액티비티(activity)

액티비티

- 안드로이드 구성 4대 컴포넌터
  - ✓ 액티비티(Activity) : 화면을 구성하는 가장 기본적인 컴포넌트
  - ✓ 서비스(Service) : 액티비티와 상관없이 백그라운드에서 동작하는 컴포넌트
  - ✓ 브로드캐스트 리시버(Broadcast Receiver) : 문자 메시지 도착, 배터리 방전, SD카드 탈부착, 네트워크 환경 변화 등이 발생하면 방송 신호 보냄
  - ✓ 컨텐트 프로바이더(Content Provider)

- 액티비티는 안드로이드 응용 프로그램을 구성하는 4가지 컴포넌트 중 하나로 가장 빈번히 사용되며 사용자를 대면한다는 면에서 실질적으로 가장 중요한 요소

액티비티

- **setContentView** 메서드
  액티비티가 생성될 때마다 호출되며, 액티비티 안에 뷰를 배치하는 명령
  액티비티 하나는 독립된 기능을 수행하며, 서로 중첩되지 않음
- 일반적으로 액티비티 하나당 XML 파일 하나를 만들어서 사용
- MainActivity.java 코드는 Activity 클래스를 상속받으므로 MainActivity.java를 액티비티라고 부름

```
public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

액티비티 추가

- 다른 액티비티를 호출하기 위한 버튼 추가하기

activity_main.xml

```
<Button
    android:id="@+id/callBtn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="액티비티  호출" />
```

액티비티 추가

- 새로운 액티비티 화면 xml 작성

activity_sub1.xml

```
<LinearLayout>
    <Button
        android:id="@+id/backBtn"
        android:text="종료" />
</LinearLayout>
```

액티비티 추가

- 새로운 액티비티 클래스 추가

SubActivity1.class

```
public class SubActivity1 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sub1);
        생략….
    }
}
```

액티비티 추가

- 추가된 액티비티 화면에서 종료 버튼을 클릭 시 처리
- finish()

  현재 액티비티를 종료시킨다.

SubActivity1.class

```
private Button backBtn;
backBtn = (Button)findViewById(R.id.backBtn);
backBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        finish();
    }
});
```

액티비티 추가

- MainActivity 에서 SubActivity1 호출하기

MainActivity.class

```
private Button callBtn;
callBtn = (Button)findViewById(R.id.callBtn);
callBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getApplicationContext(), SubActivity1.class);
        startActivity(intent);
    }
});
```

액티비티 추가

- 프로그램에 포함된 모든 액티비티는 반드시 매니페스트에 등록
  (매니페스트에 등록되지 않은 액티비티는 존재하지 않는 것으로 취급)
- 메인 액티비티는 자동 등록되지만 추가한 액티비티는 별도로 등록해야 함

AndroidManifest.xml

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".SubActivity1"></activity>
```

인텐트(Intent)

- 안드로이드 4대 컴포넌트가 상호 간에 데이터를 주고 받기 위한 메시지 객체
- 액티비티 및 컴포넌트간 수행할 작업에 대한 정보 및 작업 결과 리턴을 위해서도 사용
- 명시적 인텐트와 암시적 인텐트로 구분

명시적 인텐트와 데이터 전달하기

- 인텐트에 호출할 대상 액티비티의 이름을 명확히 지정 할 때 사용
- 주로 같은 응용 프로그램내의 서브 액티비티를 호출할 때 사용되며, 권한에 따라 외부 응용 프로그램의 액티비티도 호출 가능

```
Intent intent = new Intent(getApplicationContext(), SubActivity1.class);
startActivity(intent);
```

데이터 전달하기

- 인텐트는 액티비티간에 인수와 리턴값을 전달하는 도구로도 이용
- **Bundle** 타입의 **Extras**를 활용하며 이름과 값의 쌍으로 된 임의 타입의 정보를 원하는 개수만큼 전달 가능

Intent putExtra (String name, int value)

Intent putExtra (String name, String value)

Intent putExtra (String name, boolean value)

int getIntExtra (String name, int defaultValue)

String getStringExtra (String name)

Boolean getBooleanExtra (String name, boolean defaultValue)

데이터 전달하기

MainActivity

```
Intent intent = new Intent(getApplicationContext(), SubActivity1.class);
intent.putExtra("name", "홍길동");
intent.putExtra("age", 22);
startActivity(intent);
```

