MODULE 8

ACTIVITY LIFECYCLE

Françoise Dubisy

TABLE OF CONTENT

TABLE OF CONTENT

Stack of Activities

- Each activity operates independently of the others
- Stack of activities maintained while running the application
  - The activity on the top is the one currently being displayed
- When Back button pressed
  - The top activity is popped from the stack
  - ⇨ The previous activity becomes the current activity
  - ⇨ The previous screen is displayed
- Transition from one activity to another through intents
  - Asynchrounous message
  - Can be used to pass data from one activity to another one

Activity States

- Created
  - Transient (fast moving to the resumed state)
- Started
  - Activity is visible
  - Transient (fast moving to the resumed state)
- Running (Resumed)
  - Activity is running in the foreground
  - Activity is visible and interacts with the user

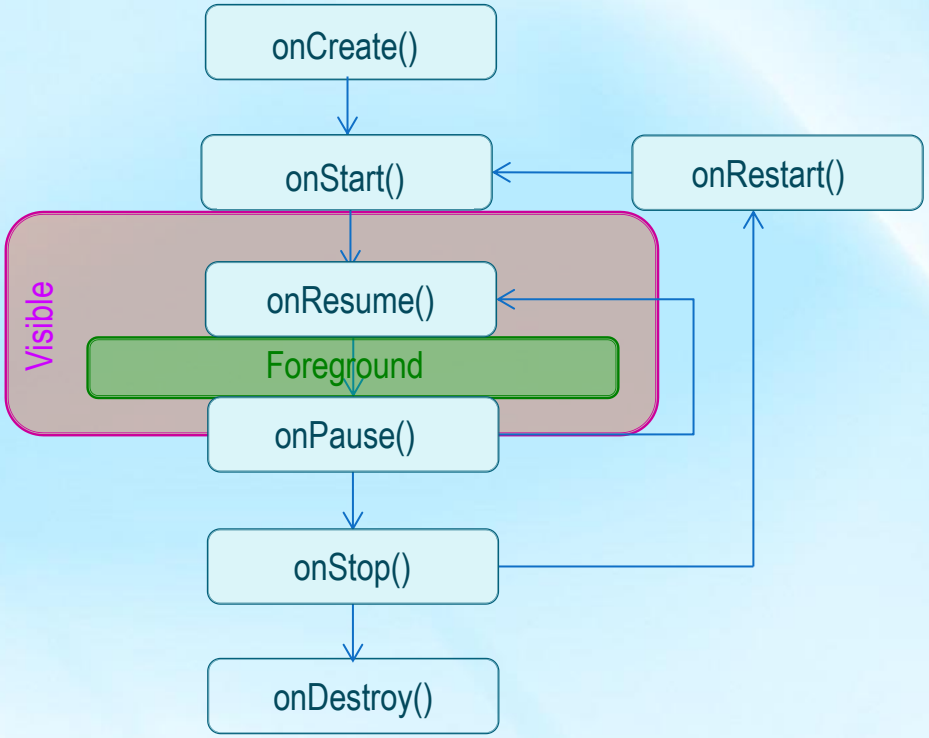
Activity States

- Paused
  - Activity is still visible but partially obscured
  - Instance is running but might be killed by the system
- Stopped
  - Activity is hidden (not visible)
  - Instance is running
    - Activity instance (state information + member variables) is retained
    - But cannot execute any code
  - Might be destroyed/killed by the system

