Activity Lifecycle

noname

Conceptual Parts

States of Activities

- Active
- Pause
- Stop
- Inactive

Active state

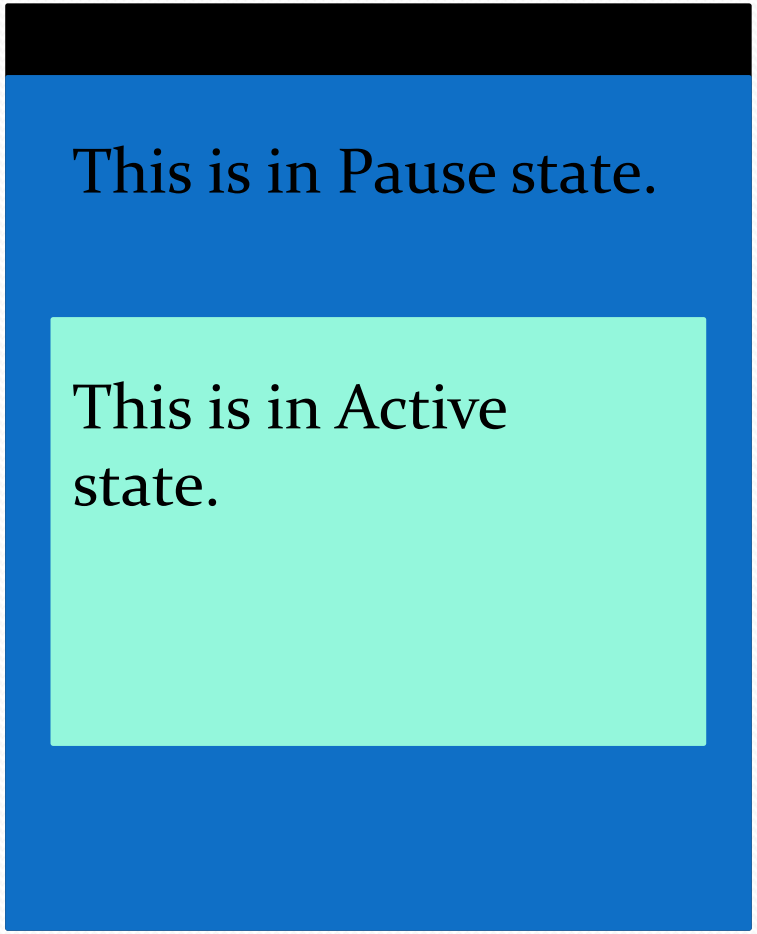
- The state that the activity is on the most foreground and having a focus.

Active state

- The condition that an activity goes into active state
  - Create -> Start -> Resume
  - (Pause state) -> Resume
  - (Stop state) -> Restart -> Start -> Resume
  - (Inactive state caused by abnormal termination) -> Create -> Start -> RestoreInstanceState -> Resume

Pause state

- The state that the activity is shown in the screen although it does not have a focus because another activity is in active state but occupies just some part of the screen or the full screen with transparency.

