Activity Life Cycle

Mobile Software
2019 Fall

Android 리소스 관리 전략

- Android 시스템은 리소스(특히 메모리)가 부족하면, 실행 중인 프로세스를 강제로 중단시킴
  - 어느 프로세스를 중단시킬 것인가?
    - 프로세스 상태(state)에 따라 우선 순위(priority)가 정해짐
    - 낮은 우선 순위 프로세스부터 중단시킴 ➔ 메모리 반환

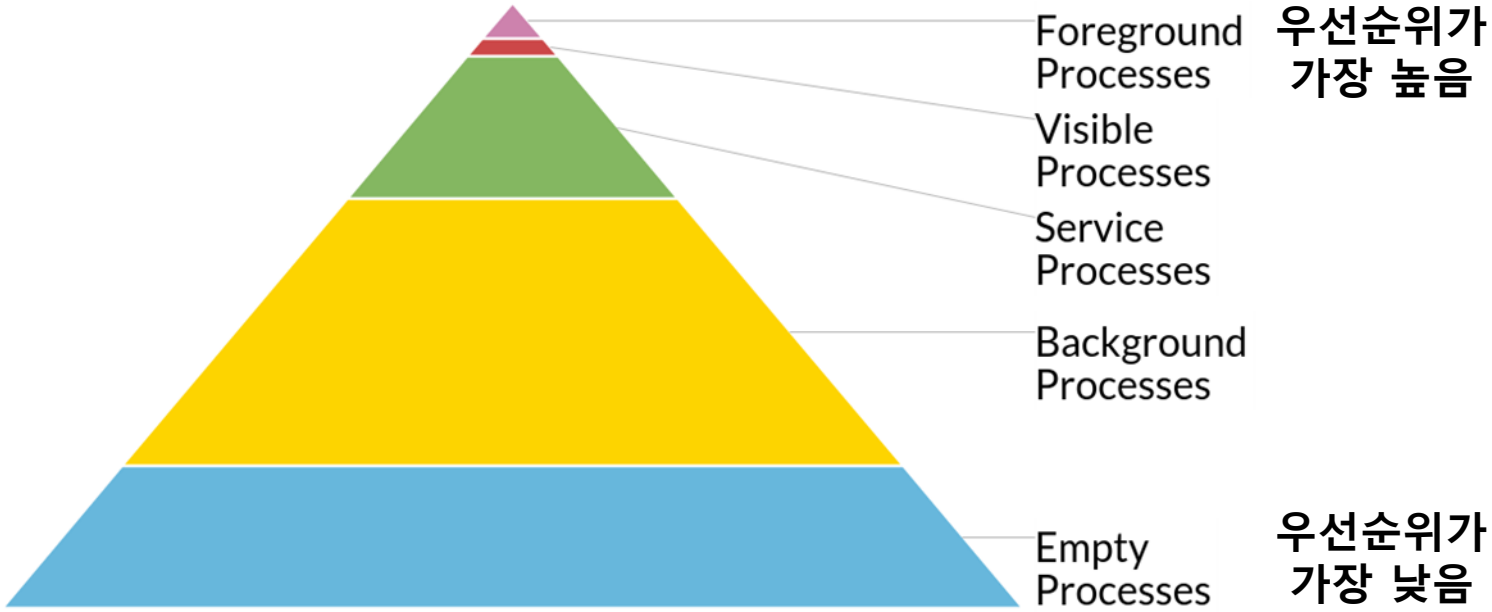


그림 출처: https://medium.com/androiddevelopers/who-lives-and-who-dies-process-priorities-on-android-cb151f39044f

Android Process States

- **Foreground process**
  - 실행 중. 사용자와 상호 작용 중.
    - onCreate, onStart, onResume 메소드 중 하나를 실행 중
    - onReceive 실행 중(broadcast receiver로써 실행 중)
- **Visible process**
  - 화면을 볼 수 있지만, 사용자와 상호작용은 하고 있지 않음.
- **Service process**
  - 현재 백그라운드에서 실행 중인 **서비스.**
- **Background process**
  - 화면이 없으며, background에서 실행 중.
- **Empty process**
  - 새롭게 실행될 App.을 처리하기 위해 대기 중.
    - 메모리를 차지하고 있음

Activity Stack (1/3)

- **Activities** in the system are managed as an **activity stack**.

