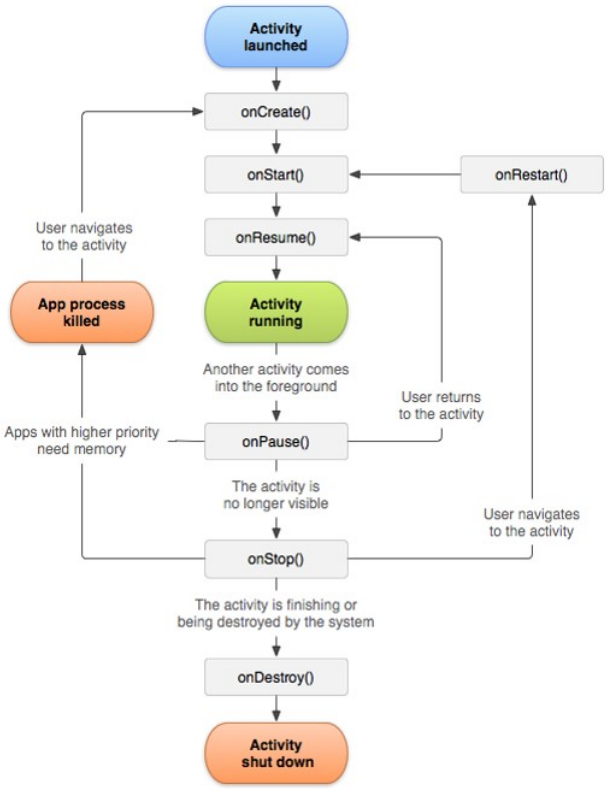
Activity의 Lifecycle



onCreate()

- Activity를 생성할 때 실행
- Activity의 전체 수명 주기 동안 한번만 발생함 ex) 데이터를 목록에 바인딩, Activity를 ViewModel과 연결, 일부 클래스 범위 변수를 인스턴스 화
- saveInstanceState 매개변수를 수신, 이는 Activity의 이전 저장 상태가 포함된 Bundle 객체.
    - Bundle이란 여러가지의 타입의 값을 저장하는 Map 클래스
    - 전에 저장된 데이터가 있으면 그 데이터를 가지고 다시 Activity를 생성한다. Bundle은 아무거나 포장할 수 있는 상자를 의미, 이 포장 박스를 이용하여 이리저리 인텐트도 오고갈 수 있고, 다양한 데이터 동신에 이용할 수 있다.

onStart()

- Activity가 시작됨 상태에 들어가면 시스템은 이 콜백을 호출함
- 이게 호출되면 Activity가 사용자에게 보여지게 됨, 앱은 포그라운드에 보내 상호작용 할 수 있도록 준비함
- 매우 빠르게 완료되고 생성되며 이 상태에 머무르지 않는다. → 바로 onResume()을 호출함.

onResume()

- Activity가 재개됨 상태에 들어가면 포그라운드에 표시되고 시스템이 onResume() 콜백을 호출함
- 이 상태에서 앱이 사용자와 상호작용함.
- 어떤 이벤트가 발생하여 앱에서 포커스가 떠날 때까지 앱이 이 상태에 머무름
- Activity의 수명 주기와 연결된 모든 수명주기 인식 구성요소는 ON_RESUME이벤트를 수신함
- 방해되는 이벤트가 발생하면 일시정지 상태가 되고 시스템이 onPause()를 콜백, 다시 재개되면 onResume()다시 호출

ON_RESUME수신 카메라 엑세스

```
public class CameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    public void initializeCamera() {
        if (camera == null) {
            getCamera();
        }
    }

    ...
}
```

onPause()

- 사용자가 Activity를 떠나는 것을 나타내는 첫번째 신호로 이 메서드 호출(Activity가 포그라운드에 있지 않다는 것, 다만 사용자가 멀티 윈도우 모드에 있을 경우에는 여전히 보일 수 있음)
- Activity의 수명 주기와 연결된 모든 수명주기 인식 구성요소는 ON_PAUSE 이벤트를 수신함
- 여기에서 수명 주기 구성요소는 구성요소가 포그라운드에 있지 않을 때 실행할 필요가 없는 기능을 모두 정지할 수 있음(*foreground: 다중 프로그래밍에서, 한 프로세스가 다른 프로세스보다 우선권을 가지고 실행하는 일)
- 멀티 윈도우 상태를 더 잘 지원하기 위해서는 onPause()일 때가 아니라 onStop()일 때를 조정하는게 좋음 (멀티 윈도우 상태에서 onPause()상태 진입하기 때문)
- onPause()는 아주 잠깐 실행되므로, 저장 작업을 수행하기에는 시간이 부족
- 이 때 애플리케이션 또는 사용자 데이터를 저장하거나, 네트워크 호출을 하거나, 데이터베이스 트랜잭션을 실행해서는 안됨
- 부하가 큰 종료 작업은 onStop()일 때 실행해야함.
- onPause()메서드의 실행이 완료되더라도, Activity가 일시정지 상태로 남아 있을 수 있다.
- 오히려 Activity는 다시 시작되거나 사용자에게 완전히 보이지 않게 될 때까지 이 상태에 머무른다.
- Activity가 다시 시작되면 시스템은 다시 한번 onResume()콜백을 호출한다.
- 일시정지됨 상태에서 재개됨 상태로 돌아오면 시스템은 Activity 인스턴스를 메모리에 남겨두고, 시스템이 onResume()을 호출할 때 인스턴스를 다시 호출한다.
- 최상위 상태가 재개됨 상태인 콜백 메서드 중에 생성된 구성요소를 다시 초기화 할 필요가 없다.

