AsyncTask

It is a light-weight asynchronous class (Params, Progress, Result).

- Threadpool: SerialExecutor + THREAD_POOL_EXECUTOR
  - Default thread pool size is 5, cache is 10
- 1 Handler(static InternalHandler requires to be loaded in MainThread)
- Abstract Class
  - MainThread: `onPreExecute()`
  - ThreadPool: `doInBackground(Params... params)`
  - MainThread: publishProgress -> `onProgressUpdate()`
  - MainThread: `onProgressUpdate(Progress... values)`
  - MainThread: `onPostExecute(Result result)`
  - MainThread: `onCancelled()`

Cancell

- Not actual cancell the task, but setting status to cancel, we can check status in `doInBackground()`. Just like `interrupt()` a Thread, we can check in the thread to end itself.
- When `cancel()`, `onCancelled()` called, and `onPostExecute()` won't be call.

Advantage

- Easy update UI, instead of passing value to MainThread Handler.

Disadvantage

- AsyncTask doesn't bind with any component's lifecycle (eg: `cancel(boolean)` when Activity/Fragment destory.
- AsyncTask has to be created in MainThread.
- `AsyncTask.execute()` has to be invoked in MainThread, and only call once.
- Memory leak
  - If a AsyncTask is a non-static inner class within Activity， then it will keep a reference of creator Activity. When the Activity destoryed before the Asynctack completed, the Activity reference will be kept in the memory. As a result, creator Activity won't be recycled by GC, and it leads to a memory leak.
- Not support stream data
- Deep callback hell

When Activity OnConfiguration changed,

- Solution 1
  - Post a `AsyncTaskResultEvent(result)` by EventBus