모바일 프로그래밍

08 액티비티 생명주기

2017 2학기

강승우

Computer Science and Engineering, KOREATECH

액티비티 생명주기 (Activity life cycle)

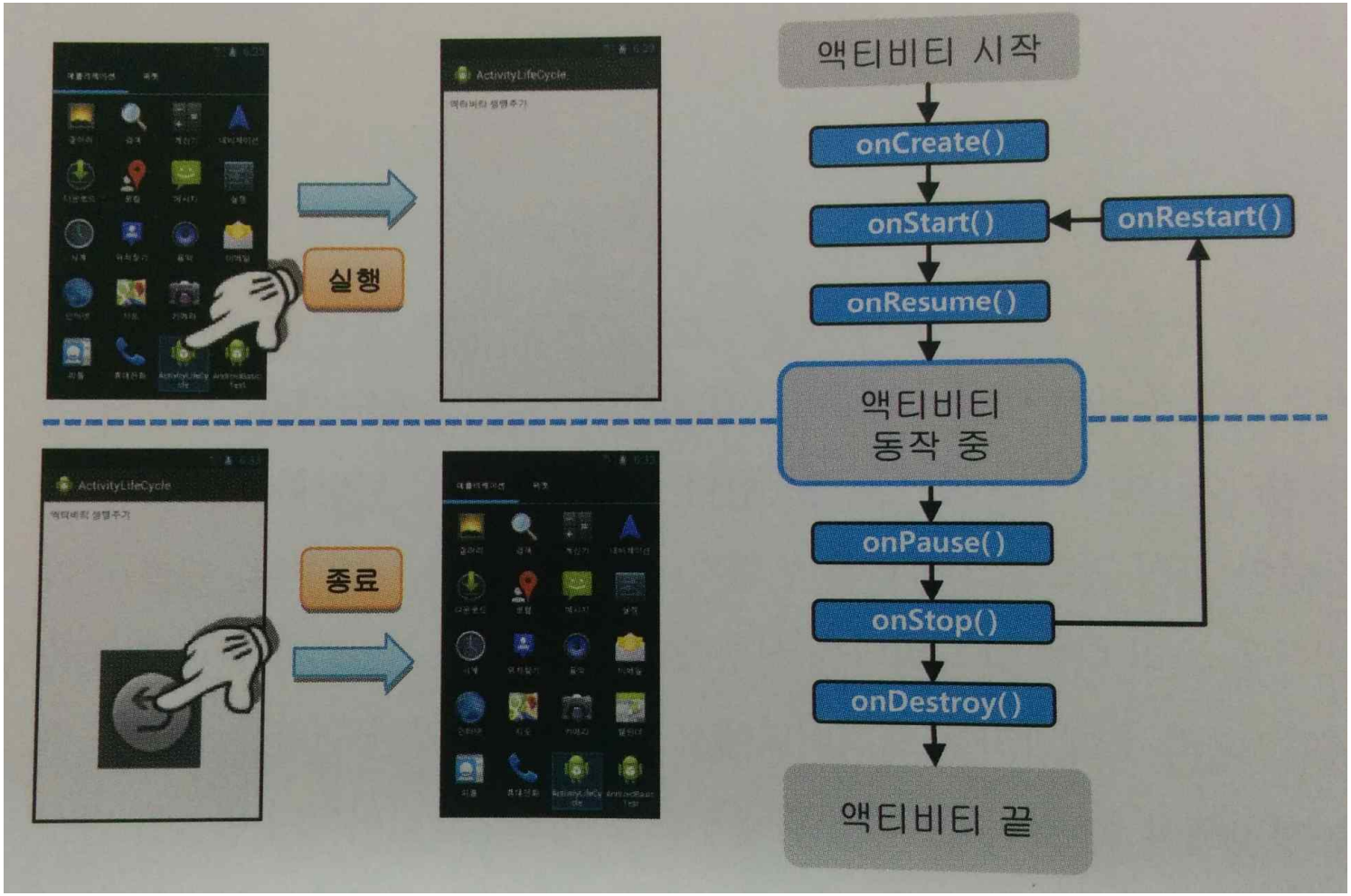
- 액티비티의 3가지 기본 상태
  - 실행 상태
    - 액티비티가 전경(foreground)에 있으며 사용자의 포커스를 가지고 있어 인터랙션을 할 수 있는 상태
  - 일시 멈춤 상태
    - 다른 액티비티가 전경에 있으며 포커스를 가지고 있지만 이전 액티비티의 일부가 여전히 화면에 보이고 있는 상태. 이때 이전 액티비티는 일시 멈춤 상태에 있음
    - 일시 멈춤 상태에 있어도 모든 상태를 유지
    - 시스템이 low memory 상태가 되면 종료될 수 있음
  - 정지 상태
    - 액티비티가 화면에서 전혀 보이지 않으며 배경(background)에 위치하고 있는 상태
    - 시스템이 메모리가 필요(low memory 상태)하면 언제든지 종료될 수 있음

액티비티 상태 변화

- 액티비티의 동적인 상태 변화
  - 액티비티는 여러 가지 내/외부적인 요인에 의해서 상태 변화 가능
    - 사용자가 어떤 액티비티에서 작업을 하던 중 전화가 오면 화면이 바뀜
    - 사용자가 어떤 액티비티 화면을 보다가 홈 버튼을 누르면 홈 화면으로 바뀜
    - 멀티태스킹 버튼(최근 사용 앱 목록 표시)을 누르고 이전에 사용한 앱을 선택하면 이전 액티비티 화면으로 바뀜
    - 스마트폰을 가로/세로 회전을 시키면 화면 사이즈가 바뀜
    - ……
- 상태 변화에 대비
  - 액티비티의 다양한 상태 변화에 맞추어 적절한 대비를 하여야 애플리케이션이 정상적인 동작을 할 수 있음

