AsyncTask in Android :

#### Android UI Main Thread : Android handles input events/tasks with a single User Interface (UI) thread and the thread is called Main thread. Main thread cannot handle concurrent operations as it handles only one event/operation at a time.

#### Concurrent Processing in Android :

If input events or tasks are not handled concurrently, whole code of an Android application runs in the main thread and each line of code is executed one after each other.

Assume if you perform a long lasting operation, for example accessing resource (like MP3, JSON, Image) from the Internet, the application goes hung state until the corresponding operation is finished.

To bring good user experience in Android applications, all potentially slow running operations or tasks in an Android application should be made to run asynchronously.

**Examples for slow running tasks :**

1. Accessing resources (like MP3, JSON, Image) from Internet
2. Database operations
3. Webservice calls

**AsyncTask :**

AsyncTask is an abstract Android class which helps the Android applications to handle the Main UI thread in efficient way. AsyncTask class allows us to perform long lasting tasks/background operations and show the result on the UI thread without affecting the main thread.

**Methods present in AsyncTas :**

**onPreExecute:**

Invoked before the task is executed ideally before doInBackground method is called on the UI thread. This method is normally used to setup the task like showing progress bar in the UI.

**doInBackground:**

Code running for long lasting time should be put in doInBackground method. When execute method is called in UI main thread, this method is called with the parameters passed.

**onProgressUpdate:**

Invoked by calling publishProgress at anytime from doInBackground. This method can be used to display any form of progress in the user interface.

**onPostExecute:**

Invoked after background computation in doInBackground method completes processing. Result of the doInBackground is passed to this method.

Note : The task can be cancelled by invoking cancel(boolean) method. This will cause subsequent calls to isCancelled() to return true. After invoking this method, onCancelled(Object) method is called instead of onPostExecute() after doInBackground() returns.