

15.1 액티비티 생명주기

15.1.1. 생명주기

활성 상태(activity running)', '일시 정지 상태(pasue)', 비활성 상태(stop)'

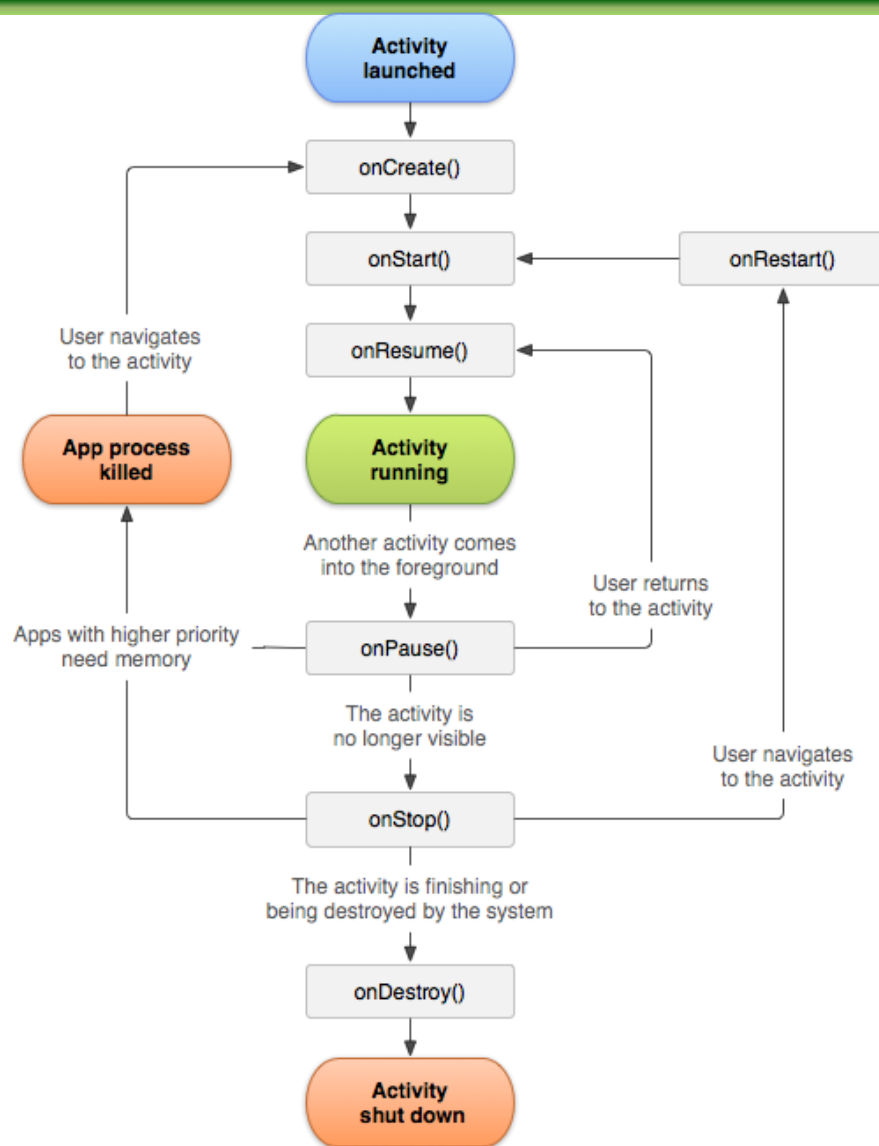
- 활성 상태(activity running): 현재 액티비티가 화면을 점유하여 출력되고 있으며 사용자 이벤트 처리가 정상으로 처리되는 상태
- 일시 정지 상태(pasue): 현재 액티비티가 일시적으로 사용이 불가능한 상태
- 비활성 상태(stop): 현재 액티비티가 다른 액티비티로 인해 화면이 완벽하게 가려진 상태

활성 상태

- onCreate() → onStart () → onResume() 함수가 호출



15.1 액티비티 생명주기



15.1 액티비티 생명주기

- setContentView()는 이전 화면을 지우면서 새로운 내용을 출력하는 함수
- 기본 화면 을 지우지 않고 그 위에 함께 출력하려면 addContentView() 함수를 이용

```
protected void onResume() {  
    super.onResume();  
    setContentView(R.layout.activity_main);  
  
    ImageView imageView=new ImageView(this);  
    //...  
    addContentView(imageView, params);  
}
```