- When ***a new activity is started***, it is placed on **the top of the stack** and becomes the ***running activity***.
  - The previous activity always remains below it in the stack, and
  - will not come to the foreground again until the new activity exits.

- If the user presses the **Back** Button,
  - the next activity on the stack moves up and becomes *active*.

Activity Stack (2/3)

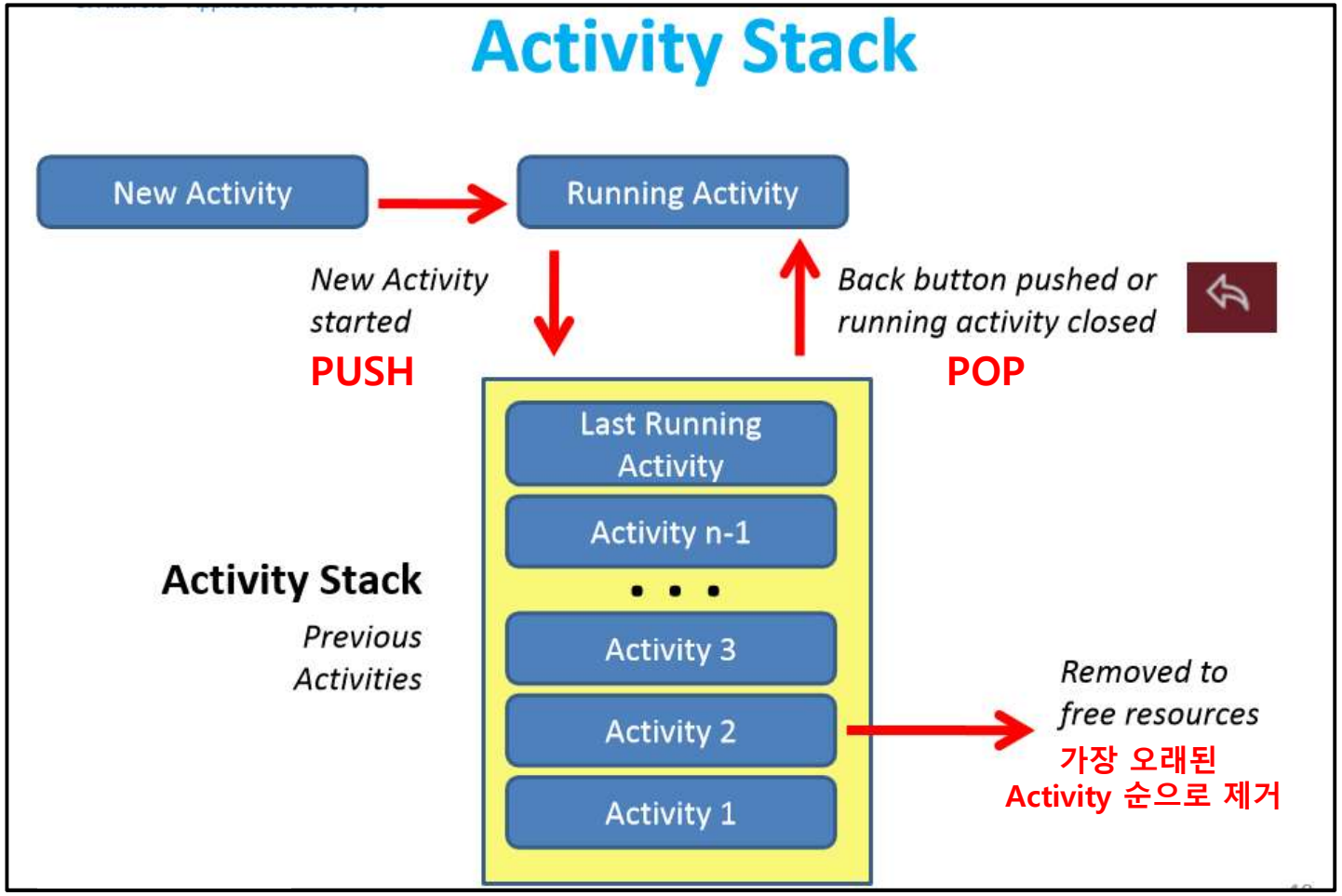


그림 출처: http://blog.appliedinformaticsinc.com/android-activity-an-overview/

Activity Stack (3/3)

- Back 키를 누르면 현재 activity를 제거하고 이전 activity로 되돌아 간다.

Activity Lifecycle (생명주기)

- Activity has a **lifecycle**.
  - **A beginning** when Android **instantiates them** to respond to intents through to an **end** when the **instances are destroyed**.
  - In **between**, they may sometimes be **active** or **inactive**, or in the case of activities, **visible** to the user or **invisible**.

Activity States (1/2)

- **Active or Running**
  - When it is in **the foreground of the screen**
    - at the top of the activity stack for the current task.
  - This is the activity that is the **focus** for the user's actions.
- **Paused**
  - If it has **lost focus** but is **still visible to the user**.
    - 투명한 activity나 화면 전체를 사용하지 않는 activity가 활성화될 경우
      - Activity가 완전히 가려지면, 해당 activity 는 중지됨
  - A paused activity is **completely alive** (*it maintains all state and member information and remains attached to the window manager*),
    - but **can be killed by the system** in extreme low memory situations.

Activity States (2/2)

- **Stopped**
  - If it is **completely obscured** by another activity.
    - It still **retains all state and member information**.
      - However, it is no longer visible to the user so its window is hidden.
    - It will often be killed by the system when memory is needed elsewhere.
    - 프로세스 종료 후보 1순위!
- **Killed**
  - This activity has been terminated by the runtime system
  - And no long present on the Activity Stack

