모바일 프로그래밍

09 서비스

2017 2학기

강승우

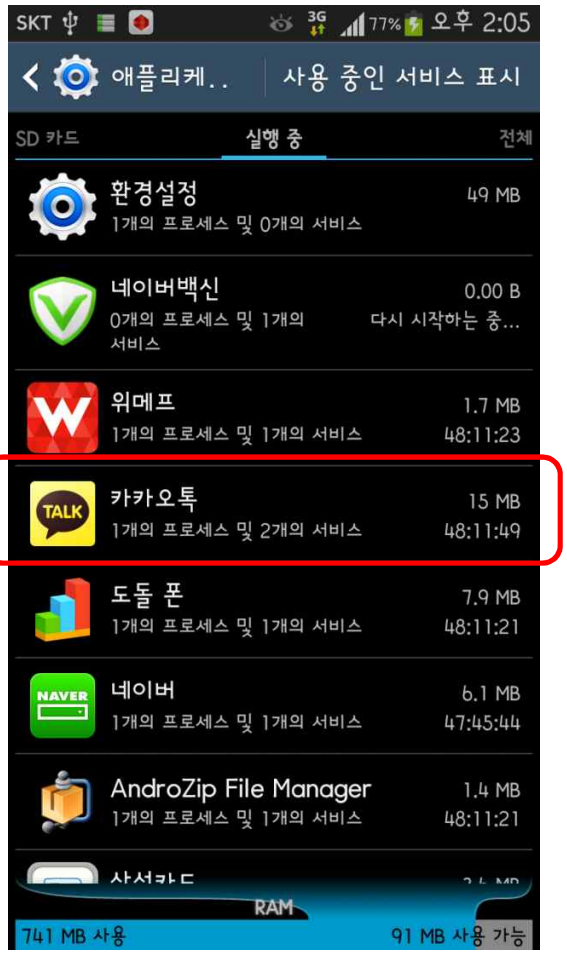
Computer Science and Engineering, KOREATECH

서비스 (Service)

- 안드로이드 애플리케이션을 구성하는 4대 component 중 하나
  - Activity
  - **Service**
  - Broadcast Receiver
  - Content Provider

- 사용자 인터페이스 없이 백그라운드에서 수행되는 기능을 담당
  - 예:
    - 음악 재생
    - 파일 다운 로드 (e.g. 앱 다운, 설치)
    - 애플리케이션 업데이트를 주기적으로 검사

실행중인 서비스



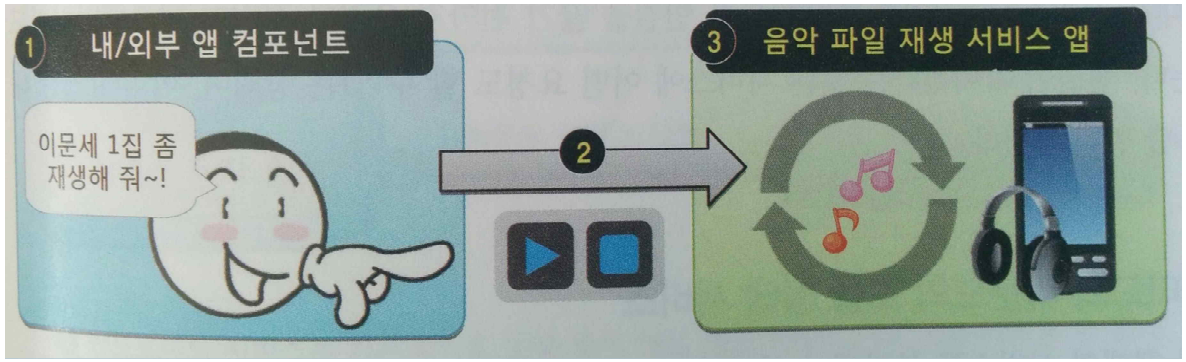
- 설정 – 애플리케이션 – 실행 중 메뉴에서 보면 왼쪽 그림과 같은 것을 볼 수 있음
  - 예
    - 카카오톡
    - 1개의 프로세스 및 2개의 서비스

서비스 특징

- 사용자 인터페이스가 없다
  - 화면에 보이는 View를 갖지 않음
- 보통 애플리케이션의 Activity에서 Service를 시작시킨다
  - 다른 애플리케이션 component에 의해서 시작되기도 함
- 한번 시작된 Service는 해당 애플리케이션의 Activity가 종료되고 다른 애플리케이션의 Activity가 시작되어도 계속해서 백그라운드에서 실행된다
- 서비스를 이용하여 서로 다른 프로세스 간 통신 (IPC: Interprocess communication) 기능을 구현할 수 있다
  - e.g., 어떤 애플리케이션의 서비스가 제공하는 기능(함수)를 다른 애플리케이션에서 호출하여 사용