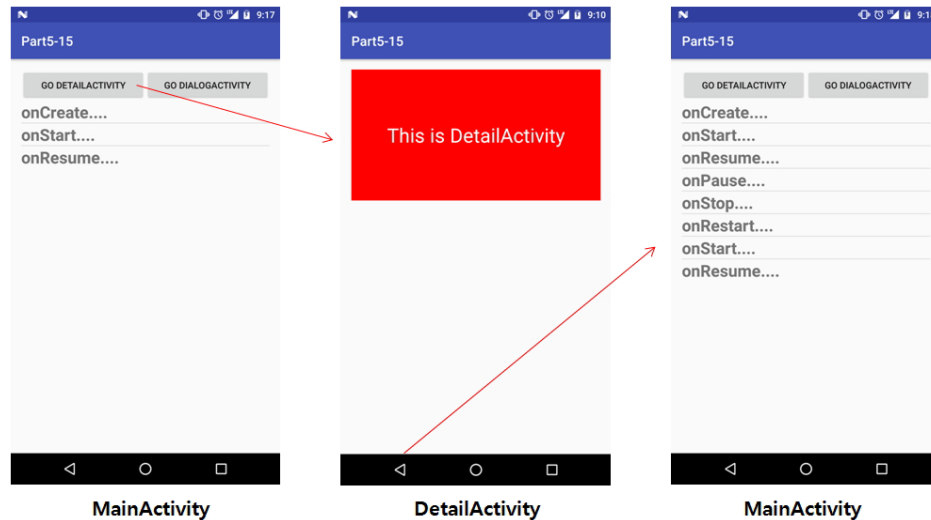


- 일시 정지 상태
- 포커스를 잃은 상태
- 애니메이션이나 CPU 소비를 야기할 수 있는 기타 지속적인 작업 정지
- GPS와 같은 센서값 수신, 서버 네트워킹 등 액티비티가 일시 정지된 동안 불필요한 동작 정지



15.1 액티비티 생명주기

- 비활성 상태
- 다른 액티비티로 인해 화면이 완전히 가려진 상태

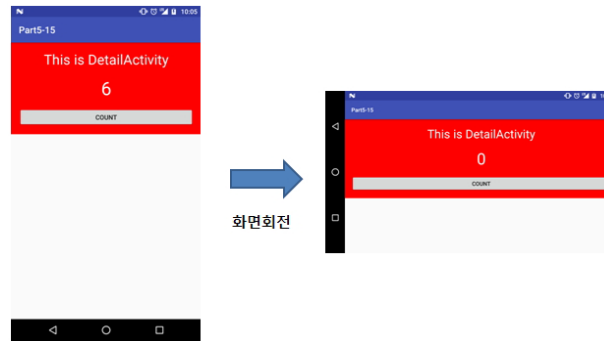


15.1.2. 액티비티 상태 저장

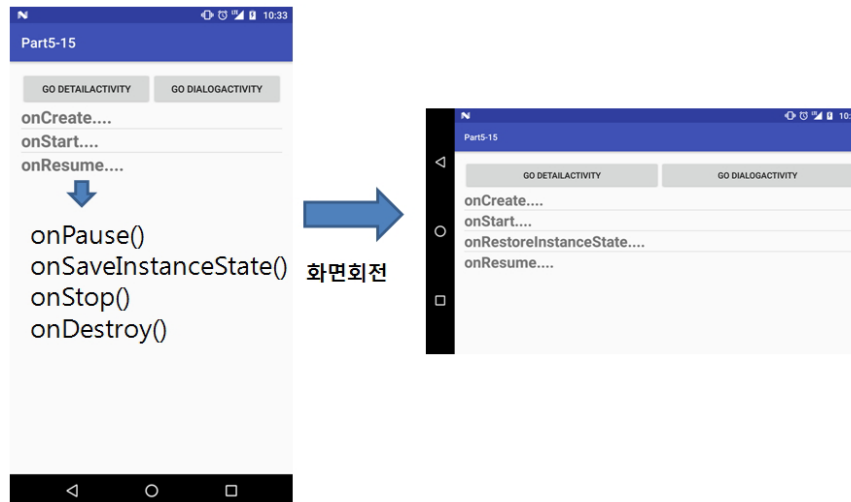
- 액티비티가 종료되더라도 계속 유지했다가 다시 액티비티가 실행될 때 그대로 이용
- 유실되는 데이터를 저장했다가 가져와야 하는 대표적인 예가 화면 회전



15.1 액티비티 생명주기



- onPause() → onSaveInstanceState() → onStop() → onDestroy()
- onCreate() → onStart() → onRestoreInstanceState() → onResume()



15.1 액티비티 생명주기

- 데이터를 저장

```
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
    outState.putString("data1", "hello");  
    outState.putInt("data2", 100);  
}
```

- Bundle은 컴포넌트의 데이터를 저장하기 위한 일종의 Map 객체
- 데이터 획득
- onCreate()와 onRestoreInstanceState ()

```
protected void onRestoreInstanceState(Bundle savedInstanceState) {  
    super.onRestoreInstanceState(savedInstanceState);  
    String data1=savedInstanceState.getString("data1");  
    int data2=savedInstanceState.getInt("data2");  
}
```



15.1 액티비티 생명주기

- onSaveInstanceState (), onRestoreInstanceState () 함수들은 API Level 21 버전부터 매개변수가 두 개인 함수가 추가

```
public void onSaveInstanceState(Bundle outState, PersistableBundle outPersistentState) {  
    super.onSaveInstanceState(outState, outPersistentState);  
    //...  
}  
  
@Override  
public void onRestoreInstanceState(Bundle savedInstanceState, PersistableBundle persistentState) {  
    super.onRestoreInstanceState(savedInstanceState, persistentState);  
    //...  
}
```

