

Activity Lifecycle

Mobile Software
2022 Fall

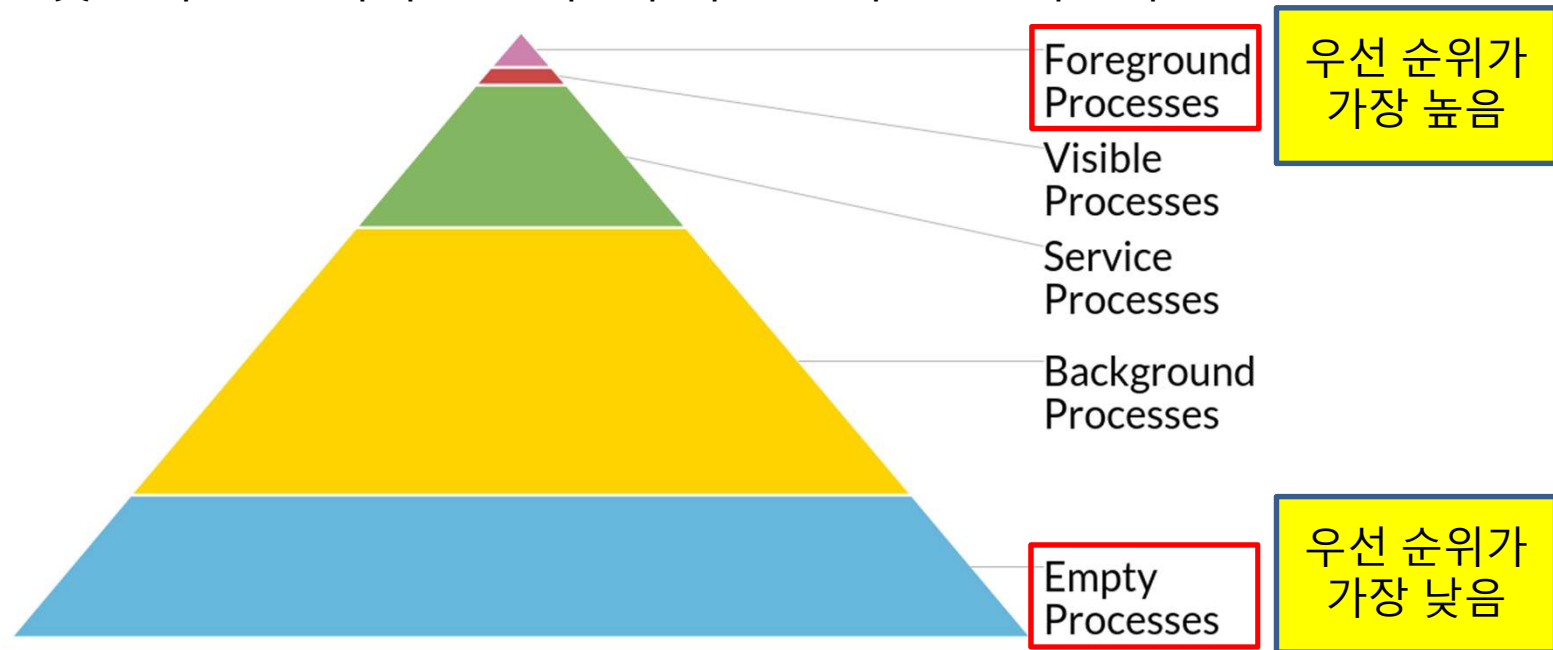
All rights reserved, 2022, Copyright by Youn-Sik Hong (편집, 배포 불허)

What to do next?

- **Activity Stack**
- Activity State
- Activity Lifecycle
- 실습 1
- 실습 2
- 실습 3
- 실습 4
- 강의 노트에 포함된 코드: 소스코드(lifecycle). hwp
 - Layout 관련 설명은 최소화. 상세 내용은 소스 코드 참조.

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 순위가 정해짐
 - 낮은 우선 순위의 프로세스부터 중단시킴 → 메모리 반환



Android Process States

- **Foreground process**
 - 실행 중 : 사용자와 상호 작용 중에 있음.
 - onCreate, onStart, onResume 메소드 중 하나를 실행 중
 - onReceive 실행 가능 : broadcast receiver(방송 수신자) 역할
- **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)

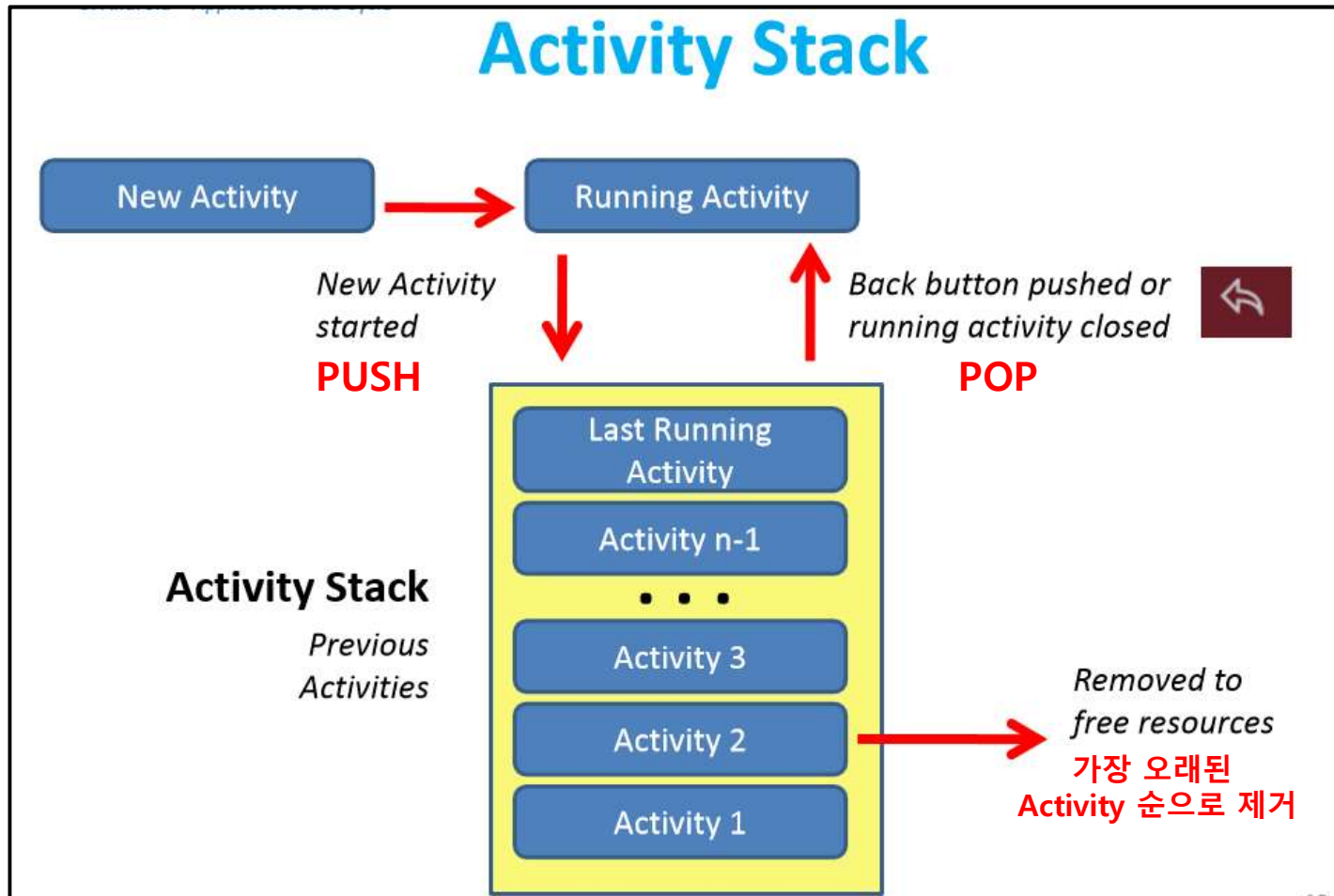
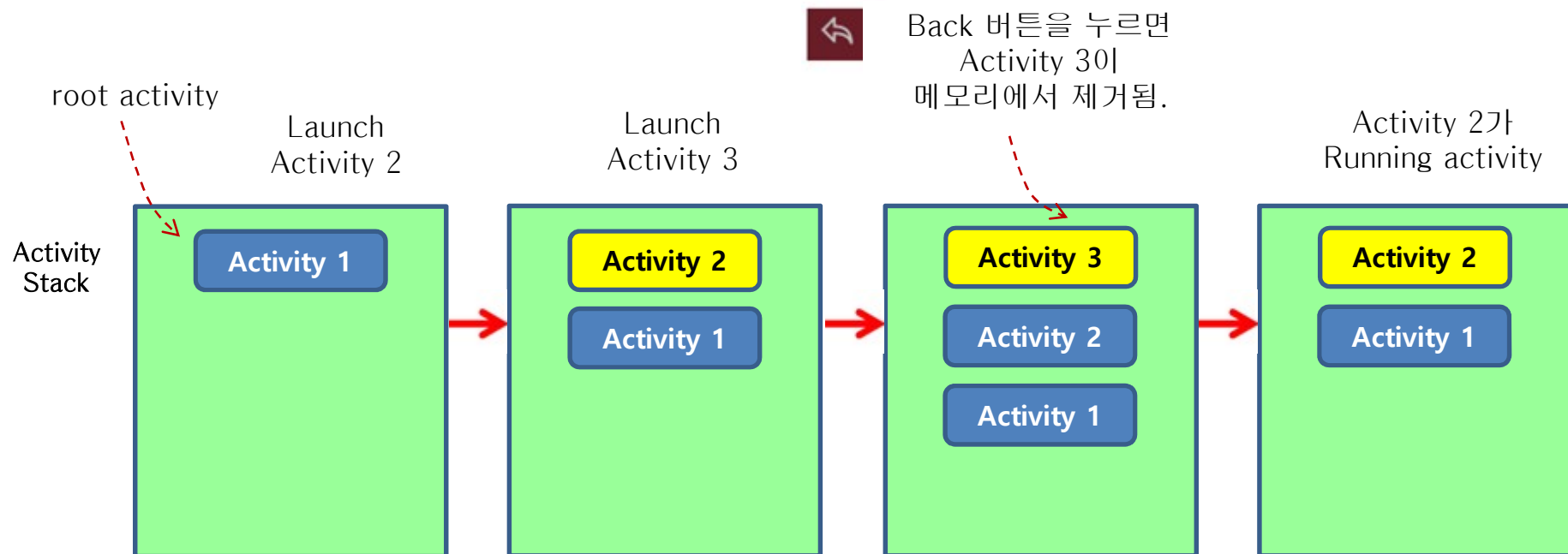