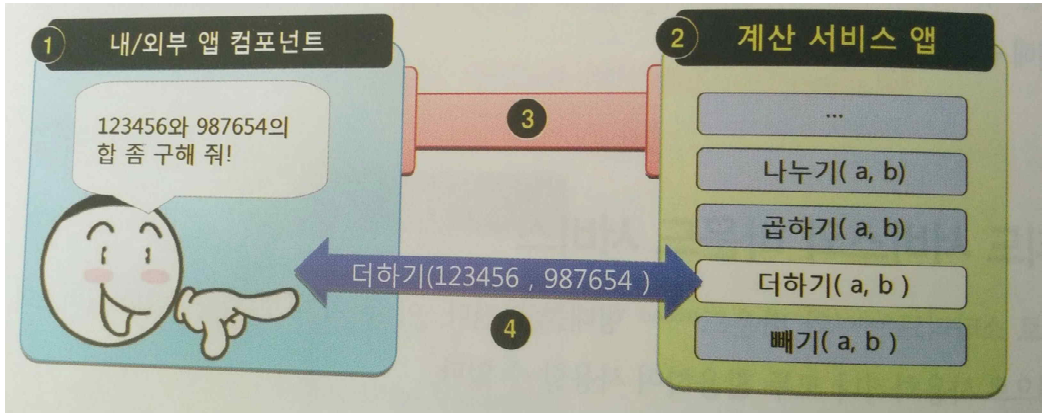
서비스 종류 – Started service

- Started service (시작 타입 서비스)
  - 보통 Activity가 startService()를 호출하여 Service 시작
    - stopService() 호출로 서비스 실행을 종료할 수 있음
    - 그 외 다른 제어는 할 수 없음
  - 시작시킨 Activity가 종료되더라도 서비스는 백그라운드에서 계속 실행될 수 있음
  - 수행할 작업이 완료되면 보통 스스로 종료
  - 호출한 Activity에게 결과를 반환할 수 없음
  - 예: 네트워크에서 파일 다운로드



서비스 종류 – Bound service

- Bound service (연결 타입 서비스)
  - 보통 Activity가 bindService()를 호출하여 Service에 연결
    - unbindService() 호출로 서비스 연결 해제
  - Activity가 연결된 Service에 어떤 요청을 하고 결과를 받을 수 있음
    - 함수 호출과 비슷
    - 서버-클라이언트와 같은 개념으로 이해할 수 있음
      - Service가 서버가 되고 요청을 하는 Activity가 클라이언트가 되는 것
  - Service는 다른 component에 연결되어 있는 동안에만 실행
    - 서비스를 요청한 component가 종료되면 연결도 끊어짐