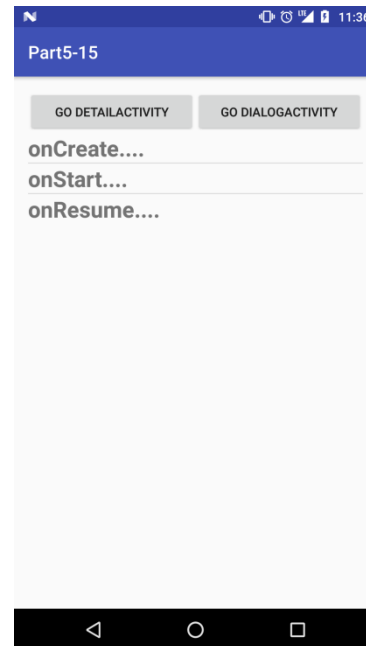
- 시스템 내부적으로 화면 회전처럼 액티비티가 종료되어야 할 때와 화면 전환처럼 종료되지 않아도 될 때를 구분
- 매개변수가 두 개인 onRestoreInstanceState () 함수는 Bundle에 저장된 데이터가 없으면 호출 안 됨



Step by Step 15-1 – Activity LifeCycle

Activity LifeCycle 테스트

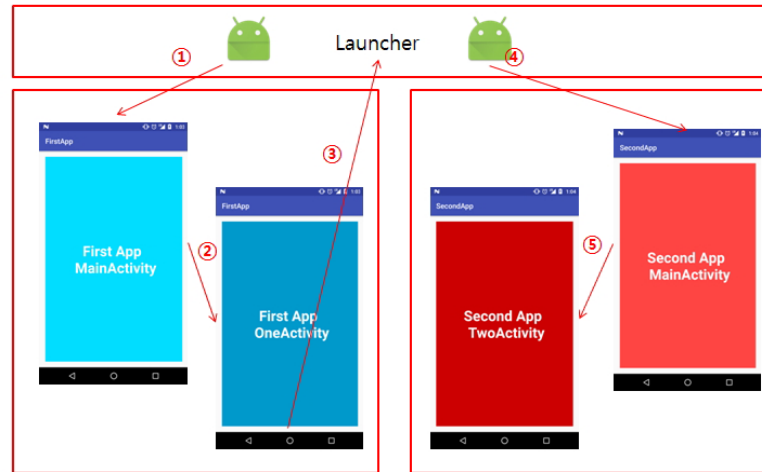
1. Module 생성
2. Activity 추가
3. 파일 복사
4. AndroidManifest.xml 작성
5. MainActivity 추가
6. 실행



15.2 태스크 관리

15.2.1. 시스템의 태스크 관리

- 태스크는 일종의 앱 실행을 위한 정보 저장공간
- 프로세스를 앱의 물리적인 실행 단위라고 표현하고, 태스크는 앱의 논리적인 실행 단위라고 표현



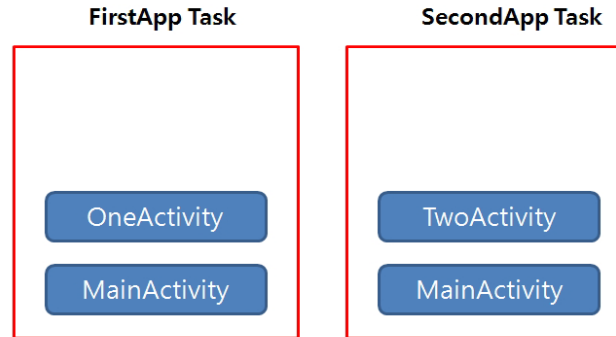
- 프로세스는 두 개가 실행

Devices		
Name		
nexus_5x-00ca488e71ca4457	Online	
com.example.firstapp	31679	
com.example.secondapp	32112	

