

ITSMAP F13 Lesson 2

Applications (Apps)

Activity + Resources

Jesper Rosholm Tørresø

Subjects Lesson 2+3

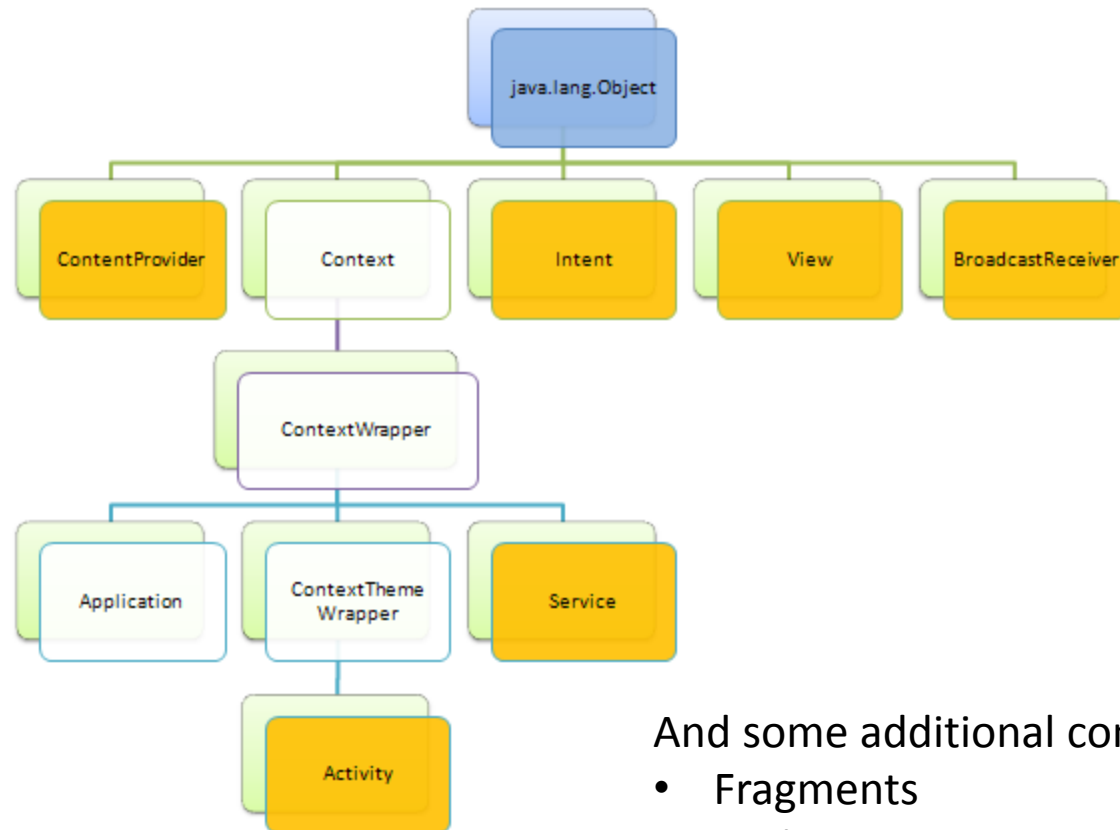
Tuesday L2

1. Android Application , the “App”
 - Principle of The Android Manifest
2. Activity Control and Life cycle, The Android App Lifercycle
 - Responding to activity changes

Thursday L3

1. Activities and resources (Externalization).
 - Using resources in layouts and manifest
 - Retrieving resources at runtime
2. Android Intents
3. Debugging and screencasts

