### Activity Lifecycle

Activities in the system are managed as an *activity stack*. 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.

An activity has essentially four states:

If an activity is in the foreground of the screen (at the top of the stack), it is *active* or *running*.

If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is *paused*. 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.

If an activity is completely obscured by another activity, it is *stopped*. 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.

If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.



There are three key loops you may be interested in monitoring within your activity:

The entire lifetime of an activity happens between the first call to [onCreate(Bundle)](https://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) through to a single final call to [onDestroy()](https://developer.android.com/reference/android/app/Activity.html" \l "onDestroy()). An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy(). For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread in onDestroy().

The visible lifetime of an activity happens between a call to [onStart()](https://developer.android.com/reference/android/app/Activity.html" \l "onStart()) until a corresponding call to [onStop()](https://developer.android.com/reference/android/app/Activity.html" \l "onStop()). During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user. Between these two methods you can maintain resources that are needed to show the activity to the user. For example, you can register a [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) in onStart() to monitor for changes that impact your UI, and unregister it in onStop() when the user no longer sees what you are displaying. The onStart() and onStop() methods can be called multiple times, as the activity becomes visible and hidden to the user.

The foreground lifetime of an activity happens between a call to [onResume()](https://developer.android.com/reference/android/app/Activity.html" \l "onResume()) until a corresponding call to [onPause()](https://developer.android.com/reference/android/app/Activity.html" \l "onPause()). During this time the activity is in front of all other activities and interacting with the user. An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

The entire lifecycle of an activity is defined by the following Activity methods. All of these are hooks that you can override to do appropriate work when the activity changes state. All activities will implement [onCreate(Bundle)](https://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) to do their initial setup; many will also implement [onPause()](https://developer.android.com/reference/android/app/Activity.html" \l "onPause()) to commit changes to data and otherwise prepare to stop interacting with the user. You should always call up to your superclass when implementing these methods.