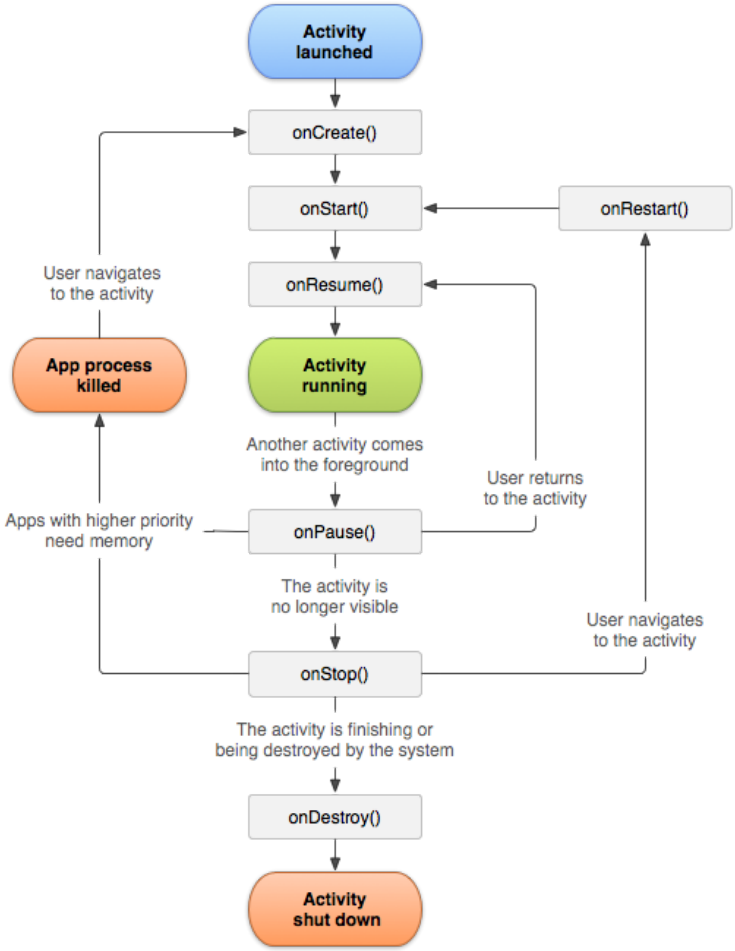
● 액티비티 상태

활성 상태 (activity running)	현재 액티비티가 화면을 점유하여 출력되고 있으며, 포커스를 가지고 있어서 사용자 이벤트에 반응할 수 있는 상태
일시 정지 상태(pasue)	현재 액티비티가 일시적으로 사용이 불가능한 상태
비활성 상태(stop)	현재 액티비티가 다른 액티비티로 인해 화면이 완벽하게 가려진 상태



▶ 활성 상태: 생성된 액티비티는 onCreate() → onStart() → onResume() 함수가 호출되면서 활성 상태가 되는 것이고, onResume() 함수 호출 전까지 액티비티 화면 내용을 출력해 주어야 한다. 일반적으로 setContentView() 함수를 이용하여 액티비티 화면을 출력한다고 하지만, 정확하게는 이 함수가 호출되는 시점은 화면이 출력되는 순간이 아니라, onResume() 함수까지 실행하고 setContentView() 함수에서 출력한 내용이 화면에 나오는 구조이다. 따라서 onResume() 함수가 실행되기까지 setContentView() 함수를 onCreate(), onStart(), onResume() 등 어디선가 호출해 주면 화면에 나온다. 하지만 onResume() 함수가 호출될 때까지 setContentView() 함수가 한 번도 호출되지 않는다면, 화면에 아무것도 나오지 않고, 에러는 발생하지는 않는다. 만약 setContentView() 함수를 반복해서 호출한다면, 화면에 출력되는 건 맨 마지막에 호출된 내용이다. setContentView()는 이전 화면을 지우면

서 새로운 내용을 출력하는 함수이기 때문이다. 만약, 기본 화면을 지우지 않고 그 위에 추가로 함께 출력하려면 addContentView() 함수를 이용하면 된다.

```
@Override
protected void onResume() {
    super.onResume();
    setContentView(R.layout.activity_main);

    ImageView imageView = new ImageView(this);
    // ...
    addContentView(imageView, params);
}
```

onResume() 함수에서 화면을 두 번 출력했다. 첫 번째는 setContentView() 함수를 이용하였고, 두 번째는 ImageView를 addContentView() 함수를 이용해서 출력했다. 이렇게 하면 첫번째 화면과 두 번째 화면이 겹쳐져서 나온다.

