Thread, Handler, AsyncTask

Thread

- Act much like usual Java Threads
- Can’t act directly on external User Interface objects (throw the Exception CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views”)
- Can’t be stopped by executing destroy() nor stop(). Use instead interrupt() or join() (by case)

Thread

- Two main ways of having a Thread execute application code:
  - Providing a new class that extends Thread and overriding its run() method.
  - Providing a new Thread instance with a Runnable object during its creation. In both cases, the start() method must be called to actually execute the new Thread.

```
class MyThread extends Thread {
    public void run() {
        // Code to execute in the new thread
    }
}

MyThread thread = new MyThread();
thread.start();
```

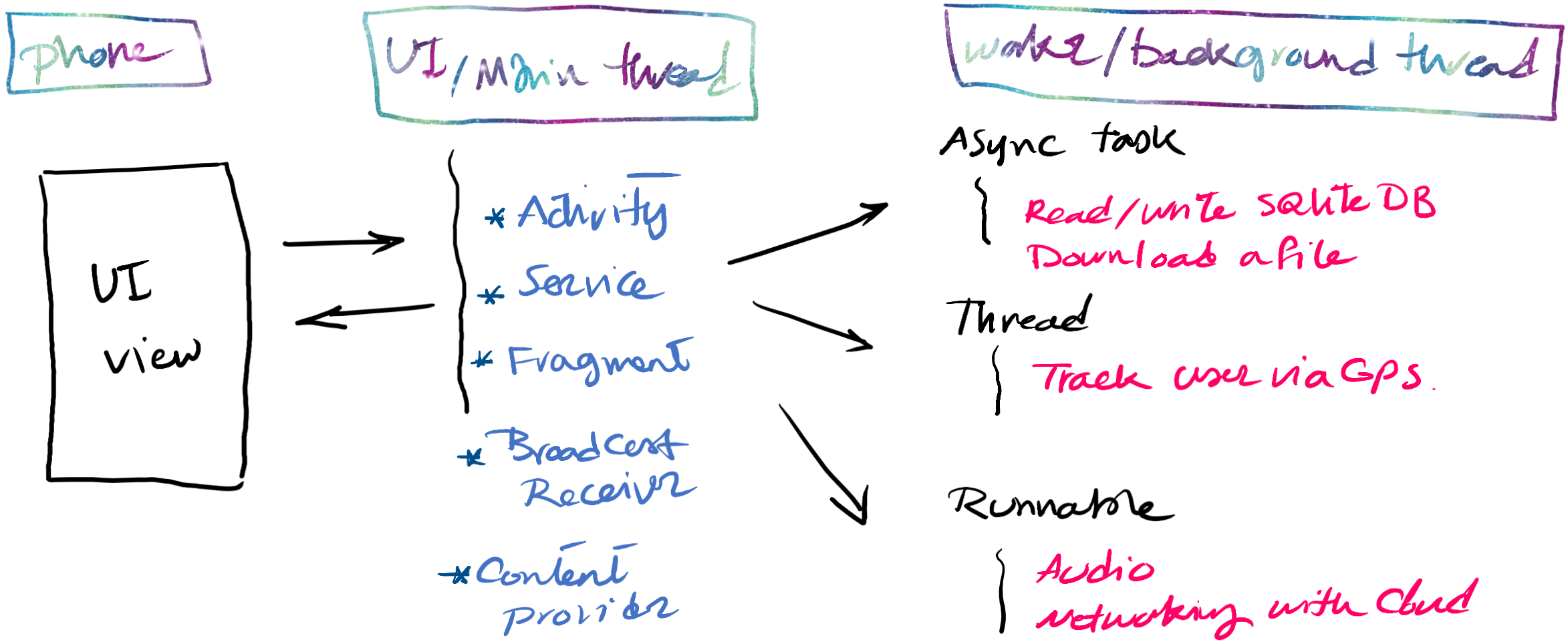
```
class MyRunnable implements Runnable {
    public void run() {
        // Code to execute in the new thread
    }
}

Thread thread = new Thread(new MyRunnable());
thread.start();
```