  → 상태 변화에 대비하여 적절한 조치를 취할 수 있도록
  액티비티 생명주기 함수를 제공

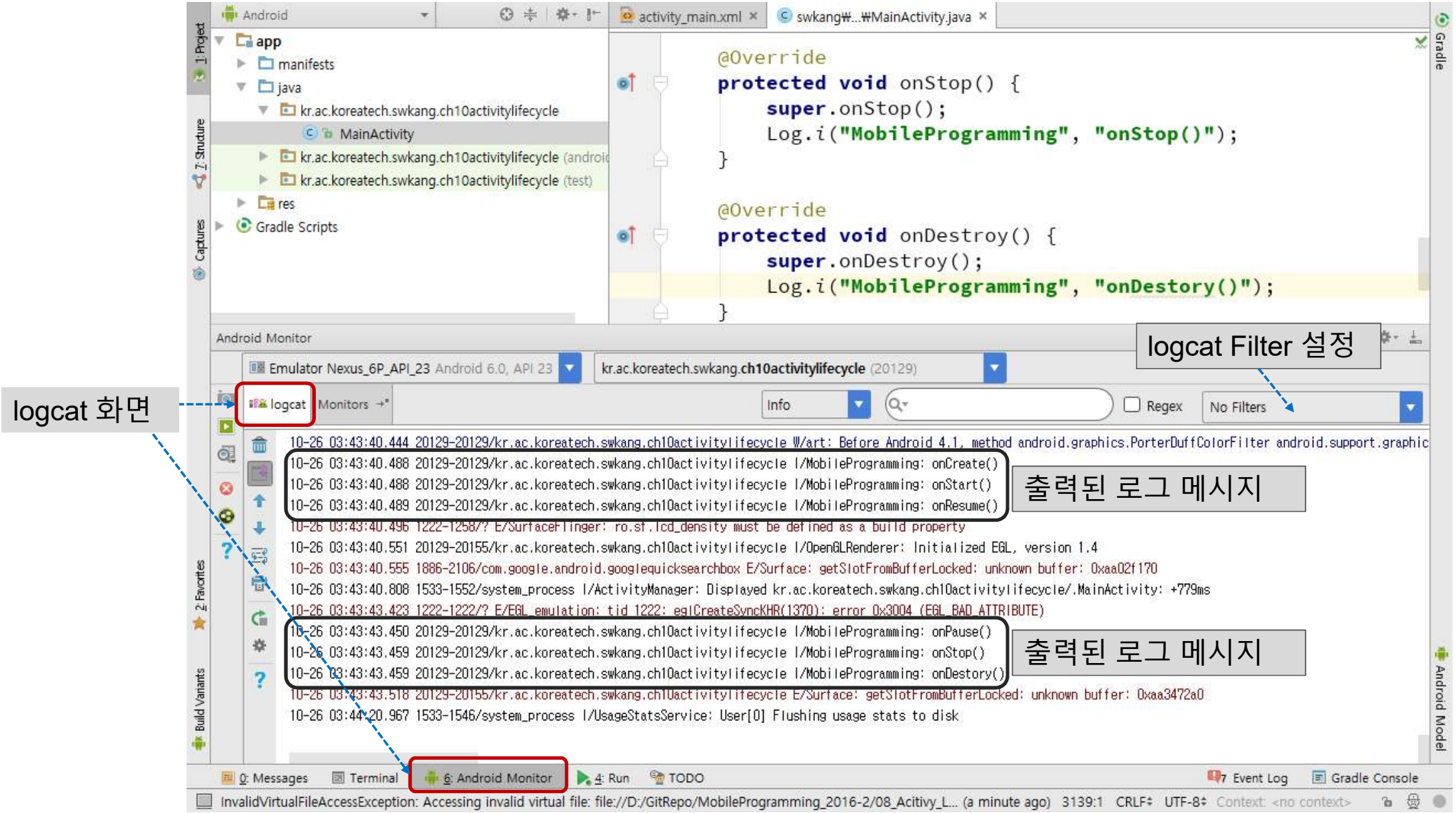
액티비티 생명주기 함수



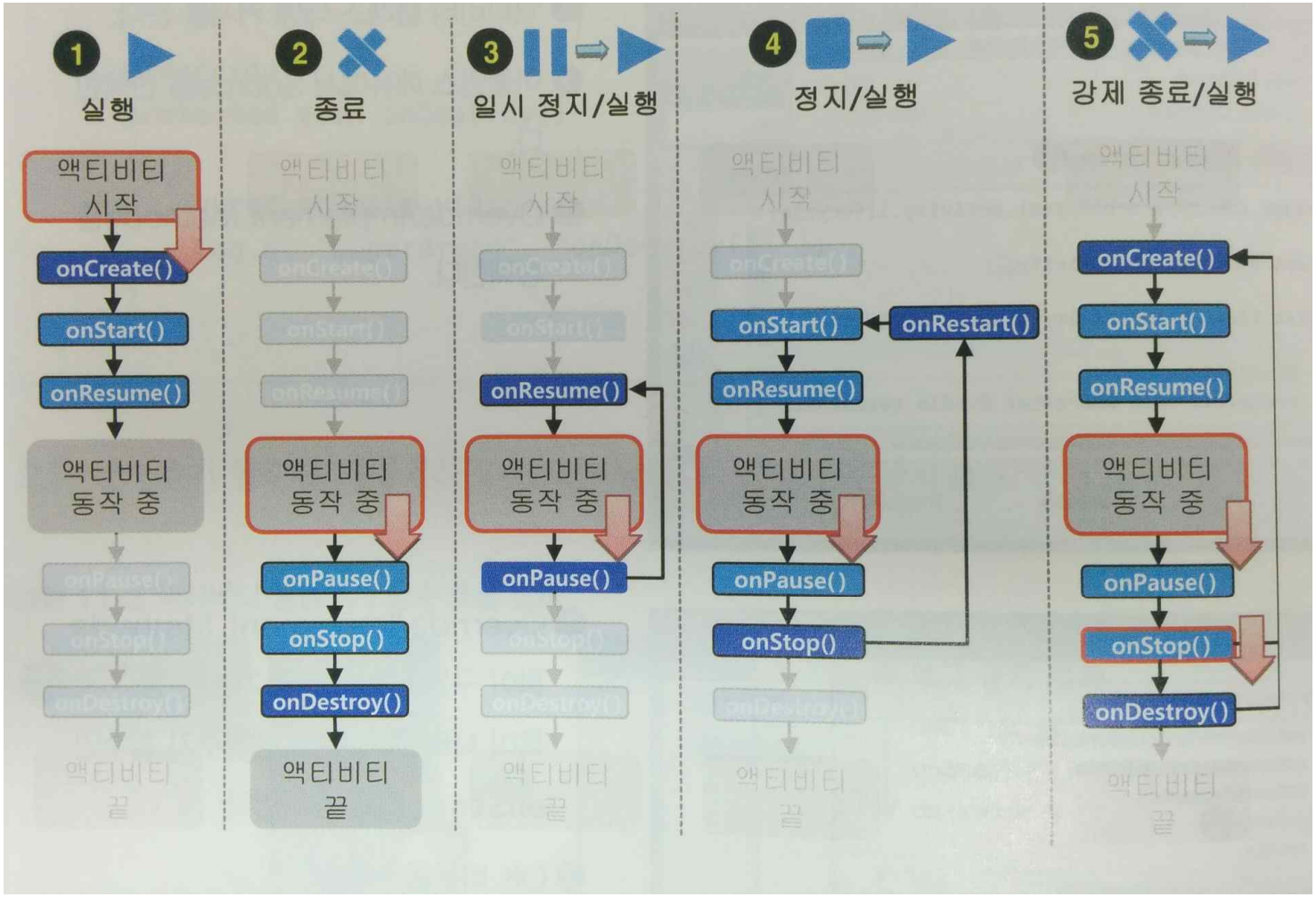
예제 프로젝트 이름

- **Ch10ActivityLifeCycle**

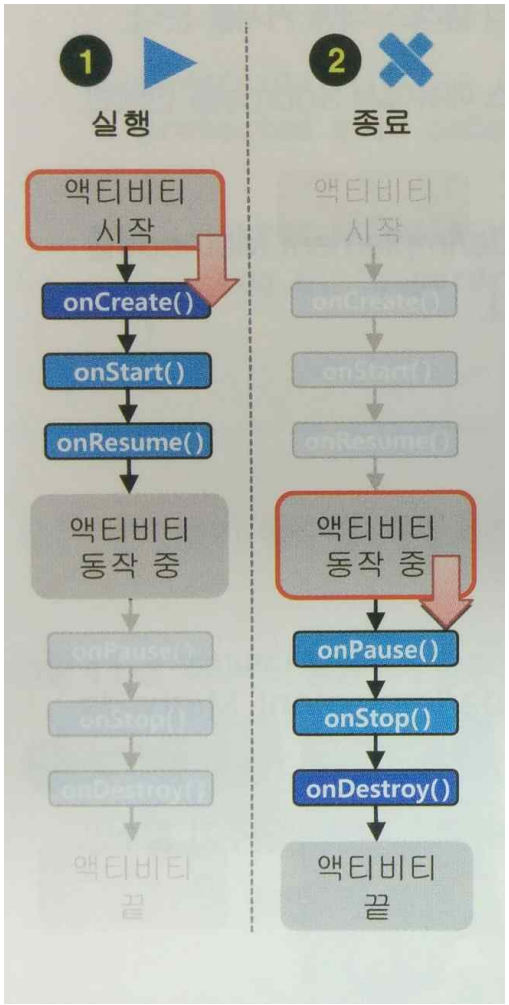
- 예제 프로젝트를 실행하고 logcat 화면에 출력되는 로그 메시지를 확인해보자