SubActivity1

```
Intent intent = getIntent();
String name = intent.getStringExtra("name");
int age  = intent.getIntExtra("age", 0);
Toast.makeText(getApplicationContext(), name + "-" + age,
                                        Toast.LENGTH_LONG).show();
```

양방향 데이터 전달하기

public void startActivityForResult (Intent intent, int requestCode)

- 두 번째 인수 : 호출한 대상을 나타내는 식별자, 리턴시에 누구에 대한 리턴인가를 구분
  호출되는 액티비티별로 고유의 번호를 붙여야 하며, **0** 이상의 중복되지 않는 정수를 넘기며, 음수일 경우 리턴받지 않음
- 호출된 액티비티가 종료 후 아래의 메서드가 호출

protected void onActivityResult (int requestCode, int resultCode, Intent data)

- 리턴값을 받으려면 메서드를 재정의
- **requestCode :** 액티비티를 호출할 때 전달한 요청 코드
- **resultCode :** 액티비티의 실행 결과
- 리턴값은 **data** 인텐트 안에 포함되어 있으므로 **data** 안에 **Extras**를 읽어 구할 수 있음

양방향 데이터 전달하기

```
Intent intent = new Intent(getApplicationContext(), SubActivity1.class);
intent.putExtra("num1", val1);
intent.putExtra("num2", val2);
startActivityForResult(intent, PLUS);
```

MainActivity

응답결과 처리 콜백함수

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {

}
```

양방향 데이터 전달하기

SubActivity1

```
Intent intent = getIntent();
final int num1 = intent.getIntExtra("num1", 0);
final int num2 = intent.getIntExtra("num2", 0);


backBtn = (Button)findViewById(R.id.backBtn);
backBtn.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view) {
        Intent rsltIntent = new Intent(getApplicationContext(), MainActivity.class);
        rsltIntent.putExtra("result", num1 + num2);
        setResult(200, rsltIntent);
        finish();
    }
});
```

암시적 인텐트(Implicit Intent, 묵시적 인텐트)

- 약속된 액션(Action)을 지정하여 안드로이드에서 제공하는 기존 응용프로그램을 실행하는 것
- 전화 거는 것을 예로 들면 전화번호를 인텐트로 넘긴 후에 전화 걸기 응용프로그램이 실행되는 것과 같음

전화걸기

```
Uri uri = Uri.parse("tel:00000000000");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);
```

웹 페이지

```
Uri uri = Uri.parse("http://www.naver.com");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

암시적 인텐트(Implicit Intent, 묵시적 인텐트)

검색

```
 Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
intent.putExtra("query", "자바");
startActivity(intent);
```

메세지

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.putExtra("sms_body", "내용  입력하세요");
intent.setType("vnd.android-dir/mms-sms");
startActivity(intent);
```

사진

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setType("image/*");
startActivity(intent);
```

액티비티의 상태

스택상의 액티비티는 아래 세 가지 상태중의 하나

실행 (active, running)