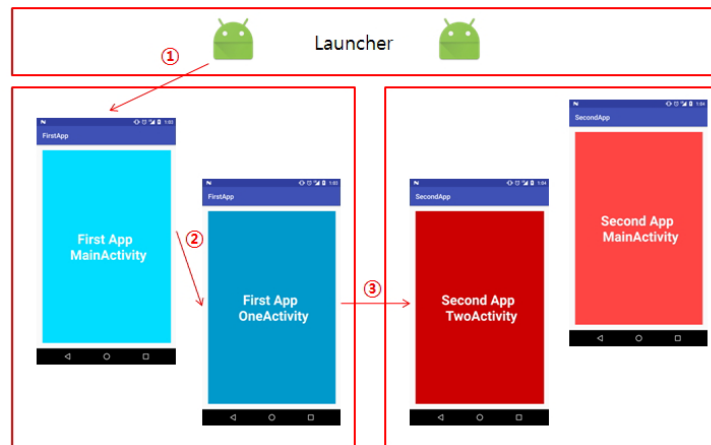


15.2 태스크 관리

- 태스크도 두개



- 두 앱 간의 연동



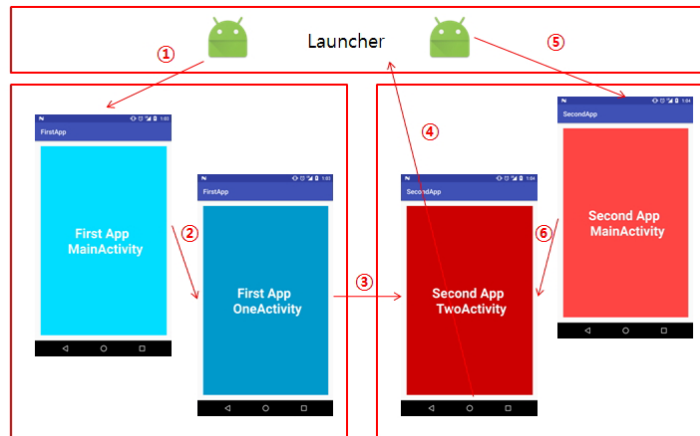
15.2 태스크 관리

두 개의 프로세스가 실행 태스크 목록은 하나

FirstApp Task

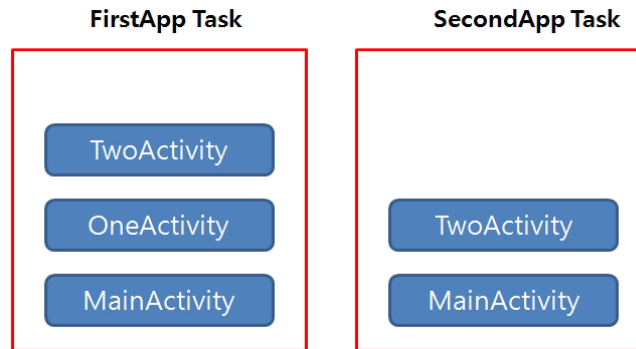


- 하나의 Activity가 두 번 실행



15.2 태스크 관리

- 액티비티 클래스는 인텐트가 발생하면 매번 객체가 생성



15.2.2. 태스크 제어 방법 1 : launchMode 속성 설정

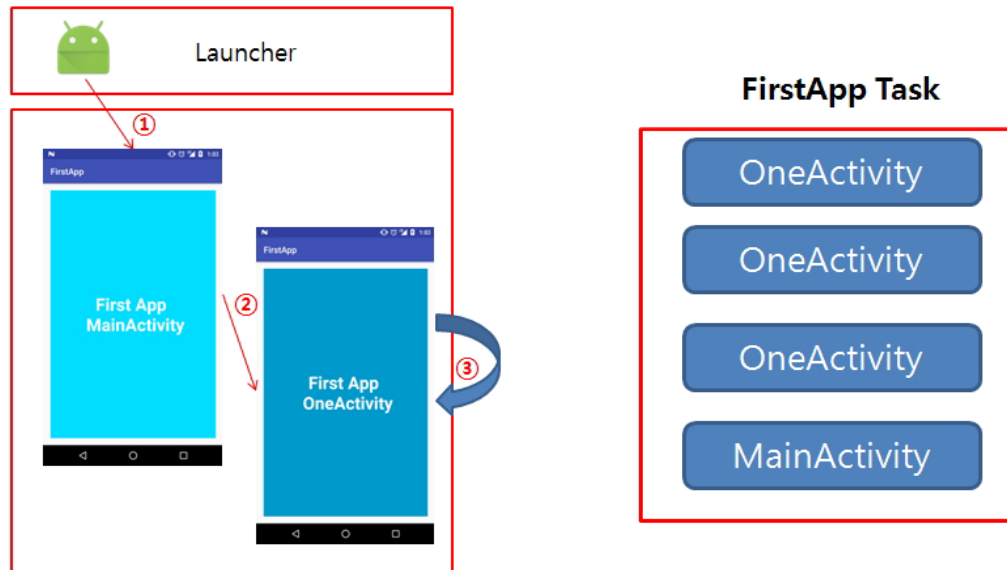
- 액티비티를 등록하는 AndroidManifest.xml 파일의 설정
- launchMode 속성값으로는 standard, singleTop, singleTask, singleInstance를 지정

```
<activity android:name=".OneActivity" android:launchMode="standard">
```