logcat 화면
logcat Filter 설정
출력된 로그 메시지
출력된 로그 메시지
Android
app
manifests
java
kr.ac.koreatech.swkang.ch10activitylifecycle
MainActivity
res
Gradle Scripts
activity_main.xml
MainActivity.java
@Override
protected void onStop() {
super.onStop();
Log.i("MobileProgramming", "onStop()");
}
@Override
protected void onDestroy() {
super.onDestroy();
Log.i("MobileProgramming", "onDestory()");
}
Android Monitor
Emulator Nexus_6P_API_23 Android 6.0, API 23
kr.ac.koreatech.swkang.ch10activitylifecycle (20129)
logcat
Monitors
Info
Regex
No Filters
10-26 03:43:40.488 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onCreate()
10-26 03:43:40.488 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onStart()
10-26 03:43:40.489 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onResume()
10-26 03:43:40.551 20129-20155/kr.ac.koreatech.swkang.ch10activitylifecycle I/OpenGLRenderer: Initialized EGL, version 1.4
10-26 03:43:40.808 1533-1552/system_process I/ActivityManager: Displayed kr.ac.koreatech.swkang.ch10activitylifecycle/.MainActivity: +779ms
10-26 03:43:43.450 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onPause()
10-26 03:43:43.459 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onStop()
10-26 03:43:43.459 20129-20129/kr.ac.koreatech.swkang.ch10activitylifecycle I/MobileProgramming: onDestory()
10-26 03:44:20.967 1533-1546/system_process I/UsageStatsService: User[0] Flushing usage stats to disk
0: Messages
Terminal
6: Android Monitor
4: Run
TODO
7 Event Log
Gradle Console
InvalidVirtualFileAccessException: Accessing invalid virtual file: file://D:/GitRepo/MobileProgramming_2016-2/08_Acitivy_L... (a minute ago) 3139:1 CRLF UTF-8 Context: <no context>
1: Project
7: Structure
Captures
2: Favorites
Build Variants
Gradle
Android Model

액티비티 상태 변화와 생명주기 함수

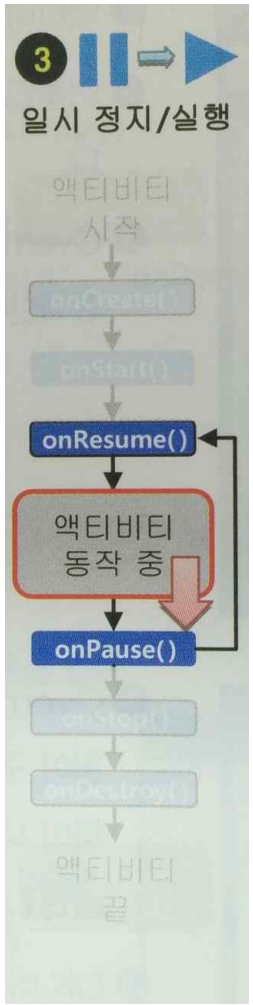


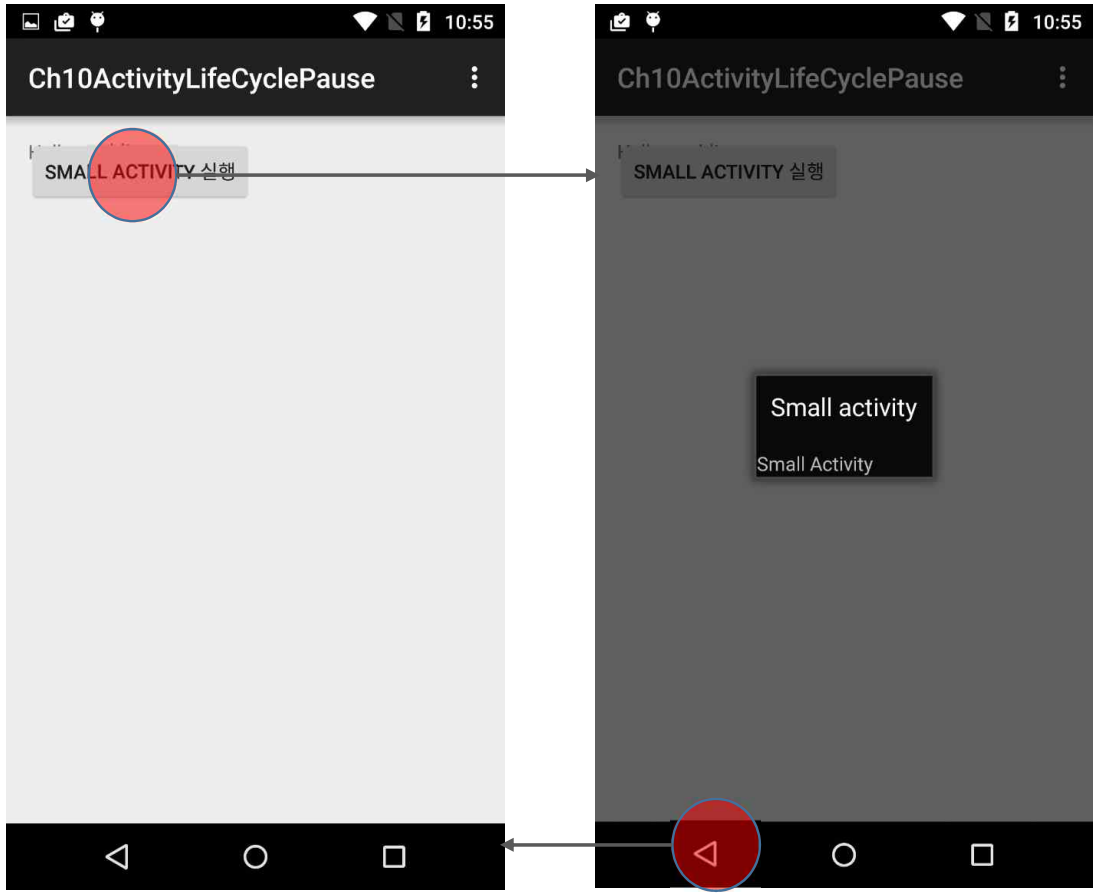
1, 2 - 액티비티 실행과 종료



- 사용자가 액티비티를 실행하고 백버튼을 눌러 종료하는 일반적인 상황
- onCreate() / onDestroy() 쌍
  - onCreate(): 한번만 실행되는 초기화 코드
    - 액티비티에서 사용하는 리소스 생성, 초기화, 사용자 인터페이스 정의, 클래스 수준의 변수 생성 및 초기화
  - onDestroy(): 실행 종료 시 사용한 리소스를 반환하는 작업 수행
- 지금까지는 onDestroy() 함수를 사용하지 않았음
  - 메모리만 사용하는 경우였기 때문에 별도의 처리가 필요 없었음
  - 파일을 사용하거나 카메라를 사용하는 등 다른 리소스 사용 시 리소스 반환(release/close)이 안 되었다면 onDestroy에서 처리 해주어야 함

3 - 액티비티 일시 정지와 재실행





예제 프로젝트 이름: Ch10ActivityLifeCyclePause

- 실행된 액티비티 뒤에 이전 액티비티가 보이는 상태
- 안드로이드에서는 이 경우 이전 액티비티로 곧 돌아갈 확률이 높다고 판단하여 일시 정지 상태로 만든다

→ onPause 함수가 호출되면 액티비티에서 진행 중인 작업을 일시 중단

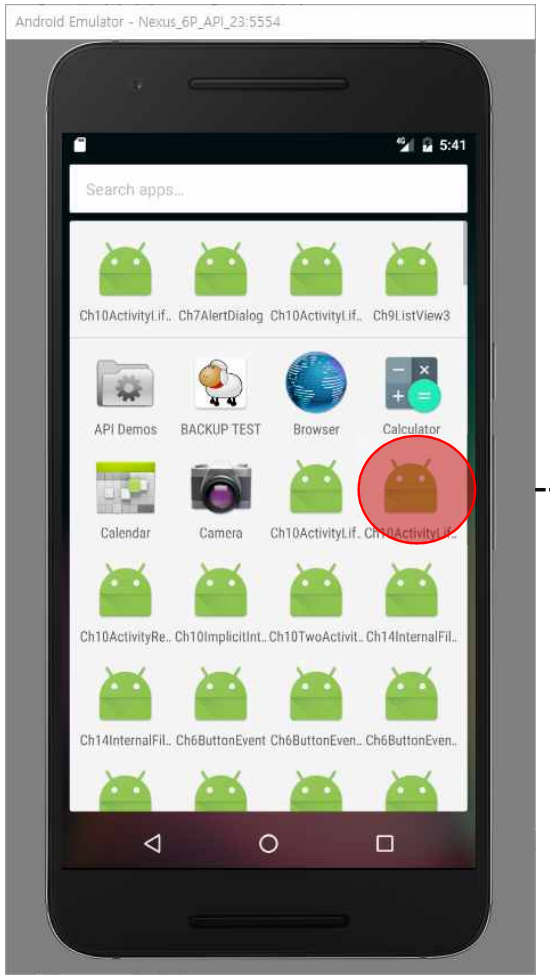
→ onResume 함수가 호출되면 중단된 작업을 재개

예)
동영상을 재생 중인 액티비티가 실행 중인데, 이 위로 작은 액티비티(예: 알림창)가 실행되어 화면을 가린다고 해보자.
이때,
onPause 함수에서 동영상 재생을 멈추고,
가린 액티비티가 종료되면
onResume 함수에서 다시 재생하도록 구현해야 할 것임