: 사용자가 직접 사용하는 상태. 스택의 제일 위에 위치하며, 화면상에도 가장 위에 위치하여 입력 포커스를 가지고 사용자의 입력을 직접 처리

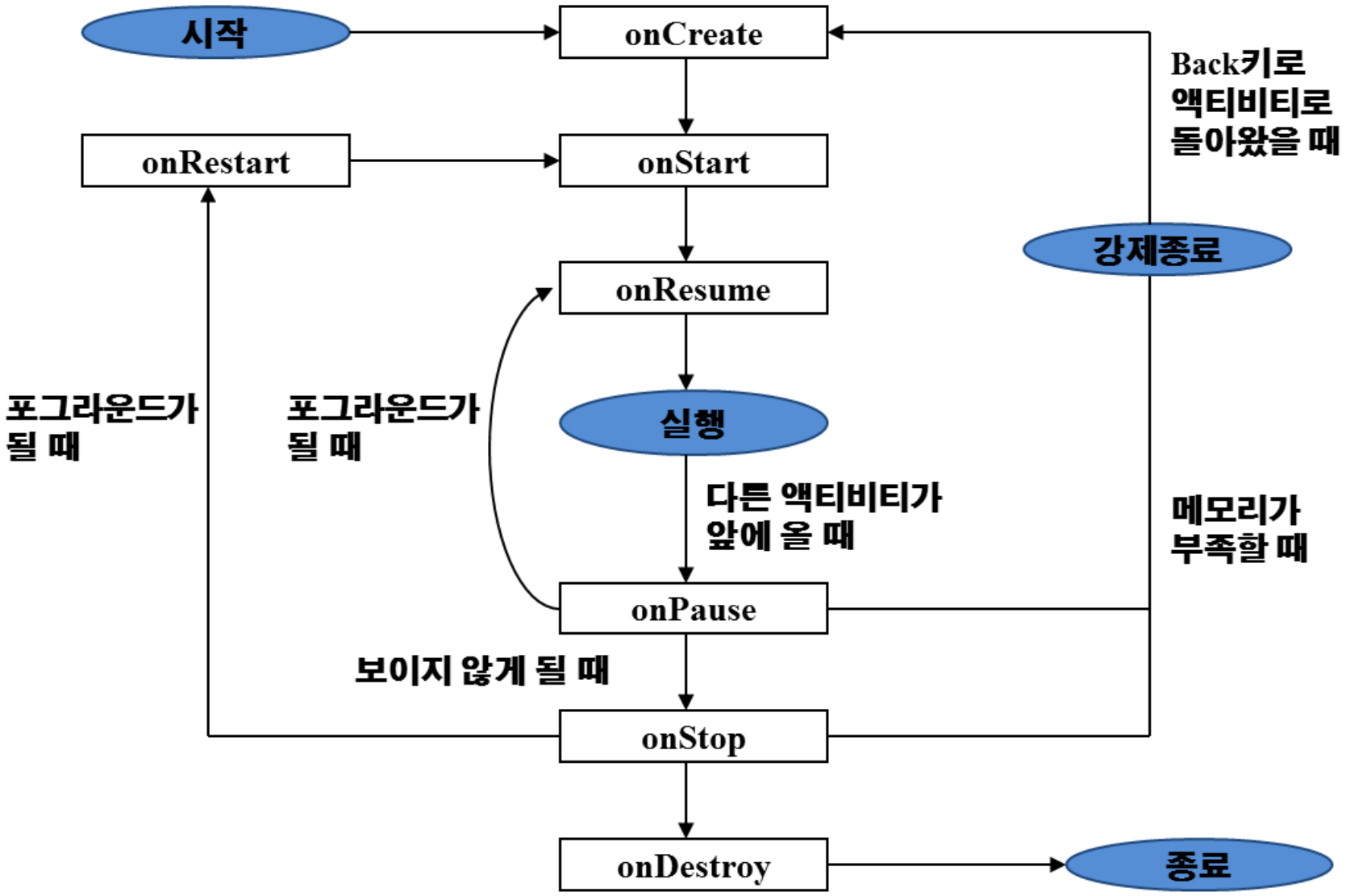
일시 정지 (pause)

: 포커스는 잃었지만 사용자에게는 보이는 상태. 위쪽에 다른 액티비티가 있지만 화면 전체를 다 가리지 않았거나 반투명한 경우에 해당한다. 살아있는 상태와 같지만 시스템에 의해 강제 종료될 수 있음

정지 (stopped)

다른 액티비티에 의해 완전히 가려진 상태. 사용자의 눈에 보이지는 않지만, 모든 정보를 다 유지하고 있어 재활성화가 가능하다. 시스템은 메모리 부족 시 정지 상태의 액티비티를 언제든지 강제 종료할 수 있음