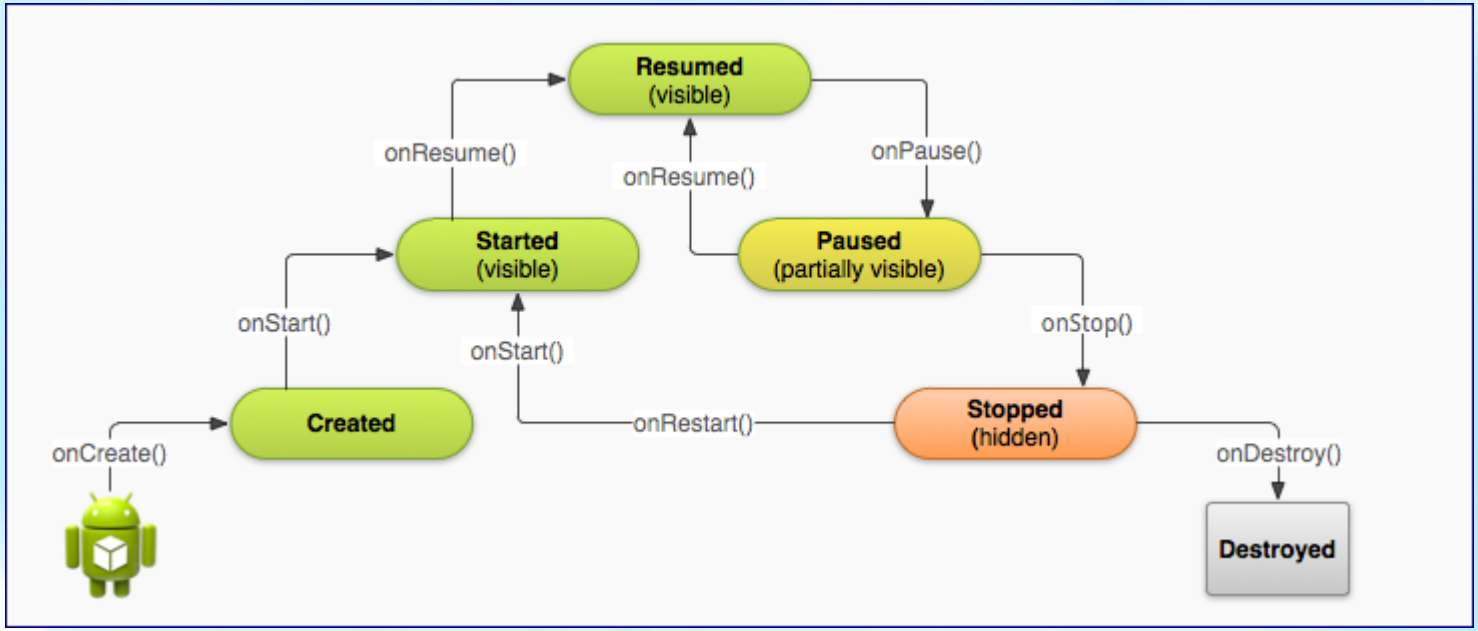
Activity States

- Destroyed (killed)
  - Activity is terminated
    - By the user
      - By pressing the Back button
      - By rotating the screen (because new layout)
    - By the system
      - Shutdown of stopped app to recover memory
    - Through a call to finish() method

Activity Lifecycle



Activity Life Cycle



Source: http://developer.android.com/training/basics/activity-lifecycle/starting.html

Activity Lifecycle

- The user should not notice if an activity which is still part of an activity stack has been terminated or not

- The developer needs
  - To store the state of the activity
  - To restore it
  - To stop any unnecessary actions if the activity is not visible anymore
    - To save system resources

onCreate Method

- onCreate()
  - Called when the activity is first created
    - Creates a new instance of the activity
  - Used to initialize the activity
    - Application startup logic : to do only once for the entire life of the activity
    - To declare the user interface (defined in an XML layout file)
      - To configure some of the UI
    - To define member variables, …

onStart Method

- onStart()
  - Called before the activity becomes visible

onResume Method

- onResume()
  - The activity becomes the foreground activity
    - The activity has the focus
    - The user starts interacting with the activity again
  - Called
    - Just after the execution of the onStart()
    - Whenever the activity appears at the top of the activity stack
      - The activity is visible again
  - To begin animations and initialize components only used while the activity has user focus
    - To initialize fields, to register listeners, to bind to services, etc

onPause Method

- onPause()
  - The activity is partially visible
    - Foreground activity obstructed by other visual components
    - E.g, a semi-transparent activity opens such as a dialog
  - The activity is stopped
    - It looses focus
  - The Activity instance is kept resident in memory
    - Will be recalled when the activity resumes
    - No need to re-initialize components

onPause Method

- onPause() (continued)
  - To stop animations or ongoing actions
    - That should not continue while paused (such as a video)
  - To minimize consumption of resources
    - To release system resources consuming CPU cycles or battery, such as broadcast receivers, handles to sensors (like GPS)
    - E.g, unregister listeners, intent receivers, unbind from services, remove system service listeners
  - Avoid performing CPU-intensive work such as writing to a database
    - Because it can slow the visible transition to the next activity
    - Instead perform heavy-load shutdown operations during onStop()