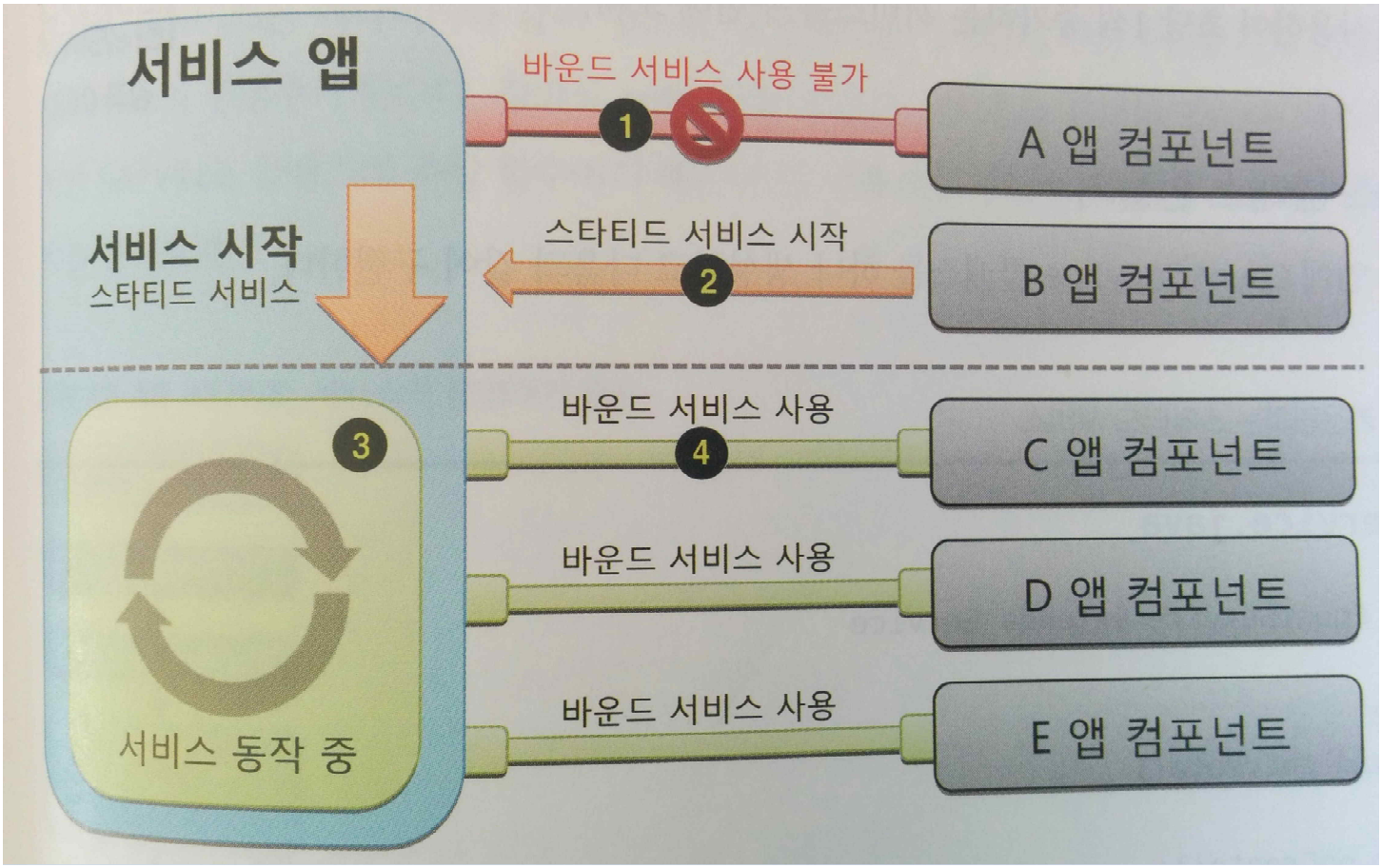


Started service와 Bound service의 조합

음악 재생 서비스

- Started service 형태로 음악 재생
- Bound service 형태로 일시 정지 등의 제어나 노래 정보 표시 등의 기능 수행

Started와 bound service의 관계



Service는 동작 중이 아닐 때는 bound service를 이용할 수 없음

Started Service

Started service의 구현 (1)

- Service 클래스를 상속받아 구현
  - https://developer.android.com/reference/android/app/Service.html
- 재정의 메소드
  - public void onCreate()
  - public int onStartCommand(Intent intent, int flags, int startId)
  - public void onDestroy()

✓public IBinder onBind(Intent intent)

- Bound service와 관련된 메소드
- Started service에서는 필요 없지만 Service 클래스의 추상 메소드이기 때문에 반드시 구현 필요
  - 빈 내용으로 구현해도 된다 → return null; 만 넣어준다

Started service의 구현 (2)

- AndroidManifest.xml에 등록
  - <service> element를 추가
    - name 속성: 서비스 클래스 이름을 지정
    - enabled 속성: true (false이면 서비스가 실행될 수 없음), 기본으로 true로 설정됨
    - exported 속성: true/false, 다른 애플리케이션의 component에서 실행할 수 있는지 여부 결정

  - https://developer.android.com/guide/topics/manifest/service-element.html

예제: 음악 재생

- MediaPlayer 클래스 이용
  - https://developer.android.com/reference/android/media/MediaPlayer.html
  - 오디오/비디오 파일이나 스트림의 재생 제어를 위한 클래스
  - MediaPlayer.create()로 객체 생성
  - start(), pause() 메소드 호출로 재생 시작 및 일시 정지
  - MediaPlayer 관련 API 가이드
    - https://developer.android.com/guide/topics/media/mediaplayer.html

- Activity 클래스에서 MediaPlayer 클래스를 이용하여 음악을 재생하는 기능을 구현해보자
  - **예제 프로젝트 이름: 08_Service/Ch12ActivityPlayer**
- Activity 실행을 종료하면 어떻게 되나?

예제: 음악 재생

- MediaPlayer 객체 생성 및 play 시작

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
```

  - res/raw/ 디렉토리에 저장된 음악 파일 재생 예

```
mediaPlayer.start();
```

  - 파일 이름은 알파벳 소문자, 숫자, 밑줄로만 되어 있어야 함

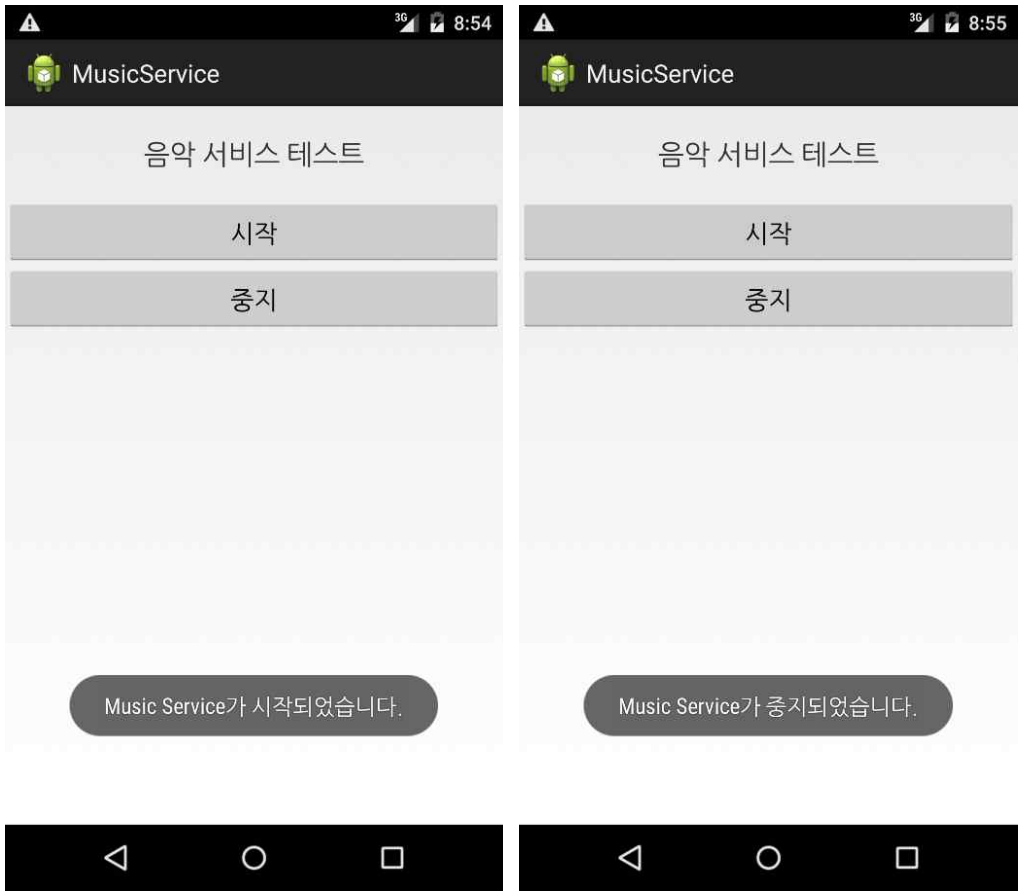
- MediaPlayer 일시 정지