Activity Lifecycle

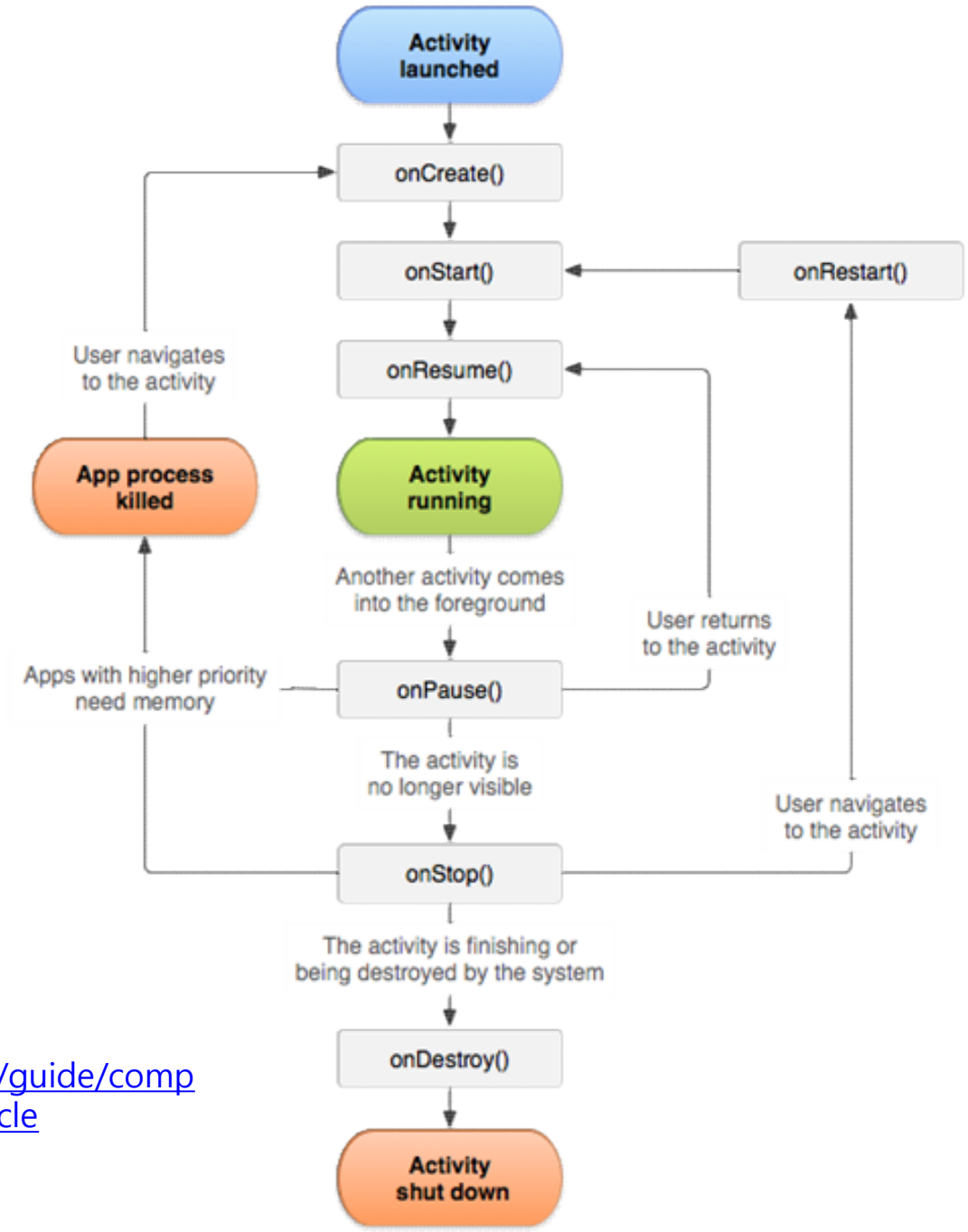


그림 출처
https://developer.android.com/guide/components/activities/activity-lifecycle

7 Lifecycle Methods (1/3)

- **callback 메소드를 구현할 때 super 클래스의 overriding 되는 메소드를 반드시 호출해야 함.**
- **onCreate**()
  - Called when the activity is **first created**.
  - This is where you should do all of your normal ***static set up***
    - create views, bind data to lists, and so on.
  - This method is ***passed a Bundle object*** containing the activity's previous state, if that state was captured.
  - **반드시 구현해야 하는 callback 메소드**

- **onRestart**()
  - Called after the activity has been stopped, just prior to it being started again.

7 Lifecycle Methods (2/3)

- **onStart()**
  - Called just before the activity becomes **visible** to the user.
- **onResume()**
  - Called just before the activity starts **interacting with the user**.
  - At this point the activity is at the **top** of the activity stack, with user input going to it.
- **onPause()**
  - Called when the system is about to start resuming another activity.
  - It is typically used to **commit unsaved changes to persistent data**, **stop animations** and other things that may be consuming CPU, and so on.

7 Lifecycle Methods (3/3)

- **onStop()**
  - Called when the activity is **no longer visible** to the user.
  - This may happen because it is being destroyed
    - because another activity has been resumed and is covering it.
- **onDestroy()**
  - Called before the activity is destroyed.
  - This is **the final call** that the activity will receive.
  - It could be called either
    - because the activity is finishing
      - called **finish**() on it
    - Or because the system is temporarily destroying this instance of the activity to save space.

onCreate 메소드의 parameter 는 왜 필요한가?