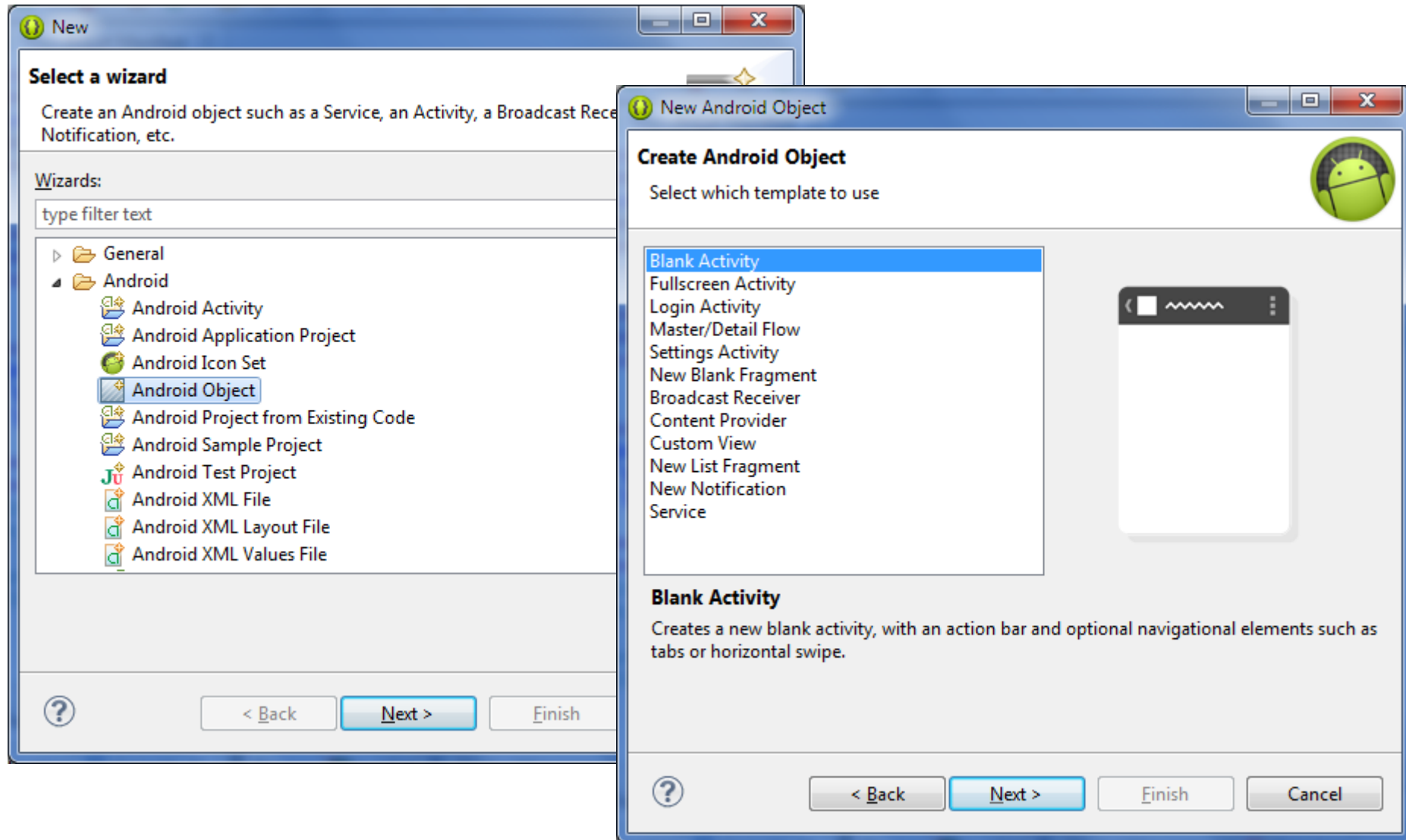
Android application architecture



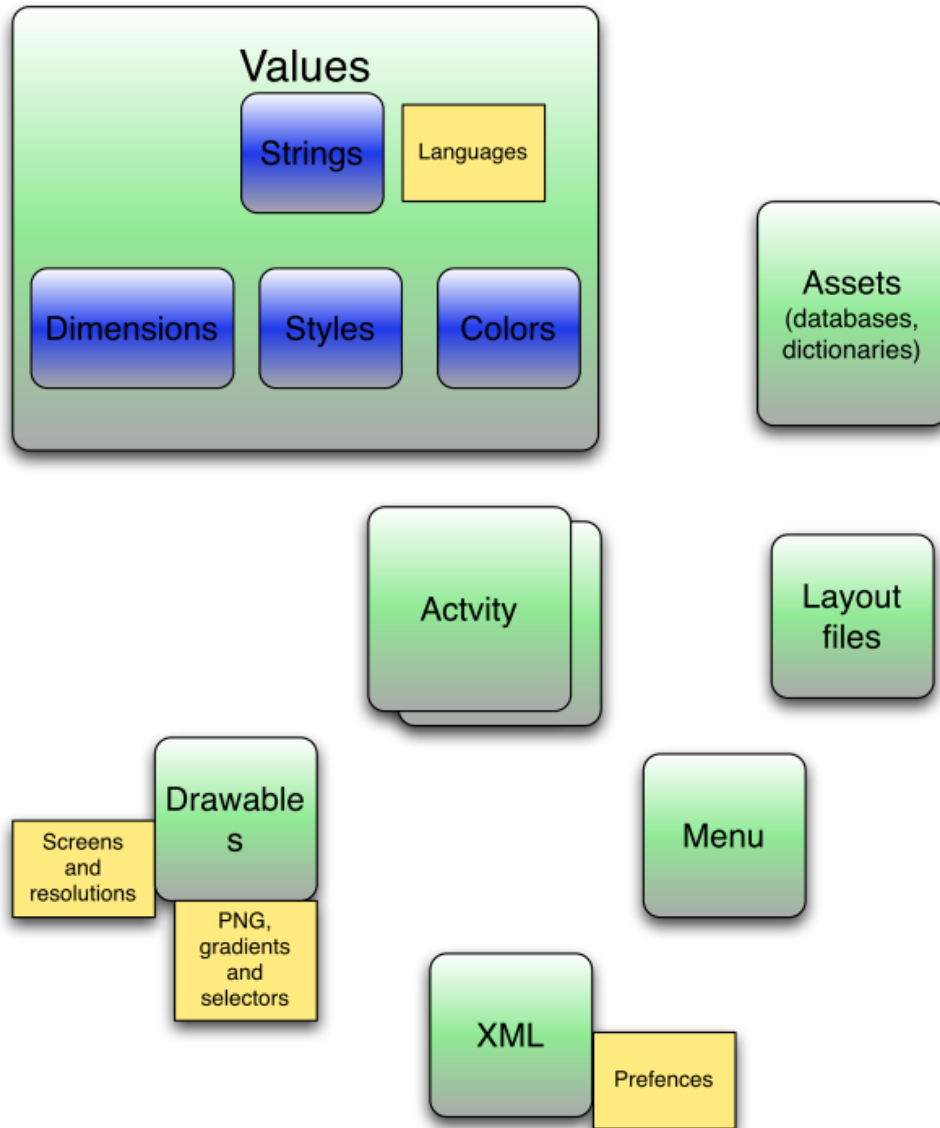
And some additional components

- Fragments
- Widget
- Notifications

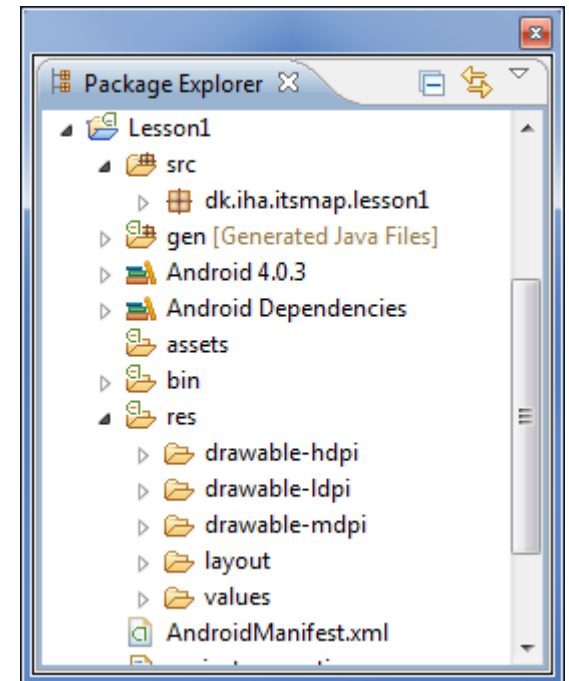
Templates in Eclipse ADT



Basic App Anatomy



Eclipse



The Android Manifest

An App' "ticket" to the Framework

- Defines the App' structure, metadata, components and requirements
- Done by an XML file in which the nodes are
 - The components (Activities, Service...)
 - Attributes/(Properties) to the App, like its name
 - Version information
 - Intent filters and permissions
 - Requirements to platform like having a camera
 -
- Uses an Android namespace with system attributes