```
mediaPlayer.pause();
```

- 사용 MediaPlayer 객체 해제

```
mediaPlayer.release();
mediaPlayer = null;
```

예제: 음악 재생 서비스



- **예제 프로젝트 이름: 08_Service/Ch12StartedService**
- Activity 화면에서 시작 버튼을 누르면 서비스가 실행되어 음악 재생 시작
- 중지 버튼을 누르면 서비스가 중지되어 음악 재생 중지

Activity에서 Service를 시작/중지

```
public void onClick(View view) {
    switch(view.getId()) {
        case R.id.start:
            Log.d(TAG, "onClick() start");
            startService(new Intent(this, MusicService.class));
            break;

        case R.id.stop:
            Log.d(TAG, "onClick() stop");
            stopService(new Intent(this, MusicService.class));
            break;
    }
}
```

MusicService 클래스

- MusicService 클래스 재정의 메소드

  startService() 메소드가 호출되면 실행되는 메소드

  - onCreate()
    - MediaPlayer 객체 생성
  - onStartCommand()
    - start() 호출

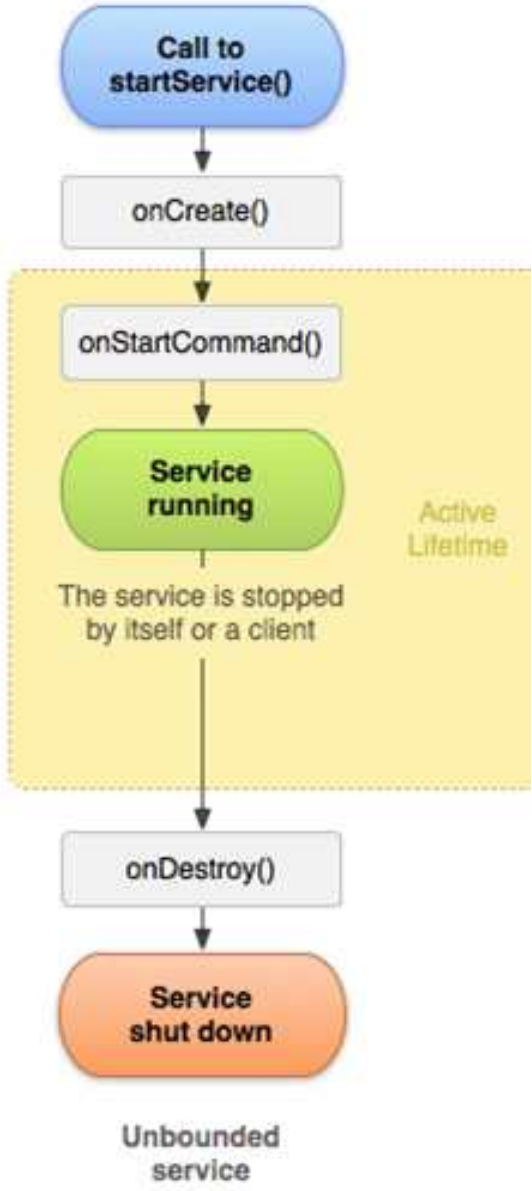
  stopService() 메소드가 호출되면 실행되는 메소드

  - onDestroy()
    - stop() 호출
    - release() 호출

  - onBind()
    - 이 예제에서는 return null; 만 포함

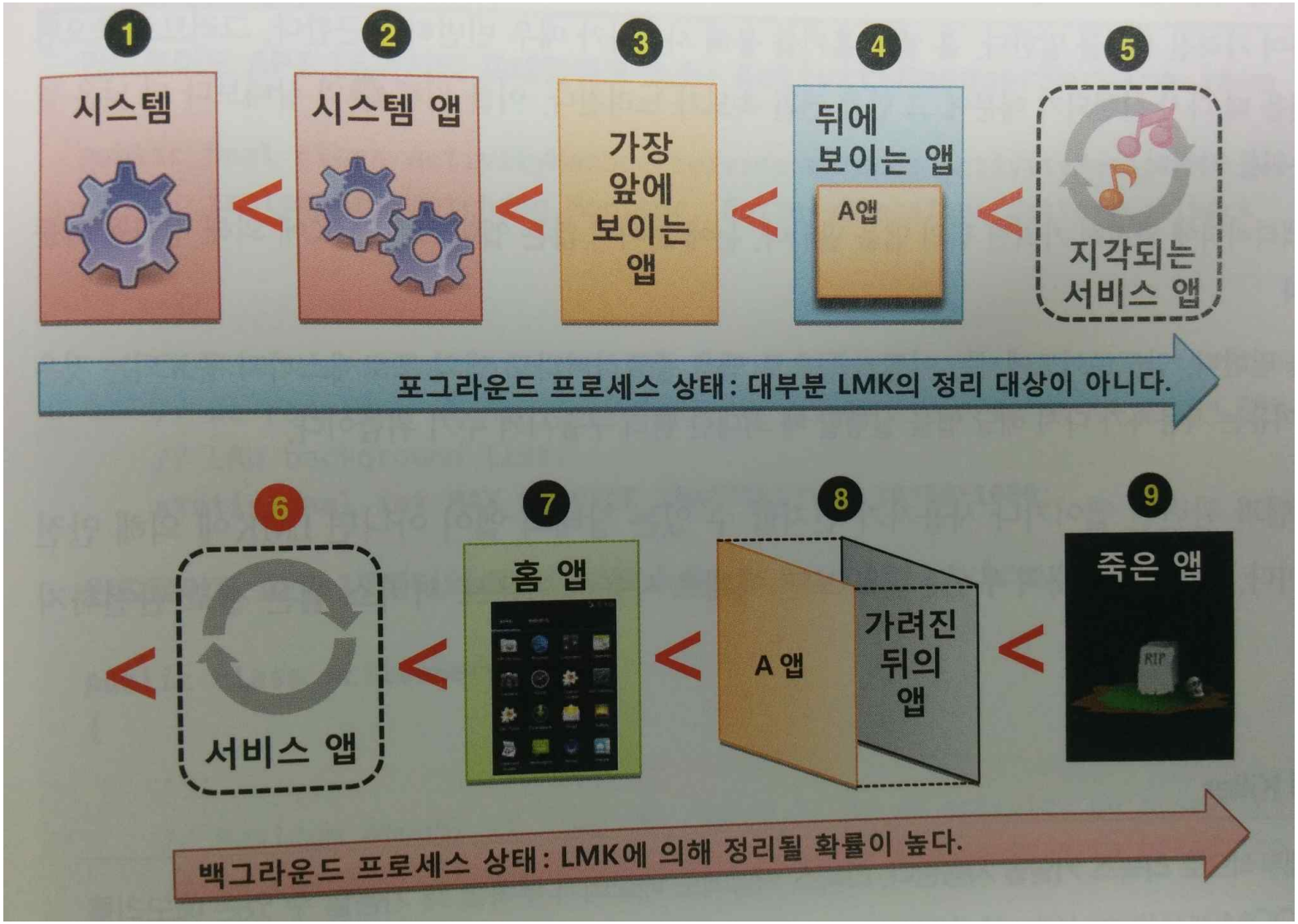
Started service 생명 주기

- startService() 호출로 서비스가 시작된 경우
  - onCreate(), onStartCommand() 메소드가 차례로 호출
  - 이미 실행 중인 Service에 대해 다시 startService() 호출할 경우 onStartCommand() 메소드만 호출
