Application Fundamentals

Runtime environment

- Applications are written in Java.
- Android SDK tools compile the code + additional data and resources into *android package.*
- All the code in single *APK* file is one application.

Runtime environment

- Android OS is multi user Linux system.
- Each application is a different user.
- System assigns unique user id to each application.
- System sets permissions for all the files in an application so that only the app's user ID can access them.
- Each process has it's own VM.
- Every application runs in its own linux process.

“Need to know basis”

- Every application can access only components that it needs to access.
- An application can request permission to access device data / HW (user contacts, phone, network access etc.)

Building Blocks

- Activities
- Services
- Content provider
- Broadcast receiver.

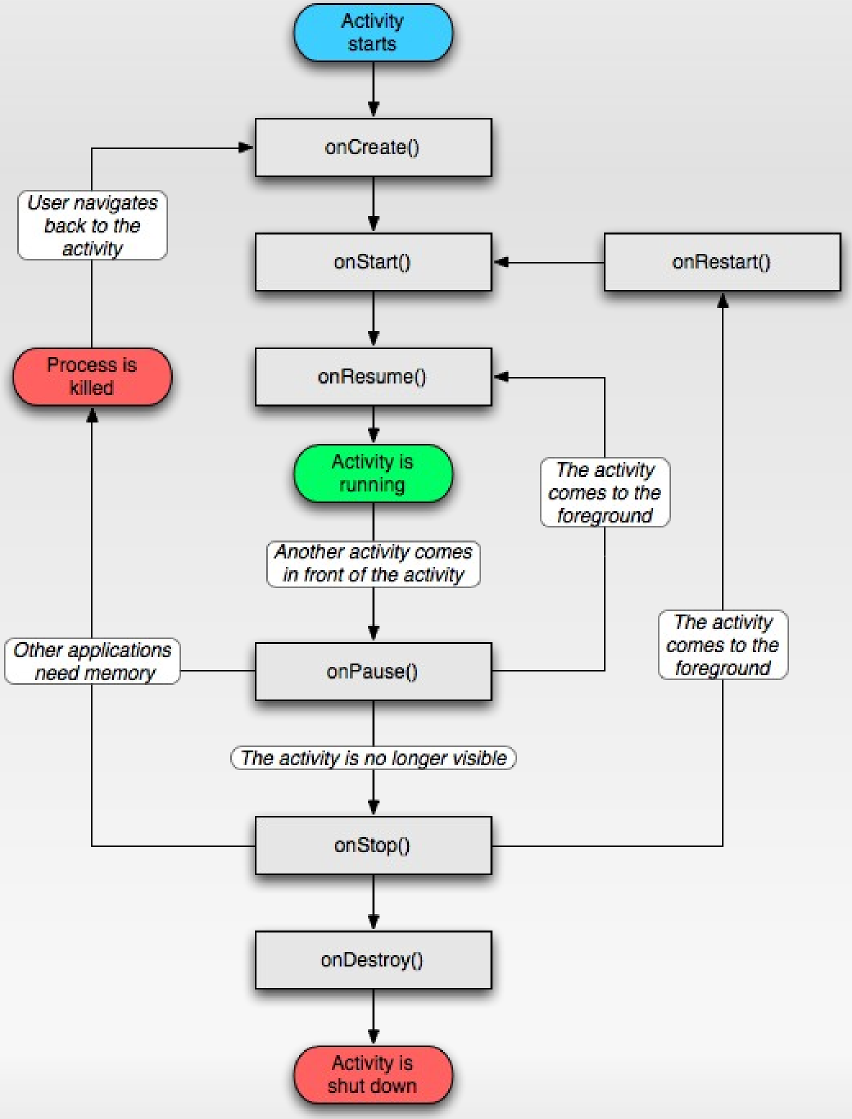
Activities

- *Activity* represents single screen with user interface.
- Application may have more than one *Activity*
  - Email app that has one activity that shows list of new emails, one for composing mail and one for reading emails.
- Each is *Activity* independent of the others.
  - Different apps can start specific activity in other app.

Activities

- Stacked like a deck of cards.
- Only one is visible.
- Only one is active.
- New *Activities* are places on top.

Activity's Life-cycle



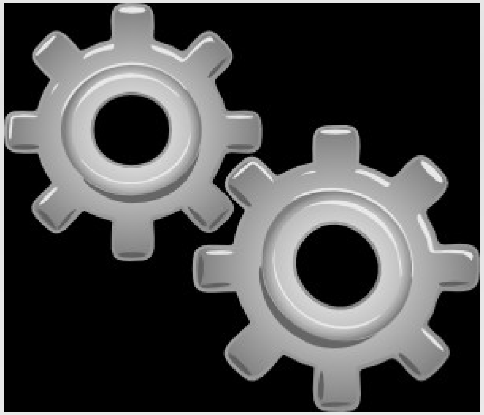
Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
The activity comes to the foreground
Other applications need memory
onPause()
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Activities states

- Active
  - At the top of the stack.
- Paused
  - Lost focus but still visible (other activity was launched but does not cover the entire screen).
- Stopped
  - Not at the top of the stack, not visible.
- Dropped
  - Killed by android system.

Services

- A *Service* runs in the background.
- *Service* Does not provide user interface.
    - Play music in the background while the user is in different application.
    - Fetch data over the network without blocking user interaction with an activity.
- Can be started by other components.