onStop()

- Activity가 사용자에게 더 이상 표시되지 않으면 중단됨 상태에 들어가고, 시스템이 onStop()콜백을 호출함.
- 새로 시작된 Activity가 화면 전체를 차지할 경우에 적용됨
- 시스템은 Activity의 실행이 완료되어 종료될 시점에 onStop()을 호출할 수도 있음
- Activity의 수명 주기와 연결된 모든 수명 주기 인식 구성요소는 ON_STOP 이벤트를 수신함
- 여기에서 수명 주기 구성요소는 구성요소가 화면에 보이지 않을 때 실행할 필요가 없는 기능을 모두 정지 할 수 있음
- onStop()메서드에서 앱이 사용자에게 보이지 않는 동안 앱은 필요하지 않은 리소스를 해제하거나 조정해야함.
- ex)애니메이션을 일시정지 하거나, 세밀한 위치 업데이트에서 대략적인 위치 업데이트로 전환 해야함
- Activity가 다시 시작되면 onRestart()호출, Activity가 실행을 종료하면 onDestroy()호출

초안 내용을 영구 저장소에 저장하는 onStop()

```
@Override
protected void onStop() {
    // call the superclass method first
    super.onStop();

    // save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    // do this update in background on an AsyncQueryHandler or equivalent
    asyncQueryHandler.startUpdate (
            mToken,  // int token to correlate calls
            null,    // cookie, not used here
            uri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
    );
}
```

onDestroy()

- Activity가 소멸되기 전에 호출
    1. 사용자가 Activity를 완전히 닫거나 Activity에서 finish()가 호출되어 Activity가 종료되는 경우
    2. 구성 변경(기기 회전 또는 멀티 윈도우 모드)로 인해 시스템이 일시적으로 Activity를 소멸시키는 경우
- Activity의 수명 주기와 연결된 모든 수명 주기 인식 구성요소는 ON_DESTROY 이벤트를 수신
- Activity가 소멸되기 전에 필요한 것을 정리 할 수 있음
- ViewModel객체를 사용하여 Activity의 관련 뷰 데이터를 포함해야 함.
- isFinishing() 메서드를 이용해 아래 두가지 시나리오를 구별
    1. Activity가 구성 변경으로 인해 다시 생성될 경우 ViewModel은 그대로 보존되어 다음 Activity 인스턴스에 전달되므로 추가 작업이 필요하지 않음
    2. Activity가 다시 생성되지 않은 경우 ViewModel은 onCleared()메서드를 호출하여 Activity가 소멸되기 전에 모든 데이터를 정리해야 함
- onDestroy()는 Activity가 수신하는 마지막 수명 주기 콜백임
- 구성변경으로 인해 onDestroy()가 호출되는 경우, 시스템이 즉시 새 Activity인스턴스를 생성한 다음, 새로운 구성에서 그 새로운 인스턴스에 대해 onCreate()를 호출함
- 여기서 아직 해제되지 않은 모든 리소스를 해제해야함.

프로세스 상태, Activity상태, 시스템이 프로세스를 종료할 가능성 사이의 상관관계

Likelihood of being killed	Process state	Activity state
Least	Foreground (having or about to get focus)	Created Started Resumed
More	Background (lost focus)	Paused
Most	Background (not visible)	Stopped
	Empty	Destroyed

Table 1. Relationship between process lifecycle and activity state

시스템은 메모리 공간은 확보하기 위해 절대 Activity를 직접 종료하지 않음

그 대신, Activity를 실행하는 프로세스를 종료하여 Activity뿐만 아니라 프로세스에서 실행되는 다른 모든 작업을 함께 소멸시킴

임시 UI상태 저장 및 복원

사용자는 구성이 변경되더라도 Activity UI의 상태가 그대로 유지되기를 원함

→ 회전, 멀티 윈도우 모드로 전환 또는 일시적으로 앱에서 다른 앱으로 전환했다가 다시 앱으로 돌아왔을 때

ViewModel, onSaveInstanceState() 및 로컬 저장소를 결합하여 사용자의 임시 UI상태를 보존해야 함(UI 상태 저장 참고)

- Activity가 정상 소멸되지 않는 경우, Activity는 다시 돌아가려고 시도하면 소멸 당시 Activity상태를 설명하는 저장된 데이터 세트를 사용해 새로운 Activity의 인스턴스를 생성한다.
- 시스템이 이전 상태를 복원하기 위해 사용하는 저장된 데이터를 인스턴스 상태라고 하며, 이는 Bundle객체에 저장된 키-값 쌍의 컬렉션이다.
- Bundle객체는 메인 스레드에서 직렬화 되어야 하고, 시스템 프로세스 메모리를 사용하므로 소량의 데이터를 보존하는 데만 적합함.

onSaveInstanceState()를 사용하여 간단하고 가벼운 UI 상태 저장

https://developer.android.com/guide/components/activities/activity-lifecycle#onsaveinstancestate%EB%A5%BC-%EC%82%AC%EC%9A%A9%ED%95%98%EC%97%AC-%EA%B0%84%EB%8B%A8%ED%95%98%EA%B3%A0-%EA%B0%80%EB%B2%BC%EC%9A%B4-ui-%EC%83%81%ED%83%9C-%EC%A0%80%EC%9E%A5