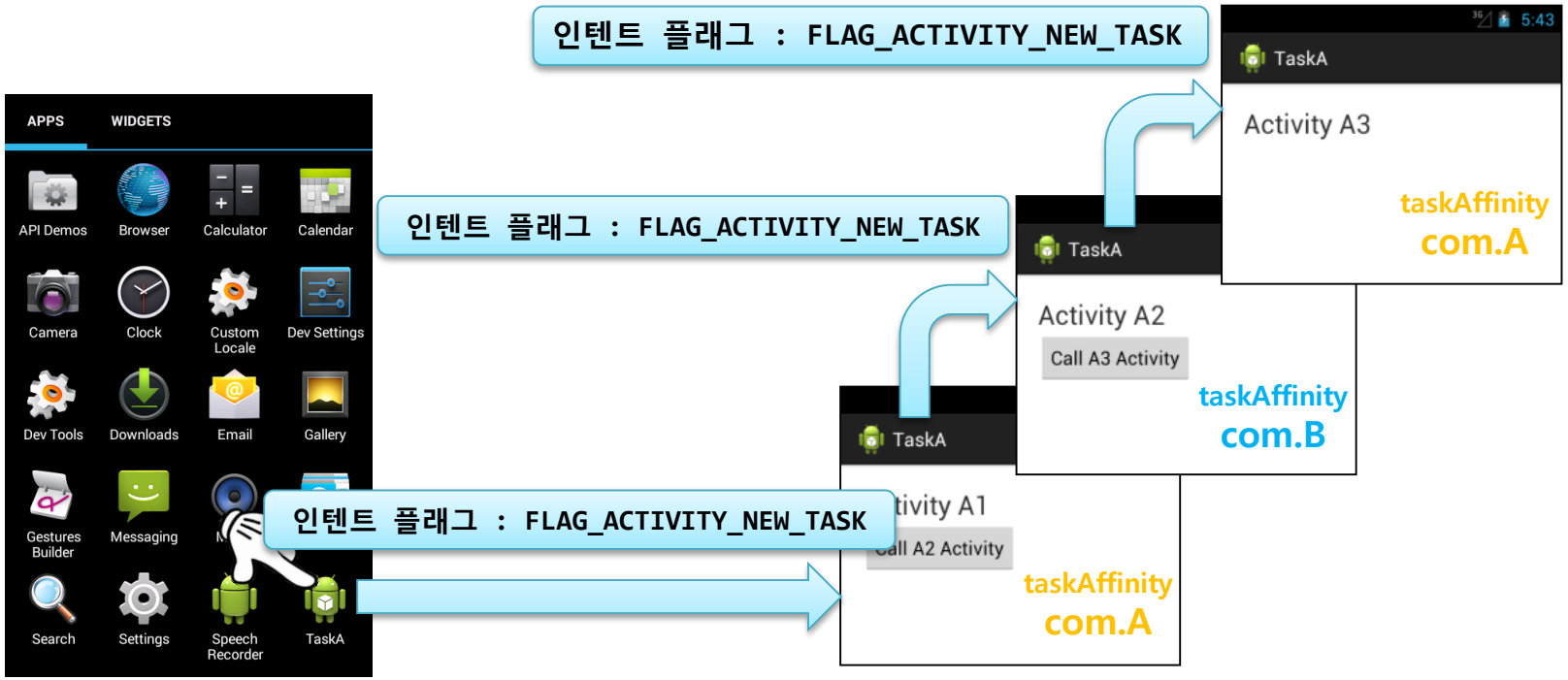
src/ActivityA1.java

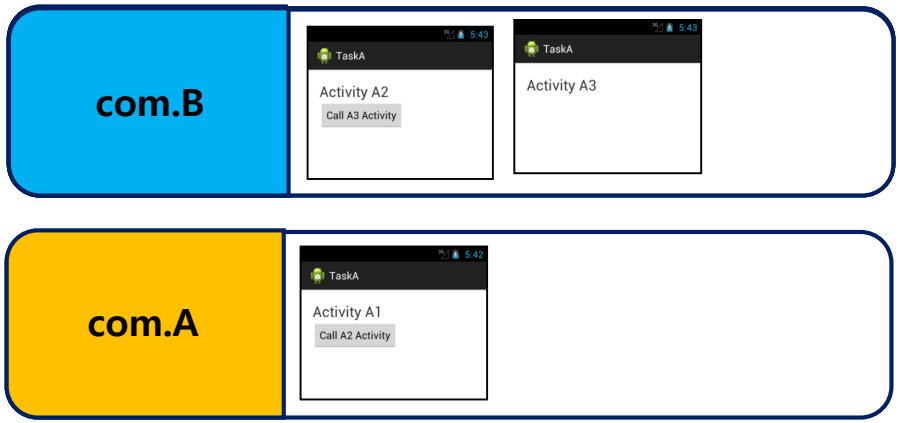
```
public class ActivityA1 extends Activity
{

    public void onClick ( View v )
    {
        Intent intent = new Intent
        intent.addFlags( In

        startActivity( intent );
    }
}
```

src/ActivityA2.java

```
public class ActivityA2 extends Activity
{

    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA3.class);
        intent.addFlags( Intent.FLAG_ACTIVITY_NEW_TASK );

        startActivity( intent );
    }
}
```

액티비티 친화력과 FLAG_ACTIVITY_NEW_TASK 인텐트 플래그

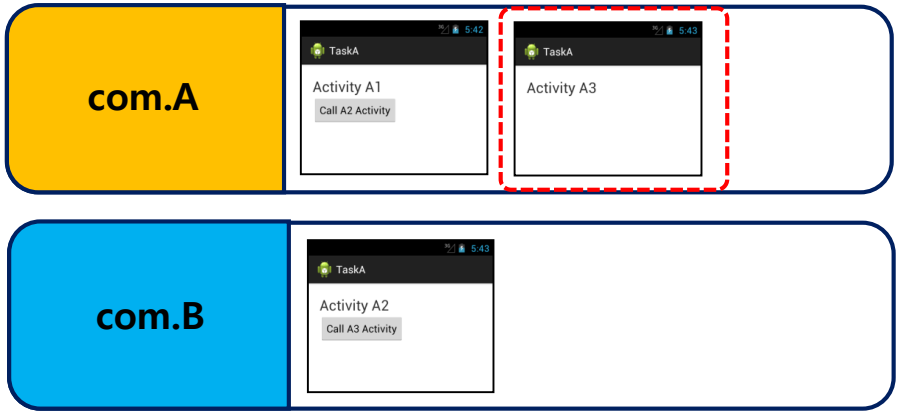


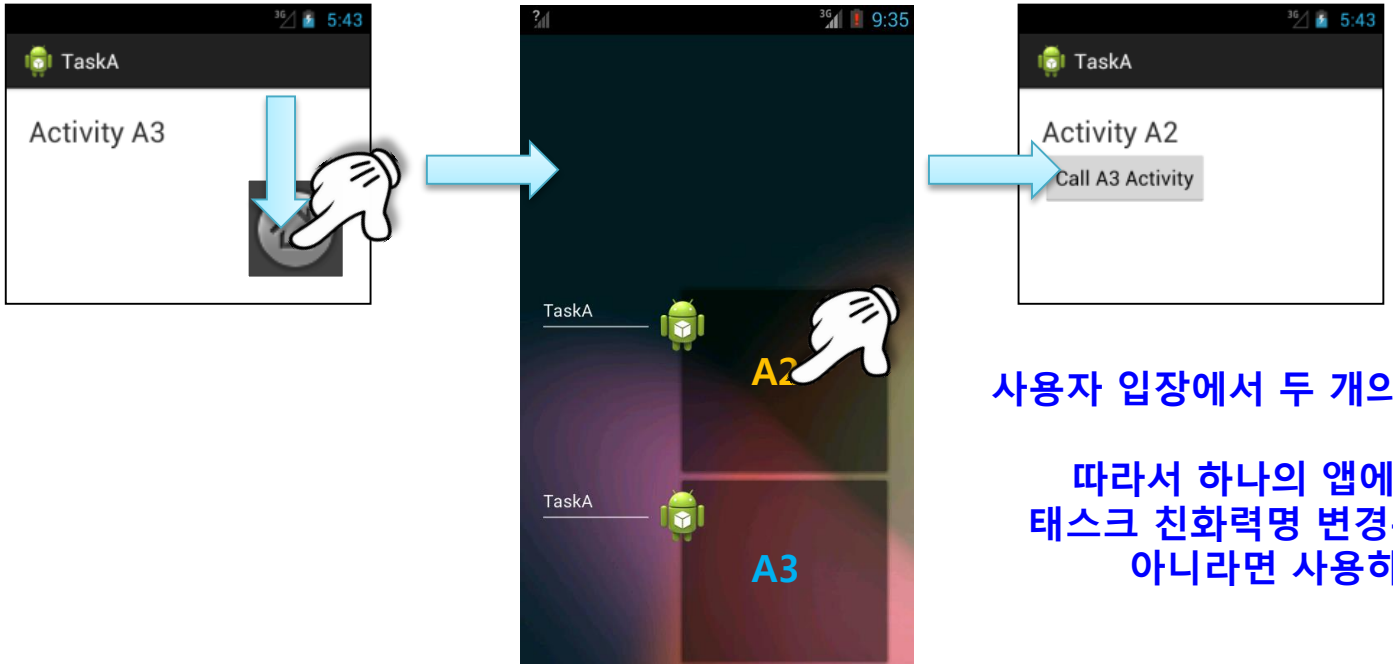


adb shell dumpsys activity activities

```
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

  Main stack:

  * TaskRecord{4201b640 #8 A com.A U 0}
    affinity=com.A
    intent={ act=android.intent.action.MAIN
             cat=[android.intent.category.LAUNCHER]
              flg=0x10200000 ...}

    * Hist #3: ActivityRecord{420aa548 u0 com.superdroid.test.TaskA/.ActivityA3}
        taskAffinity=com.A

    * Hist #2: ActivityRecord{4189f020 u0 com.superdroid.test.TaskA/.ActivityA1}
        taskAffinity=com.A

  * TaskRecord{41a55b60 #9 A com.B U 0}
    affinity=com.B
    intent={flg=0x10000000 cmp=com.superdroid.test.TaskA/.ActivityA2}

    * Hist #1: ActivityRecord{41a31220 u0 com.superdroid.test.TaskA/.ActivityA2}
        taskAffinity=com.B

  * TaskRecord{42099b90 #4 A com.android.launcher U 0}
    affinity=com.android.launcher

    * Hist #0: ActivityRecord{41aac390 u0 com.android.launcher/com.android.launcher2.Launcher}
```

■ 예제에서 태스크가 두 개로 분리되는 것은 어떤 의미가 있을까?