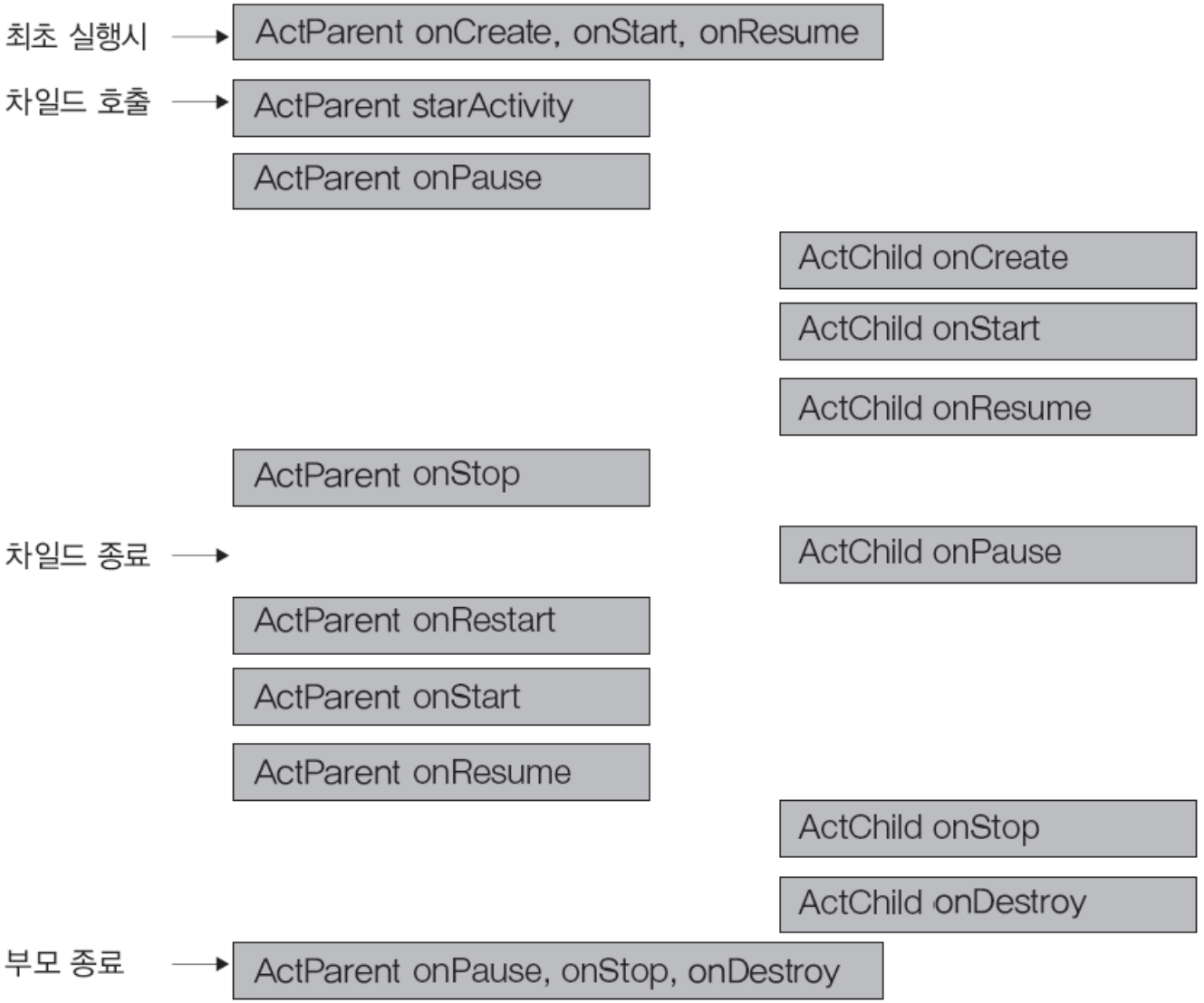
액티비티 - 생명주기


시작
onCreate
onRestart
onStart
onResume
실행
onPause
onStop
onDestroy
종료
강제종료
Back키로
액티비티로
돌아왔을 때
포그라운드가
될 때
포그라운드가
될 때
다른 액티비티가
앞에 올 때
메모리가
부족할 때
보이지 않게 될 때

액티비티 – 생명주기

메서드	해야 할 일
onCreate	액티비티를 초기화하며, 중지 후 재시작하는 경우엔 액티비티의 이전 상태 정보인 Bundle이 전달되며 이 정보대로 재초기화한다.
onRestart	재시작될 때 호출된다.
onStart	액티비티가 사용자에게 보이기 직전에 호출된다.
onResume	사용자와 상호작용을 하기 직전에 호출되며, 이 단계에서 스택의 제일 위로 올라온다.
onPause	다른 액티비티가 실행될 때 호출되며, 이 단계에서 미저장한 데이터가 있으면 저장한다.
onStop	액티비티가 사용자에게 보이지 않게 될 때 호출된다.
onDestroy	액티비티가 종료 될 때 호출된다. 시스템에 의한 강제 종료인지 finish 메서드 호출에 의해 스스로 종료하는 것인지는 isFinishing 메서드로 조사 가능하다.

액티비티 - 생명주기


최초 실행시
ActParent onCreate, onStart, onResume
차일드 호출
ActParent starActivity
ActParent onPause
ActChild onCreate
ActChild onStart
ActChild onResume
ActParent onStop
차일드 종료
ActChild onPause
ActParent onRestart
ActParent onStart
ActParent onResume
ActChild onStop
ActChild onDestroy
부모 종료
ActParent onPause, onStop, onDestroy