The XML file

AndroidManifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest android:versionCode="1" android:versionName="1.0"
    package="dk.iha.itsmap.Lesson1"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="15"/>
    <application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name">
        <activity android:label="@string/app_name"
            android:name="MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service android:name=".Lesson1Service"/>
        <provider
            android:authorities="dk.iha.itsmap.Lesson1.Lesson1contentprovider"
            android:name=".Lesson1ContentProvider"/>
    </application>
</manifest>
```

Use the *Android SDK* documentation!

Nodes in the Manifest (just click the nodes)

[<action>](#)

[<action>](#)

[<activity>](#)

[<activity-alias>](#)

[<application>](#)

[<category>](#)

[<compatible-screens>](#)

[<data>](#)

[<grant-uri-permission>](#)

[<instrumentation>](#)

[<intent-filter>](#)

[<manifest>](#)

[<meta-data>](#)

[<path-permission>](#)

[<permission>](#)

[<permission-group>](#)

[<permission-tree>](#)

[<provider>](#)

[<receiver>](#)

[<service>](#)

[<supports-gl-texture>](#)

[<supports-screens>](#)

[<uses-configuration>](#)

[<uses-feature>](#)

[<uses-library>](#)

[<uses-permission>](#)

[<uses-sdk>](#)

All the elements that can appear in the manifest file are listed in alphabetical order. These are the only legal elements; **you cannot add your own elements or attributes.**

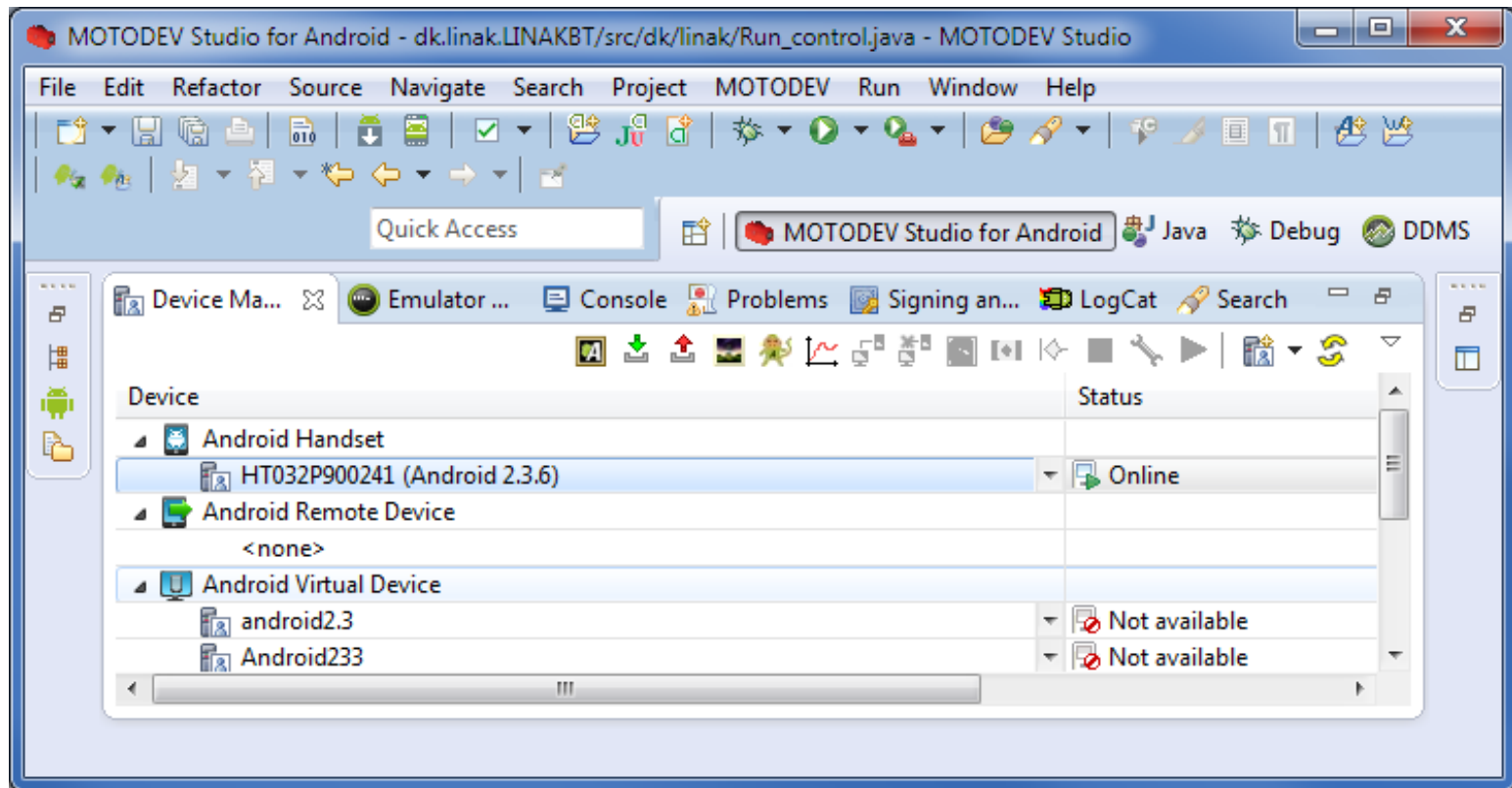
<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

Life cycle of an App/Activity

- The Android Frameworks way to control an Activity
- Life cycle control also goes for other component like Service.
- State changes calls a number of handles depending on current state and events for state change

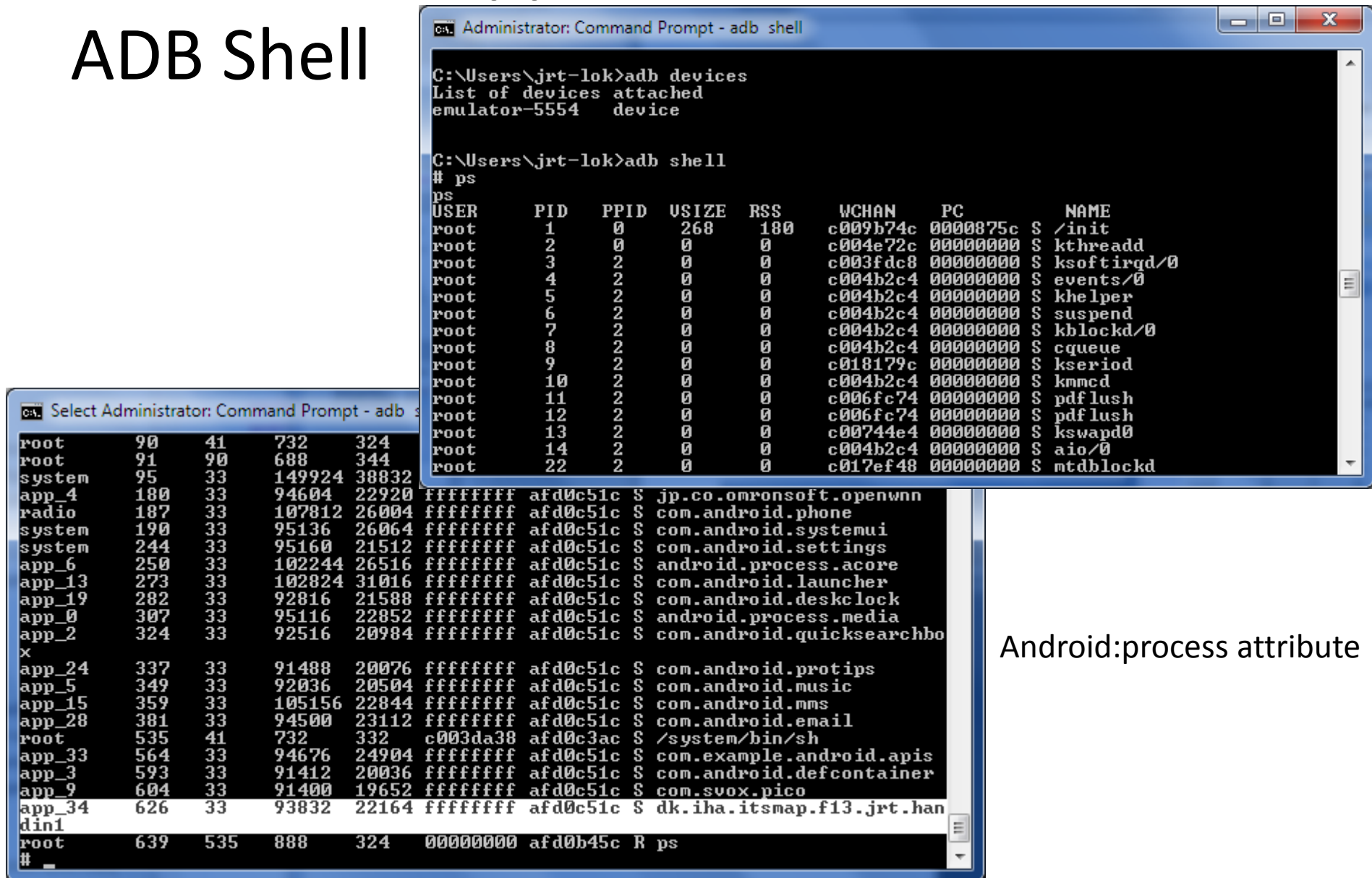
An Android App is a Linux Process

Use the ADB Shell



An Android App is a Linux Process

ADB Shell



