Android Activity

Sang Shin
Michèle Garoche
www.javapassion.com
"Learn with Passion!"

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - > http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - > http://creativecommons.org/licenses/by/2.5/

Topics

- What is an Activity?
- Declaration of Activities in manifest file
- Activities and application
- Activity life-cycle
- Activity stack

What is an Activity?

What is an Activity?

- An activity presents a visual user interface (screen) for one focused endeavor the user can undertake
- Examples - Individual activity
  - > A list of menu items users can choose from
  - > Display of photographs along with their captions
- Example - Sequence of activities
  - > A text messaging application might have one activity that shows a list of contacts to send messages to, a second activity to write the message to the chosen contact, and other activities to review old messages or change settings.

User Interaction Occurs on Views

- Visual content of the window (screen contents) is provided by a hierarchy (tree) of views
- Each view controls a particular rectangular space within the window.
  - > Parent views contain and organize the layout of their children.
  - > Leaf views (those at the bottom of the hierarchy) draw in the rectangles they control and respond to user actions directed at that space.
- Views are where the activity's interaction with the user takes place.
  - > For example, a view might display a small image or button and initiate an action when the user taps that image or button.

How Visual Content is Presented?

- Each activity is given a default window to draw
  - > Typically, the window fills the screen, but it might be smaller than the screen and float on top of other windows.
- A view hierarchy is placed within an activity's window by the *Activity.setContentView(..)* method.
  - > The content view is the View object at the root of the hierarchy.

Activities and Application

Application is Made up with Activities

- An application might consist of just one or more Activities
  - > What the activities are, and how many there are depends, of course, on the application
- Typically, one of the activities is marked as the first one (starting activity or main activity) that should be presented to the user when the application is launched.
  - > Specified in the AndroidManifest.xml
- Moving from one activity to another is accomplished by having the current activity start (call or launch) the next one

How an Activity Start Another?

- An activity is started in two ways
  - > *startActivity(Intent intent)* - Used when starting an activity (calling an activity) does not expect a result
  - > *startActivityForResult(Intent intent)* - Used when starting activity expects a result from the started activity
- The responding activity (started activity or target activity) can look at the initial intent that caused it to be launched by calling its *getIntent()* method
  - > (We have not learned Intent yet but think of an Intent as a message that gets passed around between Activities)

Declaration of Activities in the Manifest file

What is Manifest File?

- Every Activity has to be declared in the Manifest file - otherwise "Activity Not Found" error condition occurs

```
<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
    <application . . . >
        <activity android:name="com.example.project.FreneticActivity"
                android:icon="@drawable/small_pic.png"
                android:label="@string/freneticLabel"
                . . . >
        </activity>
        . . .
    </application>
</manifest>
```

Activity Lifecycle

An Activity has three states

- "Active" (or "Running") state
  - > When it is in the foreground of the screen (at the top of the activity stack for the current task).
  - > This is the activity that is the focus for the user's actions.
- "Paused" state
  - > Lost focus but is still visible to the user
  - > Another activity lies on top of it and that activity either is transparent or doesn't cover the full screen, so some of the paused activity can show through
  - > A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations

An Activity has three states

- "Stopped" state
  - > Completely obscured by another activity
  - > It still retains all state and member information.
  - > However, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.

State Transition Callback Methods

- As an activity transitions from state to state, it is notified of the change by calls to the following protected methods:
    - > void onCreate(Bundle savedInstanceState)
    - > void onStart()
    - > void onRestart()
    - > void onResume()
    - > void onPause()
    - > void onStop()
    - > void onDestroy()
- An activity must implement *onCreate()* to do the initial setup when the object is first instantiated

onCreate()

- Called when the activity is first created.
- This is where you should do all of your normal static set up
  - > Create views, bind data to lists, and so on.
- This method is passed a Bundle object containing the activity's previous state, if that state was captured
- Always followed by *onStart()*

