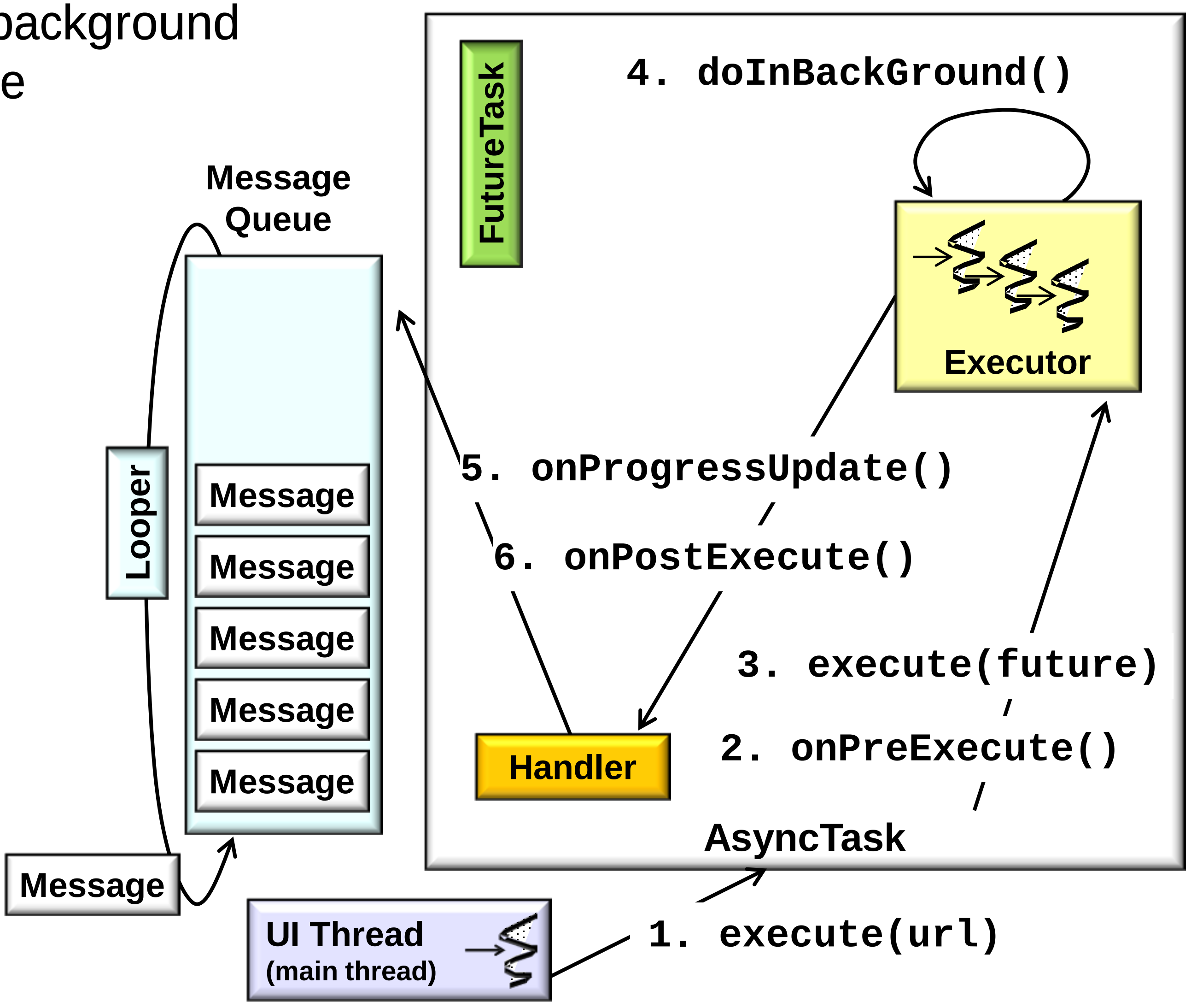
AsyncTask Usage Considerations

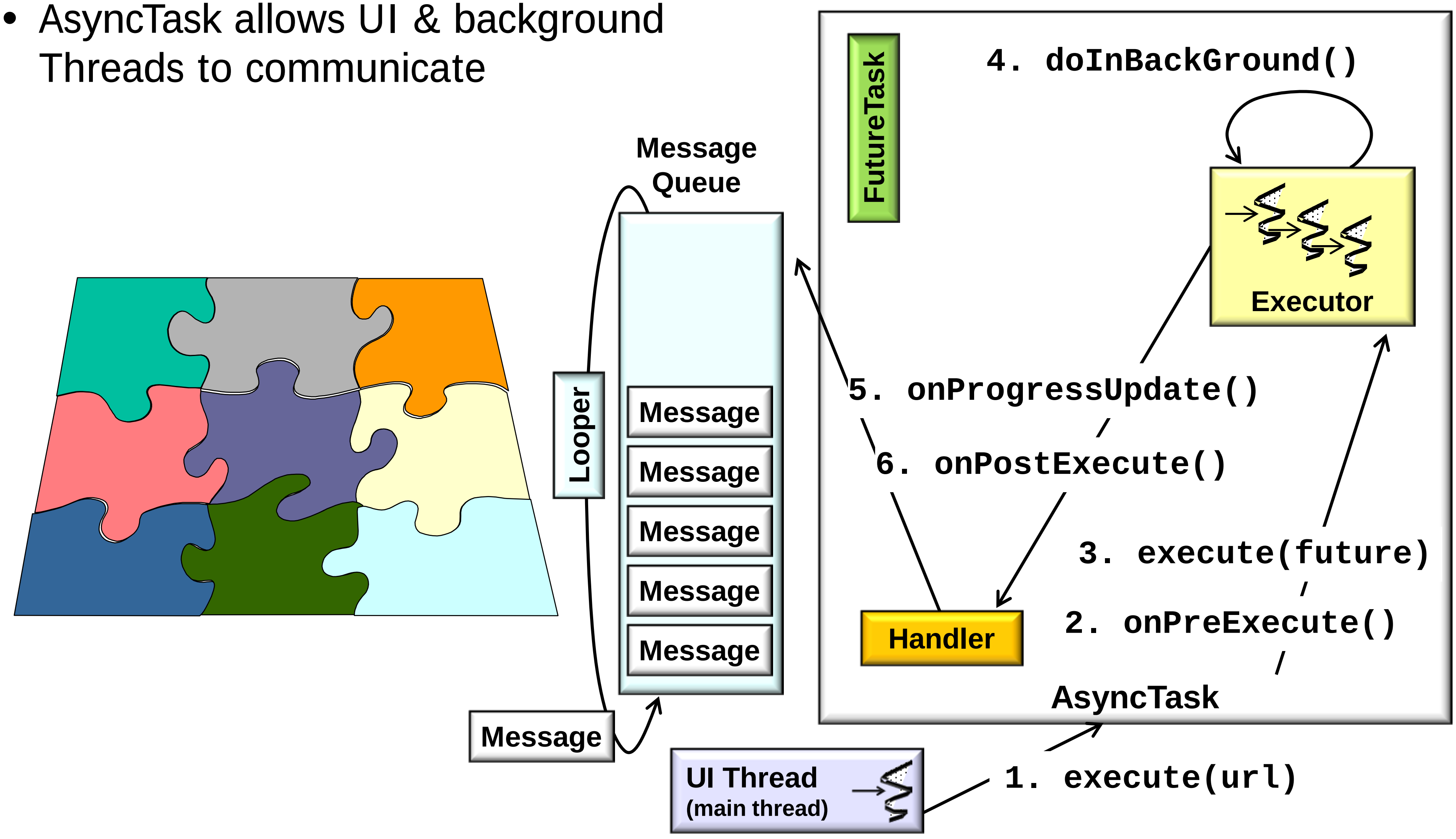