```
Administrator: Command Prompt - adb shell

C:\Users\jrt-lok>adb devices
List of devices attached
emulator-5554    device

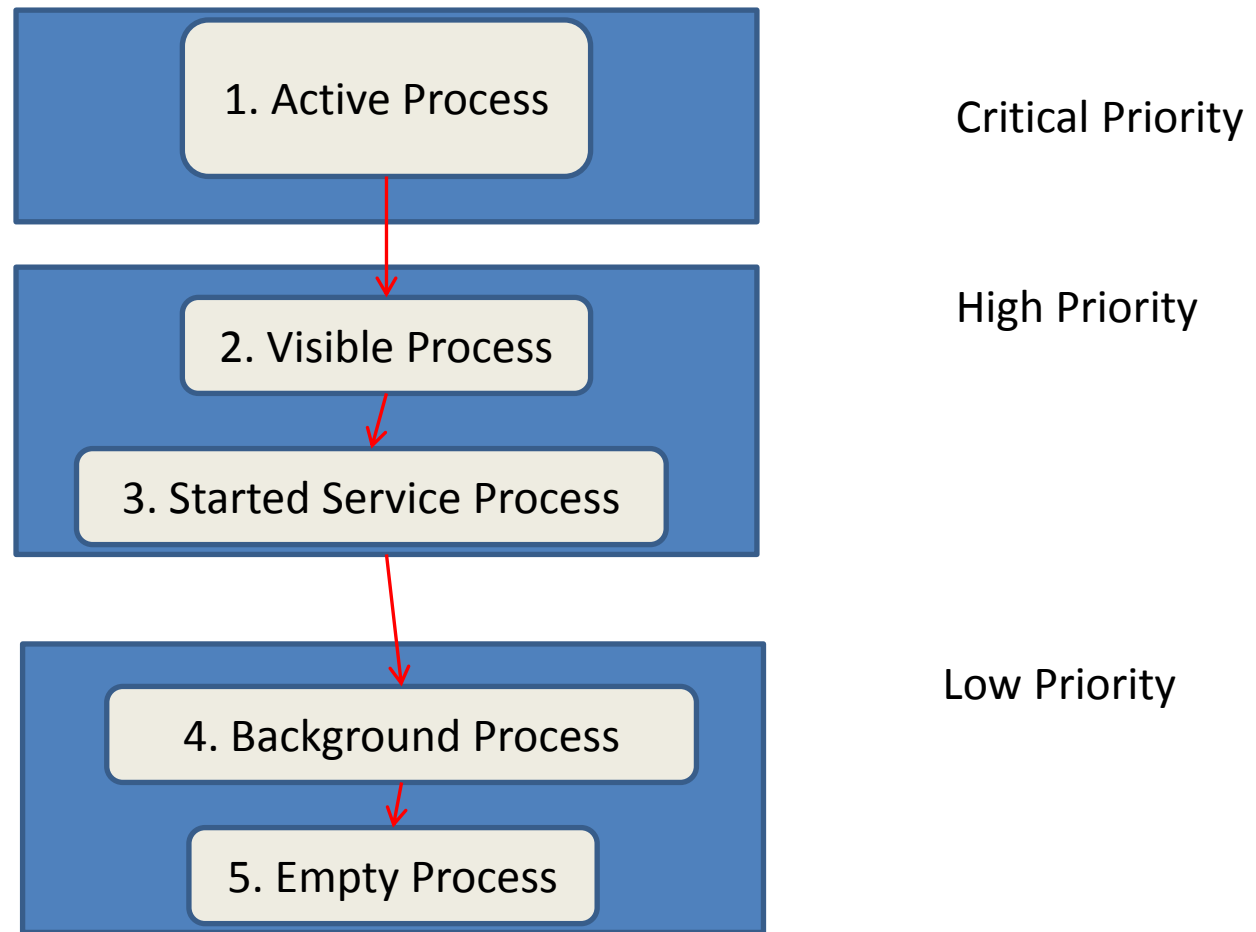
C:\Users\jrt-lok>adb shell
# ps
ps
USER      PID   PPID  USIZE  RSS    WCHAN    PC      NAME
root       1     0      268    180    c009b74c 0000875c $ /init
root       2     0       0     0      c004e72c 00000000 $ kthreadd
root       3     2       0     0      c003fdc8 00000000 $ ksoftirqd/0
root       4     2       0     0      c004b2c4 00000000 $ events/0
root       5     2       0     0      c004b2c4 00000000 $ khelper
root       6     2       0     0      c004b2c4 00000000 $ suspend
root       7     2       0     0      c004b2c4 00000000 $ kblockd/0
root       8     2       0     0      c004b2c4 00000000 $ cqueue
root       9     2       0     0      c018179c 00000000 $ kseriod
root      10     2       0     0      c004b2c4 00000000 $ kmmcd
root      11     2       0     0      c006fc74 00000000 $ pdf_lush
root      12     2       0     0      c006fc74 00000000 $ pdf_lush
root      13     2       0     0      c00744e4 00000000 $ kswapd0
root      14     2       0     0      c004b2c4 00000000 $ aio/0
root      22     2       0     0      c017ef48 00000000 $ mtblockd