15.2 태스크 관리

- standard
- 기본값
- 인텐트가 발생하면 매번 액티비티를 생성하고, 태스크 목록에 반복해서 올리겠다는 의미



15.2 태스크 관리

- singleTop

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

- 액티비티가 태스크의 최상단에 있으면 다시 생성하지 않겠다는 의미



삼성카드 SMS



삼성카드 SMS



현대카드 SMS

standard

ReadSMSActivity
현대카드

ReadSMSActivity
삼성카드

singleTop

ReadSMSActivity
현대카드

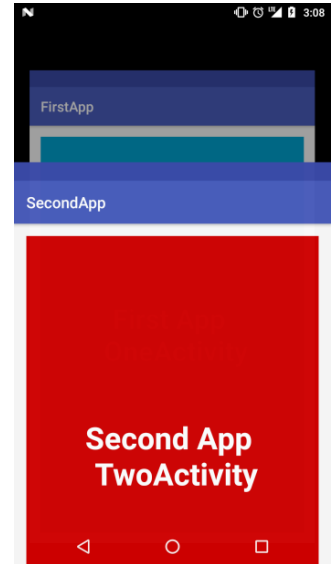
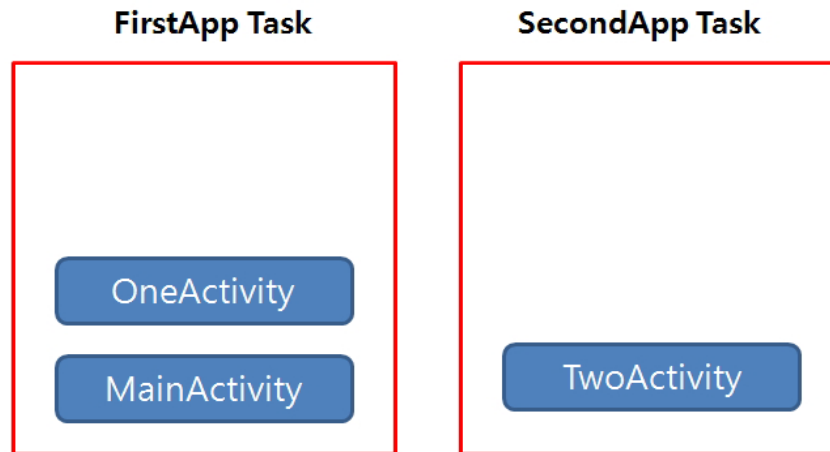
- `onNewIntent()`

```
protected void onNewIntent(Intent intent) {  
    super.onNewIntent(intent);  
    //내용 변경}
```



15.2 태스크 관리

- singleTask
- 새 로운 태스크 목록을 만들어 그곳에 액티비티 정보를 저장



15.2 태스크 관리

- `singleInstance`
- `singleInstance`가 설정된 액티비티 혼자 하나의 태스크를 차지하게 되어 이후에 수행되는 컴포넌트들이 다시 다른 태스크에 쌓이게 된다



15.2 태스크 관리

15.2.3. 태스크 제어 방법 2 : Intent 플래그 이용

- 자바 코드에서 인텐트 발생 시점에 인텐트의 플래그로 제어

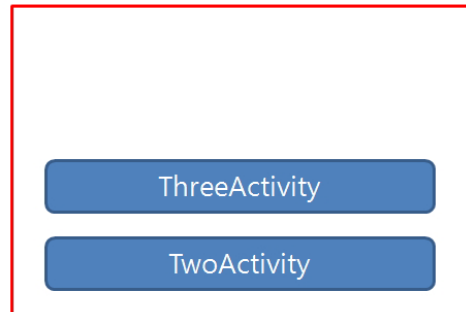
```
Intent intent=new Intent(MainActivity.this, TwoActivity.class);  
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  
startActivity(intent);
```

- FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET: (Deprecated) 태스크 목록을 리셋합니다. 즉, 이전 액티비티 정보를 삭제합니다.
- FLAG_ACTIVITY_NEW_DOCUMENT: (API Level 21)
FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET과 같음

