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)](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) through to a single final call to [onDestroy()](http://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()](http://developer.android.com/reference/android/app/Activity.html" \l "onStart()) until a corresponding call to [onStop()](http://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](http://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()](http://developer.android.com/reference/android/app/Activity.html" \l "onResume()) until a corresponding call to [onPause()](http://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)](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) to do their initial setup; many will also implement [onPause()](http://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.

 public class Activity extends ApplicationContext {  
     protected void onCreate(Bundle savedInstanceState);  
  
     protected void onStart();  
       
     protected void onRestart();  
  
     protected void onResume();  
  
     protected void onPause();  
  
     protected void onStop();  
  
     protected void onDestroy();  
 }

In general the movement through an activity's lifecycle looks like this:

| Method | | | Description | Killable? | Next |
| --- | --- | --- | --- | --- | --- |
| [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) | | | 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, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one.  Always followed by onStart(). | No | onStart() |
|  | [onRestart()](http://developer.android.com/reference/android/app/Activity.html#onRestart()) | | Called after your activity has been stopped, prior to it being started again.  Always followed by onStart() | No | onStart() |
| [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart()) | | Called when the activity is becoming visible to the user.  Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden. | No | onResume()or onStop() |
|  | [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume()) | Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it.  Always followed by onPause(). | No | onPause() |
| [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause()) | Called when the system is about to start resuming a previous activity. This is typically used to commit unsaved changes to persistent data, stop animations and other things that may be consuming CPU, etc. Implementations of this method must be very quick because the next activity will not be resumed until this method returns.  Followed by either onResume() if the activity returns back to the front, or onStop()if it becomes invisible to the user. | **Pre-**[HONEYCOMB](http://developer.android.com/reference/android/os/Build.VERSION_CODES.html#HONEYCOMB) | onResume()or onStop() |
| [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop()) | | Called when the activity is no longer visible to the user, because another activity has been resumed and is covering this one. This may happen either because a new activity is being started, an existing one is being brought in front of this one, or this one is being destroyed.  Followed by either onRestart() if this activity is coming back to interact with the user, or onDestroy() if this activity is going away. | **Yes** | onRestart()or onDestroy() |
| [onDestroy()](http://developer.android.com/reference/android/app/Activity.html#onDestroy()) | | | The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called [finish()](http://developer.android.com/reference/android/app/Activity.html#finish()) on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the [isFinishing()](http://developer.android.com/reference/android/app/Activity.html" \l "isFinishing()) method. | **Yes** | *nothing* |