Select Administrator: Command Prompt - adb shell

root      90    41     732    324
root      91    90     688    344
system    95    33   149924 38832
app_4     180   33    94604 22920 ffffffff afd0c51c $ jp.co.omronsoft.openwnn
radio     187   33   107812 26004 ffffffff afd0c51c $ com.android.phone
system    190   33    95136 26064 ffffffff afd0c51c $ com.android.systemui
system    244   33    95160 21512 ffffffff afd0c51c $ com.android.settings
app_6     250   33   102244 26516 ffffffff afd0c51c $ android.process.acore
app_13    273   33   102824 31016 ffffffff afd0c51c $ com.android.launcher
app_19    282   33    92816 21588 ffffffff afd0c51c $ com.android.deskclock
app_0     307   33    95116 22852 ffffffff afd0c51c $ android.process.media
app_2     324   33    92516 20984 ffffffff afd0c51c $ com.android.quicksearchbo
x
app_24    337   33    91488 20076 ffffffff afd0c51c $ com.android.protips
app_5     349   33    92036 20504 ffffffff afd0c51c $ com.android.music
app_15    359   33   105156 22844 ffffffff afd0c51c $ com.android.mms
app_28    381   33    94500 23112 ffffffff afd0c51c $ com.android.email
root      535   41     732    332    c003da38 afd0c3ac $ /system/bin/sh
app_33    564   33    94676 24904 ffffffff afd0c51c $ com.example.android.apis
app_3     593   33    91412 20036 ffffffff afd0c51c $ com.android.defcontainer
app_9     604   33    91400 19652 ffffffff afd0c51c $ com.svox.pico
app_34    626   33   93832 22164 ffffffff afd0c51c $ dk.iha.itsmap.f13.jrt.han
din1
root      639   535    888    324    00000000 afd0b45c R ps
#
```

Android:process attribute

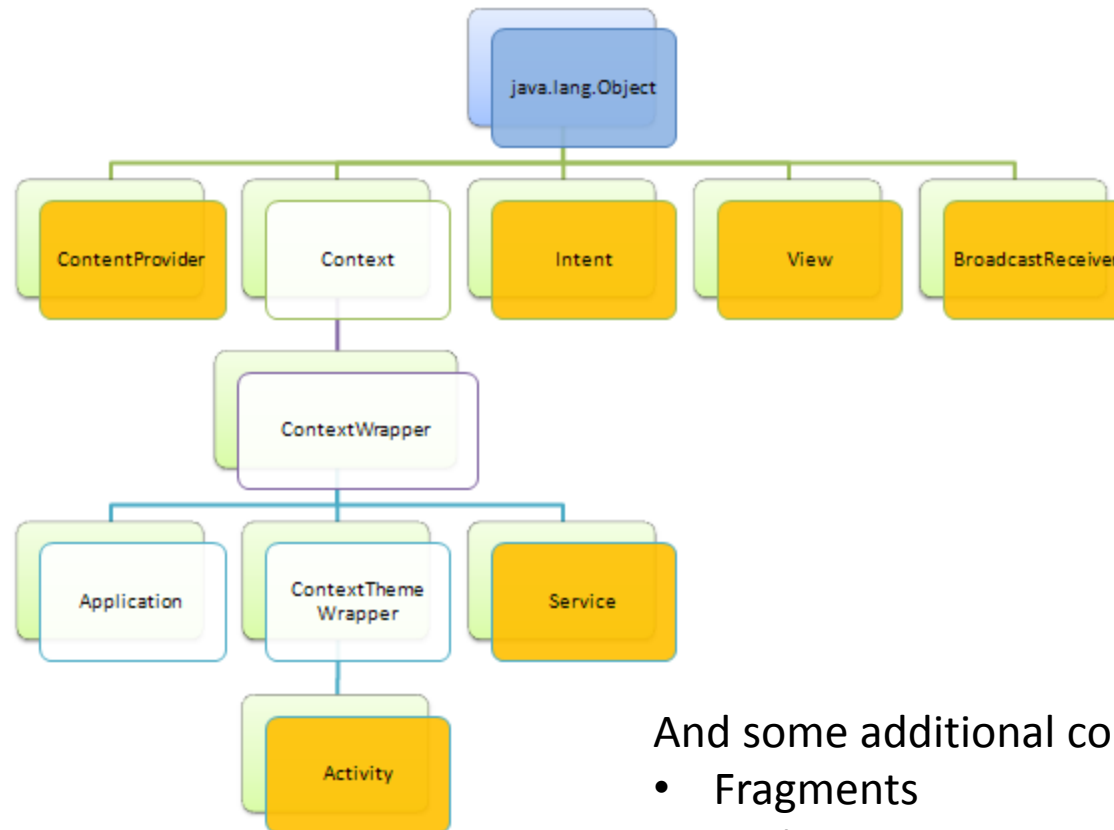
Process state and components