- Bundle **savedInstanceState**
  - 이 parameter는 어떤 정보를 저장하고 있나?
    - **Activity 의 UI state**
      - Checkbox states, user focus
      - Entered but not committed user input

- Activity가 active 상태에서 pause 상태로 바뀌기 전에
  - **onSaveInstanceState** 를 호출해서 UI state 를 저장

- 이렇게 저장된 parameter가 **onCreate** 메소드로 전달됨
  - 현재 activity 실행이 종료되면 activity stack에서 (실행이 중단된)이전 activity를 꺼냄
    - 전달된 UI 상태 값을 사용하여 이 activity의 원래 상태를 복원

another 2 callback methods

- **Activity 상태에 따라 어떤 값을 저장하거나 복원해야 할까?**
  - **Persistent state** : 작업한 데이터가 없어지지 않도록 저장
    - 데이터베이스, content provider, file 등
  - **Dynamic state** (동적 상태) : 사용자가 작업하던 내용을 저장
    - EditText창에 입력했던 내용, CheckBox 체크 여부 등.
    - 장치 구성이 바뀌면 이전 activity의 instance를 없애고 새 instance를 생성하기 때문 이전 작업 내용이 다 없어짐.

- **동적 상태 저장 및 복원에 사용하는 메소드**
  - **onSaveInstanceState** (Bundle outstate)
    - Bundle 객체에 동적 상태 저장 → onCreate, onRestoreInstanceState 메소드에 전달
  - **onRestoreInstanceState** (Bundle savedInstanceSate)
    - onStart 메소드 직후 호출

Killable States

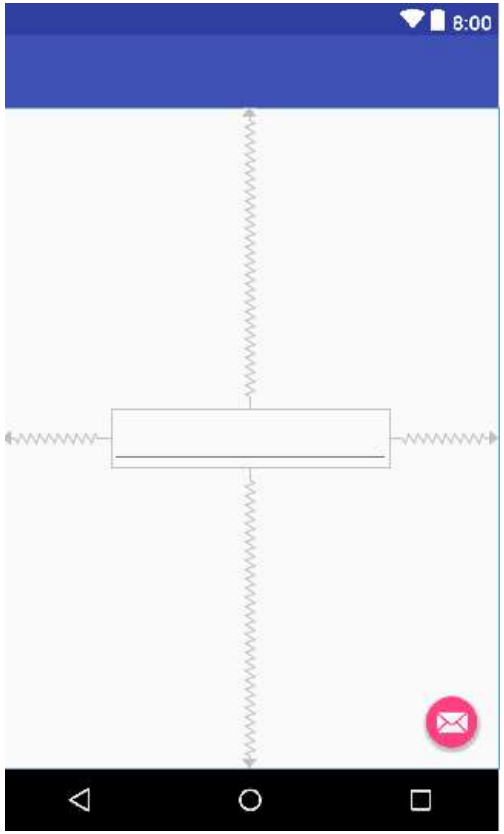
- Activities on killable states **can be terminated by the system** at any time after the method returns, ***without executing another line of the activity's code***.
  - **onPause**(), **onStop**(), and **onDestroy**()

- **onPause**() is the only one that is **guaranteed to be called before the process is killed.**
  - **onStop**() and **onDestroy**() may not be.
  - Therefore, you should use **onPause**() to write any persistent data (such as user edits) to storage.

실습 준비

- 새 프로젝트 생성
  - **Application name**
    - **StateChange**
  - **Target Android Devices**
    - **Phone and Tablet**
      - minimum SDK – **API 26** 이상
  - **Activity**
    - **Basic Activity**

- 자동 생성된 layout은 **ConstraintLayout**
  - TextView 삭제 → **EditView** 를 중앙에 배치
    - **inputType** : text
    - **text** 속성은 없앰(글자가 보이지 않음)
    - id 속성 : **editText**



실습 1: MainActivity.kt

```kotlin
class StateChangeActivity : AppCompatActivity() {
    val TAG = "StateChange"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_state_change)
        setSupportActionBar(toolbar)

        fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action",
                    .setAction("Action", null).show()
        }

        Log.i(TAG, "onCreate")
    }

    override fun onStart() {
        super.onStart()
        Log.i(TAG, "onStart")
    }
```