onStop Method

- onStop()
  - The activity is no longer visible
  - Called
    - When the activity switches to the background
    - The activity is being destroyed
  - Used
    - To shut down Time or CPU intensive operations
      - E.g, writing information to a database
  - The Activity instance is kept resident in memory
    - Will be recalled when the activity resumes
    - No need to re-initialize components

onRestart Method

- Scenarios in which activity is stopped and restarted
  - By opening the Recent Apps window and switching to a recent app
    - The activity currently in the foreground is stopped
    - If the user returns to this app from the Home screen launcher icon or the Recent Apps window, the activity restarts
  - By performing an action in an app that starts a new activity
    - The current activity is stopped when the second activity is created
    - If the user then presses the *Back* button, the first activity is restarted
  - The user receives a phone call while using the app

onDestroy Method

- onDestroy()
  - Used when
    - The activity is completed
    - The activity is about to be destroyed
  - To release last resources
    - E.g, to kill background threads created during onCreate()
    - N.B. most of the resources have generally been released with onStop()
  - Activity is completely removed from the system memory
    - Activity instance is destroyed

Saving Activity State

- The system keeps track of the current state for each View in the layout

  ⇨ no need to save and restore it!

- The state of the View objects is retained in a **Bundle**
  - Key-value pairs stored in a Bundle object
- The state of the View objects is restored
  - If the user navigates back to the same instance of the activity
  - Even if the system has to destroy the stopped activity to recover memory
    - i.e, if the system destroys the activity due to system constraints (rather than normal app behavior)

Saving Activity State

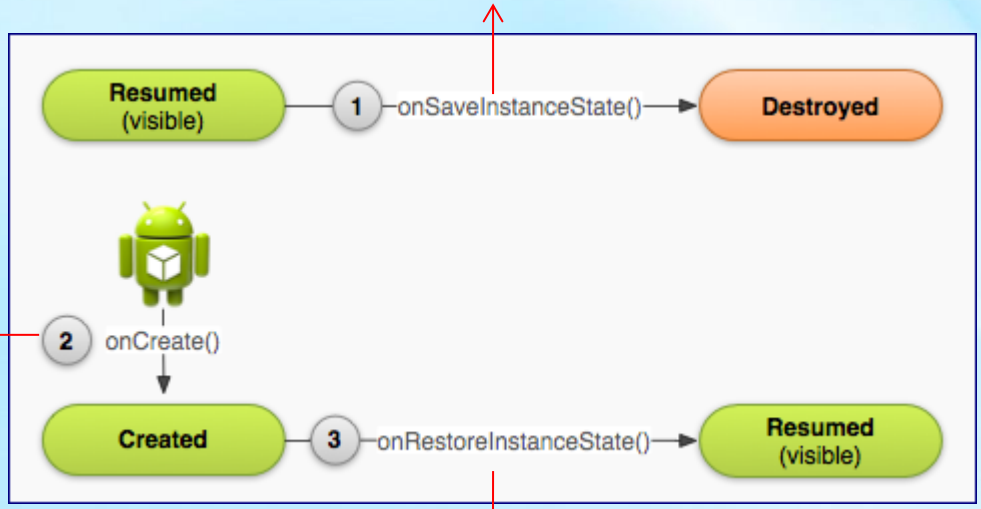
- N.B. In order for the Android system to restore the state of the views, each view must have a unique ID (android:id attribute)

onSaveInstanceState Method

- To save additional data about the activity state (to bundle)
  - **onSaveInstanceState()**
    - Method called when the user is leaving the activity
    - Argument: **Bundle** *savedInstanceState*
- Bundle data automatically used by the system
  - The same **Bundle** object will be passed to both the **onRestoreInstanceState()** and **onCreate()** methods
    - Will be used if the system must recreate the activity instance later
    - Called by the system only if the activity has been destroyed unexpectedly

onSaveInstanceState Method

Save additional information into bundle object



Same bundle object passed to those methods

Source: http://developer.android.com/training/basics/activity-lifecycle/recreating.html

onSaveInstanceState Method

- E.g, to add data to bundle

```
private double total;
```

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {

    savedInstanceState.putDouble("Total", total);

    // Always call the superclass so it can save the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

Additional information saved into bundle

onSaveInstanceState Method

- E.g, to retrieve data from bundle

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (savedInstanceState != null) {
        total = savedInstanceState.getDouble("Total");
    }
```

Data retrieved from the bundle

Webography

- http://developer.android.com/training/basics/activity-lifecycle/index.html
- http://developer.android.com/guide/components/activities.html