The app component running decides running/process state of your application

Android application architecture

The components



And some additional components

- Fragments
- Widget
- Notifications

Android Application class

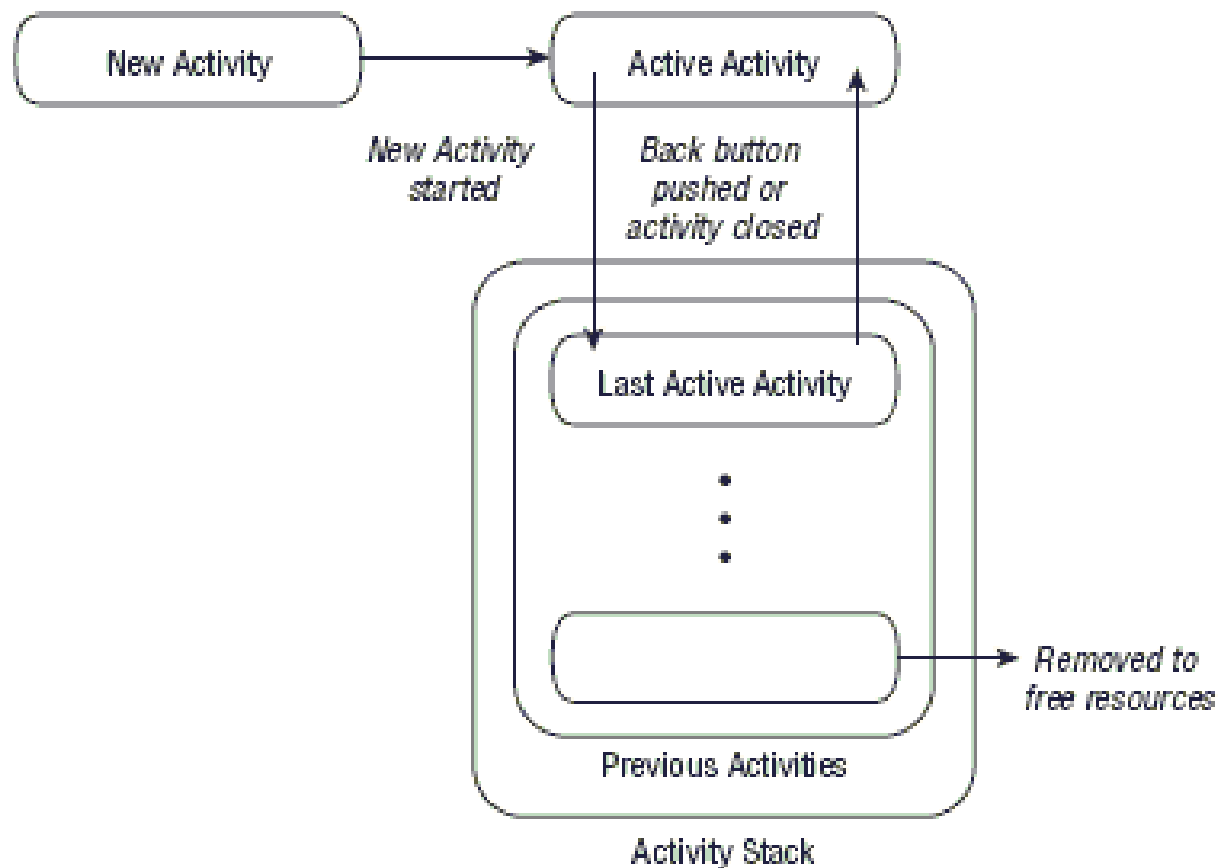
“Android Singleton”

- The “logical top level” Android component that has lifecycle and thereby some impact at process state
 - Responds to application level events from Android Frame Work (run time low level memory)
 - Transfer object between application components
 - The only component guaranteed to be instantiated through whole App lifetime
 - Manage and maintain resources for several App Components
 - Configured in the Android Manifest

Android Activity

- An important component that that let the Android Framework decide state of the process running the App