- 서비스가 시작되면
  - stopService()가 호출되거나
  - 서비스 자체에서 stopSelf()를 호출하여 종료할 때까지 계속 실행
- stopService()가 호출 되면
  - onDestroy() 호출



Started service의 생존 우선 순위

- 백그라운드에서 실행 중인 서비스는 stopService 호출을 통해 의도적으로 종료하는 경우를 제외하고 계속 실행이 될 수 있을까?
  - 아니다 (생존이 보장 되지 않는다)
  - 시스템이 사용해야 할 메모리가 부족해지면 실행 중인 프로세스를 죽이고 메모리를 확보한다
- LMK (Low Memory Killer)
  - 안드로이드에서 메모리가 부족할 때 애플리케이션 프로세스를 강제 종료하는 것
  - LMK 대상이 되는 앱의 우선 순위에 따라 차례로 kill 해가면서 메모리 확보



1. 시스템 프로세스 (패키지 매니저, 액티비티 매니저)
2. 단말 제조사에서 미리 탑재한 전화, 메시지, 카메라와 같은 앱
   <application> element persistent 속성 값이 true
3. 실행되어 화면에 보이는 앱
4. 일부가 가려져 보이는 앱
5. 사용자가 인지하는 서비스 앱 (음악 재생)
6. 일반 서비스 앱

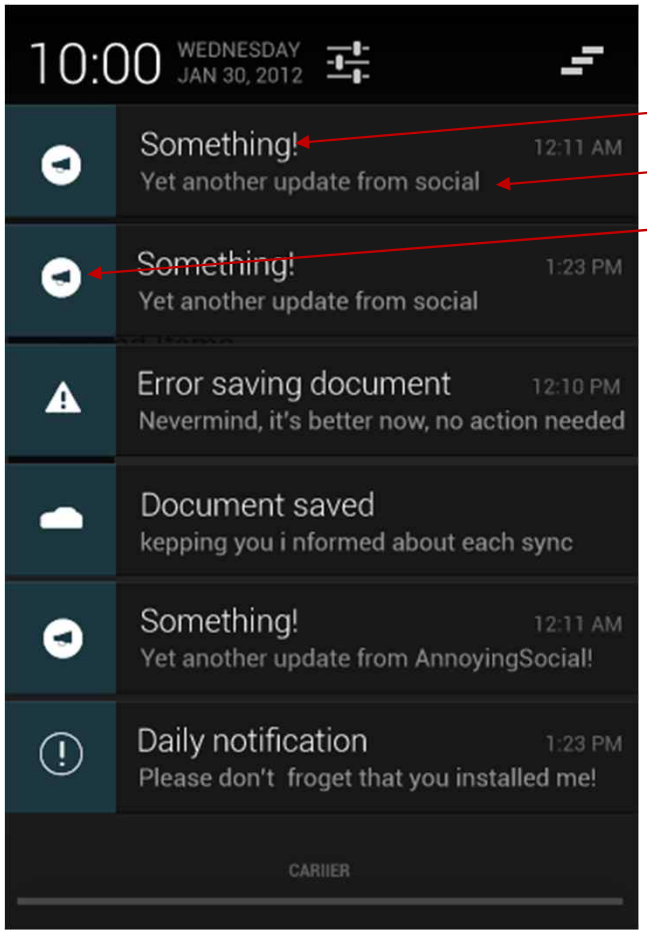
서비스의 생존 방법

- LMK의 kill 대상에 포함되지 않도록 하기
  → Foreground 프로세스로 실행시키기
  - startForeground() 메소드 호출

- Kill 된 서비스를 다시 실행되도록 만들기
  - onStartCommand() 메소드의 반환값을 필요에 맞게 지정
    - START_STICKY
    - START_NOT_STICKY
    - START_REDELIVER_INTENT

Service 클래스의 startForeground() 이용

- startForeground(int id, Notification notification)
  - Id: Notification bar에 표시할 현재 notification 객체의 id (0이 아닌 정수)
  - notification: Notification 클래스의 객체
- Foreground 실행 서비스의 표시
  - 화면 상단의 상태 바(status bar)에 startForeground()에서 설정된 notification의 아이콘 표시



- Notification 객체 생성