그림 출처: <http://blog.appliedinformaticsinc.com/android-activity-an-overview/>

Activity Stack (3/3)

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



What to do next?

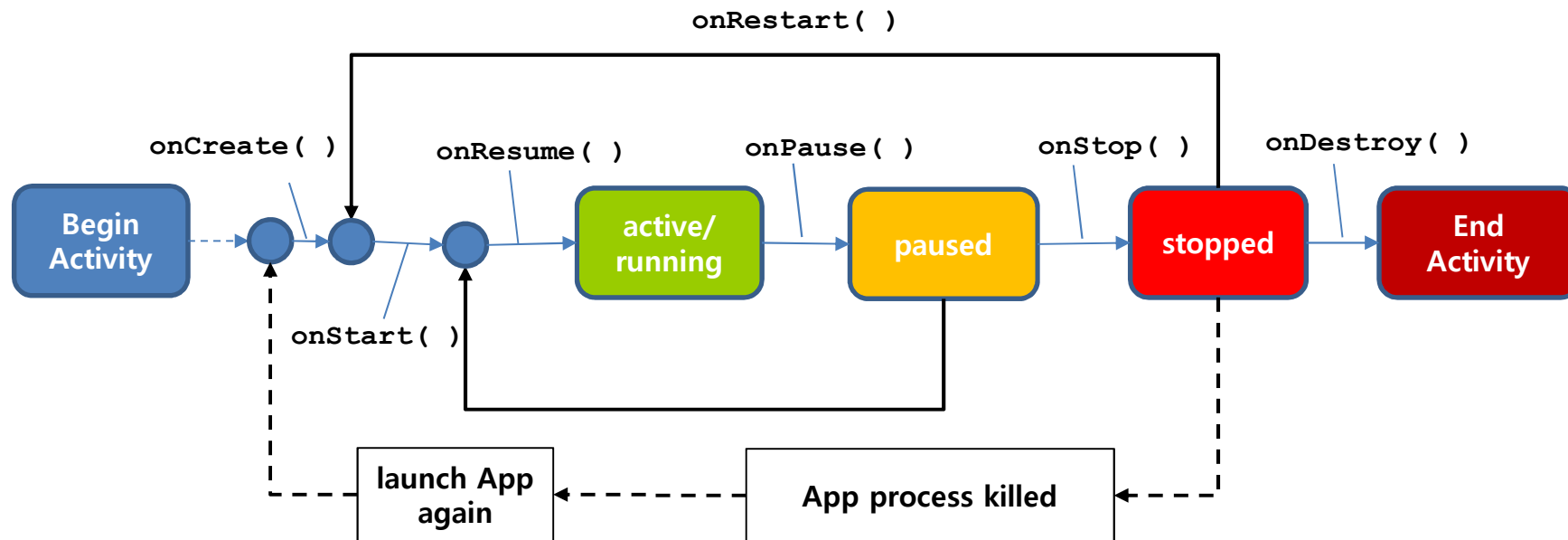
- Activity Stack
- **Activity State**
- Activity Lifecycle
- 실습 1
- 실습 2
- 실습 3
- 실습 4

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 Lifecycle (생명주기)



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

What to do next?

- Activity Stack
- Activity State
- **Activity Lifecycle**
- 실습 1
- 실습 2
- 실습 3
- 실습 4

Activity Lifecycle

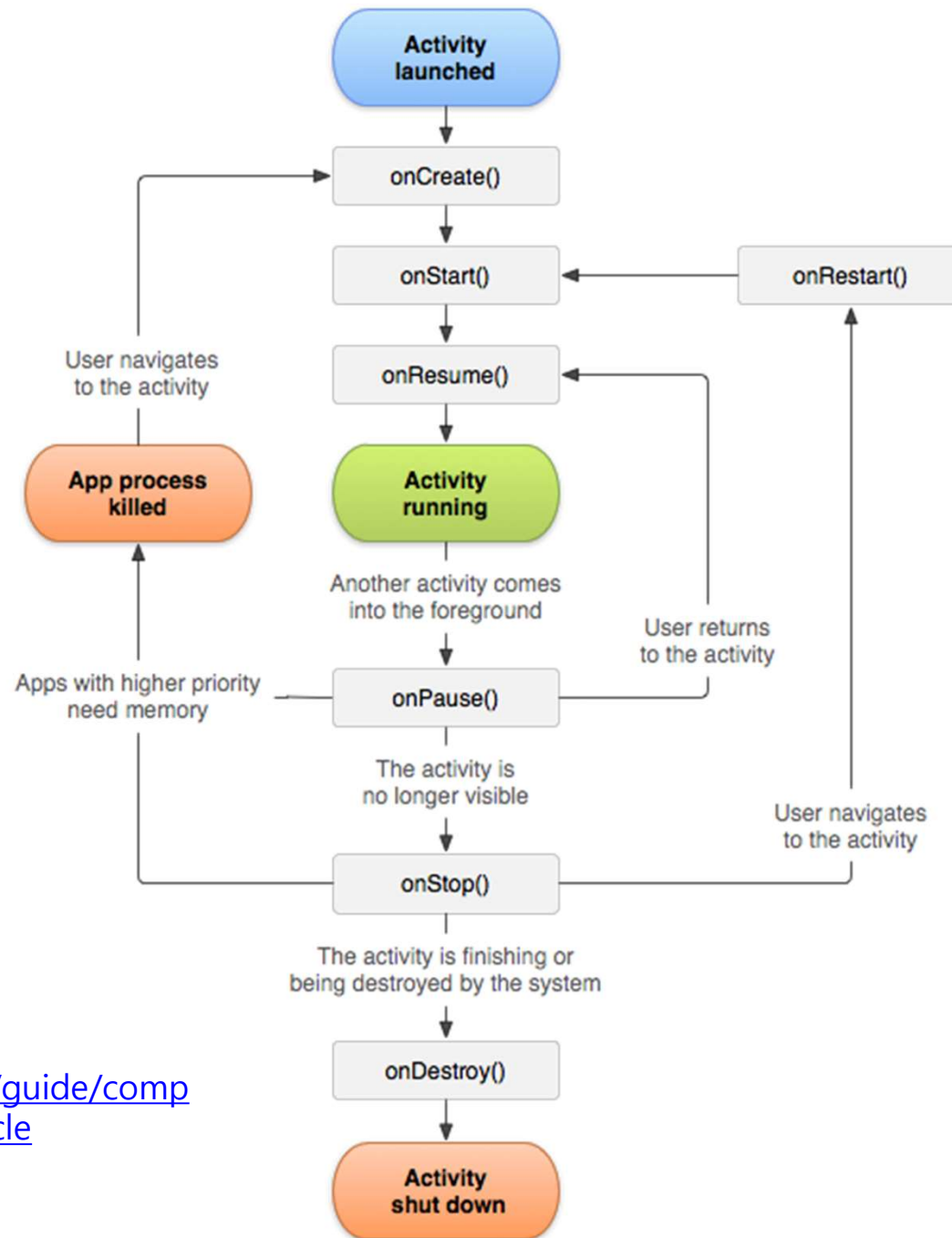


그림 출처

<https://developer.android.com/guide/components/activities/activity-lifecycle>

7 Lifecycle Methods (1/3)

- **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.