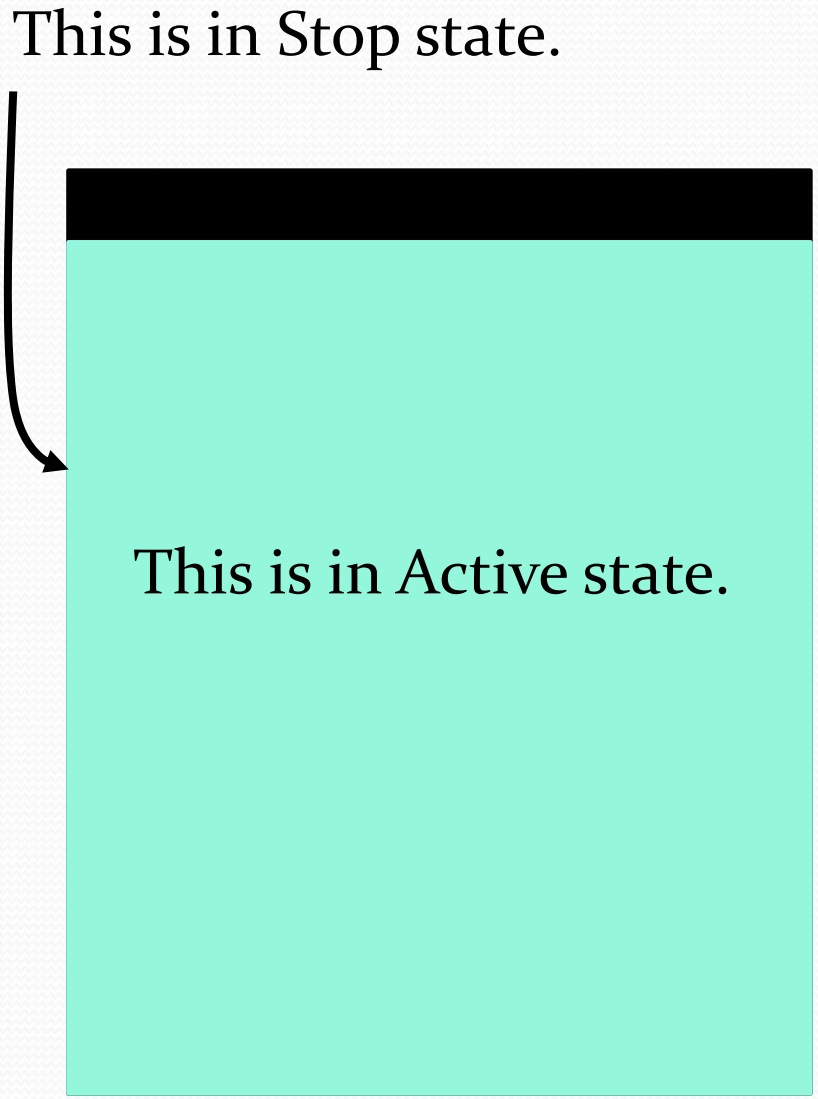


Pause state

- The condition that an activity goes into pause state
  - (Active state) -> (Some part of the activity doesn't appear) -> SaveInstanceState -> Pause

Stop state

- The state that the activity doesn't appear on the screen because of occupation of an active activity which uses the full screen without transparency.



Stop state

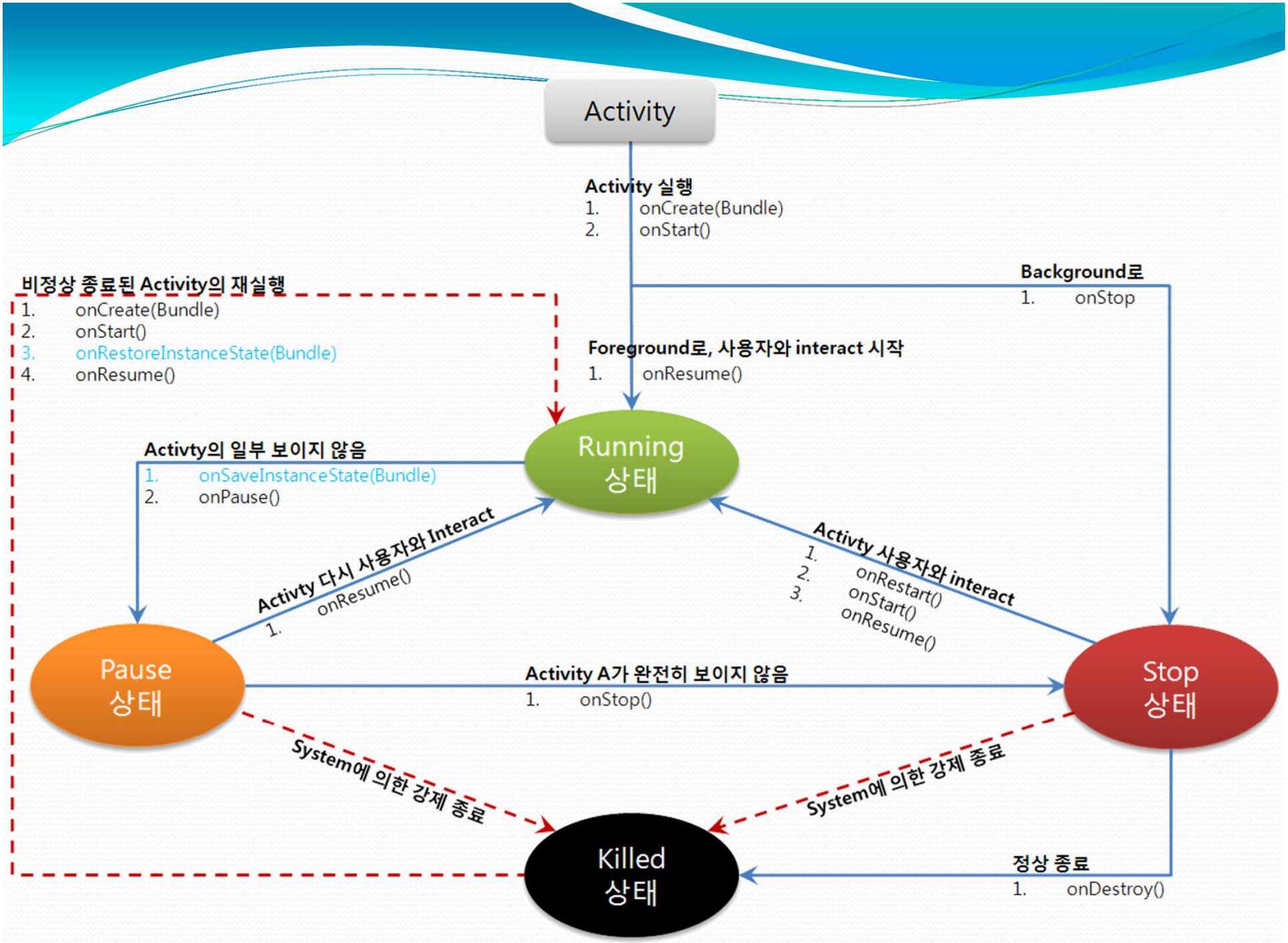
- The condition that an activity goes into stop state
  - (Pause state) -> (Any part of the activity doesn't appear) -> Stop
  - Create -> Start -> Stop

Inactive state

- The state that the activity is terminated or not started yet.
- The activity cannot appear on the screen until it is (re)started.