```
Notification noti = new
Notification.Builder(mContext)
        .setContentTitle("Music service")
        .setContentText("Service is running")
        .setSmallIcon(R.drawable.icon)
        .setContentIntent(pIntent)
        .build();
```

- setContentIntent(PendingIntent pIntent)
  - Notification을 클릭했을 때 전달되는 PendingIntent 객체를 지정
- PendingIntent
  - Intent와 유사 → activity, service 등을 실행하고자 보내는 메시지
  - PendingIntent를 생성한 component가 직접 intent를 전송하는 것이 아니라 다른 component에게 나중에 대신 보내달라고 할 때 사용

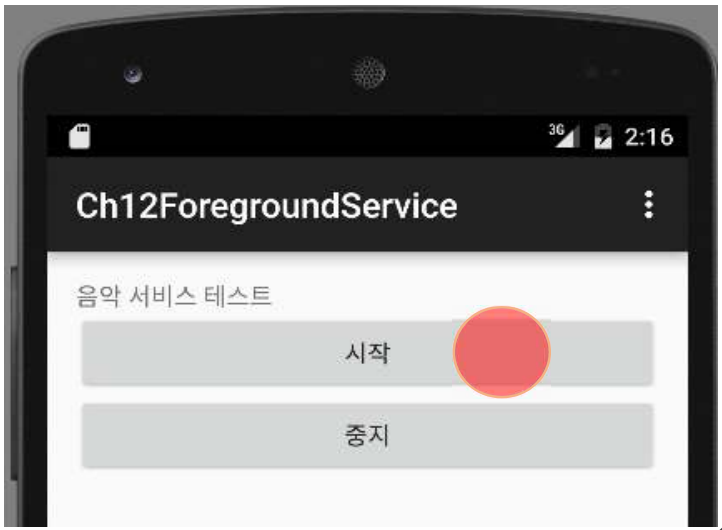
PendingIntent 사용예

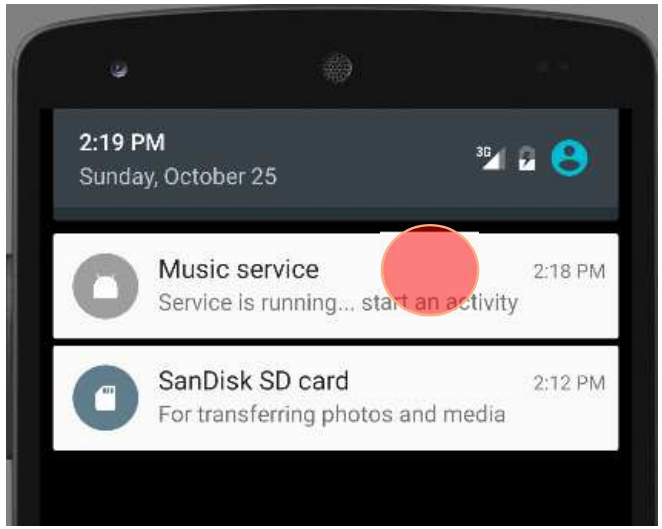
```
// 1-1. Intent 객체 생성 - MainActivity 클래스를 실행하기 위한 Intent 객체
Intent intent = new Intent(this, MainActivity.class);