▶ 일시 정지 상태: 액티비티가 여전히 화면에 보이지만 포커스를 잃은 상태이다. 대표적인 예가 다든 액티비티가 화면 전체를 가리지 않고 실행되었을 때이다. 다든 액티비티가 반투명하게 실행되거나 다이얼로그 스타일로 실행되어 여전히 자신이 화면에 보이지만 포커스를 잃은 상태이다. 일시 정지 상태로 멈출 대도 있지만, 대부분 정지 상태(onStop)로 전환되기 전에 호출되어 곧 정지될 것임을 나타내기 위해 사용된다. 일시 정지 상태가 되면 onPause() 함수가 자동으로 호출된다. 예를 들어 버든이 클릭될 때 다든 액티비티가 다이얼로그 스타일로 실행되면, onPause() 함수가 자동으로 호출된다. onPause() 함수에서는 대부분 다음의 내용을 구현할 것을 권장한다 — 1. 애니메이션이나 CPU 소비를 야기할 수 있는 기타 지속적인 작업 정지. 2. GPS와 같은 센서값 수신. 서버 네트워킹 등 액티비티가 일시 정지된 동안 불필요한 동작 정지. 액티비티는 일시 정지 상태가 되어도 정상적으로 동작해야 하는 때가 있는데, 대표적으로 다든 액티비티가 다이얼로그 스타일로 진행되어 자신이 보일 때 애니매이션에 의한 화면 변화가 지속적으로 유지돼야 하는 경우이다. 그러나 대부분은 더 이상 실행될 필요가 없는 경우이며, 이런 작업을 onPause() 함수에서 멈추었다가 다시 활성 상태로 변경될 때 동작하게 하는 제어가 필요하다. 그렇지 않으면 앱이 불필요하게 리소스를 점유하거나 메모리를 확보하거나 네트워킹을 계속 발생시켜 사용자에게 악성 앱이 되기 쉽다. (액티비티 종료 시점인 onDestory() 함수에서 하면 안 되는 이유는 화면에 우리의 앱이 안 보이더라도 액티비티는 계속 실행 상태에 있을 가능성이 크기 때문에, 사용자가 보고 있지도 않은 액티비티를 한 시간 동안이나 동작해서 불필요한 네트워크 트래픽이 계속 발생하거나 배터리를 소모하면 악성 앱이 되기 때문이다. 액티비티 작업은 사용자 화면에서 출력되지 않으면 의미 없는 것들이 대부분이므로 onPuase()와 onResume() 함수를 이용해서 제어하는 것이 좋다).

▶ 비활성 상태: 다든 액티비티로 인해 화면이 완전히 가려진 상태이다. 보통 다든 액티비티로 화면이 전환되어 안 보이는 경우인데, onPause() → onCreate() 함수까지 호출된다. 뒤로 가기 버든 등으로 화면을 가렸던 액티비티가 사라지면 다시 활성 상태로 전환되는데, 이때는 onRestart() → onStart() → onResume() 함수가 차례로 호출된다.