- 화면 전체를 차지 하지 않는 액티비티 만들기
  - 화면 중앙에 작은 다이얼로그(대화 상자) 액티비티로 실행

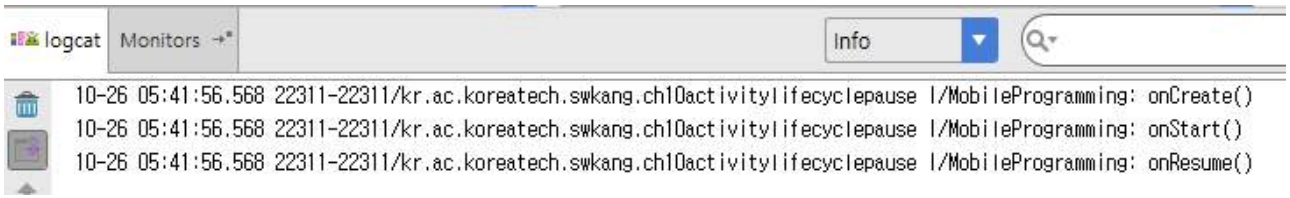
  - AndroidManifest.xml 파일에서 activity 정의 시 theme 설정을 아래와 같이 하면 대화 상자 형태의 액티비티가 됨

```
<activity android:name=".SmallActivity"
    android:label="Small activity"
    android:theme="@android:style/Theme.Dialog“>
 </activity>
```



론처 아이콘 클릭하면 → MainActivity가 실행됨

MainActivity의 생명주기 함수 onCreate() onStart() onResume() 차례로 호출됨을 알 수 있음



버튼 클릭 → SmallActivity 실행됨

MainActivity의 생명주기 함수 onPause() 호출됨

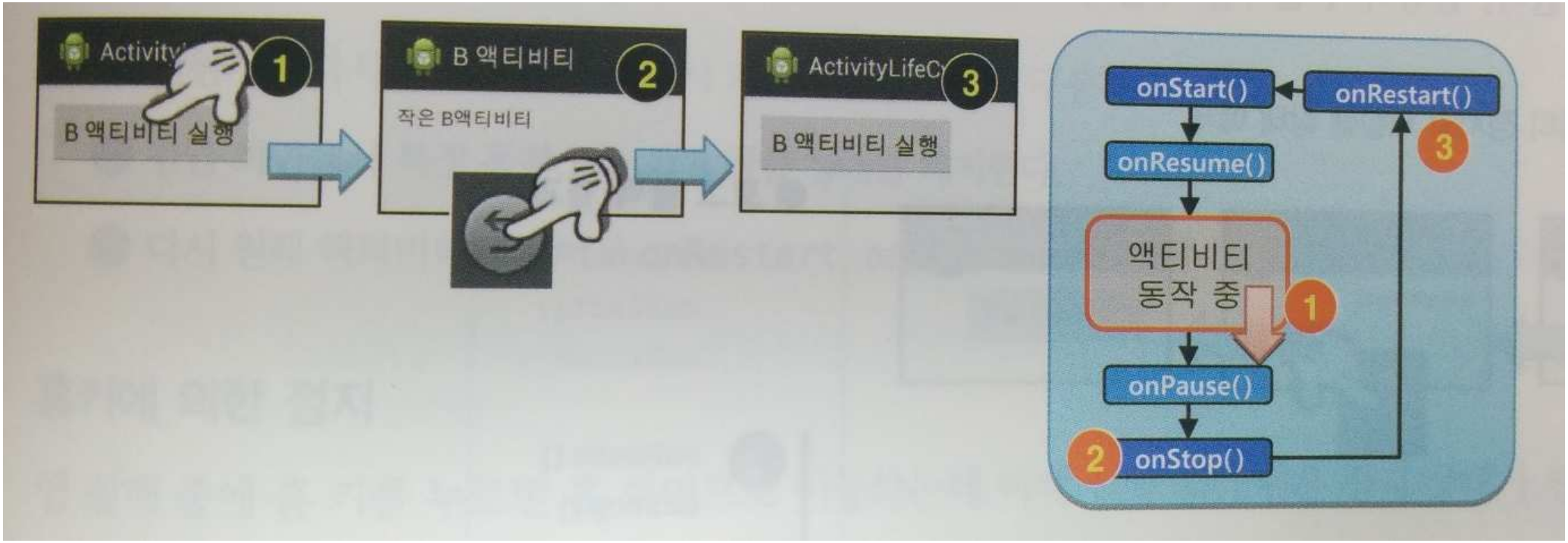
백 버튼 클릭 → SmallActivity 종료됨

MainActivity의 생명주기 함수 onResume() 호출됨


```
10-26 05:42:34.591 22311-22311/kr.ac.koreatech.swkang.ch10activitylifecyclepause I/MobileProgramming: onPause()
```

```
10-26 05:43:45.914 22311-22311/kr.ac.koreatech.swkang.ch10activitylifecyclepause I/MobileProgramming: onResume()
```

4 - 액티비티 정지와 재실행



- 앞의 예제 프로젝트(Ch10ActivityLifeCyclePause)에서 AndroidManifest.xml 파일의 android:theme="@android:style/Theme.Dialog" 를 삭제하고 실행

→ SmallActivity가 이전과 달리 전체 화면을 차지하면서 표시된다

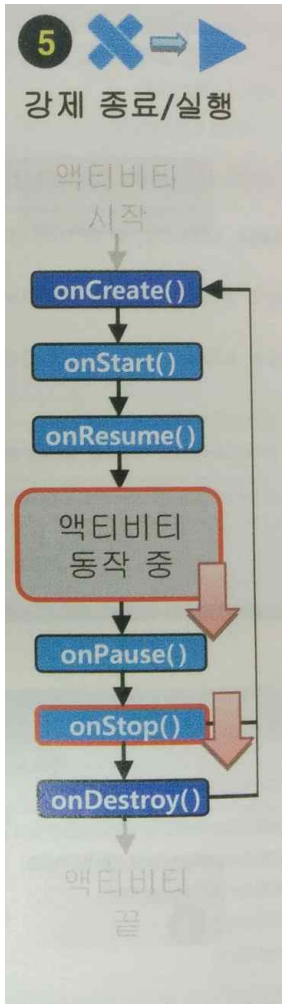
➢ logcat에 출력되는 로그 메시지로 생명주기 함수 실행을 확인해보자

액티비티 정지 상황

- 다른 액티비티 실행에 의한 정지
  - 앞의 예제와 같은 상황
- 화면 잠금에 의한 정지
  - 앱 실행 중 전원 키를 누르거나 일정 시간 터치 등의 아무 사용자 입력이 없으면 화면 잠금 상태가 된다
  - 이때 실행 중인 앱은 정지 상태가 된다
- 홈 키에 의한 정지
  - 앱 실행 중 홈 키를 누르면 홈 화면으로 이동한다
  - 이때 실행 중이던 앱은 정지 상태가 된다

