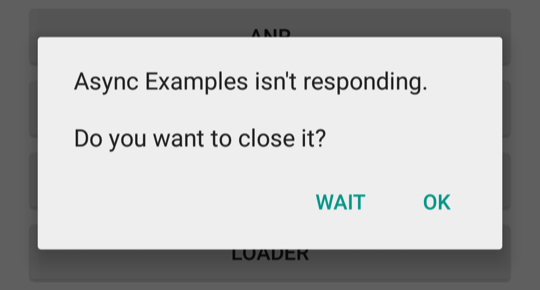
<https://developer.android.com/training/articles/perf-anr>



In any situation in which your app performs a potentially lengthy operation, **you should not perform the work on the UI thread**, but instead create a worker thread and do most of the work there. This keeps the UI thread (which drives the user interface event loop) running and prevents the system from concluding that your code has frozen.

In Android, application responsiveness is monitored by the Activity Manager and Window Manager system services. Android will display the ANR dialog for a particular application when it detects one of the following conditions:

* No response to an input event (such as key press or screen touch events) within 5 seconds.
* A [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) hasn't finished executing within 10 seconds.

<https://developer.android.com/reference/android/os/AsyncTask>

## **AsyncTask's generic types**

The three types used by an asynchronous task are the following:

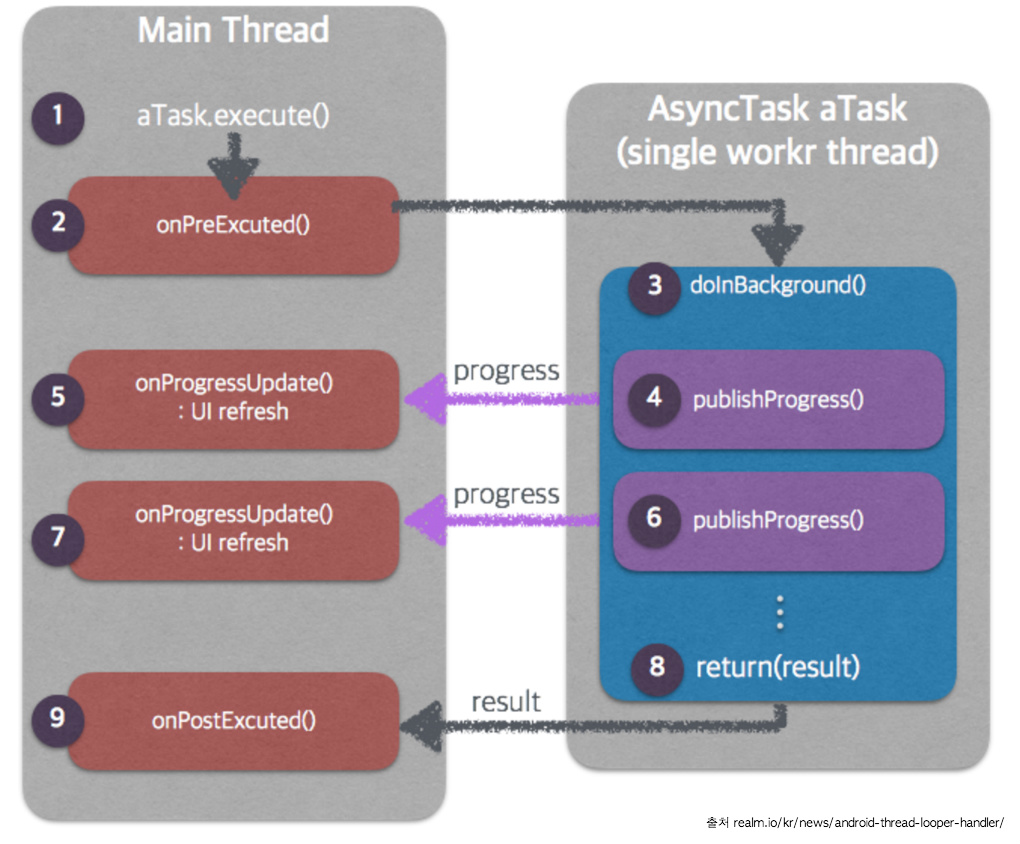
1. Params, the type of the parameters sent to the task upon execution.
2. Progress, the type of the progress units published during the background computation.
3. Result, the type of the result of the background computation.

Not all types are always used by an asynchronous task. To mark a type as unused, simply use the type [Void](https://developer.android.com/reference/java/lang/Void.html):

private class MyTask extends AsyncTask<Void, Void, Void> { ... }

## **The 4 steps**

<http://cfile23.uf.tistory.com/image/2420B240577D4A720F8136>



When an asynchronous task is executed, the task goes through 4 steps:

1. [onPreExecute()](https://developer.android.com/reference/android/os/AsyncTask.html#onPreExecute()), invoked on the UI thread before the task is executed. This step is normally used to setup the task, for instance by showing a progress bar in the user interface.
2. [doInBackground(Params...)](https://developer.android.com/reference/android/os/AsyncTask.html#doInBackground(Params...)), invoked on the background thread immediately after [onPreExecute()](https://developer.android.com/reference/android/os/AsyncTask.html#onPreExecute()) finishes executing. This step is used to perform background computation that can take a long time. The parameters of the asynchronous task are passed to this step. The result of the computation must be returned by this step and will be passed back to the last step. This step can also use [publishProgress(Progress...)](https://developer.android.com/reference/android/os/AsyncTask.html#publishProgress(Progress...)) to publish one or more units of progress. These values are published on the UI thread, in the [onProgressUpdate(Progress...)](https://developer.android.com/reference/android/os/AsyncTask.html#onProgressUpdate(Progress...)) step.
3. [onProgressUpdate(Progress...)](https://developer.android.com/reference/android/os/AsyncTask.html#onProgressUpdate(Progress...)), invoked on the UI thread after a call to [publishProgress(Progress...)](https://developer.android.com/reference/android/os/AsyncTask.html#publishProgress(Progress...)). The timing of the execution is undefined. This method is used to display any form of progress in the user interface while the background computation is still executing. For instance, it can be used to animate a progress bar or show logs in a text field.
4. [onPostExecute(Result)](https://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute(Result)), invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

<https://i.stack.imgur.com/abwYS.png>