FLAG_ACTIVITY_NEW_DOCUMENT

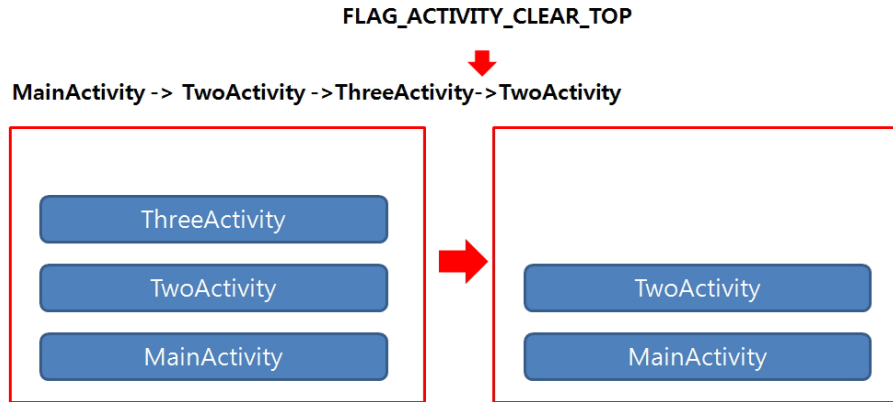


MainActivity -> TwoActivity -> ThreeActivity

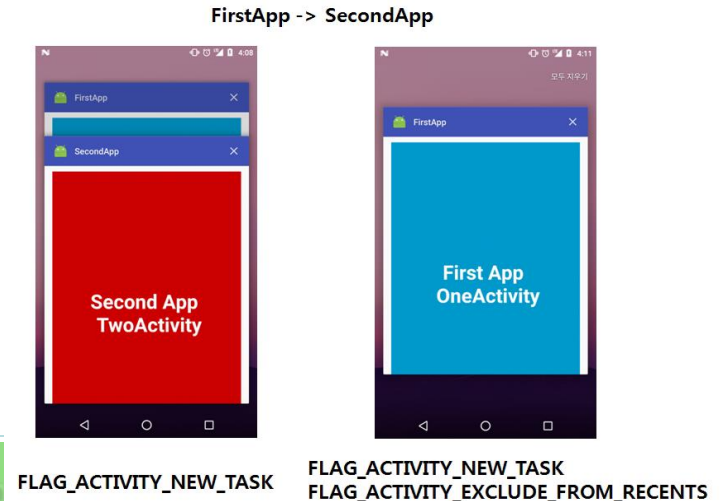


15.2 태스크 관리

- FLAG_ACTIVITY_CLEAR_TOP: 실행되는 액티비티의 상단 태스크 정보 삭제

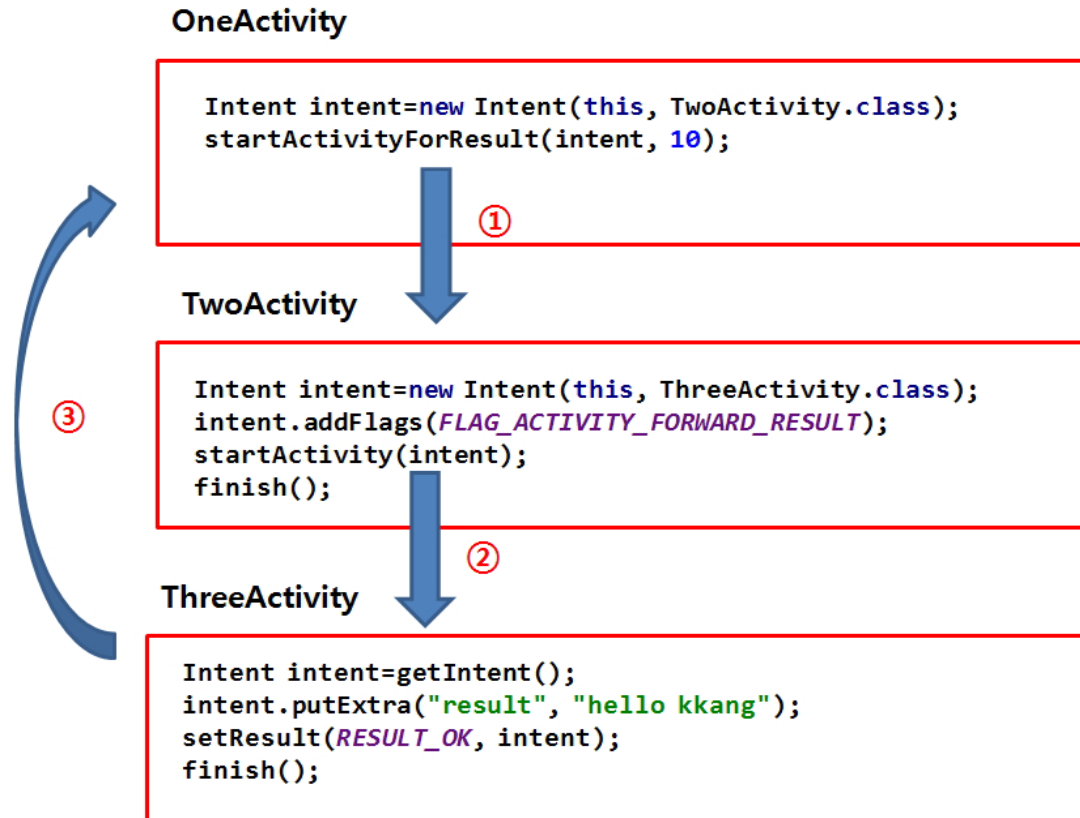


- FLAG_ACTIVITY_NEW_TASK: 새로운 태스크 목록을 만들어 액티비티 정보
- FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS: FLAG_ACTIVITY_NEW_TASK와 함께 사용되며 최근 실행한 앱 목록에 새로운 태스크의 앱이 나오지 않게 설정



15.2 태스크 관리

- FLAG_ACTIVITY_FORWARD_RESULT: 인텐트 실행 결과를 이전 액티비티에 전달



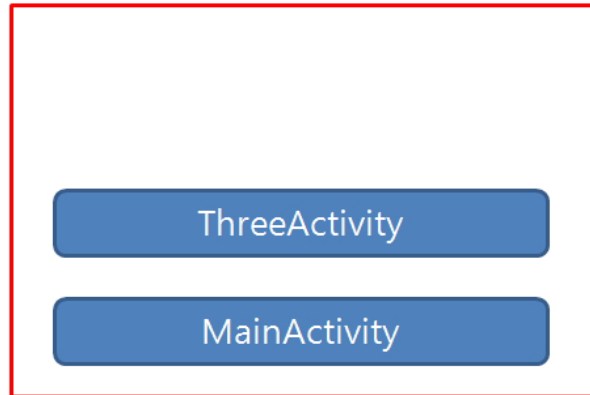
15.2 태스크 관리

- FLAG_ACTIVITY_NO_HISTORY: 액티비티를 실행하면서 태스크 목록에 저장되지 않게 설정

FLAG_ACTIVITY_CLEAR_TOP



MainActivity -> TwoActivity -> ThreeActivity



- FLAG_ACTIVITY_REORDER_TO_FRONT: 실행되는 액티비티를 태스크 목록에서 최상위로 올리기 위한 설정
- FLAG_ACTIVITY_SINGLE_TOP: 액티비티가 태스크 목록에서 최상위에 있을 때 생성하지 않는 설정



15.3 액티비티를 위한 다양한 설정

15.3.1. 키보드 제어

- 키보드 보이기와 숨김

```
InputMethodManager manager=(InputMethodManager)getService(INPUT_METHOD_SERVICE);
```