Handler

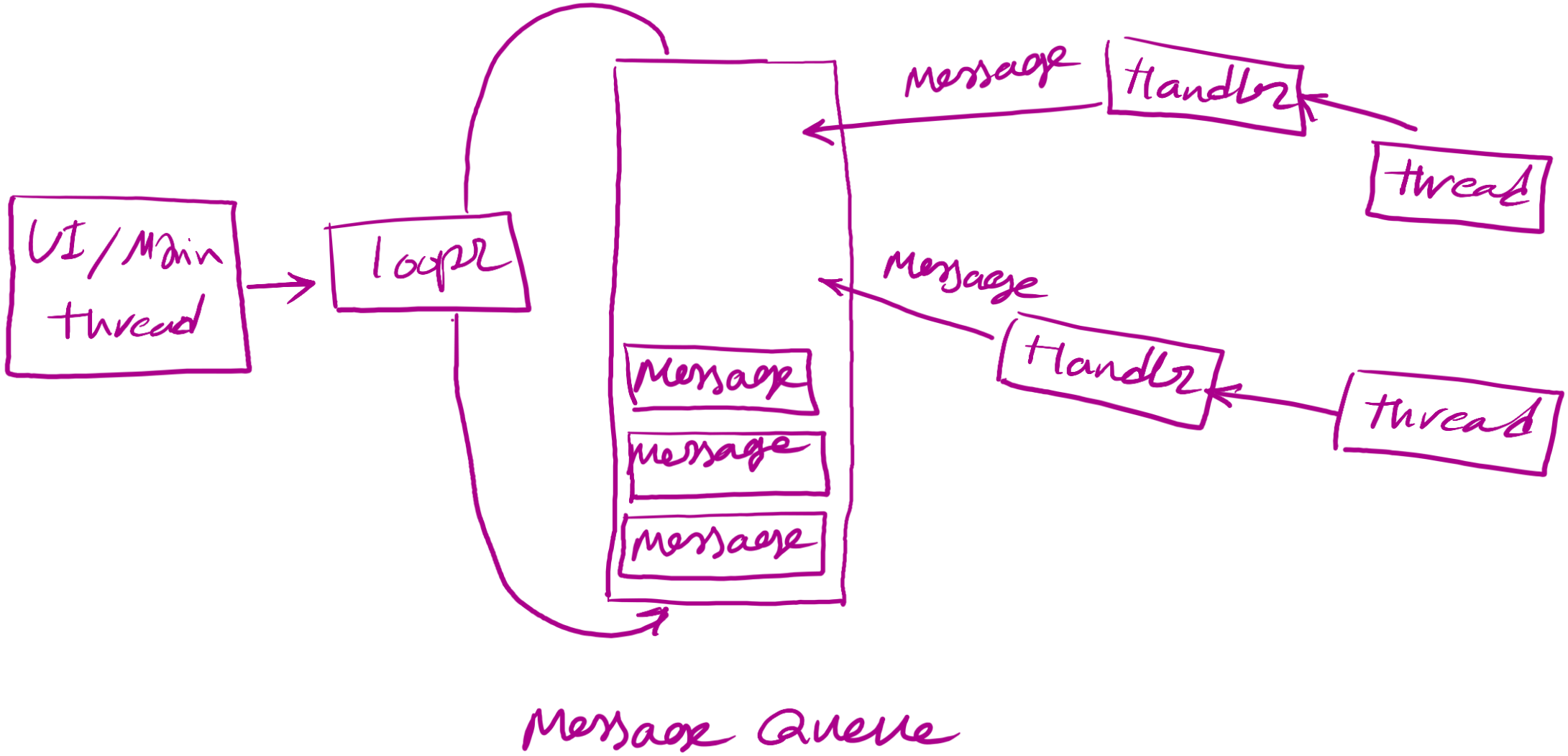
- Associated with a single thread and that thread's message queue
- Bound to the thread / message queue of the thread that is creating it
- Deliver messages and runnables to that message queue
- Execute them as they come out of the message queue

Handler

- Two main uses for a Handler:
  - To schedule messages and runnables to be executed as some point in the future
  - To add an action into a queue performed on a different thread



☑ Don't run everything on the UI thread


UI / Main thread
Looper
Message
message
message
Message
Handler
thread
Message
Handler
thread
Message Queue

AsyncTask

- Created on the UI thread and can be executed only once
- Run on a background thread and result is published on the UI thread
- The three types used by an asynchronous task are the following
  - Params, the type of the parameters sent to the task upon execution
  - Progress, the type of the progress units published during the background computation
  - Result, the type of the result of the background computation

AsyncTask

- Go through 4 steps:
  - onPreExecute(): invoked on the UI thread immediately after the task is executed
  - doInBackground(Param ...): 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