**1. Managing the application life cycle**

Android devices have limited resources, therefore the Android system is allowed to manage the available resources by terminating running processes or recycling Android components.

In addition to resource management, Android also recreates activities in case a configuration change occurs. The Configuration object contains the current device configuration, if this configuration changes activities are restarted, as they may use different resources for this configuration.

For the user of the device this should happen transparently, i.e. he should not note if an Android component have been terminated nor not.

To support this the Android platform supports lifecycle event which are called in the case of process or component termination as well as in case of a configuration change. The developer is responsible for maintaining the state of the application. He is also responsible for restore the activity *instance state*. The instance state of an activity is the nonpersistent data that needs to be passed between activities restarts during a configuration change to restore user selections.

**2. Application**

The application object is created whenever one of your Android components are started. It is started in a new process with a unique ID under a unique user. Even if you do not specify one in your AndroidManifest.xml file, the Android system creates a default object for you. This object provides the following main lifecycle methods:

* onCreate() - called before the first components of the application starts
* onLowMemory() - called when the Android system requests that the application cleans up memory
* onTerminate() - only for testing, not called in production
* onConfigurationChanged() - called whenever the configuration changes

The application object starts before any component and runs at least as long as another component of the application runs.

If the Android system needs to terminate processes it follows the following priority system.

**Table 1. Priorities**

| **Process status** | **Description** | **Priority** |
| --- | --- | --- |
| Foreground | An application in which the user is interacting with an activity, or which has an service which is bound to such an activity. Also if a service is executing one of its lifecycle methods or a broadcast receiver which runs its onReceive() method.  | 1 |
| Visible | User is not interacting with the activity, but the activity is still (partially) visible or the application has a service which is used by a inactive but visible activity.  | 2 |
| Service | Application with a running service which does not qualify for 1 or 2.  | 3 |
| Background | Application with only stopped activities and without a service or executing receiver. Android keeps them in a least recent used (LRU) list and if requires terminates the one which was least used.  | 4 |
| Empty | Application without any active components.  | 5 |

**3. Activity lifecycle**

The Android system is also allowed to recycle Android components to free up resources. This part explains which for activities, the lifecycle of other components is described in the respective part of these components.

An activity can be in different states which are described by the following table.

**Table 2. Activity state**

| **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.  |

The user should not notice if an activity which is still part of an activity stack has been terminate or not. For this the developer needs to store the state of the activity at the right point in time and restore it. He also should stop any unnecessary actions if the activity is not visible anymore to save system resources.

The Android system defines a lifecycle for activities via predefined (lifecycle) methods. The most important methods are:

**Table 3. Important Activity lifecycle methods**

| **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. This method is guaranteed to be called as of API 11.  |

The life cycle of an activity with its most important methods is displayed in the following diagram.



Android has more life cycle methods but not all of these methods are guaranteed to be called. The onDestroy() method is not guaranteed to be called, hence you typically do not use it. For more information on the other methods see [Activity lifecycle - Official documentation](http://developer.android.com/guide/components/activities.html).

**Tip**

You can safely ignore the existing of the other lifecycle methods. Applications are typically developed having other lifecycle methods.