메소드 입력 방법
Ctrl-O
또는 **Alt-Insert**

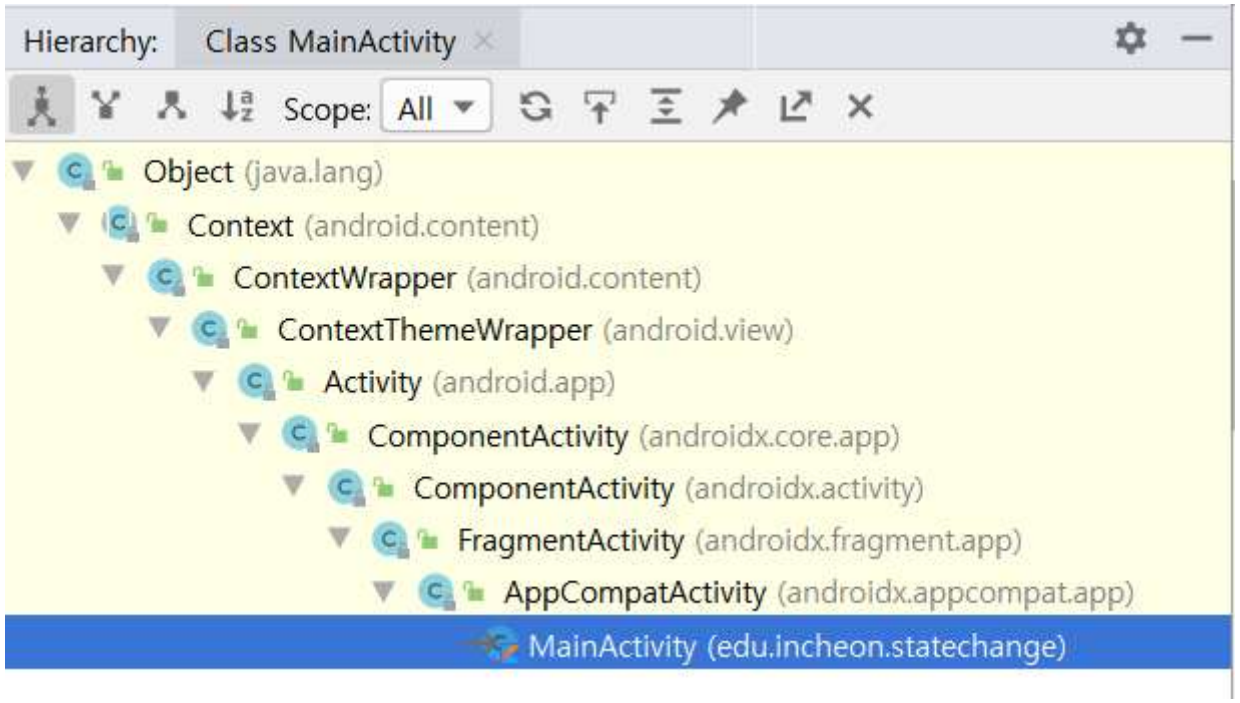
onResume
onPause
onRestart

```kotlin
override fun onDestroy() {
    super.onDestroy()
    Log.i(TAG, "onDestroy")
}

override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    Log.i(TAG, "onSaveInstanceState")
}

override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    super.onRestoreInstanceState(savedInstanceState)
    Log.i(TAG, "onRestoreInstanceState")
}
```