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.

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 새로운 instance를 생성하기 때문에 이전 작업 내용이 없어짐.
- **동적 상태 저장 및 복원에 사용하는 메소드**
 - **onSaveInstanceState** (outstate: Bundle)
 - Bundle 객체에 동적 상태 저장 → onCreate, onRestoreInstanceState 메소드에 전달
 - **onRestoreInstanceState** (savedInstanceState: Bundle)
 - onStart 메소드 직후 호출

onCreate 메소드의 parameter 는 왜 필요할까?

- **savedInstanceState:** Bundle
 - 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

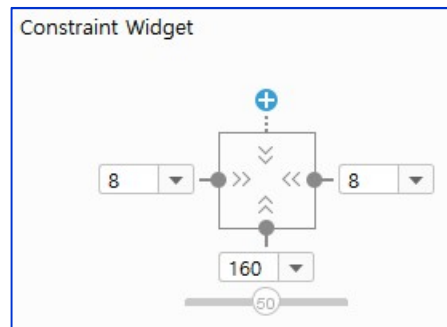
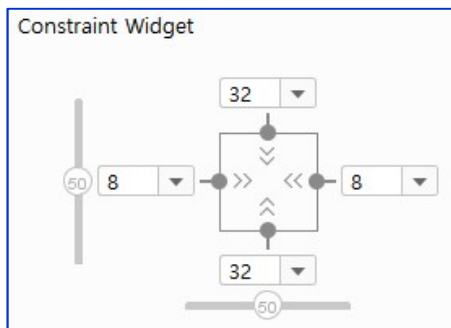
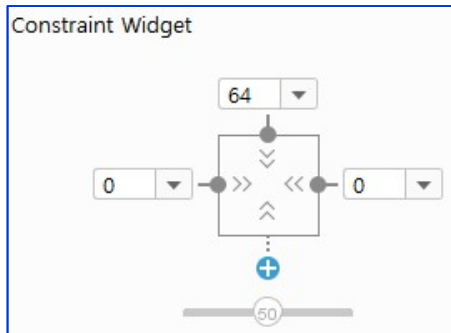
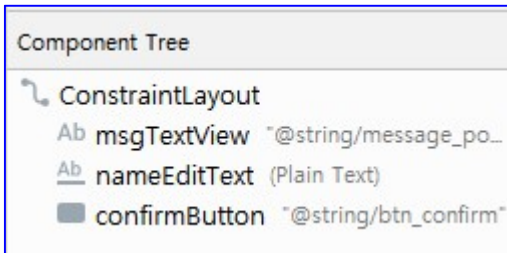
What to do next?

- Activity Stack
- Activity State
- Activity Lifecycle
- **실습 1**
 - **Lifecycle callback method**
- 실습 2
- 실습 3
- 실습 4

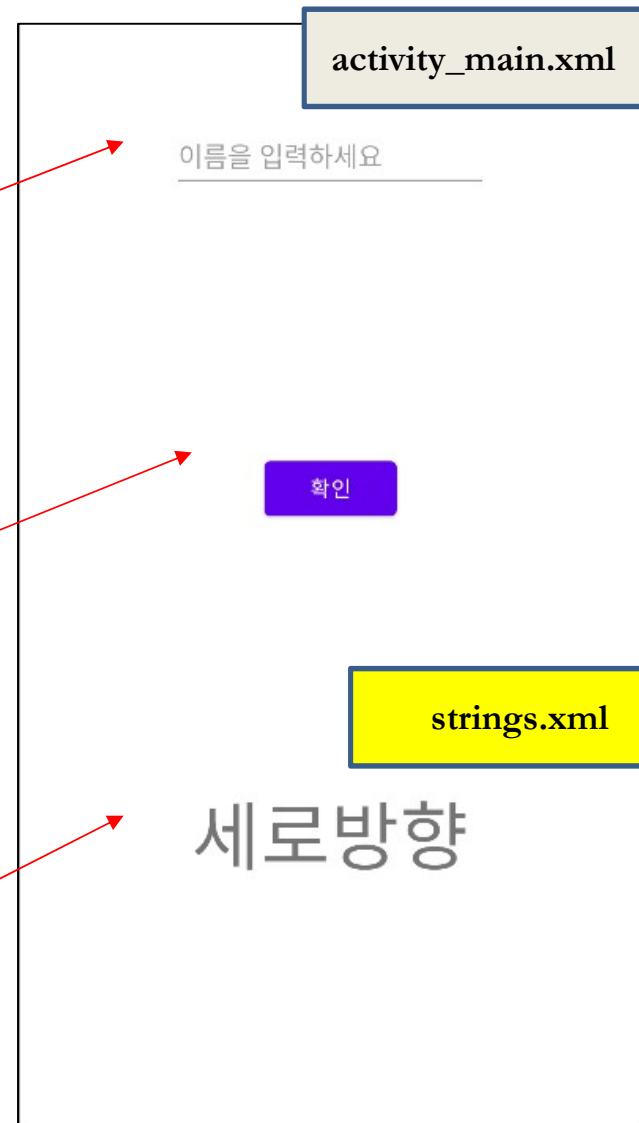
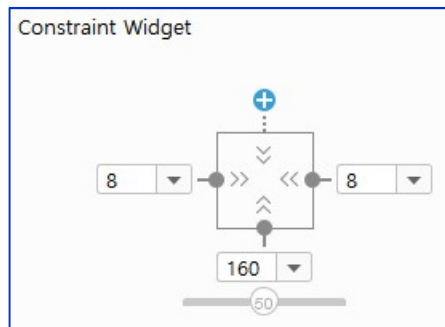
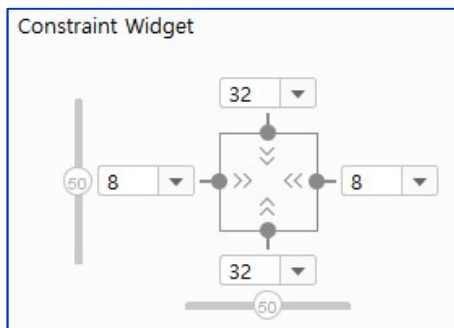
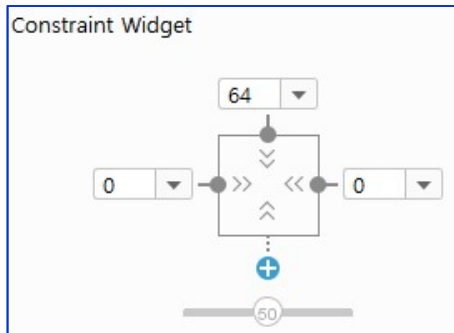
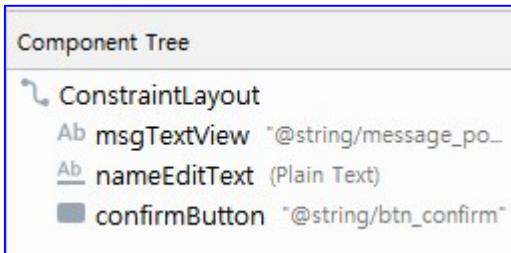
실습 프로젝트 생성

- 새 프로젝트 생성
 - Project name : Lifecycle Example
 - Package name : com.example.lifecycleexample
 - Activity : Empty Activity
 - Activity name : MainActivity.kt
 - Layout name : activity_main.xml
- 자동 생성된 XML 파일의 root view는 ConstraintLayout

실습 1: Layout - Portrait



실습 1: Layout - Portrait



Activity – callback 메소드 추가

최상위 수준 속성

- 상수(constant)
- App. 에 속한 모든 클래스에서 사용

```
private const val TAG = "Orientation"
```

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

```
    override fun onCreate(savedInstanceState: Bundle?) {
```