사용자 입장에서 두 개의 앱이 되어 버린다.

따라서 하나의 앱에서 액티비티의
태스크 친화력명 변경은 특별한 경우가
아니라면 사용하지 않는다.

1) FLAG_ACTIVITY_NEW_TASK가 필요하다면 반드시 태스크 친화력을 먼저 떠올려라

2) FLAG_ACTIVITY_NEW_TASK 플래그로 액티비티를 실행한다. 이때

실행할 액티비티의 태스크 친화력 명으로 생성된 태스크가 없다면 새로운 태스크를 생성하고
그 곳에 액티비티가 올라간다.

3) 기존에 태스크가 존재한다면 해당 태스크로 실행되는 액티비티가 올라간다.

안드로이드 앱을 개발하면서 FLAG_ACTIVITY_NEW_TASK 인텐트 플래그는 매우 빈번히 사용된다.

하지만 개발자들은 이 플래그와 태스크 친화력을 관계를 이해하지 못하는 경우가 많다.

따라서 수많은 시행착오를 겪게 된다. 꼭 이해하고 넘어가자.

■ FLAG_ACTIVITY_MULTIPLE_TASK 인텐트 플래그는 FLAG_ACTIVITY_NEW_TASK의 보조 플래그다.
따라서 FLAG_ACTIVITY_MULTIPLE_TASK 인텐트 플래그를 단독으로 사용할 수 없다.

FLAG_ACTIVITY_NEW_TASK와 FLAG_ACTIVITY_MULTIPLE_TASK를 함께 사용하여 액티비티를 실행하면 무조건 실행되는 액티비티는 무조건 새로운 태스크를 생성하고 올라간다.
즉 FLAG_ACTIVITY_NEW_TASK와 태스크 친화력의 관계를 끊고 무조건 새로운 태스크를 생성하라는 플래그다.

androidManifest.xml

```
<activity android:name=".ActivityA1" android:taskAffinity="com.A">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".ActivityA2" android:taskAffinity="com.B"/>
<activity android:name=".ActivityA3" android:taskAffinity="com.A"/>
```

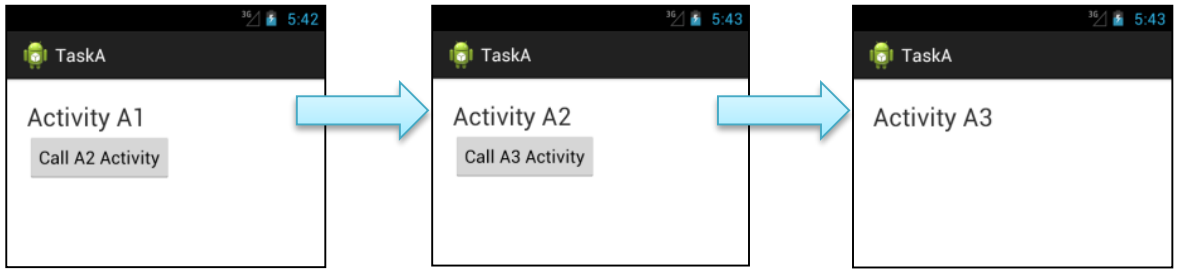
androidManifest.xml

src/ActivityA1.java

```
public class ActivityA1 extends Activity
{

    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA2.class);
        intent.addFlags( Intent.FLAG_ACTIVITY_NEW_TASK |
                         Intent.FLAG_ACTIVITY_MULTIPLE_TASK  );

        startActivity( intent );
    }
}
```

src/ActivityA2.java

```
public class ActivityA2 extends Activity
{

    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA3.class);
          intent.addFlags( Intent.FLAG_ACTIVITY_NEW_TASK |
                           Intent.FLAG_ACTIVITY_MULTIPLE_TASK  );
        startActivity( intent );
    }
}
```



adb shell dumpsys activity activities

```
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

Main stack:

  * TaskRecord{4162e198 #5 A com.superdroid.test.TaskA U 0}
    affinity=com.superdroid.test.TaskA
    * Hist #3: ActivityRecord{4162deb0 com.superdroid.test.TaskA/.ActivityA3}
        taskAffinity=com.superdroid.test.TaskA

  * TaskRecord{41683fc0 #4 A com.superdroid.test.TaskA U 0}
    affinity=com.superdroid.test.TaskA
    * Hist #2: ActivityRecord{41683cd8 com.superdroid.test.TaskA/.ActivityA2}
        taskAffinity=com.superdroid.test.TaskA

  * TaskRecord{4161b7a0 #3 A com.superdroid.test.TaskA U 0}
    affinity=com.superdroid.test.TaskA
    * Hist #1: ActivityRecord{415fdc58 com.superdroid.test.TaskA/.ActivityA1}
        taskAffinity=com.superdroid.test.TaskA
```

```
<activity android:launchMode=[ "standard"    | "singleTop"      |
                               "singleTask" | "singleInstance" ]
```

■ 액티비티 실행모드를 이용하면 액티비티의 영역별 중복 실행을 막을 수 있다.

- **standard**
  중복 허용
- **singleTop**
  탑 액티비티 위로 중복된 액티비티를 허용하지 않음
- **singleTask**
  어떤 태스크에도 중복된 액티비티를 허용하지 않음
- **singleInstance**
  어떤 태스크에도 중복된 액티비티를 허용하지 않음
  태스크내 다른 액티비티를 허용하지 않음

액티비티 실행모드 - standard

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.superdroid.test.TaskA"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="16" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity android:name=".ActivityA1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCH
            </intent-filter>
        </activity>

        <activity android:name=".ActivityA2"
                  android:exported="true" />

        <activity android:name=".ActivityA3" />

    </application>
</manifest>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.superdroid.test.TaskB"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="16" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity android:name=".ActivityB1">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

    </application>
</manifest>
```

src/ActivityA1.java

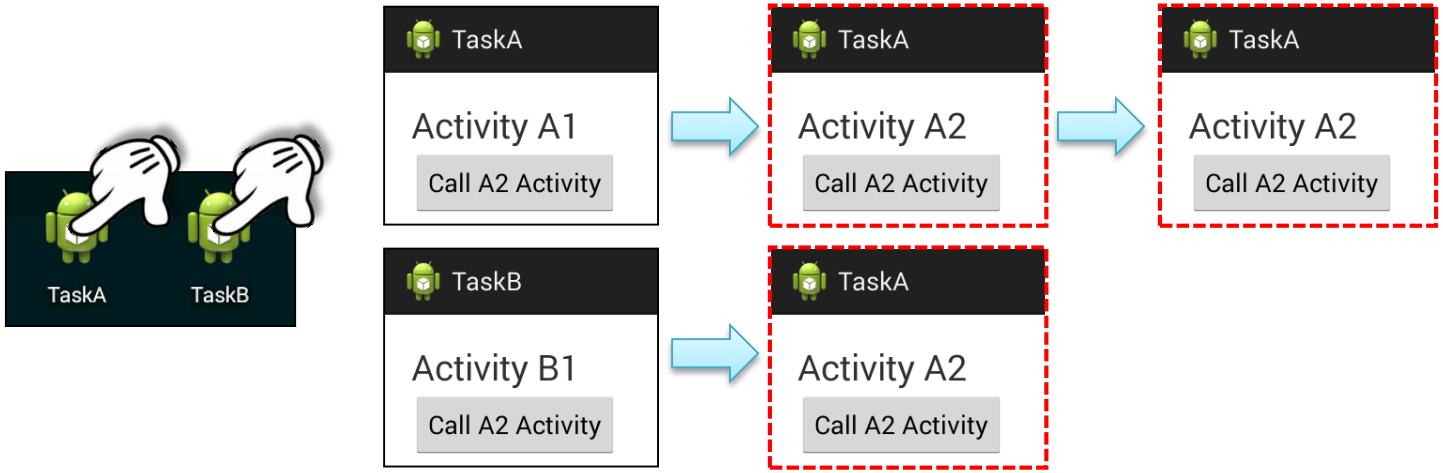
```
public class ActivityA1 extends Activity
{
    ...
    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA2.class);
        startActivity( intent );
    }
}
```

src/ActivityA2.java