➢ 앞의 예제를 실행 중인 상황에서 전원 키 or 홈 키를 누르고 logcat에 출력되는 로그 메시지를 확인해보자

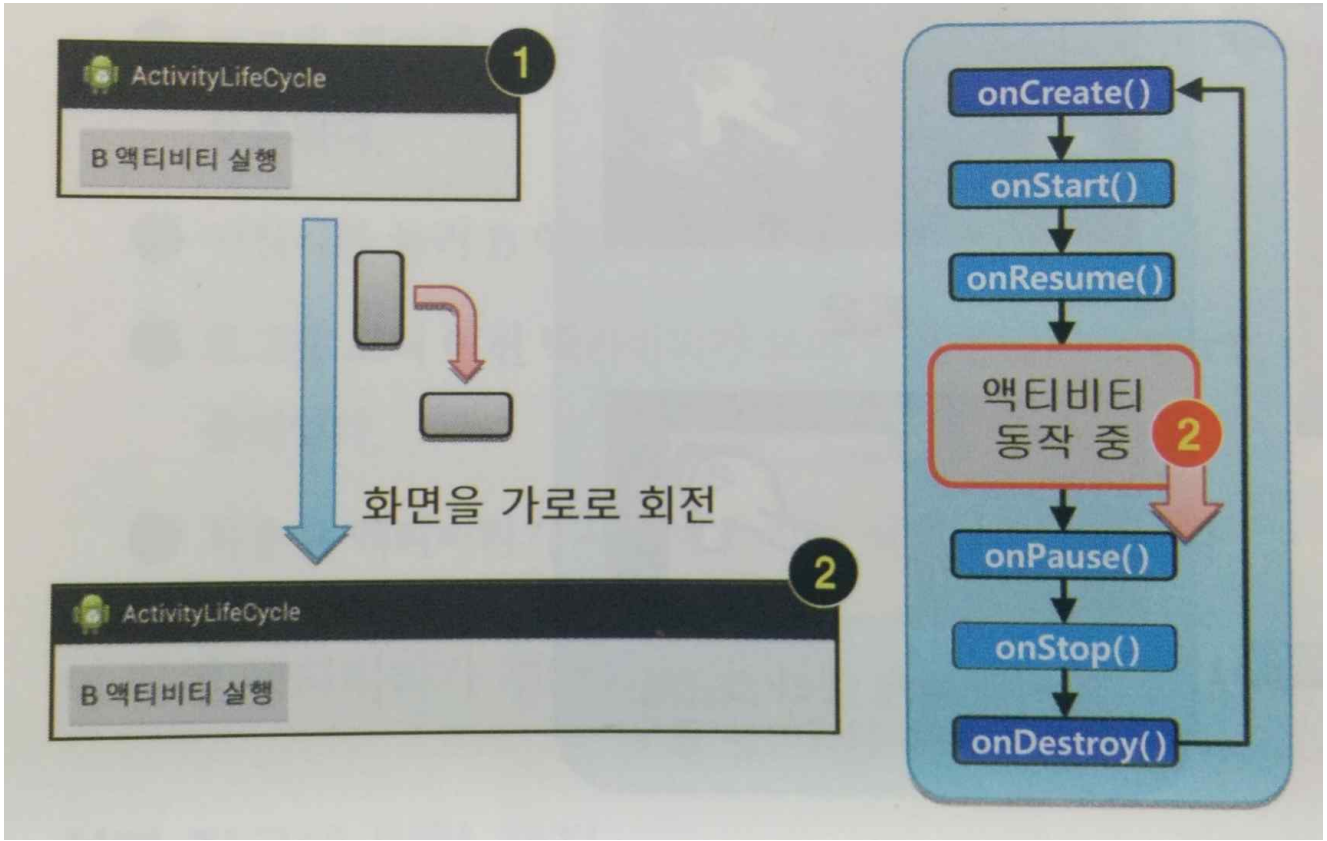
5 - 액티비티 강제 종료와 재실행



- 액티비티 강제 종료 상황
  1. 시스템 환경 변화에 의한 종료
     - 시스템 환경 설정 값이 달라지는 경우
       (예: 화면을 가로에서 세로 혹은 그 반대로 회전하는 경우)

  2. 시스템에 의한 강제 종료
     - 시스템 메모리가 부족해지는 경우
       백그라운 상태의 앱 프로세스를 안드로이드 시스템이
       강제 종료하여 메모리를 확보한다

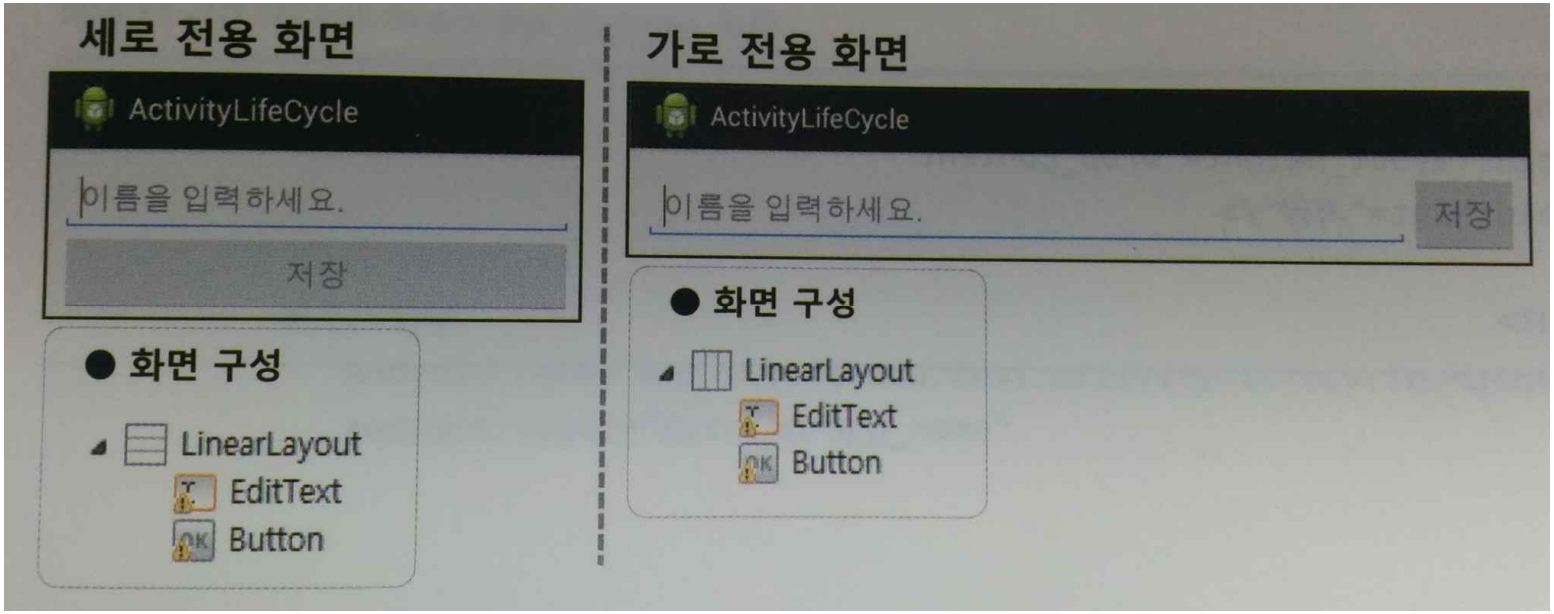
시스템 환경 변화에 의한 종료



- 화면이 회전되면 화면을 활용할 수 있는 공간이 달라진다
  →
- 화면 방향에 따라 레이아웃을 바꿔서 화면을 효과적으로 사용할 수 있다

- 화면이 전환될 때마다 다른 리소스를 적용하기 위해 안드로이드는 액티비티를 종료하고 다시 시작한다

시스템 환경 변화에 의한 종료



- 세로와 가로 전용 레이아웃을 따로 구성하여 적용하는 경우

세로 전용 레이아웃 xml 파일은 res/layout-port/activity_main.xml
가로 전용 레이아웃 xml 파일은 res/layout-land/activity_main.xml

- **예제 프로젝트 이름: Ch10ActivityLifeCycleDestroy**

- 시스템 환경 설정이 변하더라도
  액티비티를 종료하고 재실행 하도록 하지 않을 수도 있다
  - 예를 들어 가로와 세로를 구분하여 표시할 필요가 없는 화면 구성일 경우

→ 해당 Activity 클래스에 onConfigurationChanged 함수를 재정의 하여 구현한다

- onConfigurationChanged 함수 구현 시 필요 사항
  - AndroidManifest.xml 파일의 <activity> element에 configChanges 속성 추가