Content Provider

- *Content Provider*s manages a shared set of application data.
- Allows other application to
  - Retrieve data
  - Modify data.
- **Only** way to share data between applications.
- For Example: Android system has *Content Provider* that manages user's contacts.

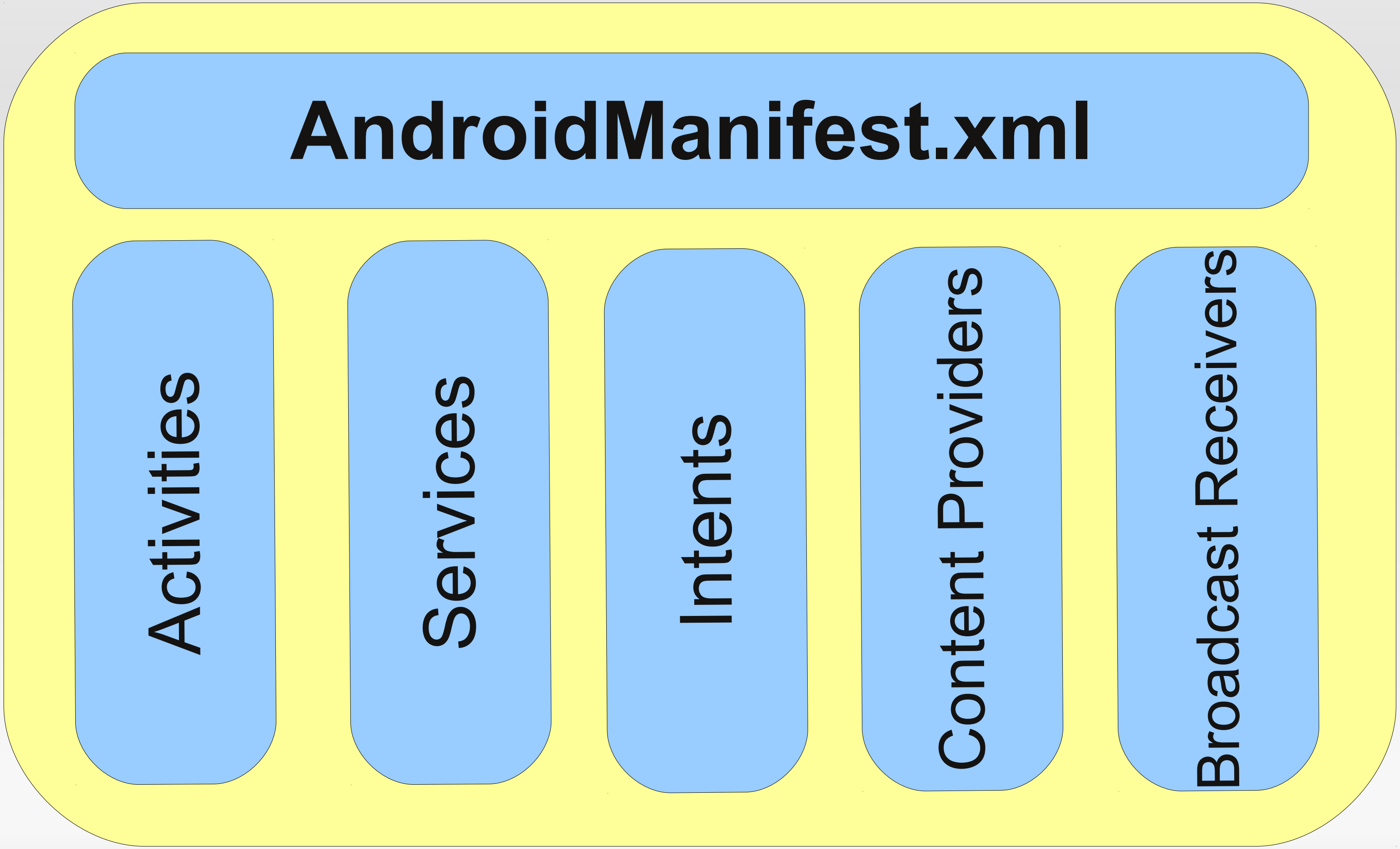
Broadcast Receiver

- *Broadcast Receiver* responds to system wide broadcast announcements.
- Many announcements originates by the system
  - Screen turned off.
  - Battery low.
  - Picture was captured.
- Application can also initiate broadcasts
  - Letting other apps know that data has been downloaded.
- May create status bar notification.

Activating Components

- *Activities, services* and *broadcast receivers* are activated by async message called ***intent***.
- ***Intents*** are used to move from *Activity* to *Activity*.
- Describes what the application wants.
  - For example: “View”, “Dial” or “Send” something.
  - May specify URI of the data to act on.

Binding it all together



AndroidManifest.xml

Activities

Services

Intents

Content Providers

Broadcast Receivers

AndroidManifest.xml

- Control file declaring application's components.
- The “glue” the specifies which *Intents* our *Activities* receive.
- Specifies permission.
- Declares minimum API level (i.e. android version)
- Minimum HW requirement (camera, BT, etc.)

Views

- *Views* are basic UI building blocks.
- Knows how to draw themselves.
- Similar to java swing.
- Responds to event.
- Can layout in code or in XML.