```
public class ActivityA2 extends Activity
{
    ...
    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA2.class);
        startActivity( intent );
    }
}
```

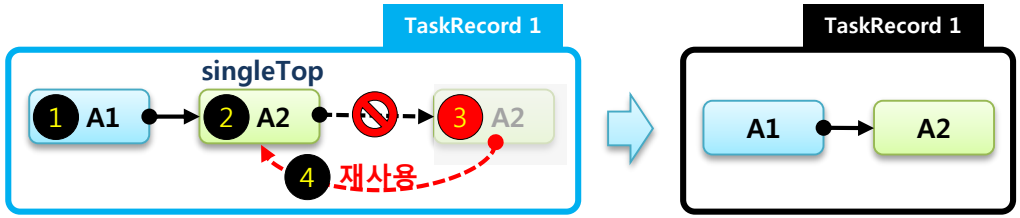
src/ActivityB1.java

```
public class ActivityB1 extends Activity
{
    @Override
    protected void onCreate( Bundle savedInstanceState )
    {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_b1 );
    }

    public void onClick ( View v )
    {
        // 다른 태스크친화력 이름을 가진 singleTask 액티비티를 실행한다.
        // ===================================================================
        Intent intent = new Intent();
        ComponentName componentName = new ComponentName(
                                "com.superdroid.test.TaskA",
                                "com.superdroid.test.TaskA.ActivityA2");
        intent.setComponent( componentName );

        startActivity( intent );
        // ===================================================================
    }
}
```



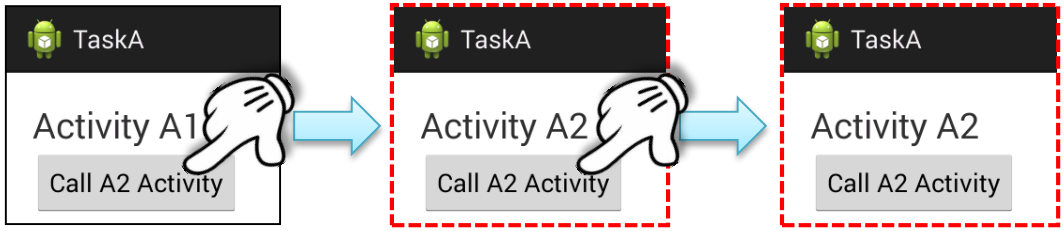
adb shell dumpsys activity activities

```
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)
  * TaskRecord{41d67d08 #8 A com.superdroid.test.TaskB U 0}
    * Hist #5: ActivityRecord{4174db70 u0 com.superdroid.test.TaskA/.ActivityA2}
    * Hist #4: ActivityRecord{4174b190 u0 com.superdroid.test.TaskB/.ActivityB1}
  * TaskRecord{41c79ce8 #4 A com.android.launcher U 0}
    * Hist #3: ActivityRecord{416f71f0 u0 com.android.launcher/com.android.launcher2.Launcher}
  * TaskRecord{420f80b8 #7 A com.superdroid.test.TaskA U 0}
    * Hist #2: ActivityRecord{41d90de8 u0 com.superdroid.test.TaskA/.ActivityA2}
    * Hist #1: ActivityRecord{41c66280 u0 com.superdroid.test.TaskA/.ActivityA2}
    * Hist #0: ActivityRecord{420e2618 u0 com.superdroid.test.TaskA/.ActivityA1}
```



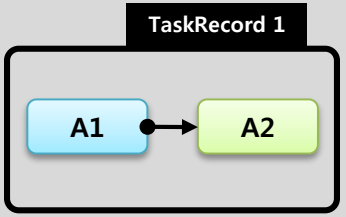
AndroidManifest.xml

```
...
<activity android:name=".ActivityA2"
          android:launchMode="singleTop" />
...
```

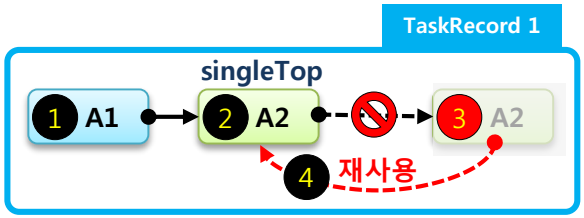


adb shell dumpsys activity activities

```
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

  * TaskRecord{41995db0 #15 A com.superdroid.test.TaskA U 0}
    * Hist #2: ActivityRecord{41d65da0 u0 com.superdroid.test.TaskA/.ActivityA2}
        ...
        fullscreen=true noDisplay=false immersive=false launchMode=1
    * Hist #1: ActivityRecord{4198bbb0 u0 com.superdroid.test.TaskA/.ActivityA1}
        ...
        fullscreen=true noDisplay=false immersive=false launchMode=0
```

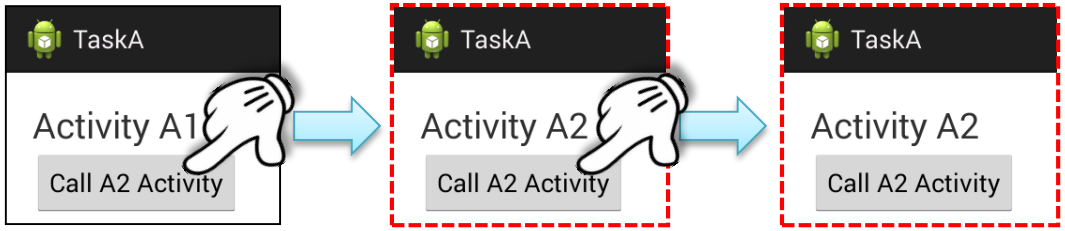


■ 재 사용의 의미 – 생명주기 변화



src/ActivityA2.java

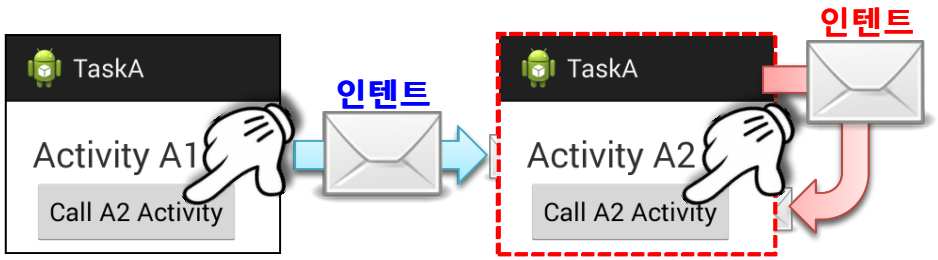
```
public class ActivityA2 extends Activity
{
    protected void onCreate( Bundle savedInstanceState )
    {
        super.onCreate( savedInstanceState );
        setContentView( R.layout.activity_a2 );
        Log.e( "superdroid", "onCreate()" );
    }

    protected void onRestart()
    {
        super.onRestart();
        Log.v( "superdroid", "onRestart()" );
    }

    protected void onStart()
    {
        super.onStart();
        Log.d( "superdroid", "onStart()" );
    }

    protected void onResume()
    {
        super.onResume();
        Log.i( "superdroid", "onResume()" );
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        Log.i( "superdroid", "onPause()" );
    }
    @Override
    protected void onStop()
    {
        super.onStop();
        Log.d( "superdroid", "onStop()" );
    }
    @Override
    protected void onDestroy()
    {
        super.onDestroy();
        Log.e( "superdroid", "onDestroy()" );
    }

    ...
}
```



● 로그 출력 결과

```
onCreate()
onStart()
onResume()
onPause()
onResume()
```

■ 재 사용의 의미 – 인텐트 재전달



● 로그 출력 결과

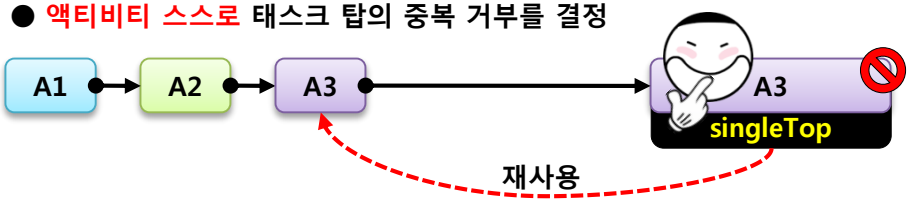
```
onCreate()
onStart()
onResume()
onPause()
onNewIntent()
onResume()
```

src/ActivityA2.java

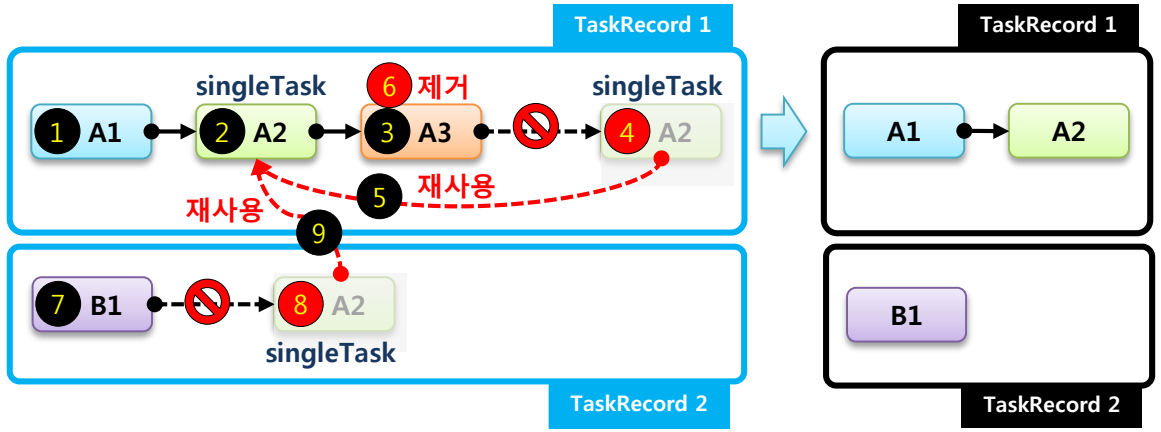
```
public class ActivityA2 extends Activity
{
    ...
    @Override
    protected void onNewIntent( Intent intent )
    {
        Log.w( "superdroid", "onNewIntent()" );

        super.onNewIntent( intent );
    }
```

■ FLAG_ACTIVITY_SINGLE_TOP 인텐트 플래그

● 액티비티 스스로 태스크 탑의 중복 거부를 결정



● 액티비티 실행하는 곳에서 태스크 탑의 중복 거부를 결정



AndroidManifest.xml