● 액티비티 상태 저장: 액티비티가 종료되어도 계속 데이터를 유지했다가 다시 실행되면 그대로 이용해야 할 때가 있다. (이는 앱이 종료되었을 때 데이터를 저장하는 방법인 데이터베이스나 파일 프로그램을 통해 데이터를 저장했다가 다시 이용하는 방법과는 다르게 액티비티 종료에 대비한 데이터 저장이다). 유실되는 데이터를 저장했다가 가져와야 하는 대표적인 예가 화면 회전이다. 스마트폰은 사용자가 화면을 회전할 수 있으며, 이때 액티비티가 종료되었다가 다시 시작된다. 정상적으로 실행되어 onResume() 함수까지 호출된 액티비티를 회전하면 onPause() → onStop() → onDestroy() 함수까지 호출되어 종료되고, 시스템에서 자동으로 다시 시작해서 onCreate() → onStart() → onResume() 함수까지 호출되어 다시 화면에 나타나기는 하지만, onResume() 함수가 실행된 후에 사용자 이벤트나 네트워킹으로 데이터가 발생하였다면 이 데이터는 모두 유실된다. 액티비티가 종료될 때 유실되면 안 되는 데이터를 저장했다가 다시 액티비티가 시작될 때 복원하여 사용하기 위한 생명주기 함수들이 있다.

onResume() 함수까지 실행된 액티비티가 화면 회전으로 출력되면, 자동으로 onPause() → onStop() → onSaveInstanceState() → onDestory() 순으로 함수가 호출되고 액티비티는 종료된다. 그리고 다시 액티비티가 시작되면서 onCreate() → onStart() → OnResetoreInstanceState() → onResume() 함수가 차례대로 실행된

다. onCreate() 함수와 onRestoreInstanceState() 함수, 그리고 onSaveInstanceState() 함수는 액티비티의 상태 관리를 위해 데이터를 저장했다가 복원하는 데 사용한다. 액티비티가 종료되는 상황에 대비하여 액티비티의 데이터를 저장해야 한다면, 우선 onSaveInstanceState() 함수를 이용한다.

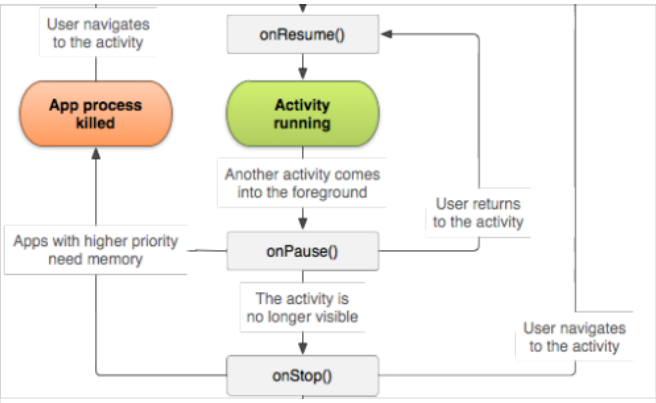
```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putString("data1", "hello");
    outState.putInt("data2", 100);
}
```

onSavedInstanceState() 함수는 onStop() 함수 후 자동으로 호출되며, 이 함수에서 액티비티의 데이터를 저장한다. 저장하는 방법은 매개변수로 전될대는 Bundle을 이용하면 되는데, 이는 컴포넌트 데이터를 저장하기 위한 일종의 Map 객체이며, 이 객체에 데이터를 key-value로 담아주면 내부적으로 파일로 저장해줌으로써 액티비티가 종료되더라도 데이터는 유실되지 않는다. 이렇게 저장한 데이터를 액티비티가 다시 시작되는 시점에 가져와서 이용할 수 있다. 이 작업에는 onCreate()와 onRestoreInstanceState() 함수를 이용한다.

```
@Override
protected void onRestoreInstanceState(Bundle saveInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    String data1 = savedInstanceState.getString("data1");
    int data2 = savedInstanceState.getInt("data2");
}
```

액티비티가 다시 시작하면서 onCreate()와 onRestoreInstance() 함수가 호출될 때 저장된 데이터를 Bundle에 담아 매개변수로 전달한다.

https://developer.android.com/guide/components/activities/activity-lifecycle?hl=ko#java



활동 수명 주기에 관한 이해 | Android 개발자 | Android De…

활동은 사용자가 전화 걸기, 사진 찍기, 이메일 보내기 또는 지도 보기와 …

developer.android.com

Resources: 깡쌤의 안드로이드 프로그래밍