// 1-2. Intent 객체를 이용하여 PendingIntent 객체를 생성 - Activity를 실행하기 위한 PendingIntent
PendingIntent pIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);

// 1-3. Notification 객체 생성
Notification noti = new Notification.Builder(this)
        .setContentTitle("Music service")
        .setContentText("Service is running... start an activity")
        .setSmallIcon(R.mipmap.ic_launcher)
        .setContentIntent(pIntent)
        .build();
```

https://developer.android.com/reference/android/app/PendingIntent.html

예제: Foreground 서비스

예제 프로젝트 이름: 08_Service/Ch12FgService





시작 버튼을 클릭하면 음악이 재생되면서 상태 바에 아이콘이 표시됨

상태 바를 내리면 notification 메시지가 표시됨

Notification 메시지를 클릭하면 Activity 실행 (왼쪽 첫 그림과 같이 Activity 화면이 표시됨)

Android의 시스템 서비스인 Notification Manager가 Activity를 실행하는 intent를 보내주는 것임

onStartCommand 메소드의 반환값

- START_STICKY
  - LMK가 서비스 앱의 프로세스를 강제 종료한 이후 가용 메모리가 확보되면 다시 서비스를 실행
  - 서비스의 생존 유지가 중요한 앱이라면 이 값을 반환
- START_NOT_STICKY
  - 가용 메모리가 확보되어도 다시 실행하지 않음
- START_REDELIVER_INTENT
  - START_STICKY와 유사하나 onStartCommand 메소드의 매개 변수의 intent 정보를 복원해 줌
  - startService를 호출할 때 intent 객체를 넘겨주고 그것을 onStartCommand에서 받는데, LMK에 의해서 강제 종료되고 다시 실행이 될 때 이전에 받았던 intent를 다시 넘겨줌
  - 서비스 생존 유지가 중요하고, 전달받은 intent 정보가 서비스 동작에 중요한 역할을 한다면 이 값을 반환

Bound service

Bound service

- 한 애플리케이션 내의 다른 component에서 Service에 연결하여 이용할 때
  - 외부 앱에는 공개하지 않고 같은 앱의 다른 component(예: 클라이언트 역할을 하는 Activity)가 Service에 정의된 메소드를 호출하는 경우