```
...
<activity android:name=".ActivityA2"
          android:launchMode="singleTask" />
...
```

src/ActivityA1.java

```
public class ActivityA1 extends Activity
{
    ...
    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA2.class);
        startActivity( intent );
    }
}
```

src/ActivityA2.java

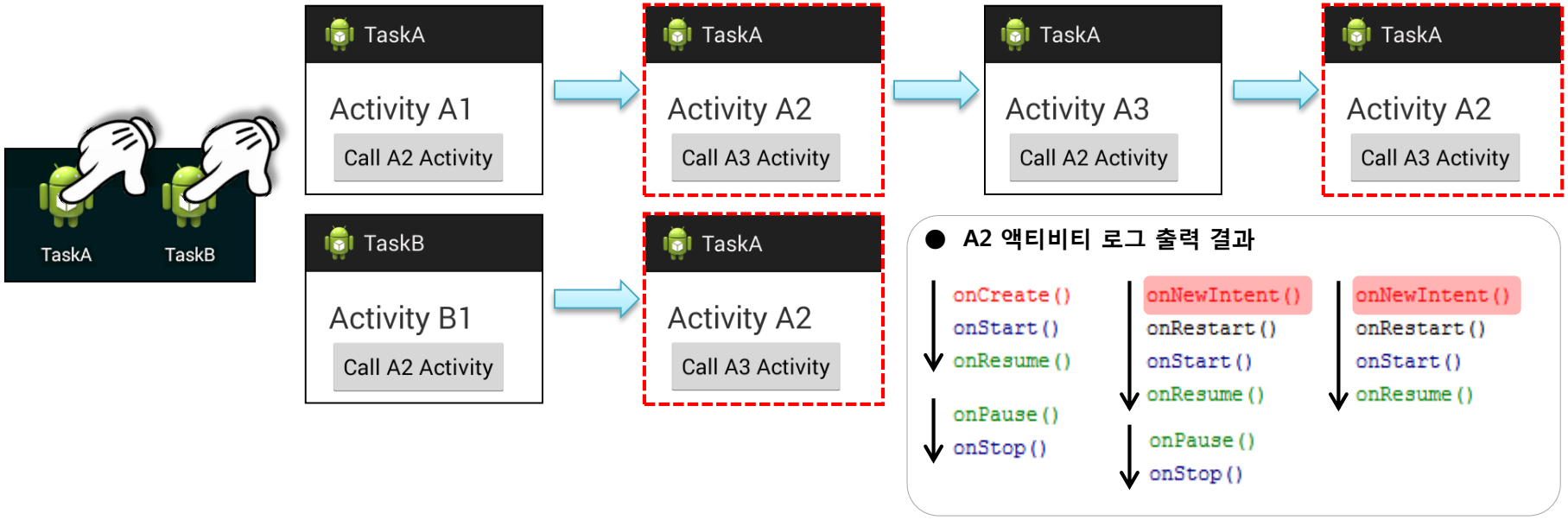
```
public class ActivityA2 extends Activity
{
    ...
    public void onClick ( View v )
    {
        Intent intent = new Intent(this, ActivityA3.class);
        startActivity( intent );
    }
}
```

src/ActivityA3.java

```
public class ActivityA3 extends Activity
{
    ...
    public void onClick ( View v )
    {
        startActivity(new Intent(this, ActivityA2.class) );
    }
}
```

src/ActivityB1.java

```
public class ActivityB1 extends Activity
{
    ...
    public void onClick ( View v )
    {
        // 다른 태스크친화력 이름을 가진 singleTask 액티비티를 실행한다.
        //
=========================================================================
        Intent intent = new Intent();
        ComponentName componentName = new ComponentName(
                                "com.superdroid.test.TaskA",
                            "com.superdroid.test.TaskA.ActivityA2");
        intent.setComponent( componentName );

        startActivity( intent );
        //
=========================================================================
    }
}
```



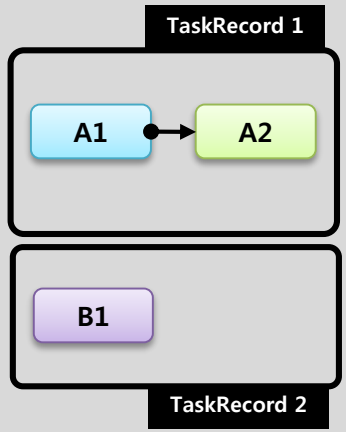
adb shell dumpsys activity activities

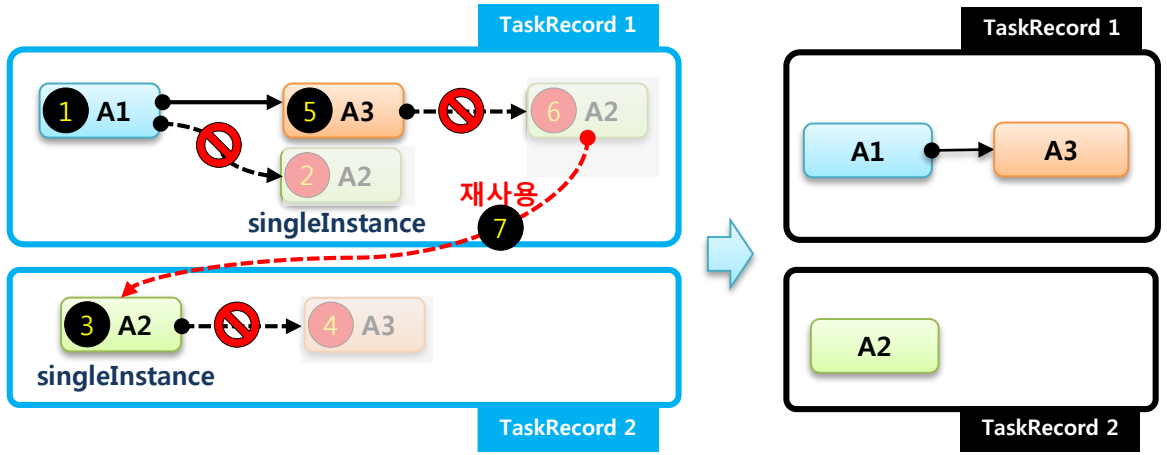
ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

Main stack:

*** TaskRecord{41472810 #4 A com.superdroid.test.TaskA U 0}**
*** Hist #2: ActivityRecord{4143c968 com.superdroid.test.TaskA/.ActivityA2}**
* Hist #1: ActivityRecord{4143c528 com.superdroid.test.TaskA/.ActivityA1}

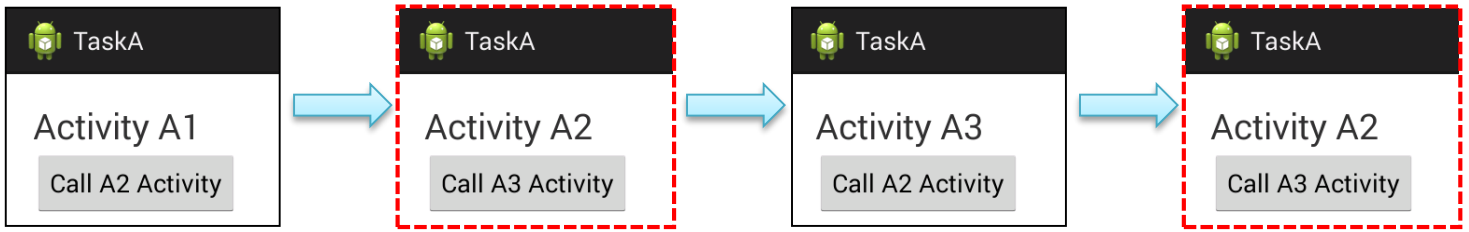
*** TaskRecord{41977810 #13 A com.superdroid.test.TaskB U 0}**
*** Hist #1: ActivityRecord{41977518 u0 com.superdroid.test.TaskB/.ActivityB1}**





AndroidManifest.xml

```
...
<activity android:name=".ActivityA1">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name=".ActivityA2"
      android:launchMode="singleInstance"/>