  → android:configChanges="orientation|screenSize"

  - 화면 방향이 바뀌거나 해상도가 변경되는 경우 액티비티 생명주기를 다시 시작하지 않고 액티비티에 재정의된 onConfigurationChanges() 함수에서 직접 처리하겠다는 의미

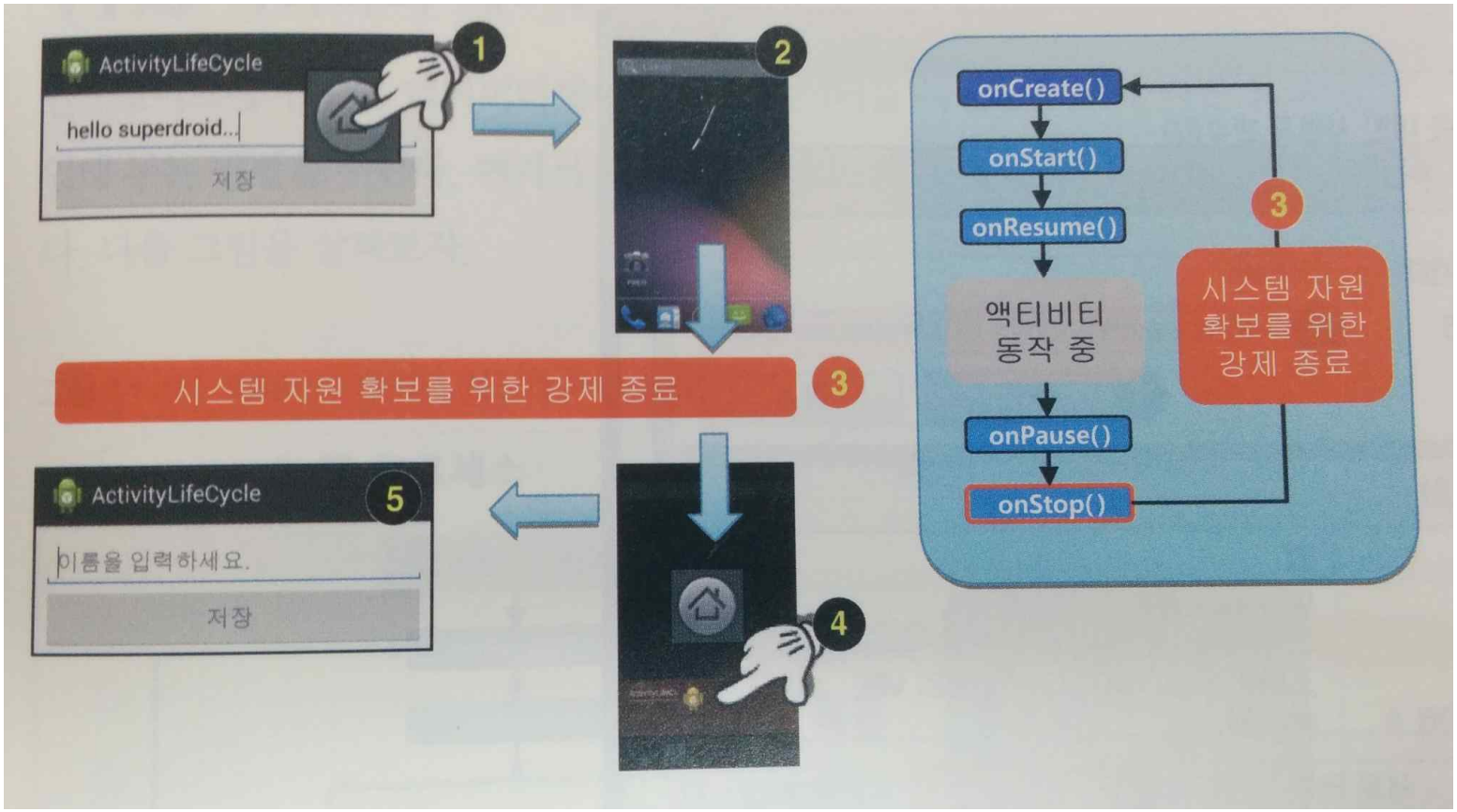
예:
```
<activity
    android:name= " .MainActivity "
    android:configChanges= " orientation|screenSize " >
```

- Activity 클래스에 onConfigurationChanged 함수 구현
- 예

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    Log.i("MobileProgramming", "onConfigurationChanged()");

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

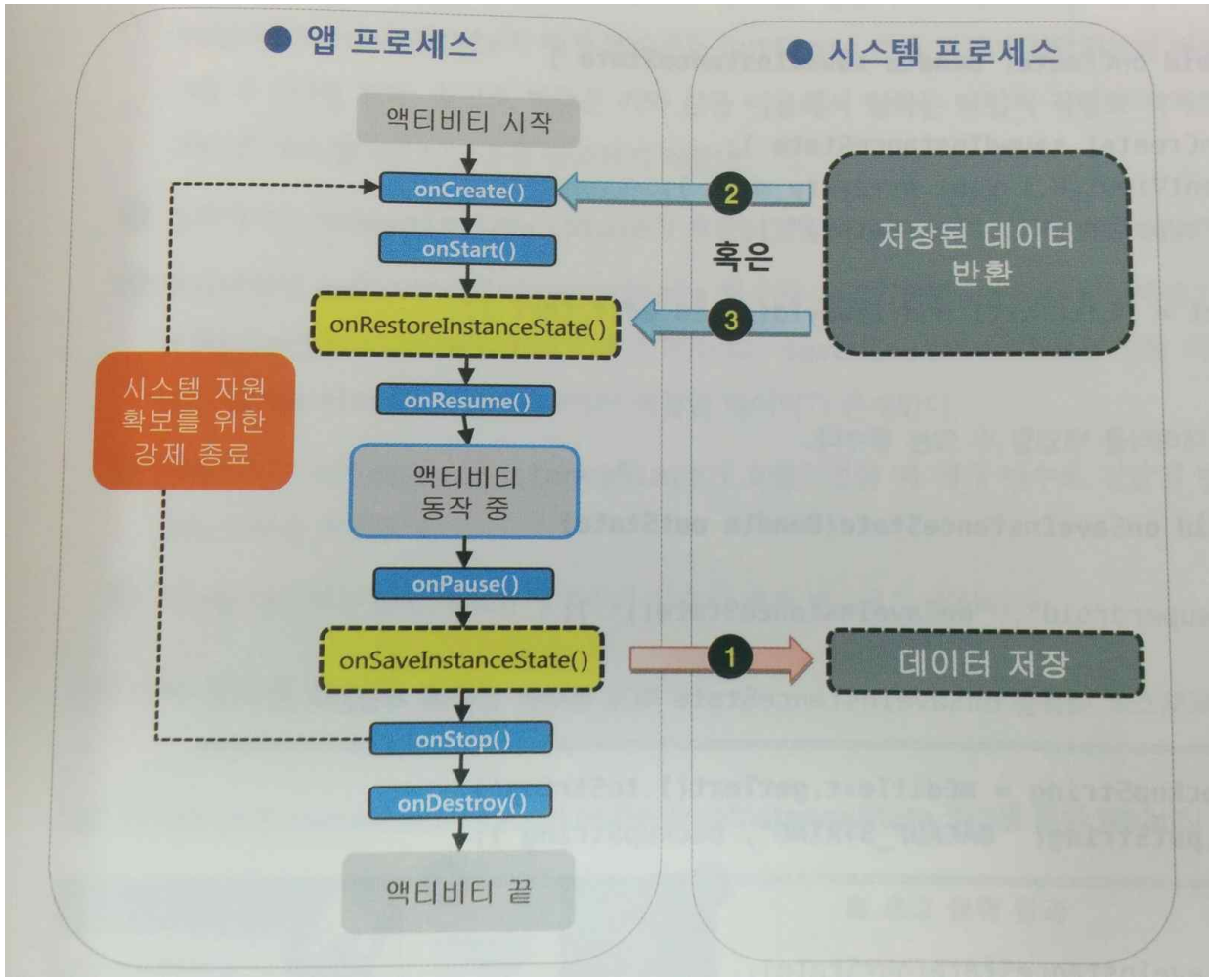
➢ 앞의 설명대로 Ch10ActivityLifeCycleDestroy 예제 프로젝트에서 AndroidManifest.xml 파일을 수정하고 위 함수 구현 후 앱을 실행시켜서 logcat에 출력되는 로그 메시지를 확인해보자

