**Thread**

A thread is a concurrent unit of execution. It has its own call stack. There are two methods to implement threads in applications.

One is providing a new class that extends Thread and overriding its run() method. The other is providing a new Thread instance with a Runnable object during its creation. A thread can be executed by calling its "start" method. You can set the "Priority" of a thread by calling its "setPriority(int)" method.

A thread can be used if you have no affect in the UI part. For example, you are calling some web service or download some data, and after download, you are displaying it to your screen. Then you need to use a Handler with a Thread and this will make your application complicated to handle all the responses from Threads.

A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue. Each thread has each message queue. (Like a To do List), and the thread will take each message and process it until the message queue is empty. So, when the Handler communicates, it just gives a message to the caller thread and it will wait to process.

If you use Java threads then you need to handle the following requirements in your own code:

Synchronization with the main thread if you post back results to the user interface No default for canceling the thread No default thread pooling No default for handling configuration changes in Android

**AsyncTask**

AsyncTask enables proper and easy use of the UI thread. This class allows performing background operations and publishing results on the UI thread without having to manipulate threads and/or handlers. An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread.

AsyncTask will go through the following 4 stages:

**onPreExecute()**

Invoked on the UI thread before the task is executed

**doInbackground(Params..)**

Invoked on the background thread immediately after onPreExecute() finishes executing.

**onProgressUpdate(Progress..)**

Invoked on the UI thread after a call to publishProgress(Progress...).

**onPostExecute(Result)**

Invoked on the UI thread after the background computation finishes.