<activity android:name=".ActivityA3" />
...
```



● A2 액티비티 로그 출력 결과

onCreate()
onStart()
onResume()

onPause()
onStop()

onNewIntent()
onRestart()
onStart()
onResume()

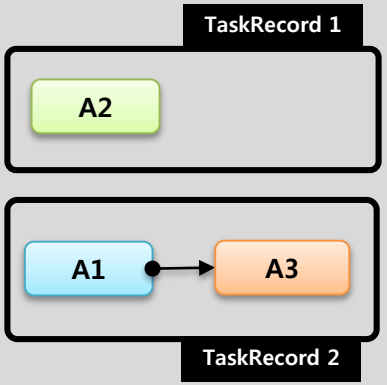
adb shell dumpsys activity activities

ACTIVITY MANAGER ACTIVITIES (dumpsys activity activities)

Main stack:

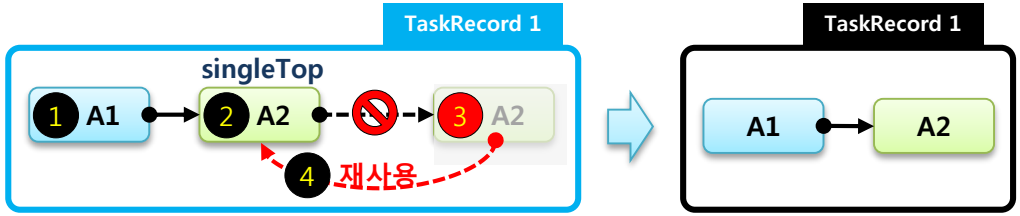
*** TaskRecord{4192f268 #18 A com.superdroid.test.TaskA U 0}**
*** Hist #3: ActivityRecord{41916e78 u0 com.superdroid.test.TaskA/.ActivityA2}**
fullscreen=true noDisplay=false immersive=false launchMode=3

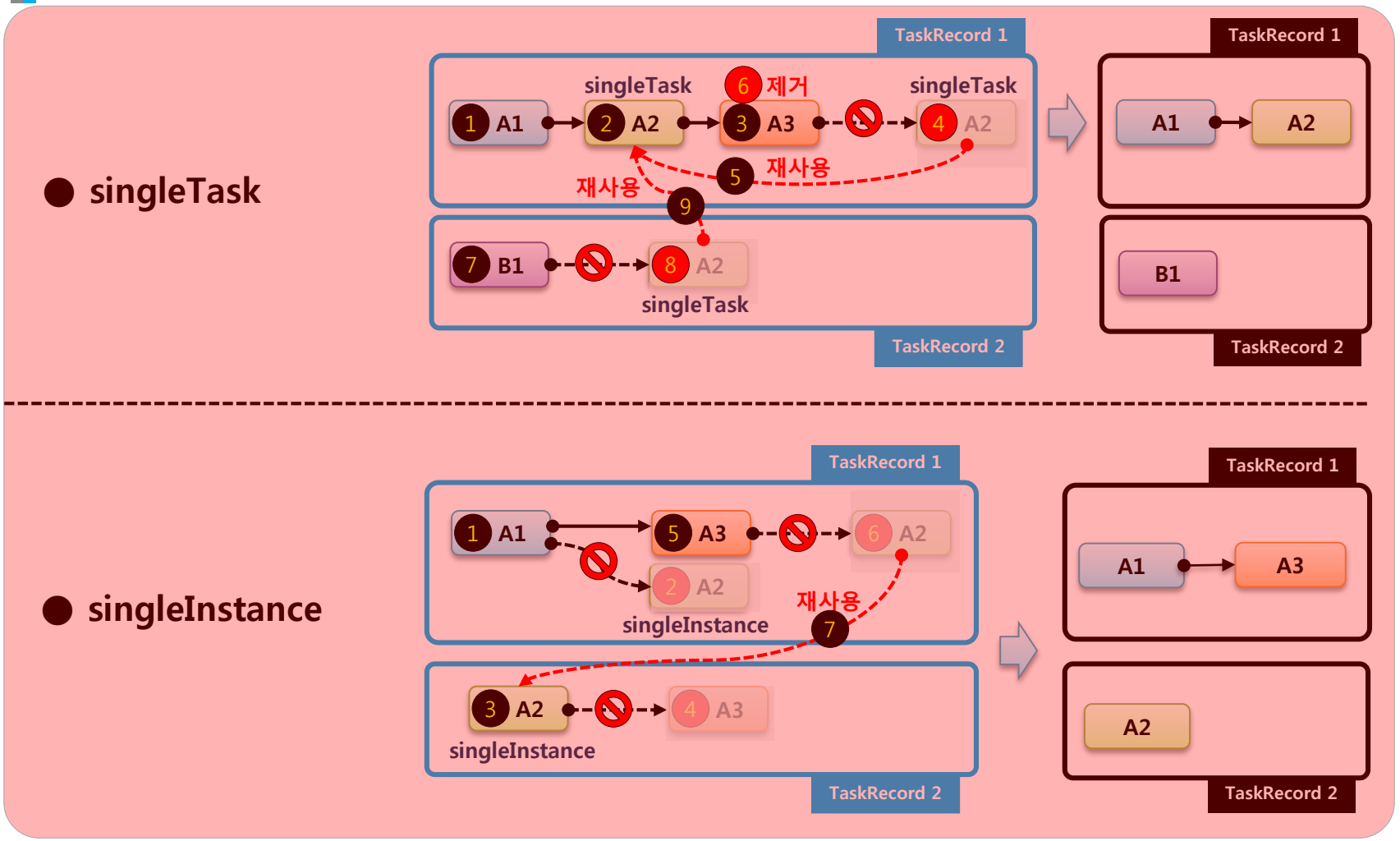
*** TaskRecord{41afdbb0 #17 A com.superdroid.test.TaskA U 0}**
*** Hist #2: ActivityRecord{41937fa0 u0 com.superdroid.test.TaskA/.ActivityA3}**
*** Hist #1: ActivityRecord{418d4ec0 u0 com.superdroid.test.TaskA/.ActivityA1}**



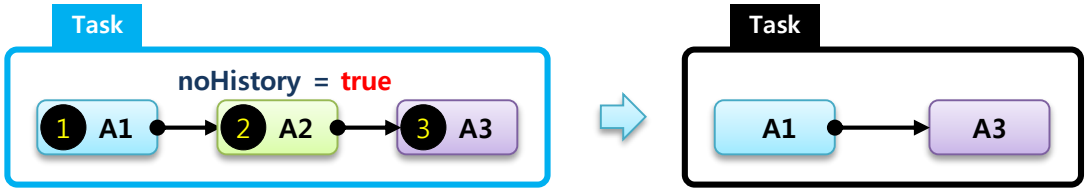
● standard

● singleTop

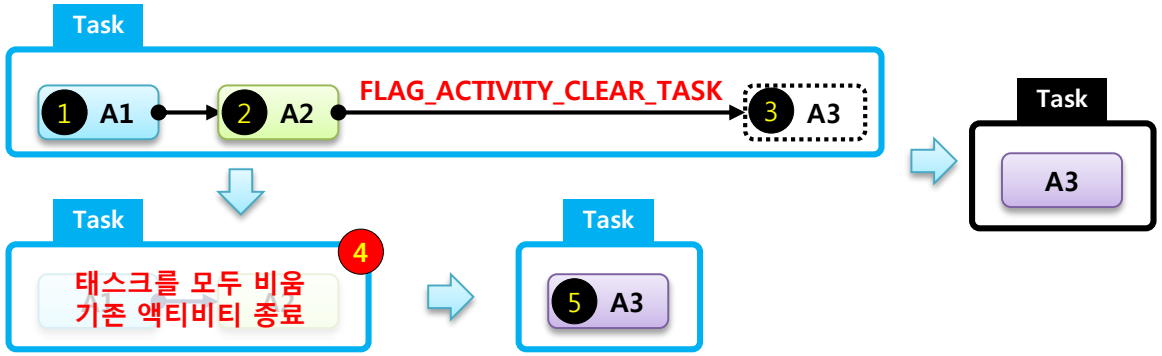



singleTask
TaskRecord 1
singleTask
6 제거
singleTask
1 A1
2 A2
3 A3
4 A2
재사용
5 재사용
9
7 B1
8 A2
singleTask
TaskRecord 2
TaskRecord 1
A1
A2
B1
TaskRecord 2
singleInstance
TaskRecord 1
1 A1
5 A3
6 A2
2 A2
singleInstance
재사용
7
3 A2
4 A3
singleInstance
TaskRecord 2
TaskRecord 1
A1
A3
A2
TaskRecord 2