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

```
        setContentView(R.layout.activity_main)
```

```
        Log.d(TAG, "onCreate() 호출")
```

```
    override fun onStart() {
```

```
        super.onStart()
```

```
        Log.d(TAG, "onStart() 호출")
```

```
    override fun onResume() {
```

```
        super.onResume()
```

```
        Log.d(TAG, "onResume() 호출")
```

```
    override fun onPause() {...}
```

```
    override fun onStop() {...}
```

```
    override fun onDestroy() {...}
```

```
}
```

MainActivity.kt

7개의 lifecycle
method 중
유일하게 인자가 있음

콜백 메소드를 구현할 때
super 클래스의
overriding 되는 메소드를
반드시 호출

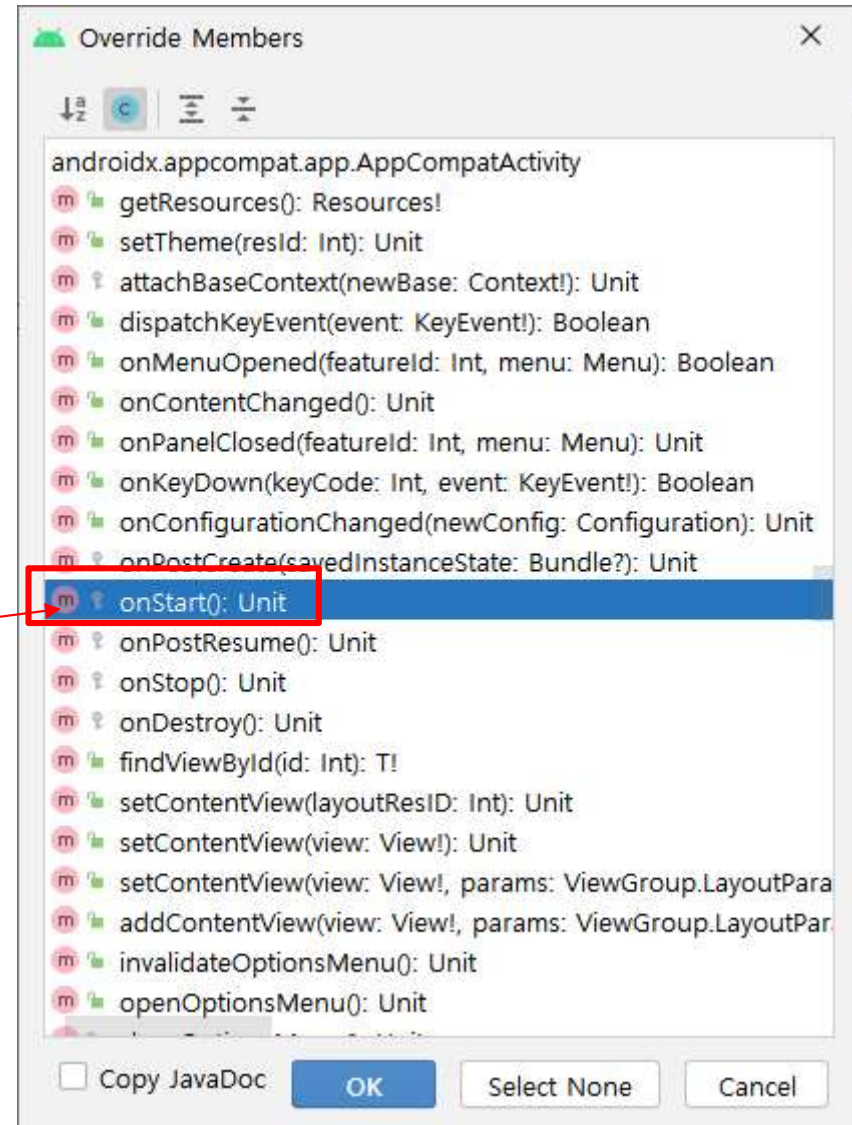
잠깐! override method 추가

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
}
```

Shortcut-Key : Alt + Insert

onStart() 메소드 검색 방법

“o” → “on” → “ons” → “onst”
입력하면서
필요한 method 검색



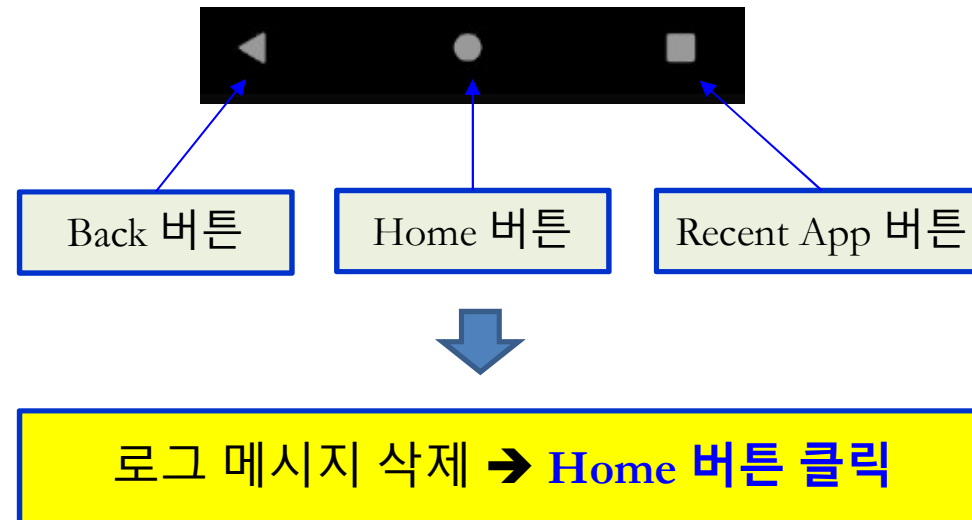
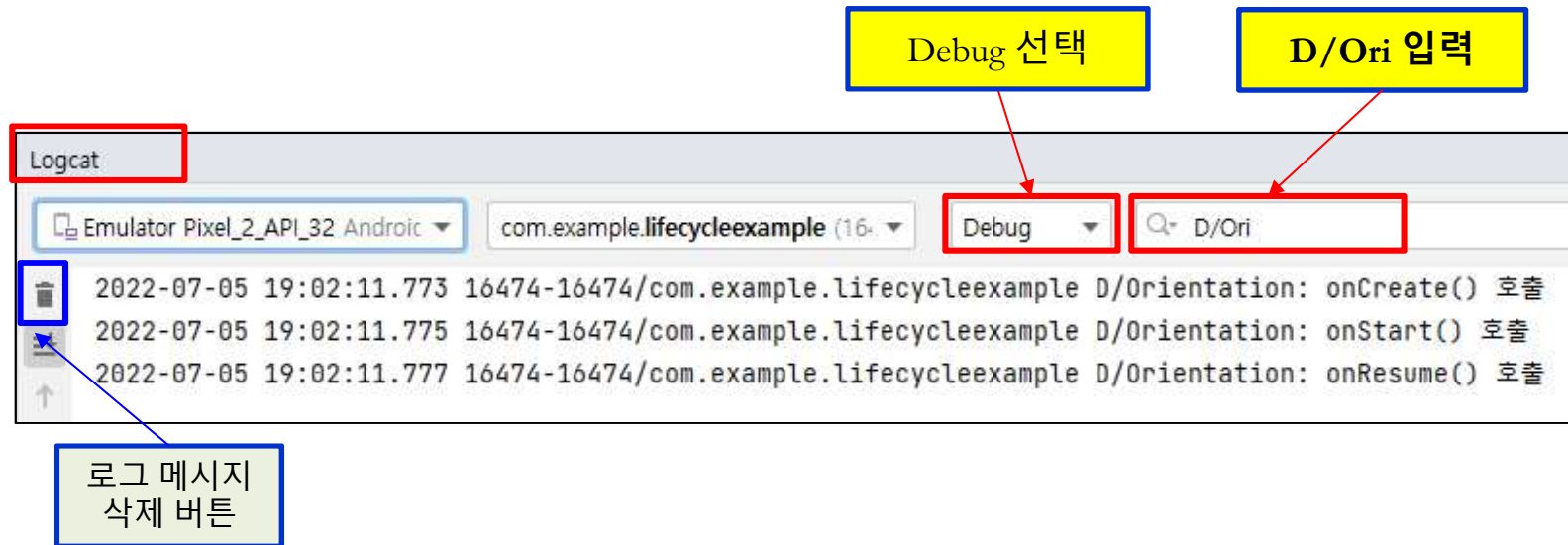
잠깐! Log와 LogCat

- 개발 중인 App 코드를 디버깅하는 간단한 방법
 - 단말이나 AVD에서 동작을 확인하기 전 단계에서 사용
 - 속성 등 확인하고 싶은 값을 콘솔(LogCat 창)에 출력
 - Callback 메소드가 언제 호출되는지 확인도 가능