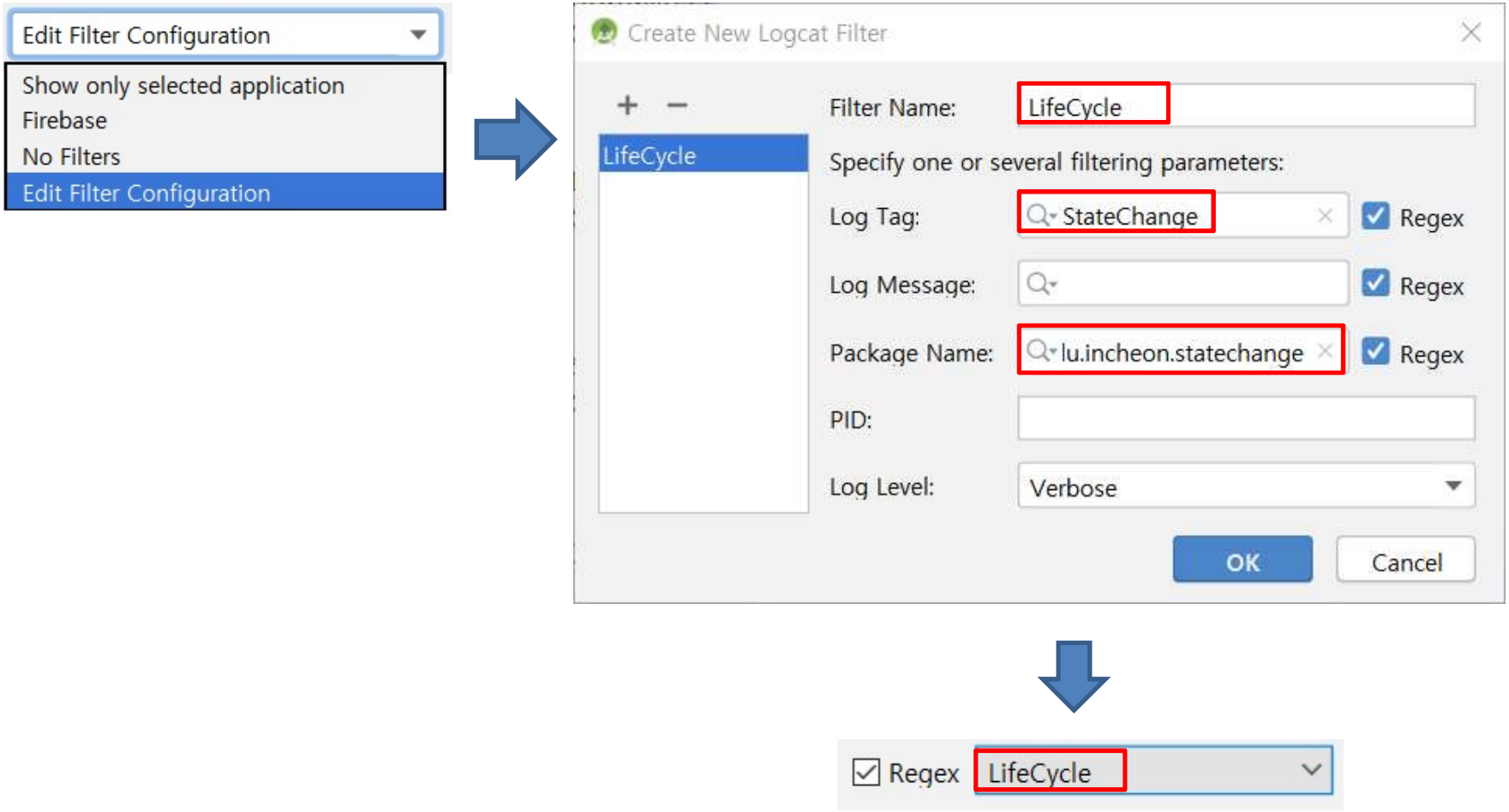
잠깐! Class hierarchy

```
class MainActivity : AppCompatActivity()
```

