Async Task

Responsiveness & threads

app should be responsive. It should atleast "feel" fast.

Strategies used

1) Show progress Indicators
2) prefetch the data
3) Design screens to fit data.
4) Limit size of network responses.
5) Smooth Animations. - avoid garbage collection

"Don't block the main thread".

Main thread

- single thread where UI gets drawn.
- Activity callbacks, UI drawings are handled on main thread.

Framework calls certain methods of your app based on central message loop

Examples of main-thread events:

1) onCreate, onResume calls.
2) when UI views are drawn.
3) when user touch-event is handled.
4) Callbacks from networking and other services.

Message Queue

while loop to respond to events.

```
while:
    mainThread
        .handleNextEvent().
```

framework
FW schedules or reschedule the Event Queue.

- Respond to Touches
- Respond to Systems
- Update UI
- Resume Activity

Responsiveness

more code on main thread, less time to update UI. ∴ response slows down.

Handler

- allows you to send and process Message & Runnable objects associated with thread's message queue.

2 main uses of handler.

1) to schedule messages & runnables to be executed.
2) to enque an action to be performed on different thread than your own.

Responsiveness

1) on main thread do as little, as fast, work, as possible.
2) Move longer tasks to background threads.
3) Main thread highest priority.

Thread

- should have resource control.
- should have object name as Thread.stop could be dangerous & deprecated..

AsyncTask. — it is a subclass.
Not recommended for long-running threads.

methods in class.

1) onPreExecute
must implement 2) doInBackground (Params...)
3) onProgressUpdate (T update)
4) onPostExecute(R result).

implemented usually as private subclass.

extends AsyncTask.

private class DownloadImage <URL, Integer, Bitmap>.
template parameter. progress Result.

• execute to handle the triggering.