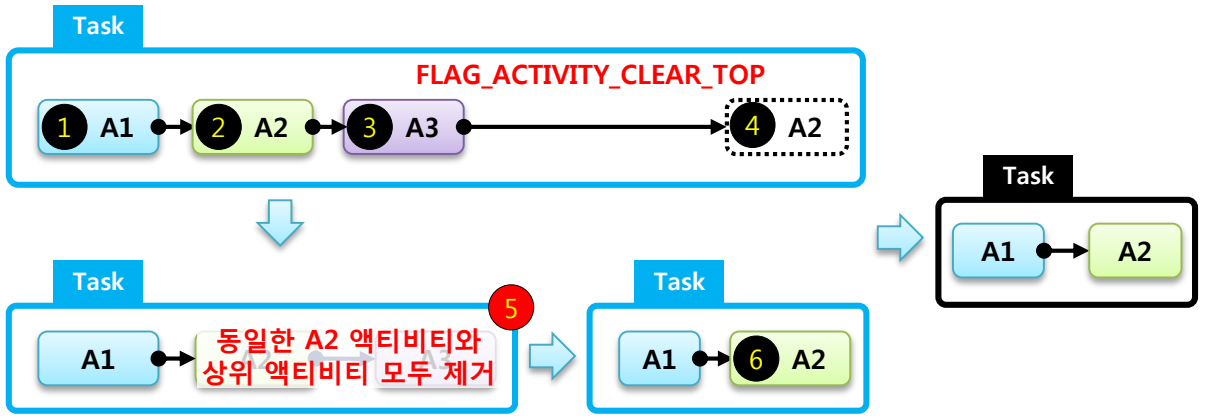
비권장 속성

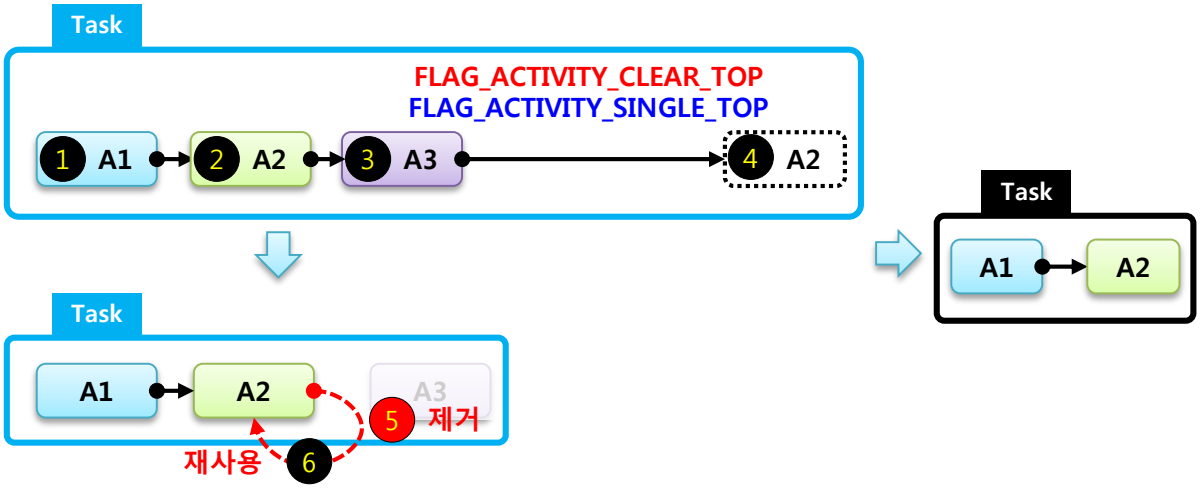

Task
noHistory = true
1 A1
2 A2
3 A3
Task
A1
A3

FLAG_ACTIVITY_CLEAR_TASK 인텐트 플래그

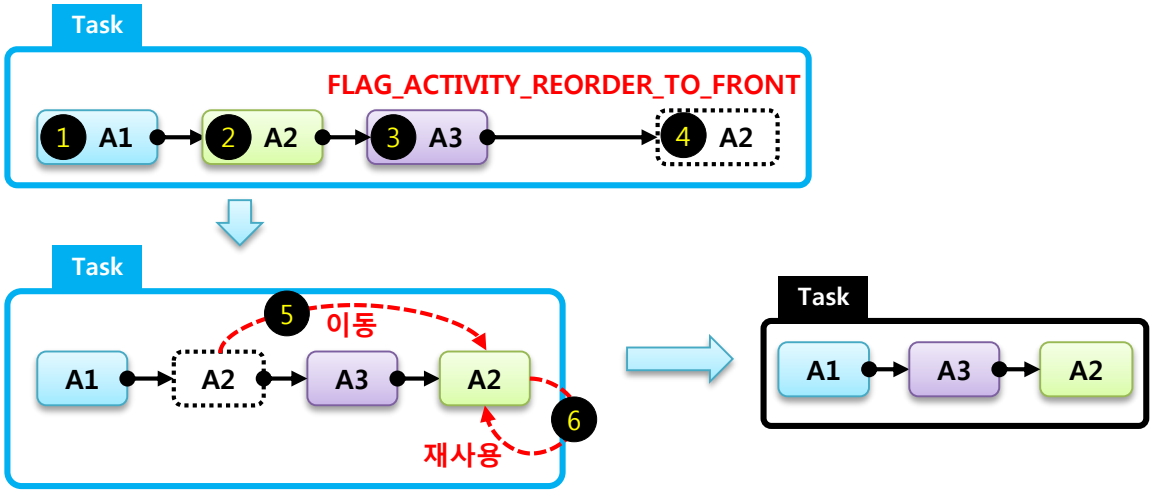


FLAG_ACTIVITY_CLEAR_TOP 인텐트 플래그





FLAG_ACTIVITY_REORDER_TO_FRONT 인텐트 플래그



finishOnCloseSystemDialogs = "true"

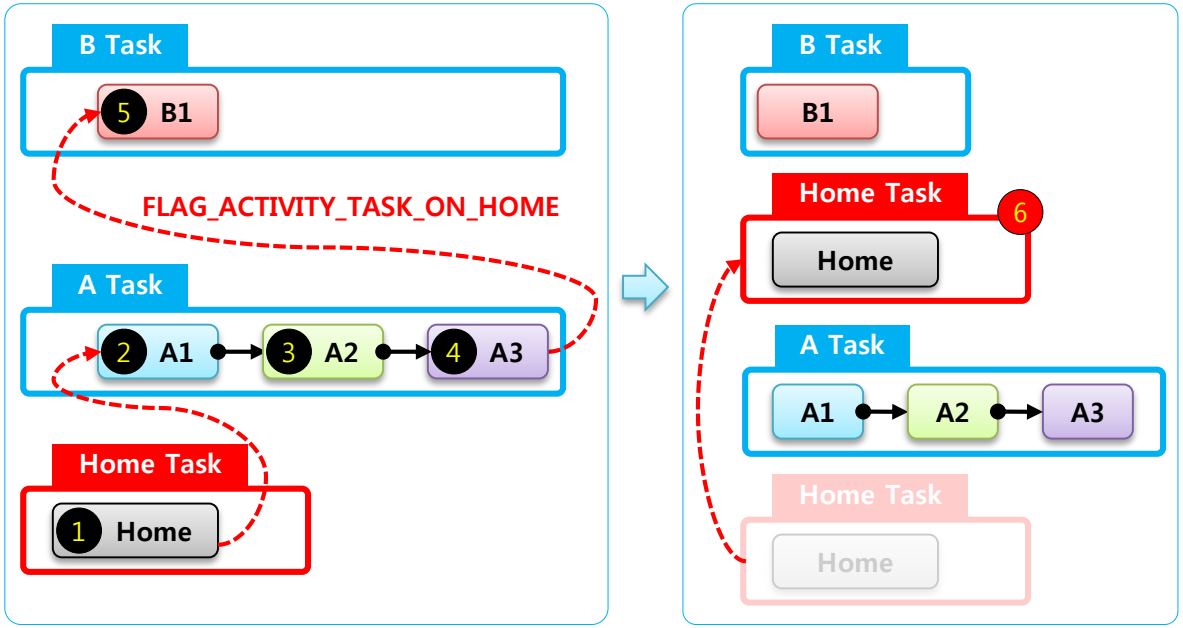
시스템 다이얼로그 종류는 아래와 같다.

- 홈 키를 눌렀을 때 실행되는 홈 화면
- 홈 키를 길게 눌렀을 때 실행되는 최근 실행 앱 리스트 다이얼로그
- 화면이 꺼지고 전원 키를 눌렀을 때 실행되는 잠금 화면
- 전원 키를 길게 눌렀을 때 실행되는 종료 다이얼로그
- 전화가 걸려왔을 때 실행되는 통화 화면
- 메뉴 키를 길게 눌렀을 때 실행되는 각종 검색 화면
  (검색 화면이 실행되지 않고 아무런 반응이 없어도 종료된다)
- 기타 등등

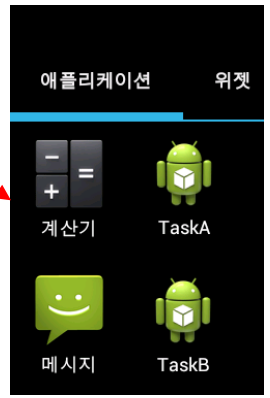
이 속성은 거의 사용되지 않는다.

또한 사용자 의도가 아니라면 어떤 경우에도 액티비티가 자동으로 종료되는 것은 바람직하지 않다.

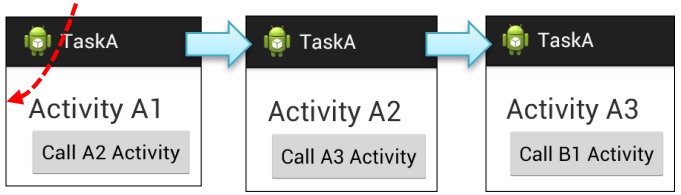
그러므로 해당 액티비티 속성은 꼭 필요한 상황에만 사용하자.



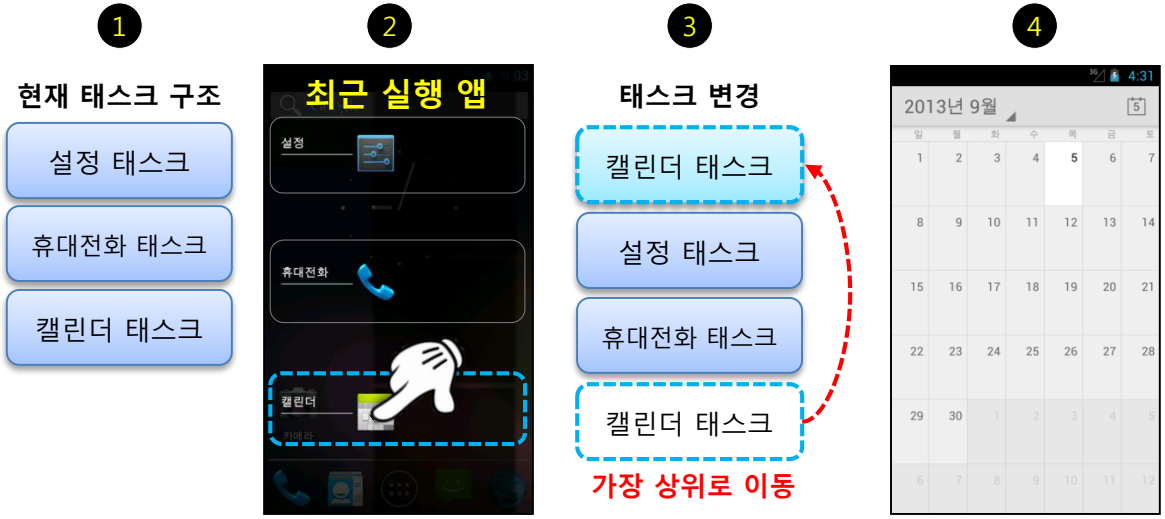
앱이 끝난 느낌을 준다.



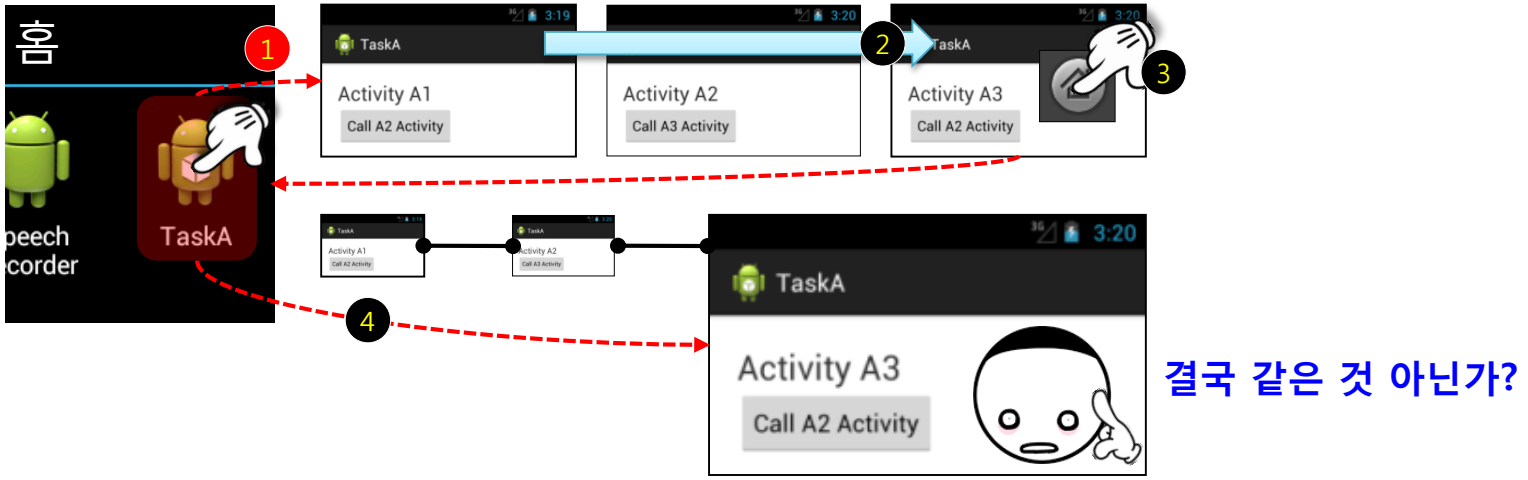
앱이 계속 진행되는 느낌을 준다.



■ 최근 실행 앱 리스트에서 앱을 실행할 때



■ 홈에서 앱을 실행할 때

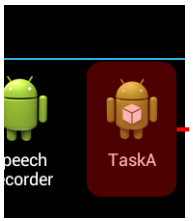


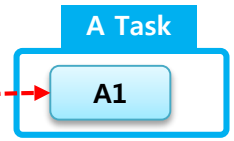
결국 같은 것 아닌가?

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그

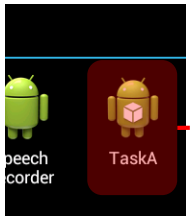
■ 홈에서 특정 앱을 실행하는 인텐트

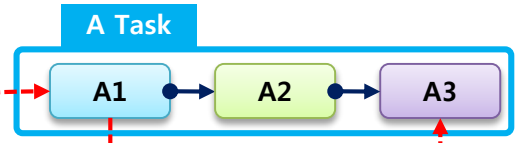
```
Intent intent = new Intent();