```
Log.d(TAG, "onCreate() 호출")
```

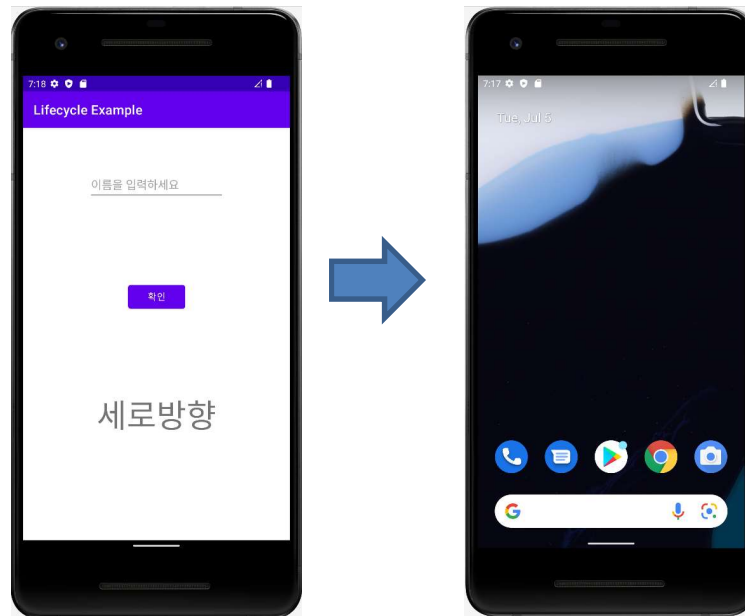
- Log 클래스
 - **Log 레벨** – 메시지 중요도나 사용 목적을 구분
 - **d**(debug), **i**(info), **v**(verbose)
 - **e**(error)
 - **w**(warning)
 - **TAG** – 구분을 위한 문자열(누가 출력을 요청했는가?)
 - **문자열** – 출력하려는 내용
 - 변수 등은 string template를 사용해 문자열에 포함.

상태 변화: Activity 생성 (1/4)



상태 변화: Activity 멈춤 (2/4)

```
Logcat
Emulator Pixel_2_API_32 Android com.example.lifecycleexample (16) Debug D/Ori
2022-07-05 19:16:42.790 16784-16784/com.example.lifecycleexample D/Orientation: onPause() 호출
2022-07-05 19:16:43.281 16784-16784/com.example.lifecycleexample D/Orientation: onStop() 호출
```



Activity가
화면에 보이지 않지만
완전히 destroy되지는
않았다!

로그 메시지 삭제 → Recent App. 버튼 클릭

상태 변화: Activity 재실행 (3/4)

```
Logcat
Emulator Pixel_2_API_32 Android com.example.lifecycleexample (16) Debug D/Ori
2022-07-05 19:23:59.495 16970-16970/com.example.lifecycleexample D/Orientation: onStart() 호출
2022-07-05 19:23:59.495 16970-16970/com.example.lifecycleexample D/Orientation: onResume() 호출
2022-07-05 19:23:59.496 16970-16970/com.example.lifecycleexample D/Orientation: onResume() 호출
```



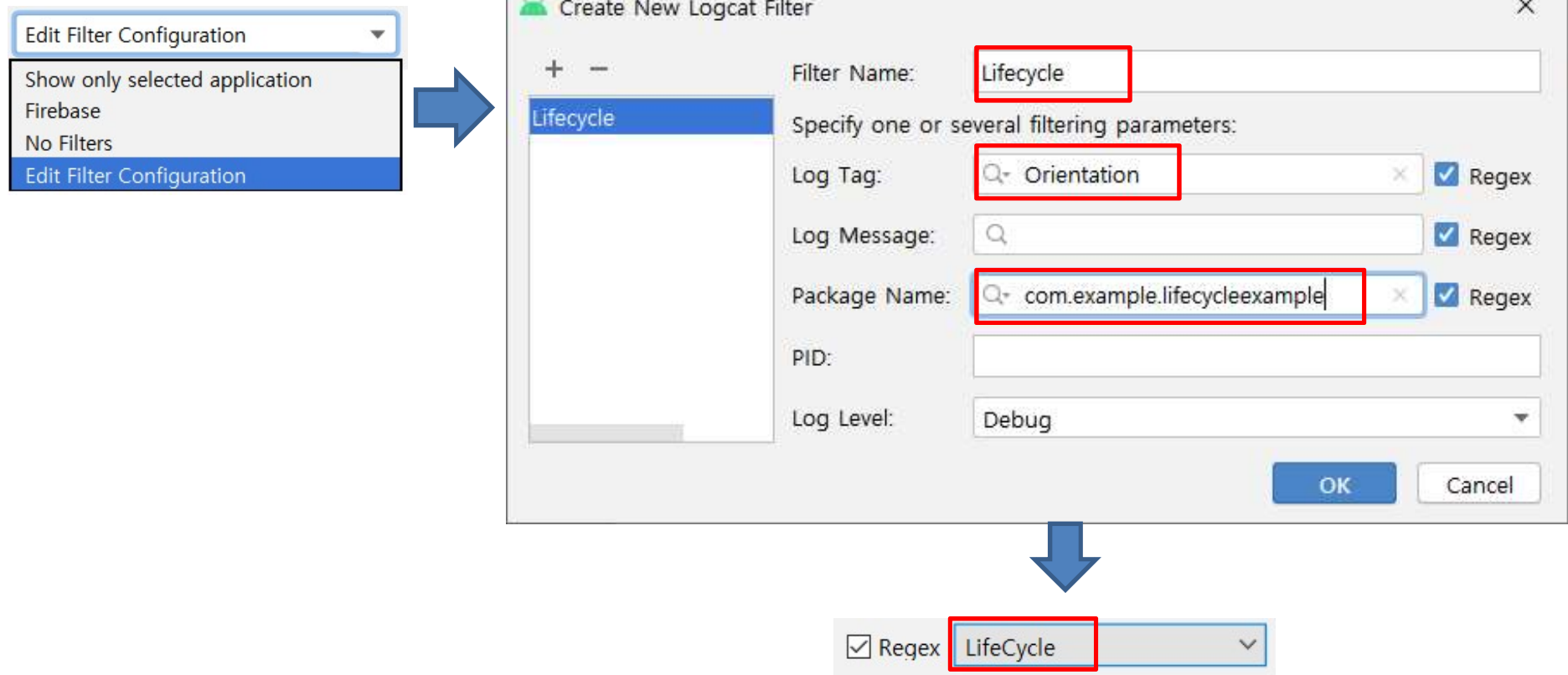
상태 변화: Activity 종료 (4/4)

