AsyncTask

Threads -> It just series of instructions that execute in a particular linear sequence.

UI thread -> All the things happening on the screen needs to happen on UI thread. App in running state handles the UI thread. At a time there can only be one app managing UI thread. Only UI thread can change things on the screen.

AsyncTask is made using Threads and handlers. In Asynctask we do not need to manage threads. Like we dont have to stop the threads or start new threads. All of this is handled by AsyncTask for us. It is like a wrapper class for threads and handler.

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the java.util.concurrent package such as Executor, ThreadPoolExecutor and FutureTask.

To use AsyncTask, we create a class that extends from asynctask. We can call execute on the object of the class to start the work in different thread.

An AsyncTask streamlines the following common background process:

1. **Pre** – Execute code on the UI thread before starting a task (e.g show ProgressBar)
2. **Backgroud** – Run a background task on a thread given certain inputs (e.g fetch data)

3. **Updates** – Display progress updates during the task (optional)
4. **Post** – Execute code on UI thread following completion of the background task (e.g show data). This take in the argument, work done by the background thread.

In java code AsyncTask can be used as follows –

```
  private class AsyncTaskDemo extends AsyncTask<String,
Void, String> {
      protected void onPreExecute() {
            // Runs on the UI thread before
doInBackground
            // Good for toggling visibility of a
progress indicator
            Log.e("AsyncTaskDemo", "onPreExecute");
        }
        protected String doInBackground(String...
strings) {
            // Some long-running task in backgorund

            long startTime = System.currentTimeMillis();

            //This task will not block UI thread as it
is being done in background thread
            while (System.currentTimeMillis() -
startTime<5000);

            Log.e("AsyncTaskDemo", "doInBackground says
" + strings[0]);
            return "Hi";
        }
        protected void onPostExecute(String result) {
            // This method is executed in the UIThread
            // with access to the result.
            Log.e("AsyncTaskDemo", "onPostExecute: " +
result);

        }
```

```
    }
```

Now to call this in Activity, we have to simply makes it object and call execute on it.

```
AsyncTaskDemo asyncTaskDemo = new AsyncTaskDemo();
asyncTaskDemo.execute("Hello");
```