// ① 실행될 패키지 액티비티를 명시적으로 지정
intent.setClassName( "com.superdroid.test.TaskA",
                     "com.superdroid.test.TaskA.ActivityA1");
// ② 해당 태스크의 가장 탑 액티비티를 복귀하는 역할
intent.setAction( Intent.ACTION_MAIN );
intent.addCategory( Intent.CATEGORY_LAUNCHER );
// ③ 해당 패키지의 태스크에서 실행되고, 태스크 내에 정리해야 할 액티비티가
//    존재한다면 정리하는 역할
intent.addFlags( Intent.FLAG_ACTIVITY_NEW_TASK );
intent.addFlags( Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED );
startActivity( intent );
```





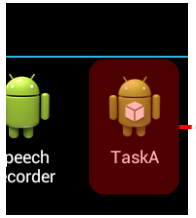
명시적 인텐트로 MAIN 액션과 LAUNCHER 카테고리가 존재하는 A1 액티비티 실행



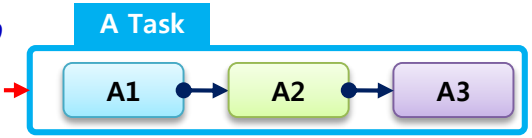


이미 태스크에 MAIN 액션과 LAUNCHER 카테고리를 가진 액티비티가 존재하면 기존 태스크의 탑 액티비티가 활성화

그렇다면 FLAG_ACTIVITY_RESET_TASK_IF_NEEDED로 태스크 내 어떤 액티비티가 정리될까?

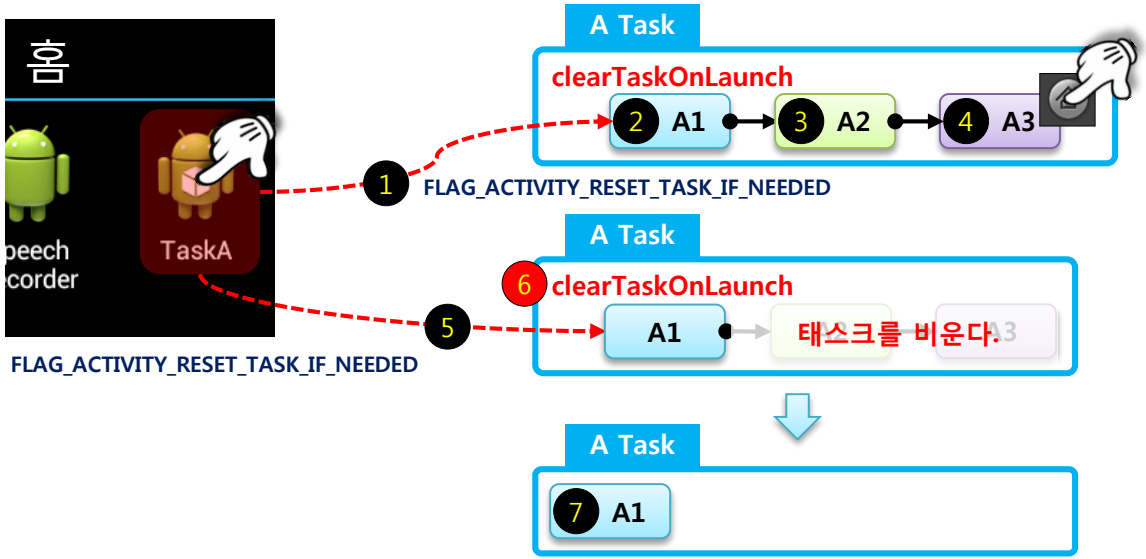


FLAG_ACTIVITY_RESET_TASK_IF_NEEDED



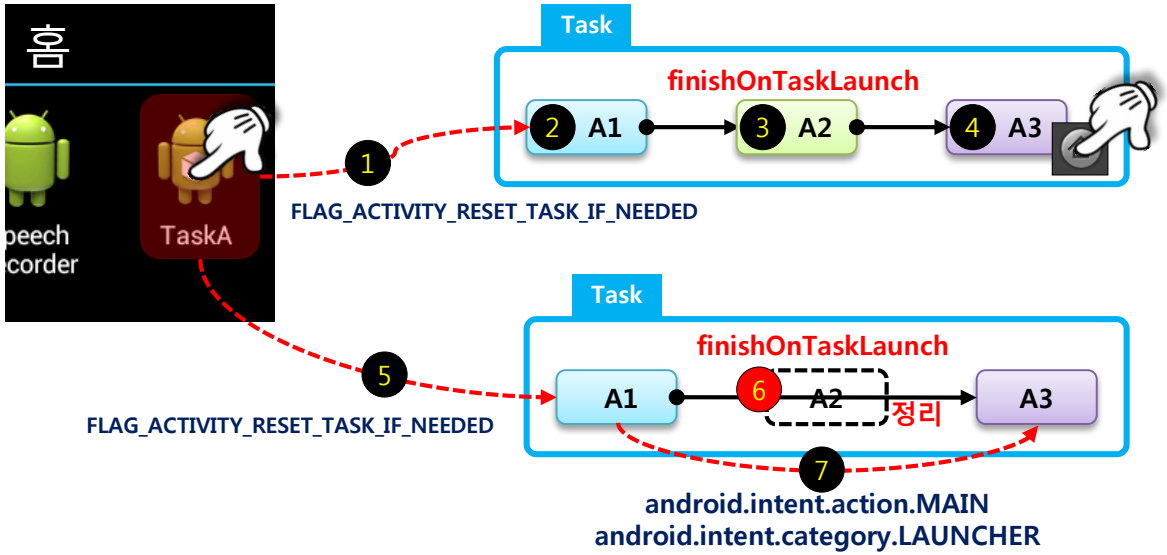
태스크를 정리한다.

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그 – clearTaskOnLaunch 액티비티 속성



clearTaskOnLaunch 속성은 루트 액티비티에만 설정할 수 있다.