Lifecycle Lifetime

- Entire lifetime
  - > Between the first call to *onCreate()* through to a single final call to *onDestroy()*.
- Visible lifetime
  - > Between a call to *onStart()* until a corresponding call to *onStop()*
- Foreground lifetime
  - > Between a call to *onResume()* until a corresponding call to *onPause()*
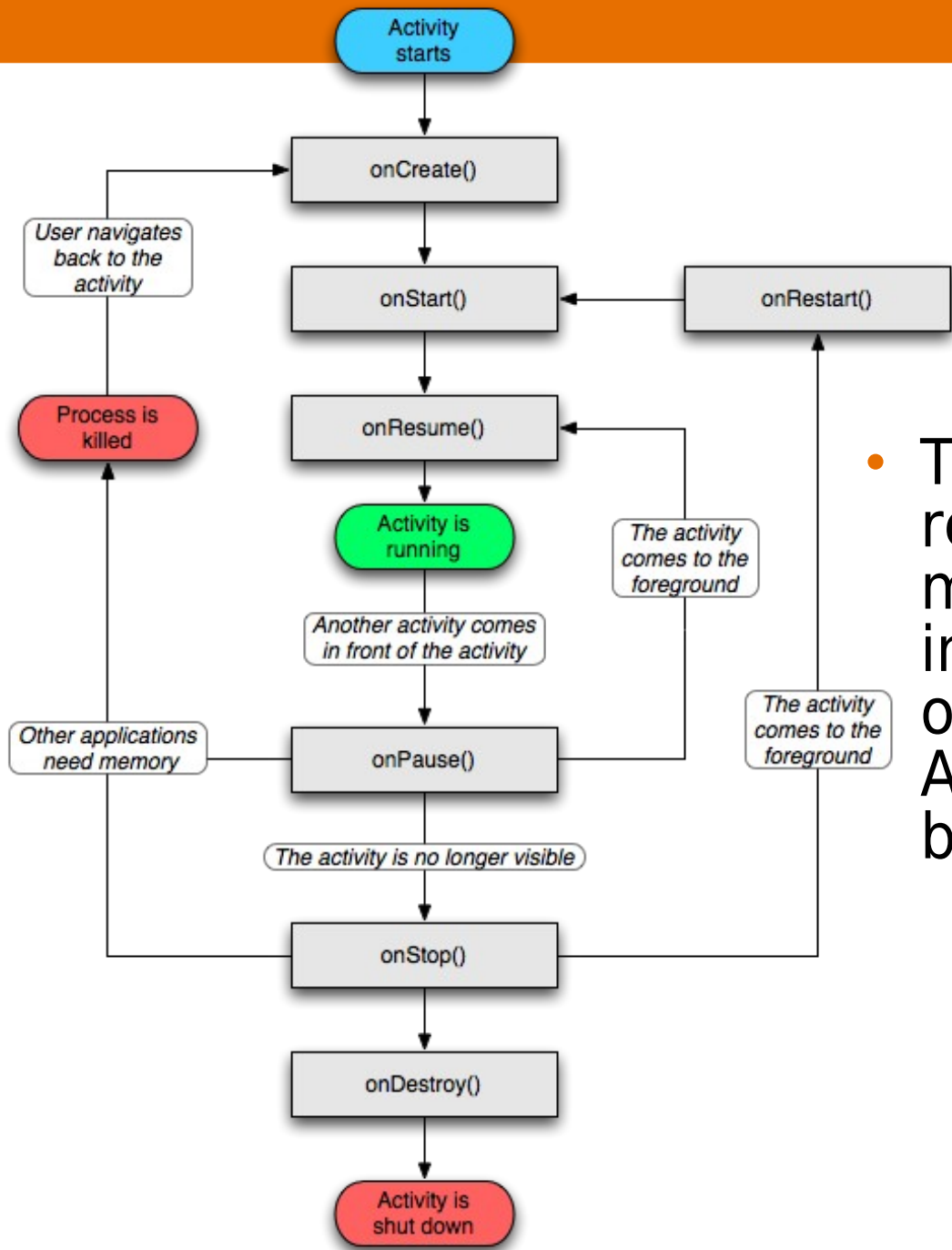
- The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states.

19

Activity Stack

Activity Stack

- Activities in the system are managed as an activity stack.
    - > It is typical a user will be switching activities while using a mobile device
    - > Example scenario: A user is reading emails, then phone call comes in. The user takes the call (activity switch occurs) and when the phone conversation is done, goes back to reading emails (activity switch occurs)
- When a new activity is started, it is placed on the top of the stack and becomes the running activity
    - > The previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"