Android Developer Fundamentals

Unit 1

Android Activity:

Almost in all languages a program starts from **main()** function.

Similarly, the Android system initiates its program with in an **Activity.**

The Activity class provides a number of callbacks that allow the activity to know that a state has changed:

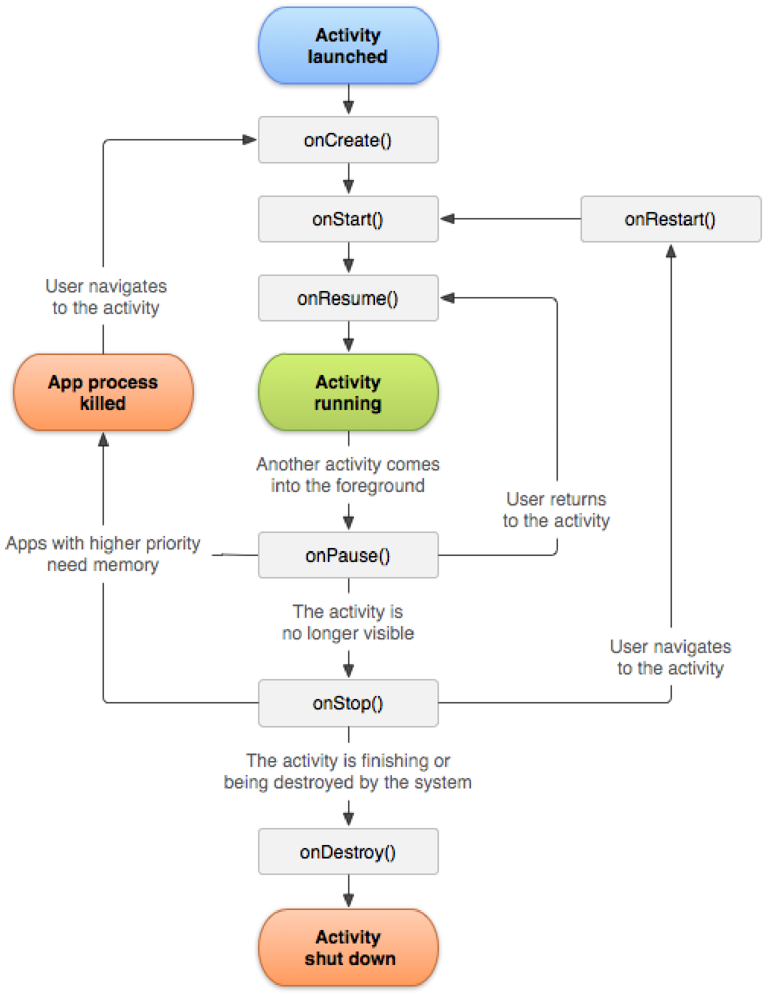
- that the system is creating,
- stopping,
- or resuming an activity,
- or destroying the process in which the activity resides.

Good implementation of the lifecycle callbacks can help ensure that your app avoids:

- Crashing if the user receives a phone call or switches to another app while using your app.
- Consuming valuable system resources when the user is not actively using it.
- Losing the user's progress if they leave your app and return to it at a later time.
- Crashing or losing the user's progress when the screen rotates between landscape and portrait orientation.

  To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks:

  onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy()


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

States in Android Activity:

An activity can be in different states depending how it is interacting with the user.

State	Description
Running	Activity is visible and interacts with the user.
Paused	Activity is still visible but partially obscured, instance is running but might be killed by the system.
Stopped	Activity is not visible, instance is running but might be killed by the system.
Killed	Activity has been terminated by the system of by a call to its finish() method.

Methods or callbacks in Android Activity:

If an activity is in the foreground, and the user taps the **Back** button, the activity transitions through the onPause(), onStop(), and onDestroy()callbacks.

Method	Purpose
onCreate()	Called then the activity is created. Used to initialize the activity, for example create the user interface.
onResume()	Called if the *activity* get visible again and the user starts interacting with the activity again. Used to initialize fields, register listeners, bind to services, etc.
onPause()	Called once another activity gets into the foreground. Always called before the *activity* is not visible anymore. Used to release resources or save application data. For example you unregister listeners, intent receivers, unbind from services or remove system service listeners.
onStop()	Called once the activity is no longer visible. Time or CPU intensive shut-down operations, such as writing information to a database should be down in the onStop() method.
onDestroy()	This callback is called before the activity is destroyed by the system.
onRestart()	This callback is called when the activity restarts after stopping it.

Activity State Changes

Different events, some user-triggered and some system-triggered, can cause an Activity to transition from one state to another.

There are a number of events that can trigger a configuration change. Perhaps the most prominent example is a change between portrait and landscape orientations. Other cases that can cause configuration changes include changes to language or input device.

When a configuration change occurs, the activity is destroyed and recreated. If you wish to preserve transient-state data for the activity, you must override the onSaveInstanceState() method to save the data, and then use onCreate() or onRestoreInstanceState() callbacks to recreate the instance state.

If an activity is in the foreground, and the user taps the **Back** button, the activity transitions through the onPause(), onStop(), and onDestroy() callbacks.

Assume for example the user scrolled through a ListView with thousands of items and the activity is recreated. Losing the position in the list is annoying for the user, hence the position should be restored.
The onSaveInstanceState() can be used to store the instance state.