```
Logcat
Emulator Pixel_2_API_32 Android com.example.lifecycleexample (18 Debug D/Ori
2022-07-05 19:43:51.204 18409-18409/com.example.lifecycleexample D/Orientation: onPause() 호출
2022-07-05 19:43:51.641 18409-18409/com.example.lifecycleexample D/Orientation: onStop() 호출
```



Activity가
메모리에서
완전히 제거됨!!

잠깐! LogCat 창에서 Filter 만들기



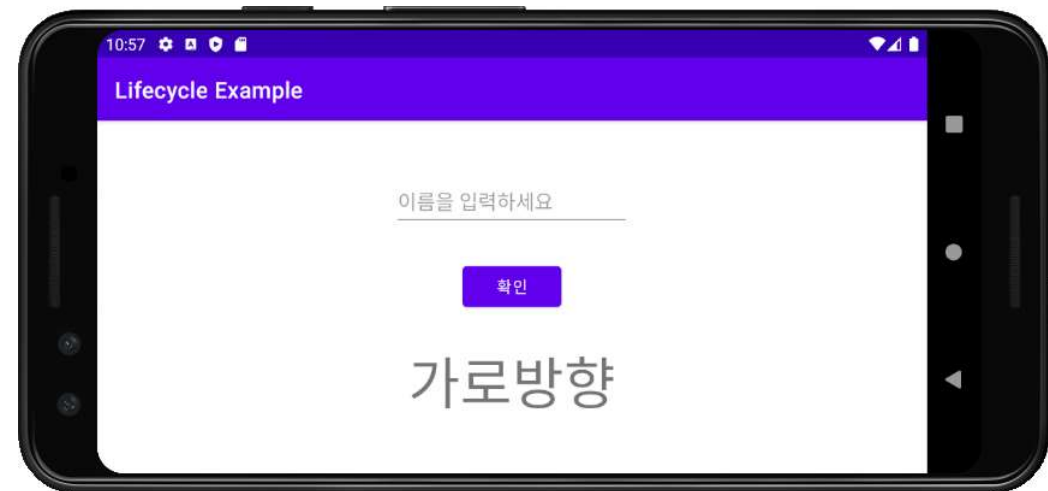
What to do next?

- Activity Stack
- Activity State
- Activity Lifecycle
- 실습 1
- **실습 2**
 - 화면 회전
- 실습 3
- 실습 4

실습 2 : 화면 회전



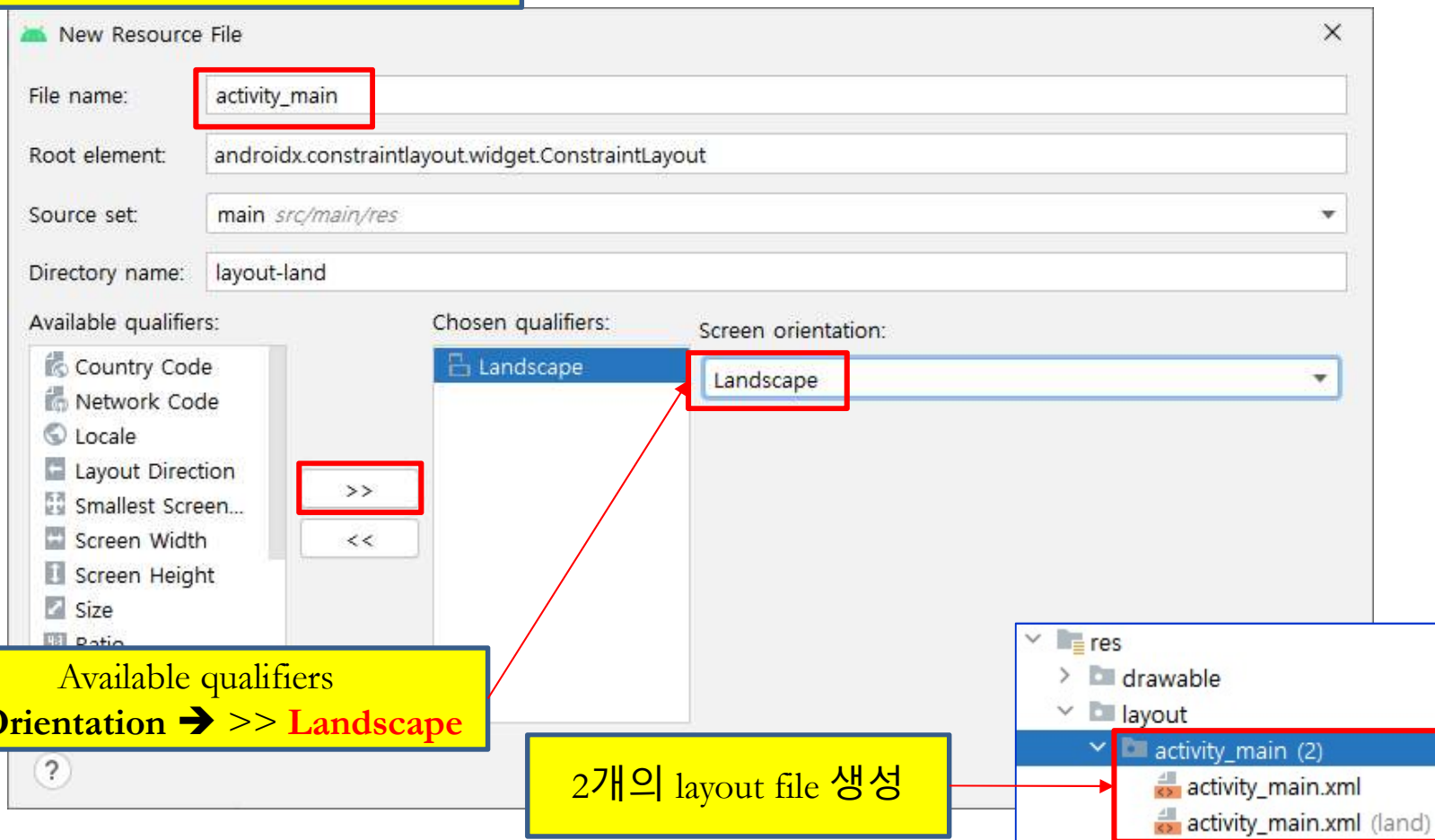
Portrait mode



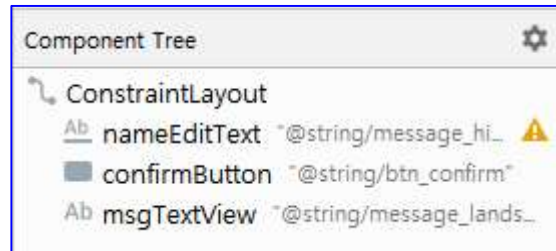
Landscape mode

실습 2: Landscape layout 추가

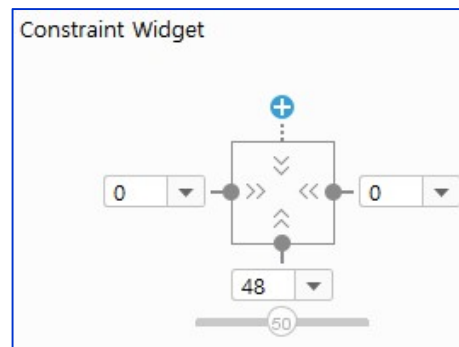
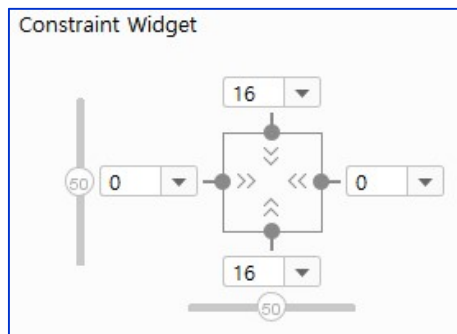
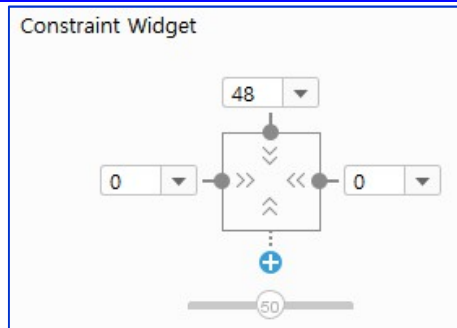
App > res > **layout** > 오른쪽 버튼 >
New > Layout Resource File



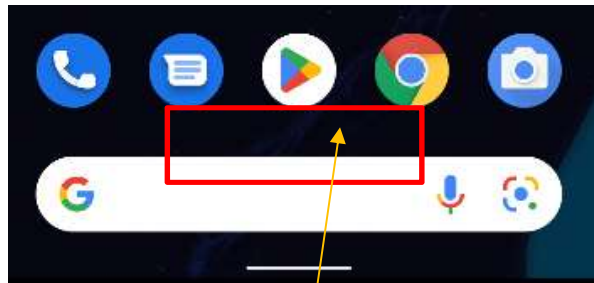
실습 2: Layout - Landscape



activity_main.xml(land)

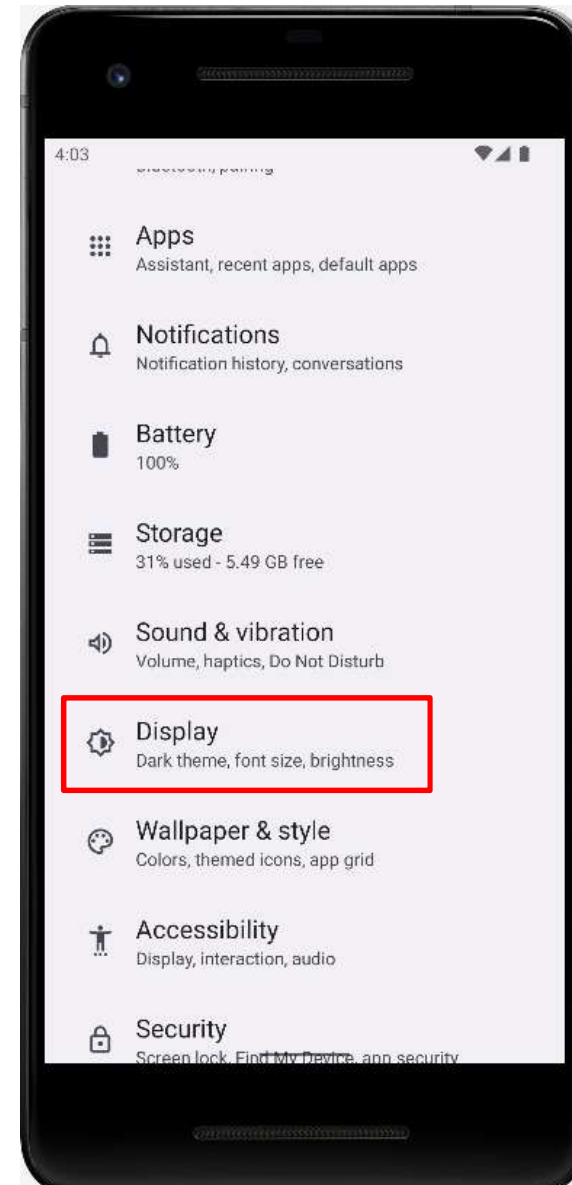
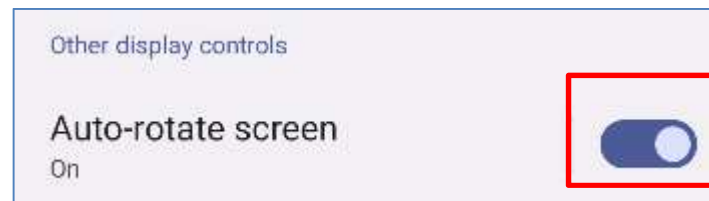
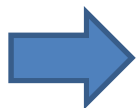
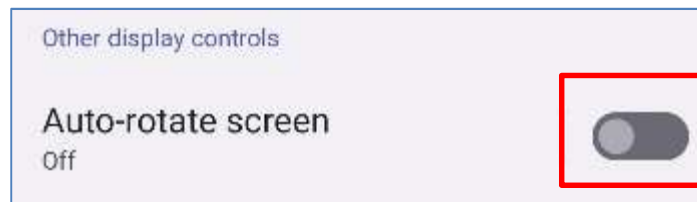


잠깐! 화면 회전



위쪽으로 drag

Settings
> Display



실습 2: 실행 결과 (1/2)

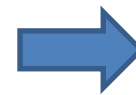
```
Logcat
Emulator Pixel_2_API_32 Android com.example.lifecycleexample (18) Debug
2022-07-05 20:02:58.715 18931-18931/com.example.lifecycleexample D/Orientation: onCreate() 호출
2022-07-05 20:02:58.716 18931-18931/com.example.lifecycleexample D/Orientation: onStart() 호출
2022-07-05 20:02:58.717 18931-18931/com.example.lifecycleexample D/Orientation: onResume() 호출
2022-07-05 20:03:07.988 18931-18931/com.example.lifecycleexample D/Orientation: onPause() 호출
2022-07-05 20:03:07.990 18931-18931/com.example.lifecycleexample D/Orientation: onStop() 호출
2022-07-05 20:03:08.001 18931-18931/com.example.lifecycleexample D/Orientation: onDestroy() 호출
2022-07-05 20:03:08.072 18931-18931/com.example.lifecycleexample D/Orientation: onCreate() 호출
2022-07-05 20:03:08.073 18931-18931/com.example.lifecycleexample D/Orientation: onStart() 호출
2022-07-05 20:03:08.074 18931-18931/com.example.lifecycleexample D/Orientation: onResume() 호출
```



실습 2: 실행 결과 (2/2)