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그 – finishOnTaskLaunch 액티비티 속성

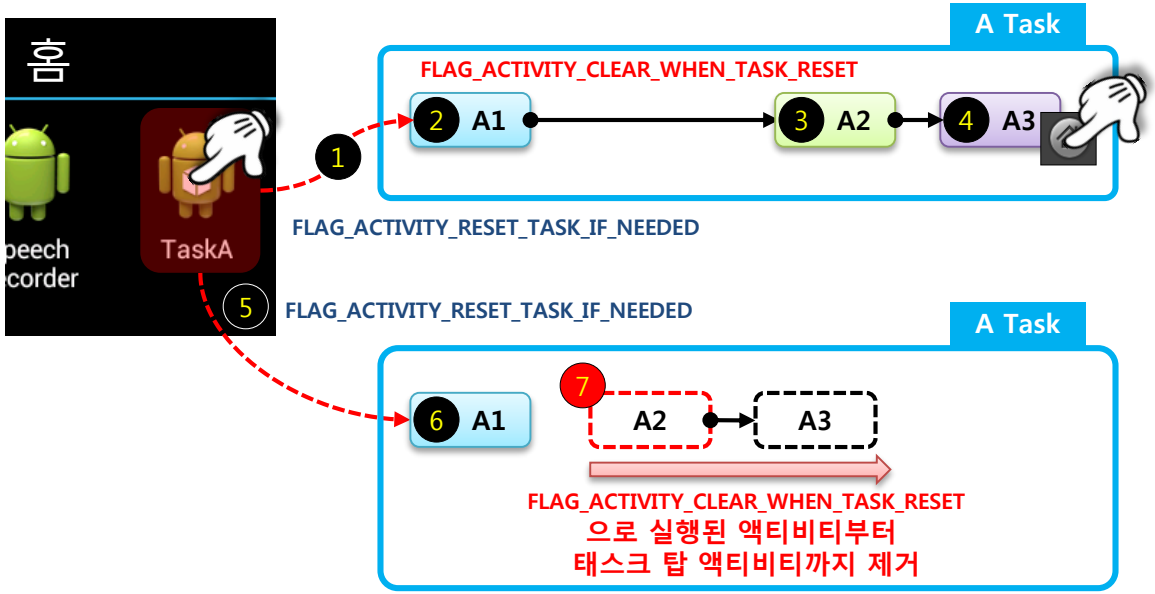


finishOnTaskLaunch 속성은 noHistory와 비슷해 보이다.

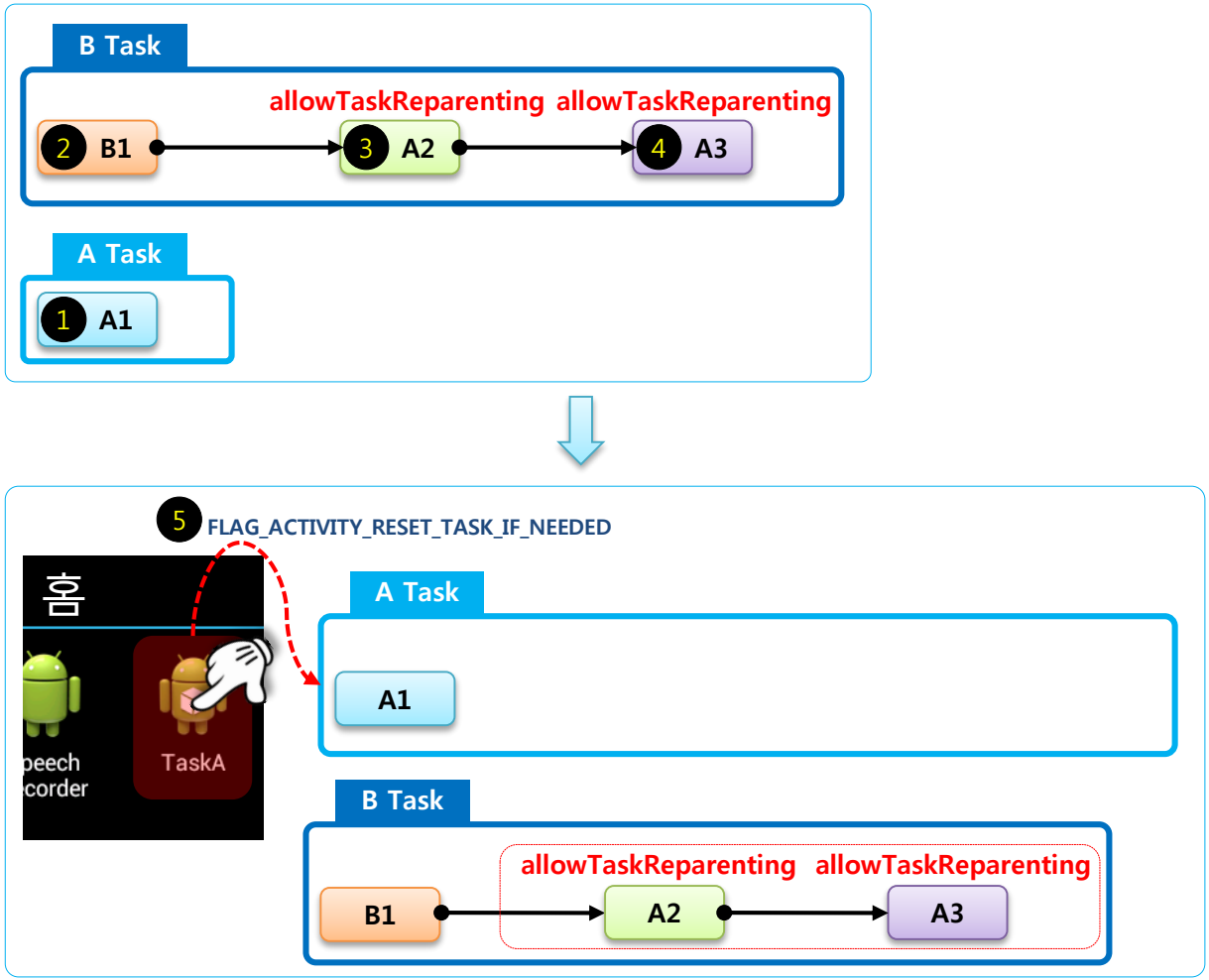
**루트 액티비티는 적용되지 않고,
FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그에 의해
정리된다는 점이 다르다.**

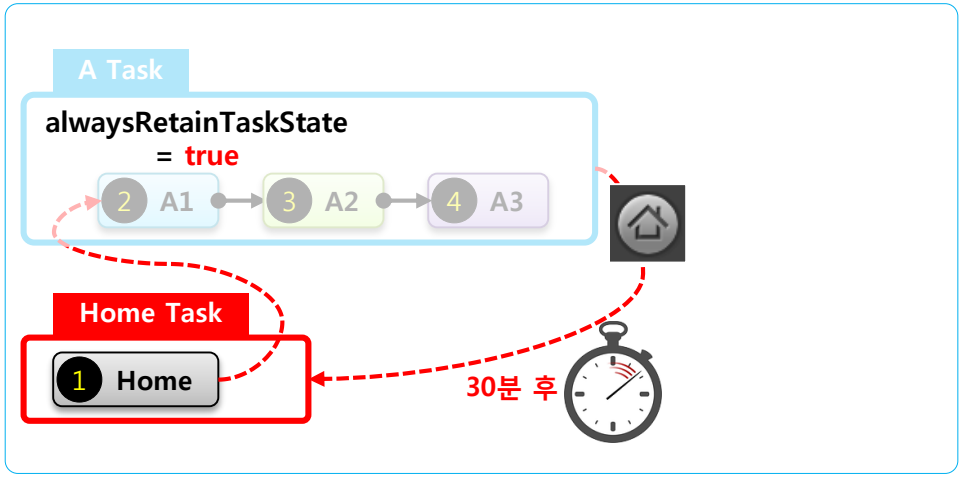
**참고로 태스크 내에 FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 플래그로 정리되는
액티비티 중 루트 액티비티는 제외다.
그 이유는 홈에서 루트 액티비티가 진입점이기 때문이다.**

FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그 – FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET 인텐트 플래그



FLAG_ACTIVITY_RESET_TASK_IF_NEEDED 인텐트 플래그 - allowTaskReparenting 액티비티 속성





안드로이드 프레임워크 API ~10 : ActivityStack.java

```
...
// How long until we reset a task when the user returns to it.  Currently
// 30 minutes.
static final long ACTIVITY_INACTIVE_RESET_TIME = 1000*60*30;
...
```

안드로이드 프레임워크 API 11~ : ActivityStack.java

```
...
// How long until we reset a task when the user returns to it.  Currently
// disabled.
static final long ACTIVITY_INACTIVE_RESET_TIME = 0;
...
```