시스템에 의한 강제 종료

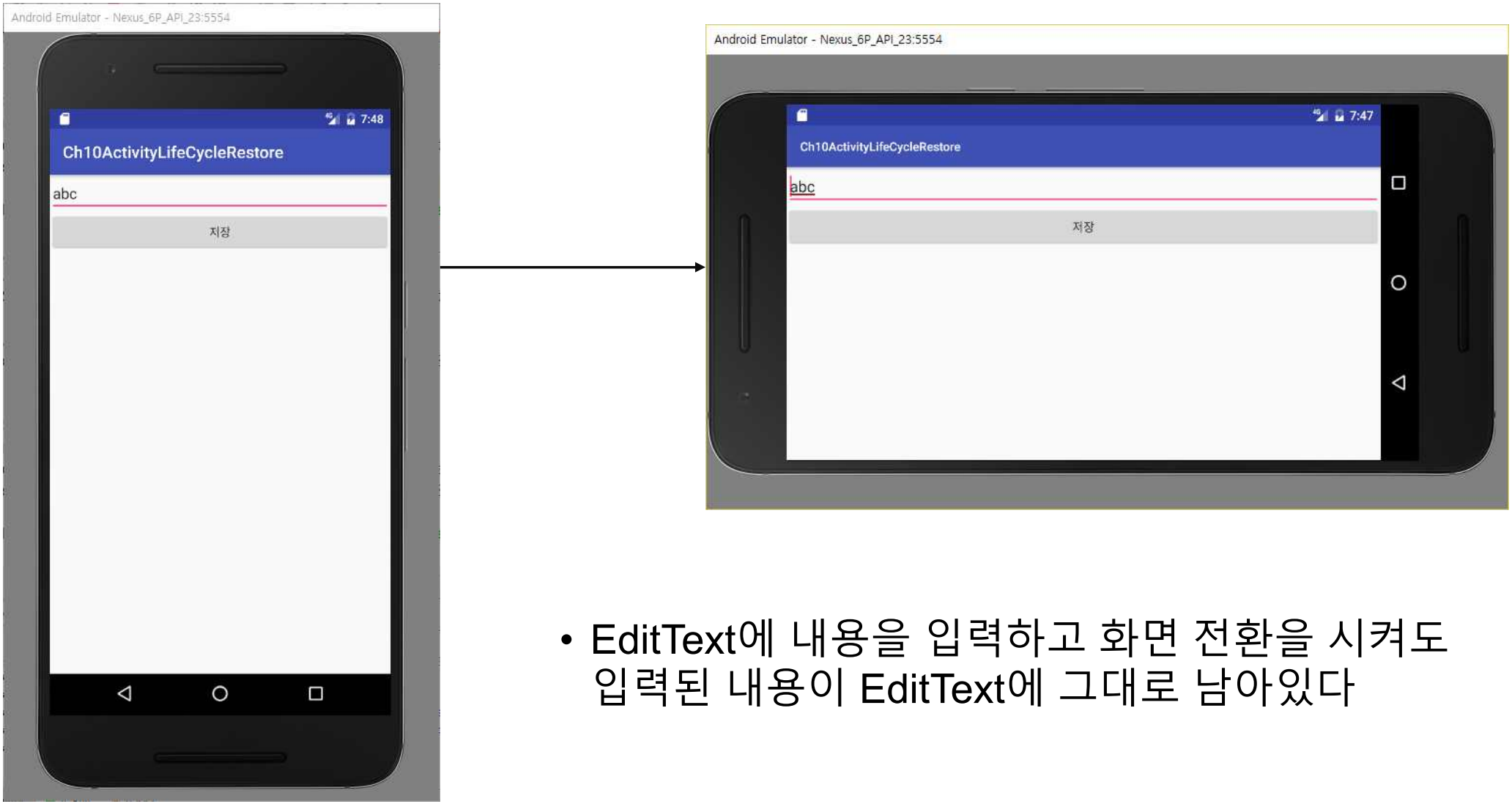


- 이처럼 강제 종료가 되는 경우 onDestroy()가 호출되지 않는다
- 이전에 사용자가 입력했던 데이터를 저장해두지 않으면 손실된다

액티비티 데이터 복원



- 액티비티를 관리하는 액티비티 매니저를 실행하는 시스템 프로세스에 데이터를 백업해 둘 수 있다
- 액티비티 재실행 시 백업된 데이터를 가지고 와서 이전 상태를 복원한다
- Activity 클래스의 onSaveInstanceState() / onRestoreInstanceState() 함수를 재정의하여 구현한다
- **예제 프로젝트 이름: Ch10ActivityLifeCycleRestore**



- EditText에 내용을 입력하고 화면 전환을 시켜도 입력된 내용이 EditText에 그대로 남아있다

onSaveInstanceState 함수 예제

```
// 액티비티 데이터를 백업하는 함수
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    Log.i("Mobile Programming", "onSaveInstanceState()");

    // EditText 내용을 onSaveInstanceState의 매개 변수인 Bundle 클래스 객체, outState에 저장
    // 먼저 EditText에 입력된 텍스트를 가지고 와서 String 객체로 변환한다
    String backupString = mEditText.getText().toString();

    // Bundle 객체도 Intent와 비슷하게, key-value 쌍으로 데이터를 저장한다
    outState.putString("Backup_string", backupString);
}
```

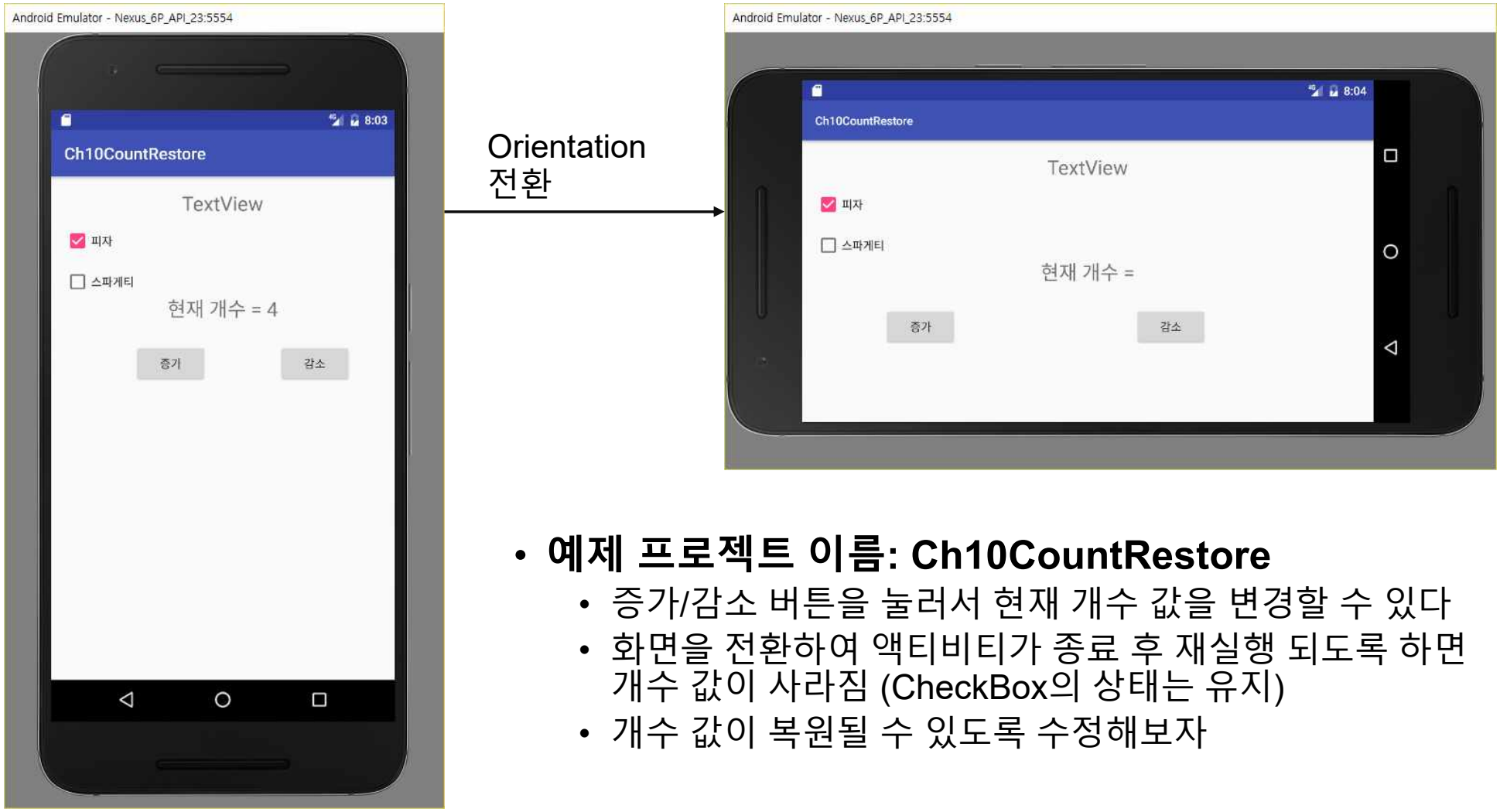
onRestoreInstanceState 함수 예제

```
// 백업한 데이터를 전달받아 복원하는 함수
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);

    Log.i("Mobile Programming", "onRestoreInstanceState()");

    // 만약 매개 변수인 Bundle 객체가 null이 아니면,
    // 해당 액티비티에서 백업한 데이터가 존재한다는 것을 의미
    // Bundle 객체에 백업된 데이터를 가지고 와서 EditText 내용을 복원한다
    if(savedInstanceState != null) {
        // Bundle 객체에 데이터를 저장할 때 사용했던 key 값을 가지고 데이터를 얻는다
        String text = savedInstanceState.getString("Backup_string");

        // 해당 데이터를 EditText에 설정한다
        mEditText.setText(text);
    }
}
```