```
Logcat
Emulator Pixel_2_API_32 Android com.example.lifecycleexample (18) Debug
2022-07-05 20:06:08.859 18931-18931/com.example.lifecycleexample D/Orientation: onPause() 호출
2022-07-05 20:06:08.865 18931-18931/com.example.lifecycleexample D/Orientation: onStop() 호출
2022-07-05 20:06:08.868 18931-18931/com.example.lifecycleexample D/Orientation: onDestroy() 호출
2022-07-05 20:06:08.912 18931-18931/com.example.lifecycleexample D/Orientation: onCreate() 호출
2022-07-05 20:06:08.913 18931-18931/com.example.lifecycleexample D/Orientation: onStart() 호출
2022-07-05 20:06:08.914 18931-18931/com.example.lifecycleexample D/Orientation: onResume() 호출
```

오른쪽으로 90도 회전
(원래 화면)



Do it yourself (1) : level-easy

- 실습 2에서
 - EditText 에 "HONG"(여러분 성)을 입력
 - 90도 회전
 - EditText 창에 이전에 입력한 글자가 나타났을까?

- 레이아웃 파일 2개 모두에서

- EditText 의 속성에
 - **saveEnabled** 속성을 **false**로 지정
- EditText 에 "HONG"(여러분 성)을 입력
- 90도 회전 ➡ EditText 에 이전에 입력한 글자가 나타났을까?
- 다시 90도 회전 ➡ EditText 에 이전에 입력한 글자가 나타났을까?



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


What to do next?

- Activity Stack
- Activity State
- Activity Lifecycle
- 실습 1
- 실습 2
- **실습 3**
 - **상태 저장 및 복원**
- 실습 4

2개의 callback method 추가

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

MainActivity.kt

```
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        Log.d(TAG, "onCreate() 호출")  
    }
```

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


parameter가 1개입니다.

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

```
    override fun onStart() {...}  
    override fun onResume() {...}  
    override fun onPause() {...}  
    override fun onStop() {...}  
    override fun onDestroy() {...}
```

```
}
```

Do it yourself (2) : level-medium

- 실습 목적
 - Back 버튼을 누르거나 화면을 회전하면 **완전히 새로운 activity instance 가 생성**
 - 이전 activity instance의 상태 저장이나 복원이 불가능
- Missions to be completed
 - 아래와 같이 실행시킬 때 **onSaveInstanceState와 onRestoreInstanceState 메소드는 언제 호출되는가?**
 - LogCat 창의 값을 지우려면 →  클릭
 - 첫 번째 실습
 - App 실행 → Back 버튼을 누른다
 - 대기 중인 App을 다시 실행시킨다.
 - Home 버튼을 누른다.
 - 두 번째 실습
 - App 실행 → 90도 회전시킨다(landscape mode)
 - 다시 90도 회전해 원래 화면으로 돌아온다(portrait mode)

Hint : 아래 사이트의 설명을 참고하세요.

<https://developer.android.com/guide/components/activities/activity-lifecycle#restore-activity-ui-state-using-saved-instance-state>

실습 3: onSaveInstanceState (1/2)

- onSaveInstanceState 메소드
 - 화면 회전으로 이전 Activity가 완전히 없어지기 전에
 - **onStop** 메소드 호출 시점과 **onDestroy** 메소드 호출 시점 사이에서 호출.
- 매개변수 **outState: Bundle**
 - Activity 소멸 전에 값을 outState에 저장할 수 있음.
 - **Bundle**
 - 어떠한 type의 데이터도 (**이름, 값**) 형태로 저장할 수 있음.
 - outState 객체는 **onCreate** 메소드의 매개변수로 전달됨.
 - onCreate는 Activity가 새롭게 만들어질 때 제일 처음 호출됨.

실습 3: onSaveInstanceState (2/2)

```
override fun onSaveInstanceState(outState: Bundle) {  
    super.onSaveInstanceState(outState)  
  
    var str = nameEditText.text.toString()  
    Log.d(TAG, "onSaveInstanceState::strValue = $str")  
    outState.putString(KEY_NAME, str)  
}
```

putString (이름, 값)

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    nameEditText = findViewById(R.id.nameEditText)  
    Log.d(TAG, msg: "onCreate() 호출")  
  
    if (savedInstanceState?.isEmpty == false) {  
        var str:String? = savedInstanceState?.getString(KEY_NAME)  
        nameEditText.setText(str)  
    }  
}
```

private const val KEY_NAME = "name"

getString (이름)

Bundle 클래스
key-value pair 형태로 데이터를 저장

실습 3: Activity

```
private val TAG = "Orientation"
private val KEY_NAME = "first_name"
private lateinit var nameEditText: EditText

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    nameEditText = findViewById(R.id.nameEditText)
    Log.d(TAG, msg: "onCreate() 호출")

    if (savedInstanceState?.isEmpty == false) {
        var str:String? = savedInstanceState?.getString(KEY_NAME)
        nameEditText.setText(str)
    }
}

override fun onSaveInstanceState(outState: Bundle) {...}
override fun onRestoreInstanceState(savedInstanceState: Bundle) {
    super.onRestoreInstanceState(savedInstanceState)
    Log.d(TAG, msg: "onRestoreInstanceState() 호출")

    var str = savedInstanceState.getString(KEY_NAME)
    Toast.makeText(applicationContext, text: "복원한 문자열은 $str",
        Toast.LENGTH_SHORT).show()
}
```

EditText 객체를 선언
단, 초기화는 나중에 함.

EditText 객체에
초기값 할당.