Ctrl-H



LogCat Panel에서 Filtering



실습 1

홈 버튼 클릭

Logcat

Emulator Pixel_2_API_28 Android | edu.incheon.statechange (7087) | Verbose

```
2019-09-15 08:44:09.804 7087-7087/edu.incheon.statechange I/StateChange: onCreate
2019-09-15 08:44:09.816 7087-7087/edu.incheon.statechange I/StateChange: onStart
2019-09-15 08:44:09.817 7087-7087/edu.incheon.statechange I/StateChange: onResume
2019-09-15 08:44:47.076 7087-7087/edu.incheon.statechange I/StateChange: onPause
2019-09-15 08:44:47.152 7087-7087/edu.incheon.statechange I/StateChange: onStop
2019-09-15 08:44:47.168 7087-7087/edu.incheon.statechange I/StateChange: onSaveInstanceState
```

다시 실행

```
2019-09-15 08:46:12.790 7087-7087/edu.incheon.statechange I/StateChange: onRestart
2019-09-15 08:46:12.791 7087-7087/edu.incheon.statechange I/StateChange: onStart
2019-09-15 08:46:12.794 7087-7087/edu.incheon.statechange I/StateChange: onResume
```

90도 회전

```
2019-09-15 08:46:47.256 7087-7087/edu.incheon.statechange I/StateChange: onPause
2019-09-15 08:46:47.261 7087-7087/edu.incheon.statechange I/StateChange: onStop
2019-09-15 08:46:47.262 7087-7087/edu.incheon.statechange I/StateChange: onSaveInstanceState
2019-09-15 08:46:47.263 7087-7087/edu.incheon.statechange I/StateChange: onDestroy
2019-09-15 08:46:47.371 7087-7087/edu.incheon.statechange I/StateChange: onCreate
2019-09-15 08:46:47.376 7087-7087/edu.incheon.statechange I/StateChange: onStart
2019-09-15 08:46:47.378 7087-7087/edu.incheon.statechange I/StateChange: onRestoreInstanceState
2019-09-15 08:46:47.379 7087-7087/edu.incheon.statechange I/StateChange: onResume
```

실습 1-2

- 아래와 같이 실행시킬 때 Lifecycle state 변화 확인
  - LogCat 창의 값을 지우려면 ➔ 🗑 클릭
  - **첫 번째 연습**
    - App 실행
    - Back 버튼을 누른다
    - 대기 중인 App을 다시 실행시킨다.
    - Home 버튼을 누른다
  - **두 번째 연습**
    - App 실행
    - 90도 회전한다(landscape mode)
    - 다시 90도 회전해 원래 화면으로 돌아온다(portrait mode)

- 실습 목적
  - Back 버튼을 누르거나 화면을 회전하면 **완전히 새로운 activity instance 가 생성**
    - 이전 activity instance의 상태 저장이나 복원이 불가능

Quiz

- 실습 1의 예제에
  - EditText 에 "**test**"를 입력
  - **90도 회전**
  - EditText 창에 이전에 입력한 글자가 나타났을까?

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:saveEnabled="false"
    android:ems="10"
```

- 레이아웃 파일에서
  - EditText 의 속성에
    - **saveEnabled** 속성을 **false**로 지정
  - EditText 에 "test"를 입력
  - 90도 회전
  - EditText 에 이전에 입력한 글자가 나타났을까?

실습 2: MainActivity.kt

Bundle 클래스 ➔
Key-value pair 형태로 데이터를 저장

```
<EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="text"
    android:saveEnabled="false"
```

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    Log.i(TAG, "onSaveInstanceState")

    val userText = editText.text
    outState.putCharSequence("savedText", userText)
}

override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    super.onRestoreInstanceState(savedInstanceState)
    Log.i(TAG, "onRestoreInstanceState")

    val userText = savedInstanceState?.getCharSequence("savedText")
    editText.setText(userText)
}
```

잠깐! 코딩 스타일

What's different from the previous code?

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    Log.i(TAG, "onSaveInstanceState")

    val userText = editText.text.toString()
    outState.putString("savedText", userText)
}

override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    super.onRestoreInstanceState(savedInstanceState)
    Log.i(TAG, "onRestoreInstanceState")

    if (savedInstanceState != null) {
        val userText = savedInstanceState.getString("savedText")
        editText.setText(userText)
    }
}
```