- showSoftInput(View view, int flags): 키보드 보임
- hideSoftInputFromWindow(IBinder windowToken, int flags): 키보드 숨김

```
if(v==hideBtn){  
    manager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),  
    InputMethodManager.HIDE_NOT_ALWAYS);  
}else if(v==showBtn) {  
    manager.showSoftInput(editText, InputMethodManager.SHOW_IMPLICIT);  
}
```

- toggleSoftInput(int showFlags, int hideFlags):

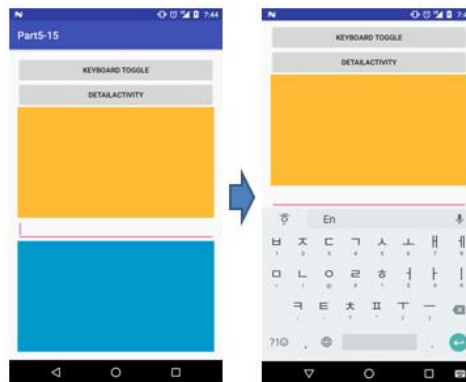


15.3 액티비티를 위한 다양한 설정

키보드로 액티비티 화면 조정

- 기본(설정 안 된 경우): `adjustUnspecified`와 `stateUnspecified`가 적용
 - `adjustPan`: 키보드가 올라올 때 입력 EditText에 맞춰 화면을 위로 올림
 - `adjustResize`: 키보드가 올라올 때 액티비티의 크기 조정
 - `adjustUnspecified`: 시스템이 알아서 상황에 맞는 옵션 설정
 - `stateHidden`: Activity 실행 시 키보드가 자동으로 올라오는 것 방지
 - `stateVisible`: Activity 실행 시 키보드가 자동으로 올라옴
 - `stateUnspecified`: 시스템이 적절한 키보드 상태를 설정하거나 테마에 따라 설정
-
- `adjustPan`

```
<activity android:name=".Lab2Activity" android:windowSoftInputMode="adjustPan">
```

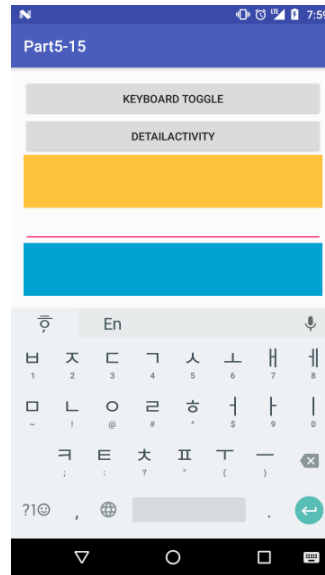


15.3 액티비티를 위한 다양한 설정

- 초기에 키보드가 자동으로 올라오게 설정

```
<activity android:name=".Lab2Activity" android:windowSoftInputMode="adjustPan|stateVisible">
```

- adjustResize



```
<activity android:name=".Lab2Activity" android:windowSoftInputMode="adjustResize|stateHidden">
```



15.3 액티비티를 위한 다양한 설정

15.3.2. 다중 창 지원

- 다중 창(Multi Window)은 API Level 24(Android 7.0)에 추가된 기능
- 스크린을 분할하여 한 화면에 두 앱을 동시에 띄울 수 있는 기능



Overview Button Long Click

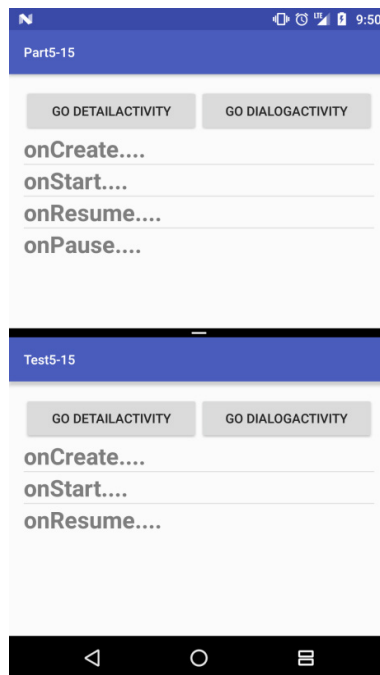


Overview 화면에서 Activity Title Long Click



15.3 액티비티를 위한 다양한 설정

- 다중 창 모드가 되었을 때의 생명주기



- 앱에서 다중 창을 지원하고 싶지 않으면 액티비티의 `resizeableActivity` 속성값을 `false`로 지정

```
<activity android:name=".Lab2Activity"          android:windowSoftInputMode="adjustResize|stateHidden"
          android:resizeableActivity="false">
```



15.3 액티비티를 위한 다양한 설정

다중 창과 관련된 상황을 파악

- `isInMultiWindowMode()`: 액티비티가 다중 창 모드에 있는지 확인
- `onMultiWindowModeChanged()`: 액티비티가 다중 창 모드로 들어가거나 나올 때 콜백 함수

@Override

```
protected void onResume() {  
    super.onResume();  
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {  
        if(isInMultiWindowMode()){  
            showToast("onResume....isInMultiWindowMode...yes...");  
        }  
    }  
}
```