- 다른 애플리케이션 프로세스가 Service에 연결하여 이용할 때

→Binder 객체를 생성하여 서비스를 사용하고자 하는 클라이언트에게 넘겨주어야 함

→클라이언트는 Binder를 이용하여 Service에 정의된 메소드를 호출할 수 있음

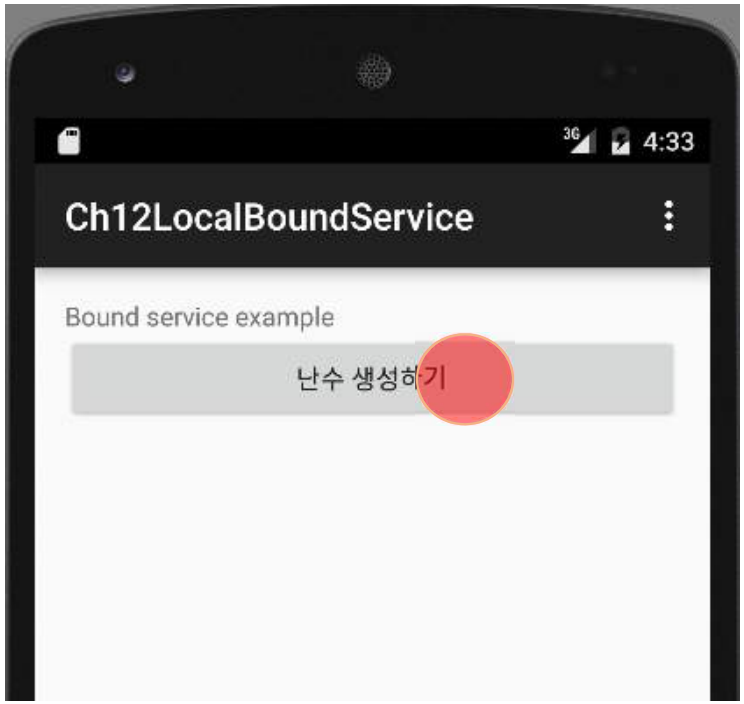
내부 Bound service 구현

- Service 클래스
  - Binder 클래스를 상속 받는 내부 클래스 정의 / 클래스 객체 생성
  - Service에서 클라이언트에게 제공할 메소드 정의
  - onBind(), onUnbind() 메소드 재정의
    - Public IBinder onBind(Intent intent)
    - public boolean onUnbind(Intent intent) : 필수는 아님
- Activity 클래스
  - bindService(), unbindService() 호출
  - ServiceConnection 인터페이스를 구현하는 객체 생성
    - onServiceConnected(), onServiceDisconnected() 메소드 구현
    - onServiceConnected(): 서비스에 연결되었을 때 호출되는 callback 메소드
    - onServiceDisconnected(): 서비스와 연결이 해제되었을 때 호출되는 callback 메소드

https://developer.android.com/reference/android/os/IBinder.html
https://developer.android.com/reference/android/os/Binder.html

예제: 난수 생성 서비스

- **예제 프로젝트 이름: 08_Service/Ch12LocalBoundService**

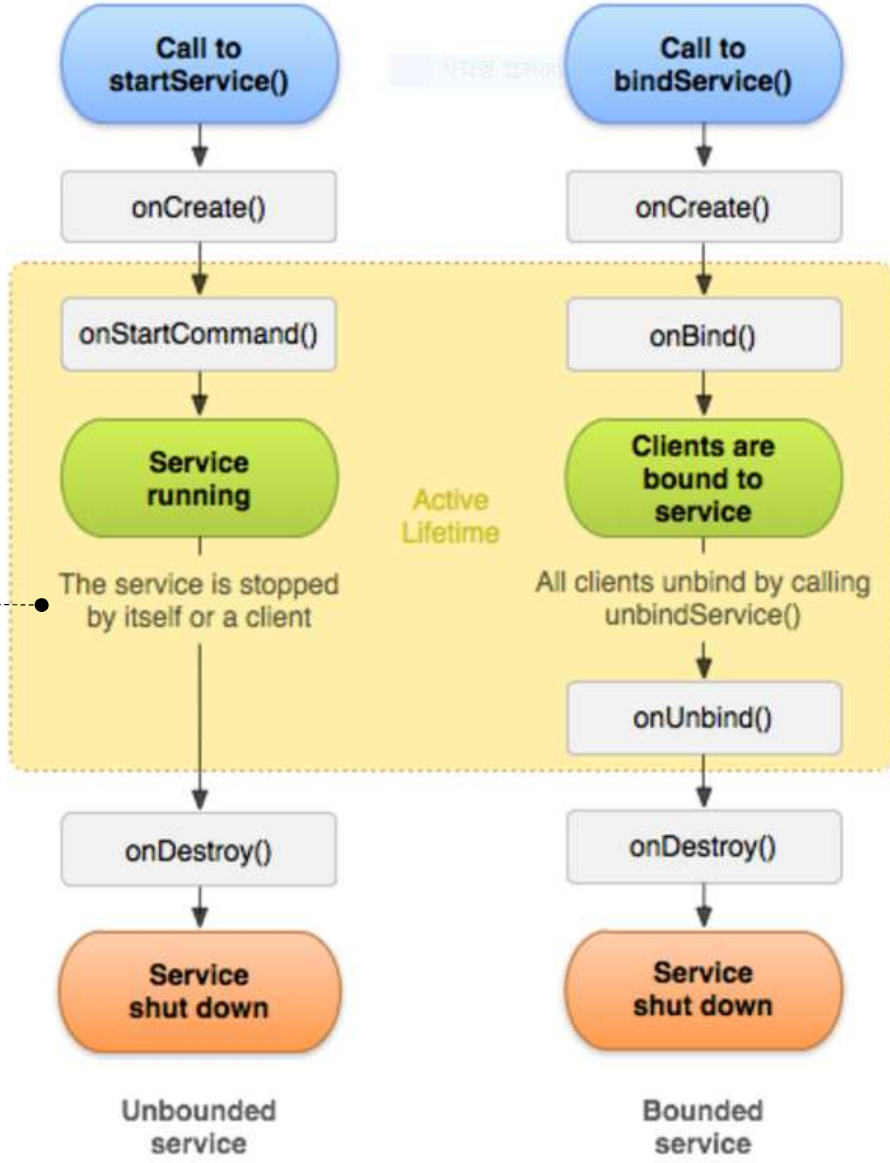


LocalService 클래스 (Service를 상속받는 클래스)
- Binder 클래스를 상속받는 내부 클래스(LocalBinder) 정의
- LocalBinder 클래스 객체(mBinder) 생성
- onBind 메소드에서 mBinder 객체 반환
- 난수 발생 메소드(getRandomNumber) 구현

- Activity가 실행되면 Service에 연결 (bindService)
- Service에 연결되면 Binder 객체를 참조하여 Service 객체를 얻음
- Service 객체를 참조하여 난수 발생 메소드 호출
- 난수가 발생 되면 토스트 메시지 표시

Service 생명주기

다른 component에서 stopService() 호출
Service 자체에서 stopSelf() 호출



https://developer.android.com/guide/components/services.html?hl=ko#Lifecycle