- Android controls the Lifecycle of an Activity
- and all other Android components like Service.

Activity Stack



Activity Stack + Life cycle

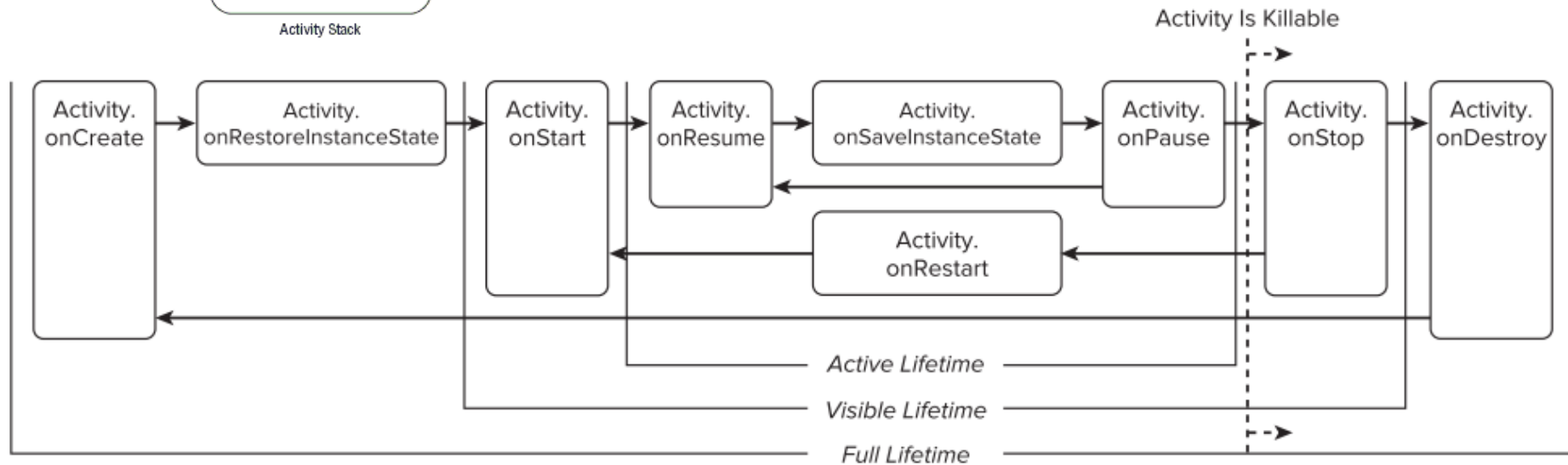
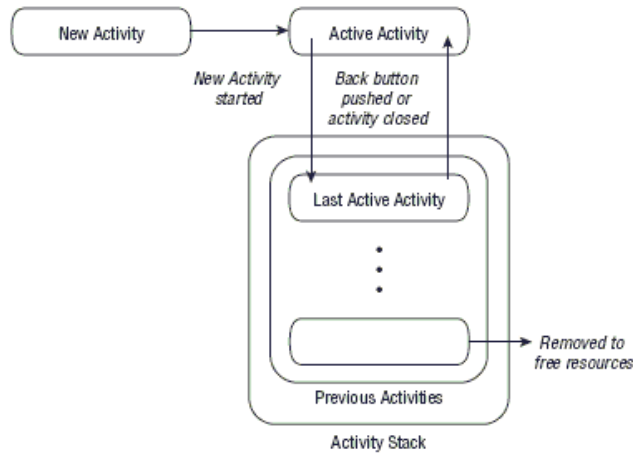
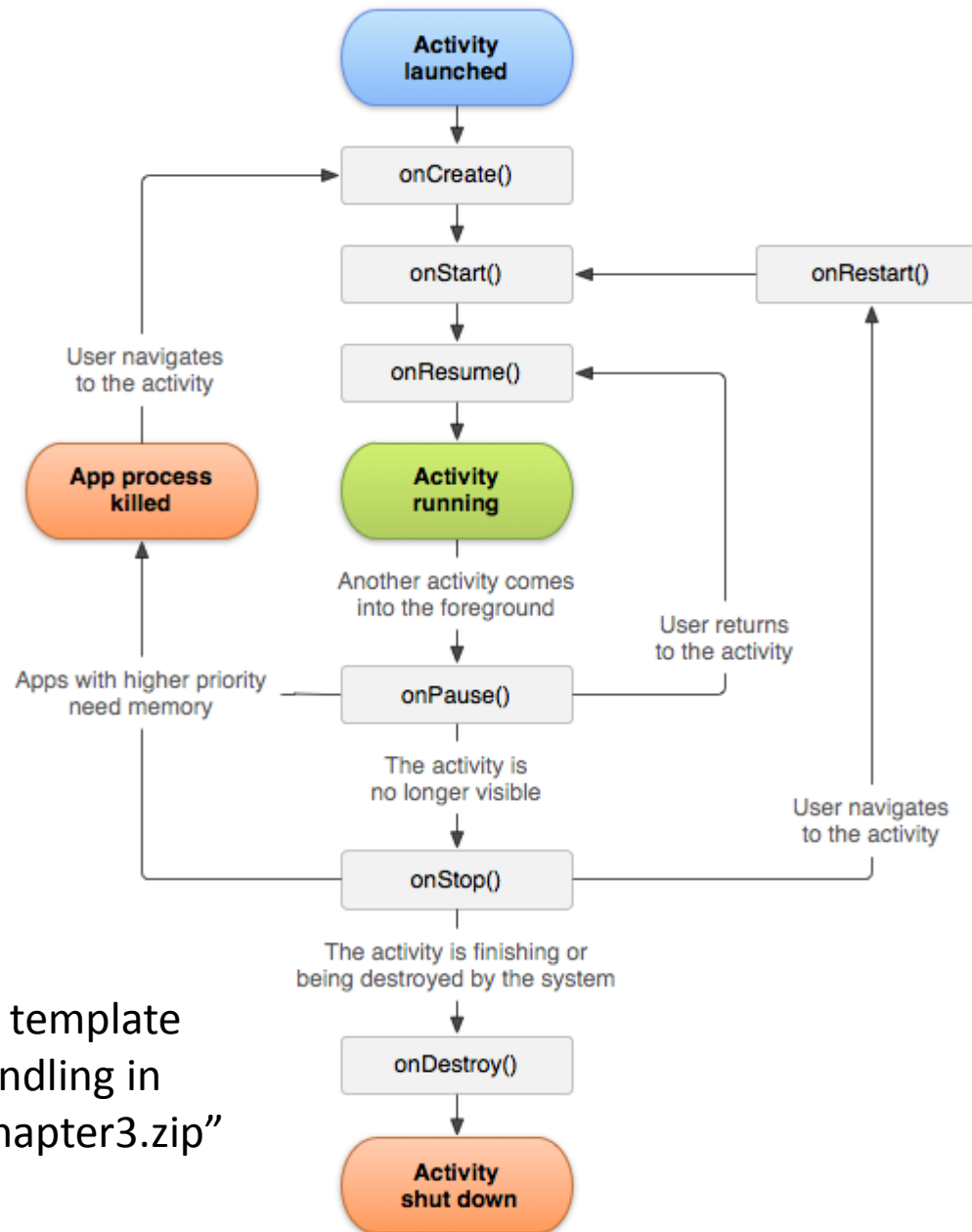


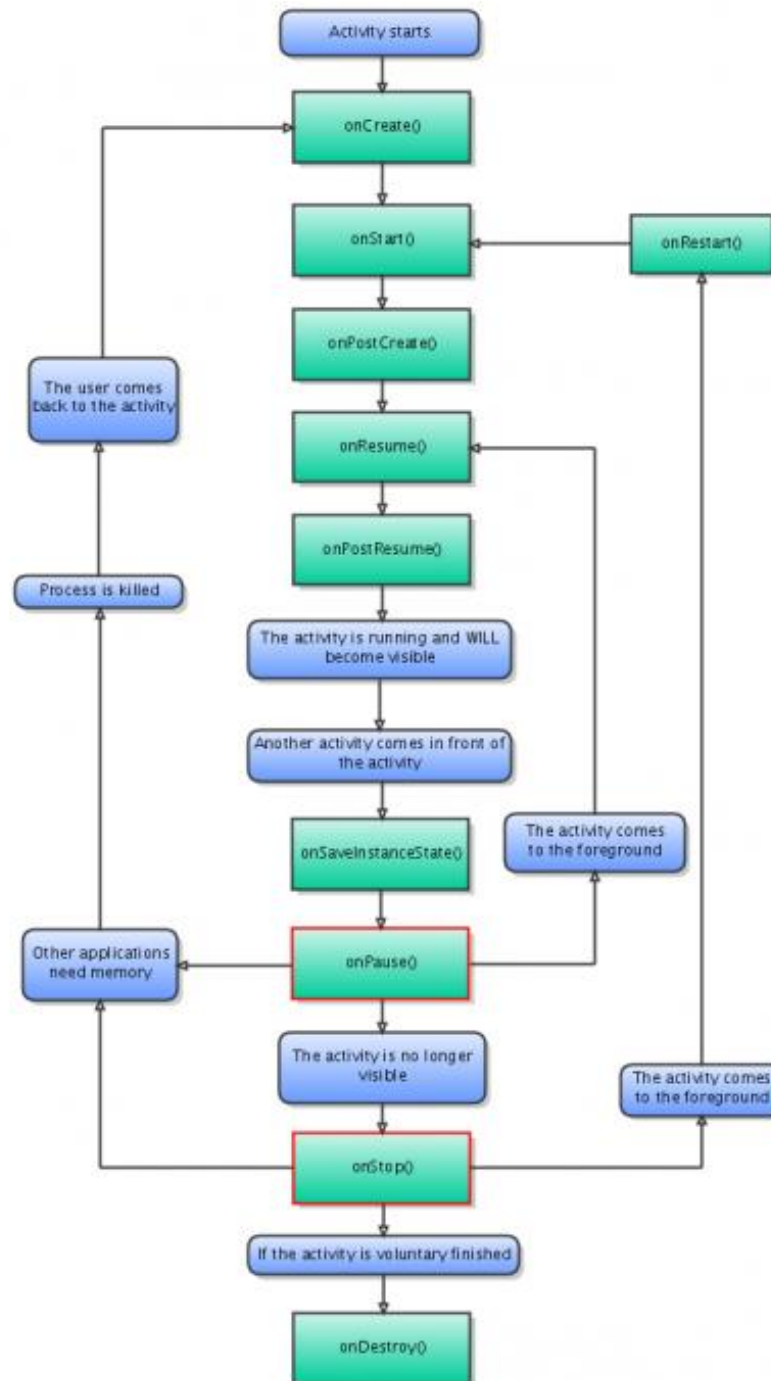
FIGURE 3-6



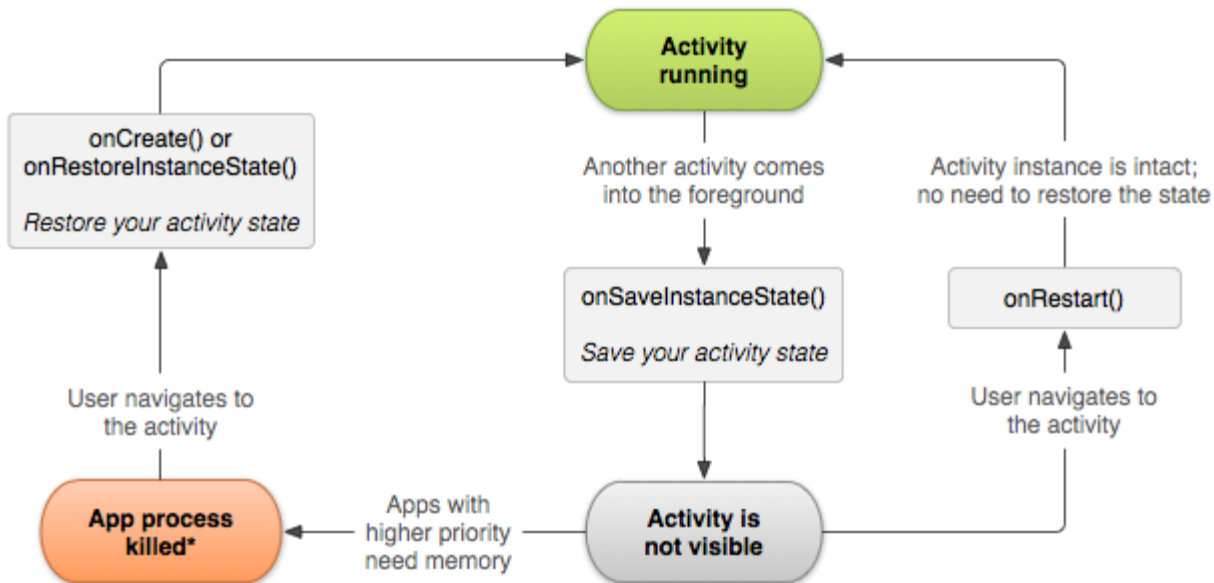
Find a Activity template
with state handling in
Lesson 2 files “chapter3.zip”

<http://developer.android.com/guide/topics/fundamentals/activities.html>

Android Activity Life Cycle with Instance Save/Restore:

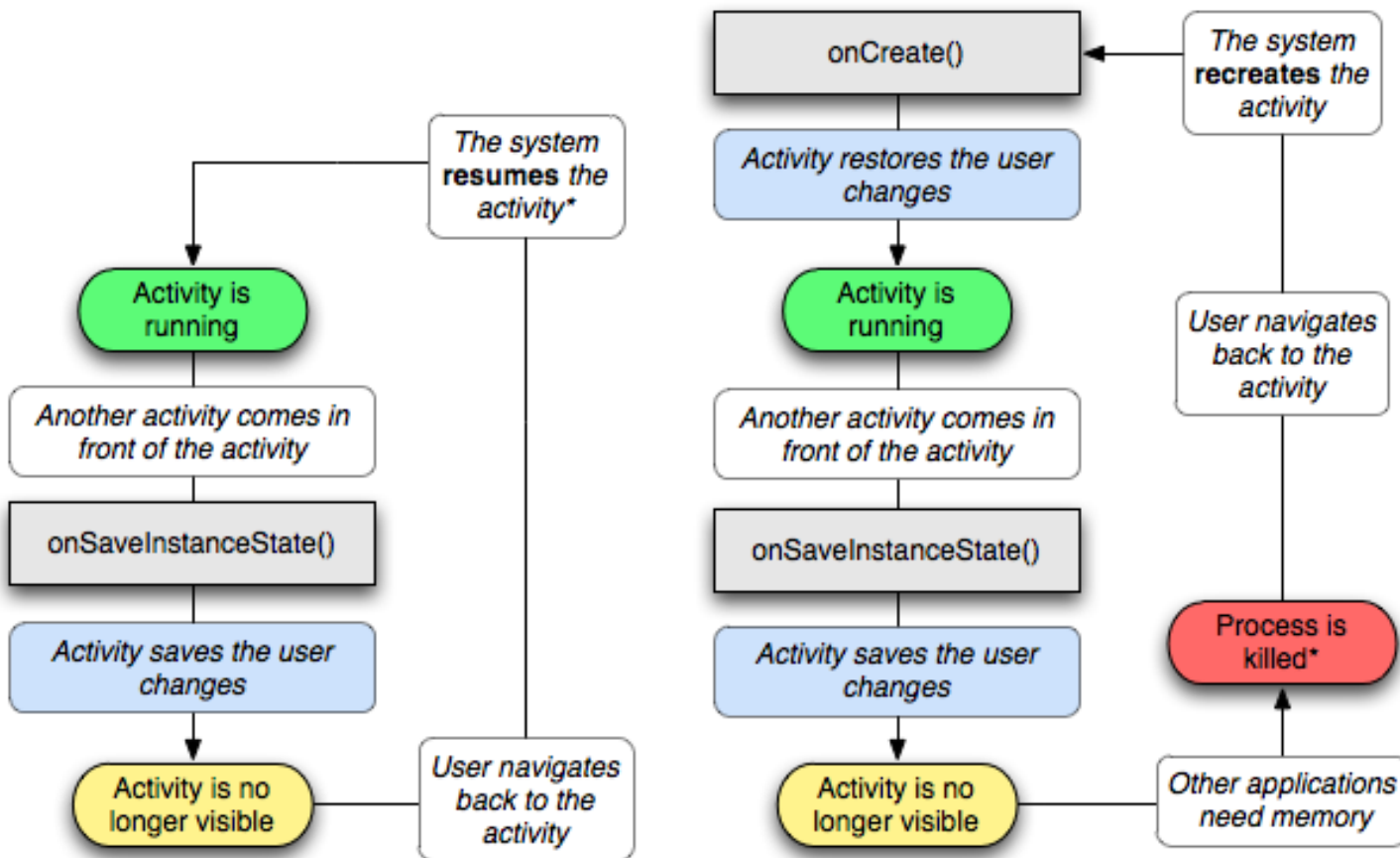


Saving activity state



*Activity instance is destroyed, but the state from `onSaveInstanceState()` is saved

Detailed Instance Save/Restore Diagram



* There's no need to restore state, because the activity is intact

* User changes are lost