복원된 값을
Toast 창에 출력

잠깐! companion object

```
private const val TAG = "Orientation"
private const val KEY_NAME = "name"

class MainActivity : AppCompatActivity() {
```



```
private const val TAG = "Orientation"

class MainActivity : AppCompatActivity() {
    companion object {
        const val KEY_NAME = "name"
    }
}
```

```
class MainActivity : AppCompatActivity() {
    companion object {
        const val KEY_NAME = "name"
        const val TAG = "Orientation"
    }
}
```

클래스 바깥에 선언하면
App. 에 속한 모든 클래스에서 사용

Kotlin에서 object 키워드는
singleton을 의미
→ singleton이란
클래스의 instance(객체)가
하나 밖에 없는 클래스.

companion object
singleton 객체이면서
동반 객체.
→ MainActivity 클래스와
함께 가는
친구 객체.
→ 클래스에서 companion object는
한 개만 정의할 수 있음.

Do it yourself (3) : level-easy

- 실습 3에서 아래 내용을 확인

- EditText 에 "HONG"을 입력

- **90도 회전**

- EditText 창에 이전에 입력한 글자가 나타났을까?

- **다시 반대 방향으로 90도 회전**

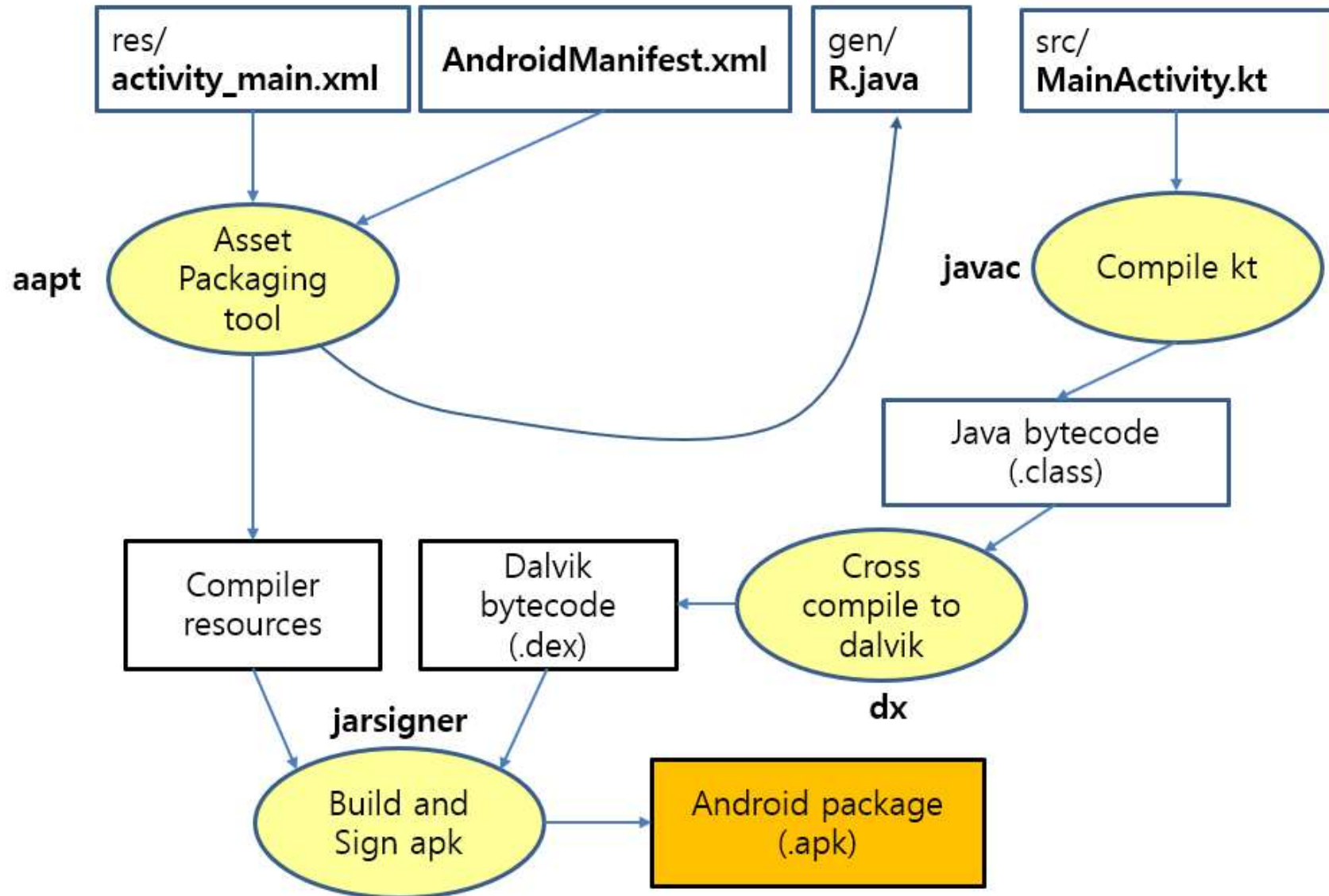
- EditText 창에 이전에 입력한 글자가 나타났을까?

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


What to do next?

- Activity Stack
- Activity State
- Activity Lifecycle
- 실습 1
- 실습 2
- 실습 3
- **실습 4**
 - **ViewModel을 사용한 동적 상태 저장 및 복원**

안드로이드 앱 생성 과정



dependencies 속성 추가

lifecycle-extensions API 라이브러리 추가

아티팩트	공개 버전	출시 후보	베타 버전	알파 버전
lifecycle-*	2.4.1	2.5.0-rc02	-	-
lifecycle-viewmodel-compose	2.4.1	2.5.0-rc02	-	-

Gradle Scripts
build.gradle (Project: RecyclerViewExample)
build.gradle (Module: RecyclerViewExample.app)

dependencies 블록 안에 아래 문장 추가

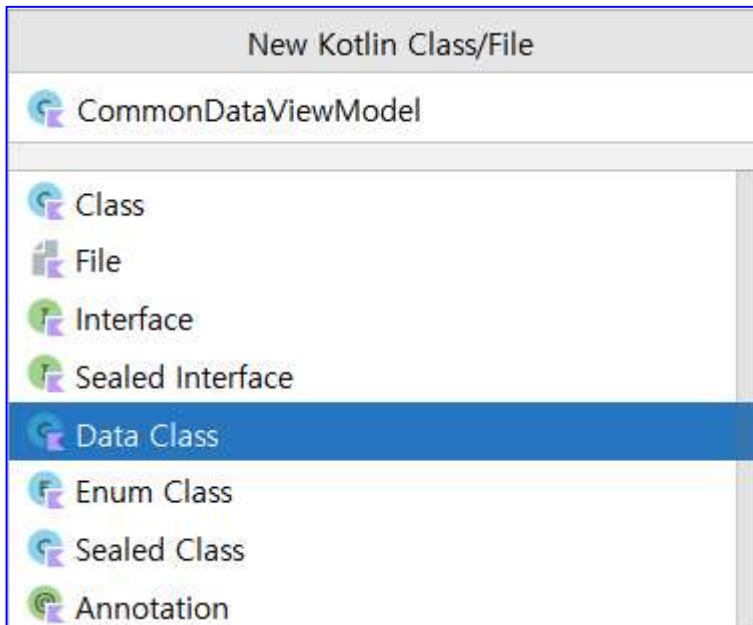
```
dependencies {  
    implementation 'androidx.core:core-ktx:1.7.0'  
    implementation 'androidx.appcompat:appcompat:1.4.2'  
    implementation 'com.google.android.material:material:1.6.1'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
}
```

Sync Now

Ignore these changes

Gradle이 동기화하도록 Sync Now 클릭!

ViewModel 클래스



```
import androidx.lifecycle.ViewModel

class CommonDataViewModel : ViewModel() {
    var inputString = ""

    fun saveString(str:String) {
        inputString = str
    }
}
```

ViewModel을 사용한 동적 상태 저장

```
private const val TAG = "Orientation"

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val dataViewModel =
            ViewModelProvider( owner: this)[CommonDataViewModel::class.java]

        var nameEditText: EditText = findViewById(R.id.nameEditText)
        var confirmButton: Button = findViewById(R.id.confirmButton)
        Log.d(TAG, msg: "onCreate() 호출")

        nameEditText.setText(dataViewModel.inputString)
        confirmButton.setOnClickListener { it: View!
            dataViewModel.saveString(nameEditText.text.toString())
            nameEditText.setText(dataViewModel.inputString)
        }
    }
}
```

```
import androidx.lifecycle.ViewModel

class CommonDataViewModel : ViewModel() {
    var inputString = ""

    fun saveString(str: String) {
        inputString = str
    }
}
```