@Override

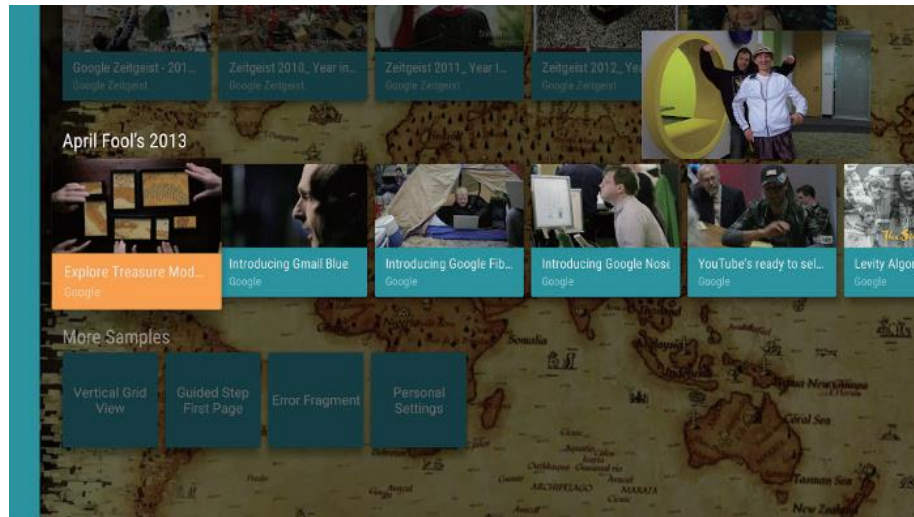
```
public void onMultiWindowModeChanged(boolean isInMultiWindowMode) {  
    super.onMultiWindowModeChanged(isInMultiWindowMode);  
    showToast("onMultiWindowModeChanged....."+isInMultiWindowMode);  
}
```



15.3 액티비티를 위한 다양한 설정

15.3.3. Picture In Picture (Android Oreo 추가사항)

- API Level 26(Android Oreo)에서는 Android TV뿐 아니라 스마트폰용 앱에서도 PIP 기능을 구현 가능



- PIP를 적용하려면 액티비티에 `supportsPictureInPicture` 설정이 필요

```
<activity android:name=".MainActivity" android:supportsPictureInPicture="true">
</activity>
```

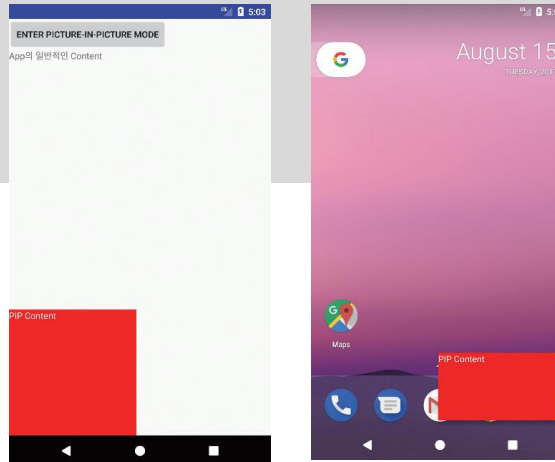


15.3 액티비티를 위한 다양한 설정

- supportsPictureInPicture 설정이 true로 지정되면 resizableActivity 속성은 자동으로 무시
- isInPictureInPictureMode(): 액티비티가 PIP 모드인지 확인
- onPictureInPictureModeChanged(): PIP 모드로 전환되거나 PIP 모드에서 해제될 때 호출
- enterPictureInPictureMode(): 액티비티를 PIP 모드로 전환

```
PictureInPictureParams.Builder pipBuilder = new PictureInPictureParams.Builder();  
enterPictureInPictureMode(pipBuilder.build());
```

```
@Override  
public void onPictureInPictureModeChanged( boolean isInPictureInPictureMode, Configuration newConfig) {  
    if(isInPictureInPictureMode){  
        scrollView.setVisibility(View.GONE);  
    }else {  
        scrollView.setVisibility(View.VISIBLE);  
    }  
}
```



15.3 액티비티를 위한 다양한 설정

15.3.4. 기타 설정

- 화면 회전

```
<activity android:name=".Lab3Activity" android:screenOrientation="portrait">
```

```
<activity android:name=".Lab3Activity" android:configChanges="orientation|screenSize">
```

```
public void onConfigurationChanged(Configuration newConfig) {  
    super.onConfigurationChanged(newConfig);  
    if(newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){  
        showToast("portrait.....");  
    }else {  
        showToast("landscape.....");  
    }  
}
```

태스크 제어

- alwaysRetainTaskState: 루트 액티비티에 설정하는 속성으로 true로 지정하면 오랜 시간이 지나도 태스크 내의 모든 액티비티 유지
- clearTaskOnLaunch: 루트 액티비티에 설정하는 속성으로 true면 이 앱이 다시 활성 상태가 될 때 루트 액티비티를 제외한 나머지 액티비티 제거
- finishOnTaskLaunch: 개별 액티비티에 모두 설정 가능하며 속성이 true면 앱이 다시 활성 상태가 될 때 이 속성이 설정된 액티비티 제거



Step by Step 15-2 – Activity 설정

Activity의 설정을 테스트

1. Activity 생성
2. 파일 복사
3. Lab15_2Activity 작성
4. AndroidManifest.xml 작성
5. Lab15_2Activity.java 실행