참고 사항

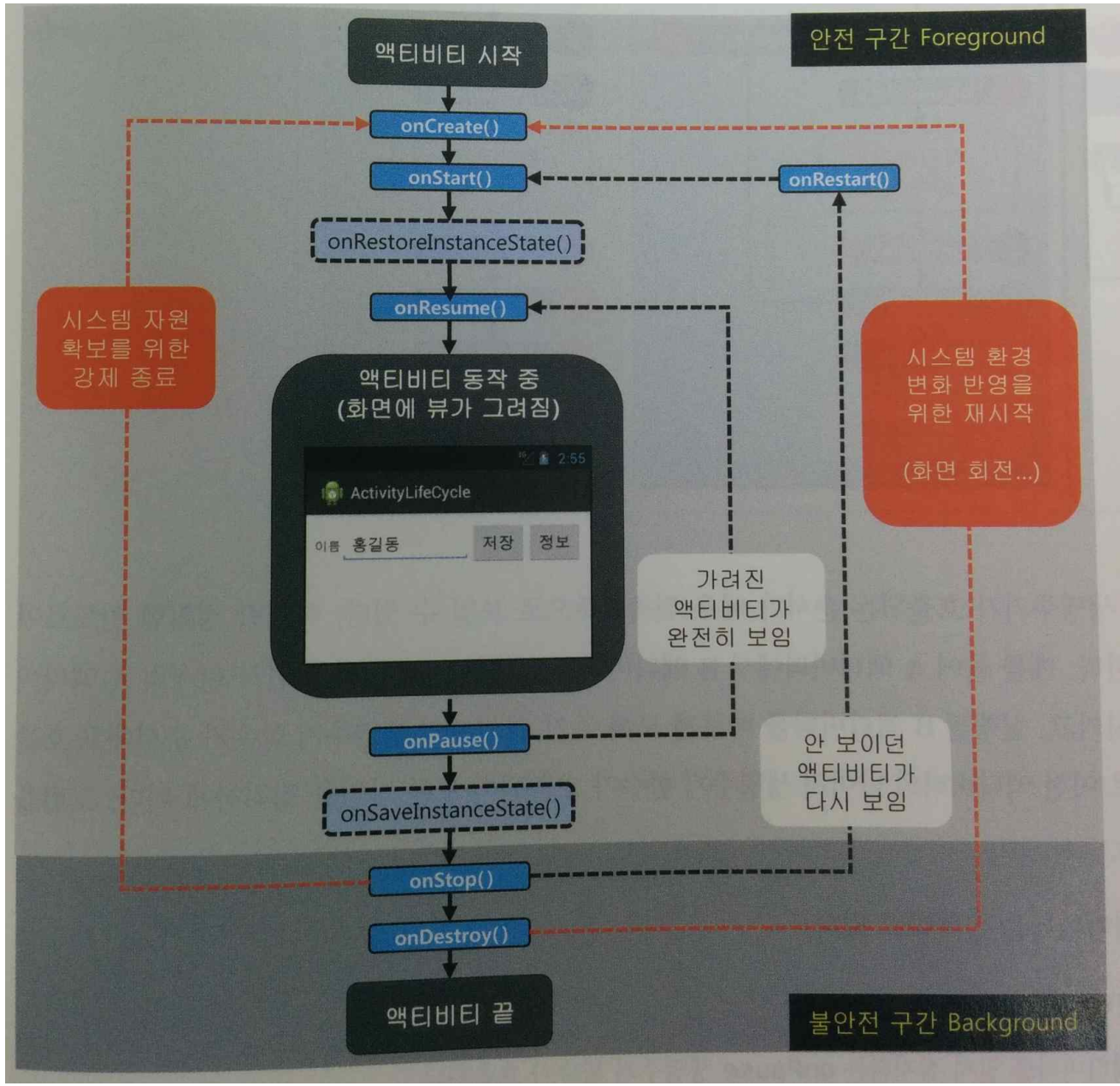
- 자동 백업/복원
  - 뷰 중에는 액티비티에 onSaveInstanceState, onRestoreInstanceState 함수를 구현하지 않아도 자동으로 백업과 복원 기능을 가진 것이 있음
  - 대표적으로 TextView와 그에 파생된 하위 뷰
  - 단, 뷰의 ID를 설정해 주어야만 함
  - 뷰 클래스 내부에 이미 해당 함수가 존재하고 액티비티 생명주기에 따라 뷰의 해당 함수가 호출되기 때문
  - 자동 복원 기능을 사용하지 않는다면, setSaveEnabled(false) 함수를 호출한다
    - 혹은 xml element의 attribute를 추가, android:saveEnabled = “false”
- 백업 데이터 용량
  - 용량이 큰 데이터는 백업되지 않을 수도 있다
  - 프로세스간 데이터를 전달할 수 있는 용량에 제한이 있음 (예: 1MB)
  - 이 경우 파일로 저장하고 파일 경로만 백업을 한 후, 액티비티 재실행 시 이 경로를 참조하여 데이터를 복원한다

데이터 저장 및 복원 예제



- **예제 프로젝트 이름: Ch10CountRestore**
  - 증가/감소 버튼을 눌러서 현재 개수 값을 변경할 수 있다
  - 화면을 전환하여 액티비티가 종료 후 재실행 되도록 하면 개수 값이 사라짐 (CheckBox의 상태는 유지)
  - 개수 값이 복원될 수 있도록 수정해보자

생명주기 정리


액티비티 시작
안전 구간 Foreground
onCreate()
onStart()
onRestart()
onRestoreInstanceState()
onResume()
시스템 자원
확보를 위한
강제 종료
액티비티 동작 중
(화면에 뷰가 그려짐)
2:55
ActivityLifeCycle
이름 홍길동
저장
정보
가려진
액티비티가
완전히 보임
시스템 환경
변화 반영을
위한 재시작
(화면 회전...)
onPause()
안 보이던
액티비티가
다시 보임
onSaveInstanceState()
onStop()
onDestroy()
액티비티 끝
불안전 구간 Background

두 액티비티 간 생명주기 함수 호출