Inactive state

- The condition that an activity goes into inactive state
  - (Pause state) -> (Abnormal termination by system)
  - (Stop state) -> (Abnormal termination by system)
  - (Stop state) -> Destroy


Activity
Activity 실행
1. onCreate(Bundle)
2. onStart()
Background로
1. onStop
비정상 종료된 Activity의 재실행
1. onCreate(Bundle)
2. onStart()
3. onRestoreInstanceState(Bundle)
4. onResume()
Foreground로, 사용자와 interact 시작
1. onResume()
Running 상태
Activty의 일부 보이지 않음
1. onSaveInstanceState(Bundle)
2. onPause()
Activty 다시 사용자와 Interact
1. onResume()
Activty 사용자와 interact
1. onRestart()
2. onStart()
3. onResume()
Pause 상태
Activity A가 완전히 보이지 않음
1. onStop()
Stop 상태
System에 의한 강제 종료
System에 의한 강제 종료
Killed 상태
정상 종료
1. onDestroy()

Activity Stack

- Activities are stacked.
- The top activity in the stack is being in active state.
- This is why the back button exists in Android phones and works as we expected.

Application Termination Priority

- Mobile phones have restricted resources, hence the system terminates the applications whose termination priority is high.
- An application's priority is determined by its activity which has the highest priority.
- Forced termination priority
  - Stop state: 1st
  - Pause state: 2nd
  - Active state: never

Development Parts

Callback functions

- Android system provides us with callback functions as handler for changing of states.
- What we have to do is just implementing the callback functions.
- **Caution: We have to call the original callback function of super class.**

onCreate(Bundle)

- When?
  - When the activity is being created.
- What to do?
  - Setting the layout of the activity with setContentView().
  - Creating views.
  - Binding between views and data.
  - Restoring flowing(?) states from Bundle.
- Followed by…
  - onStart()

onCreate(Bundle)

- What is Bundle?
  - Will be explained later.

onRestart()

- When?
  - When the activity's state is being changed from stop to active.
- What to do?
  - Similar to onCreate function.
- Followed by…
  - onStart()

onStart()

- When?
  - When the activity is becoming visible.
- What to do?
  - Making the activity to be able to be shown.
- Followed by…
  - onResume(): When the activity goes to the foreground.
  - onStop(): When the activity is becoming hidden.

onResume()

- When?
  - Before the activity's state is changed to active. At this point, the activity is on the top of the activity stack.
- What to do?
  - Preparing for correct working of the activity.
  - e.g. Occupying a camera device resource.
- Followed by…
  - onPause()

onPause()

- When?
  - **Before** the activity's state is changed to pause.
- What to do?
  - Generally, undoing of onResume function.
  - Releasing occupation.
  - Cleaning data.
  - Saving data to prepare for abnormal termination.
- Followed by…
  - onResume(): When the activity is activated again.
  - onStop()
- Properties
  - This function's return is guaranteed, and abnormal termination can be done after the return.

onStop()

- When?
  - **Before** the activity's state is changed to stop.
- What to do?
  - Well…
- Followed by…
  - onRestart()
  - onDestroy()
- Properties
  - This function's return is **not** guaranteed so abnormal termination can be done at anytime.

onDestroy()

- When?
  - **Before** the activity's state is changed to inactive.
- What to do?
  - Generally, undoing of onCreate function.
- Properties
  - This function's return is **not** guaranteed so abnormal termination can be done at anytime.

onSaveInstanceState(Bundle)

- When?
  - When it is needed to save the state of the activity because of following abnormal termination(includes **rotation of the screen**).
  - Before calling onPause function.
- What to do?
  - Saving flowing(?) states to Bundle.
  - e.g. Saving the state of very complex calculation.
- Followed by…
  - onPause()

onSaveInstanceState(Bundle)

- Example of using Bundle
  - @Override
    public void onSaveInstanceState(Bundle b) {
    b.putInt(“2^31-1", 2147483647);
    b.putString(“name", “GodSiva”);
    super.onSaveInstanceState(b);
    }

onRestoreInstanceState(Bundle)

- When?
  - When the activity is being restored after abnormal termination(includes **rotation of the screen**).
  - After calling onCreate and onStart functions, and Before calling onResume function.
- What to do?
  - Restoring flowing(?) states from Bundle.
- Followed by…
  - onResume()

finish(), finishActivity()

- finish()
  - Finishing the current activity itself.
- finishActivity()
  - Finishing another activity.
- Both of the cases cause onDestroy callback function.

Thank you