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate



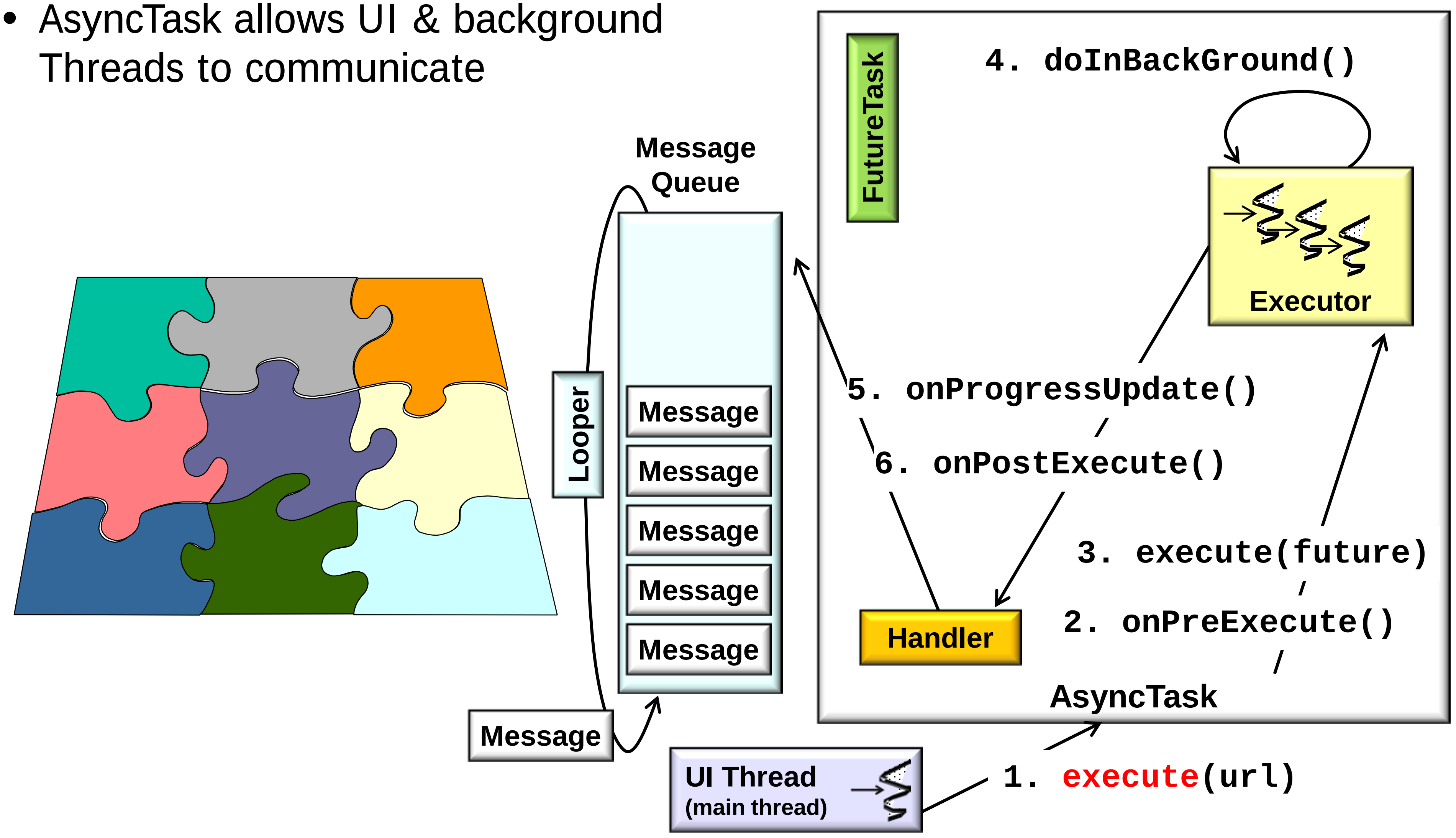
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate



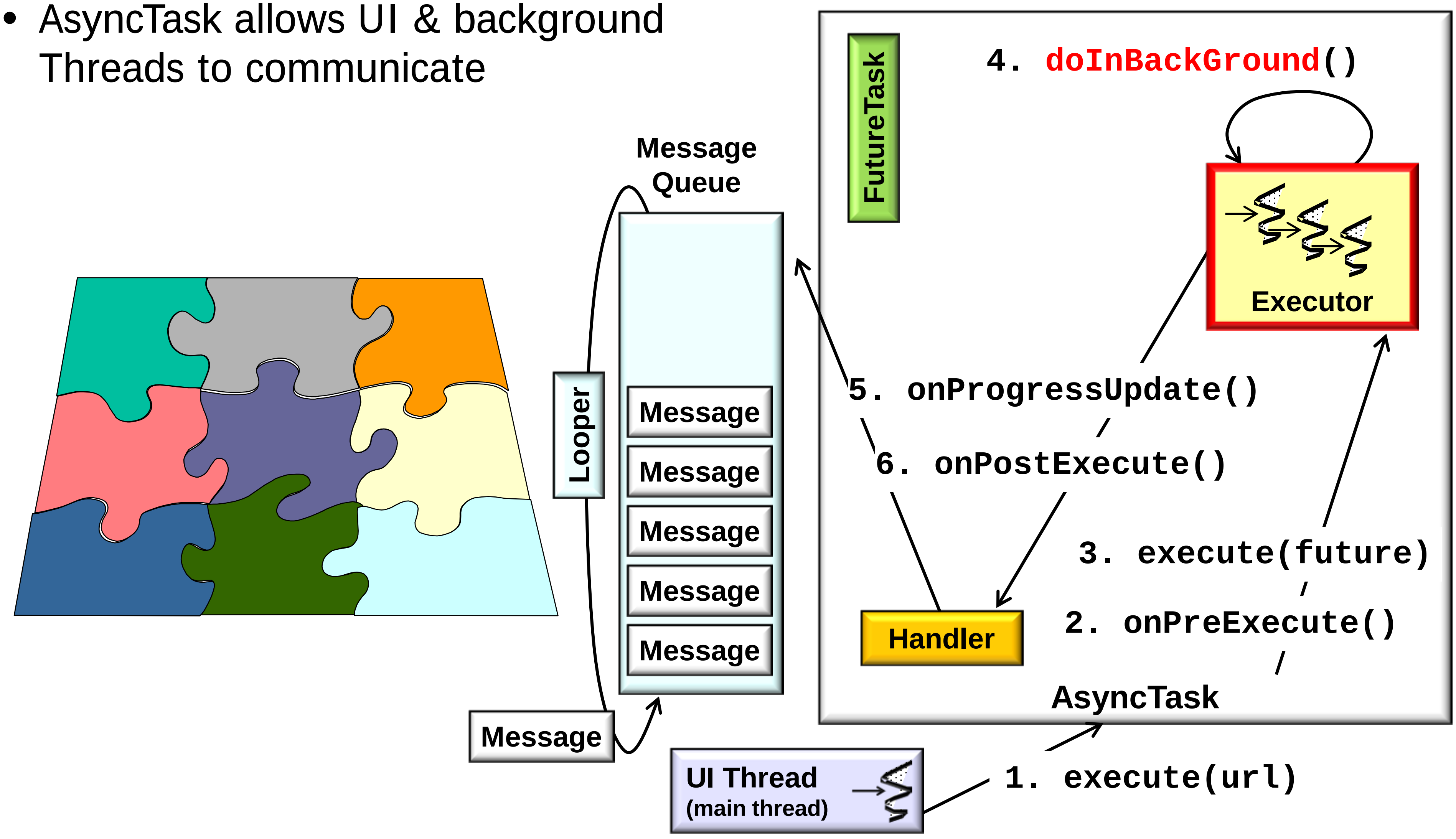
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate



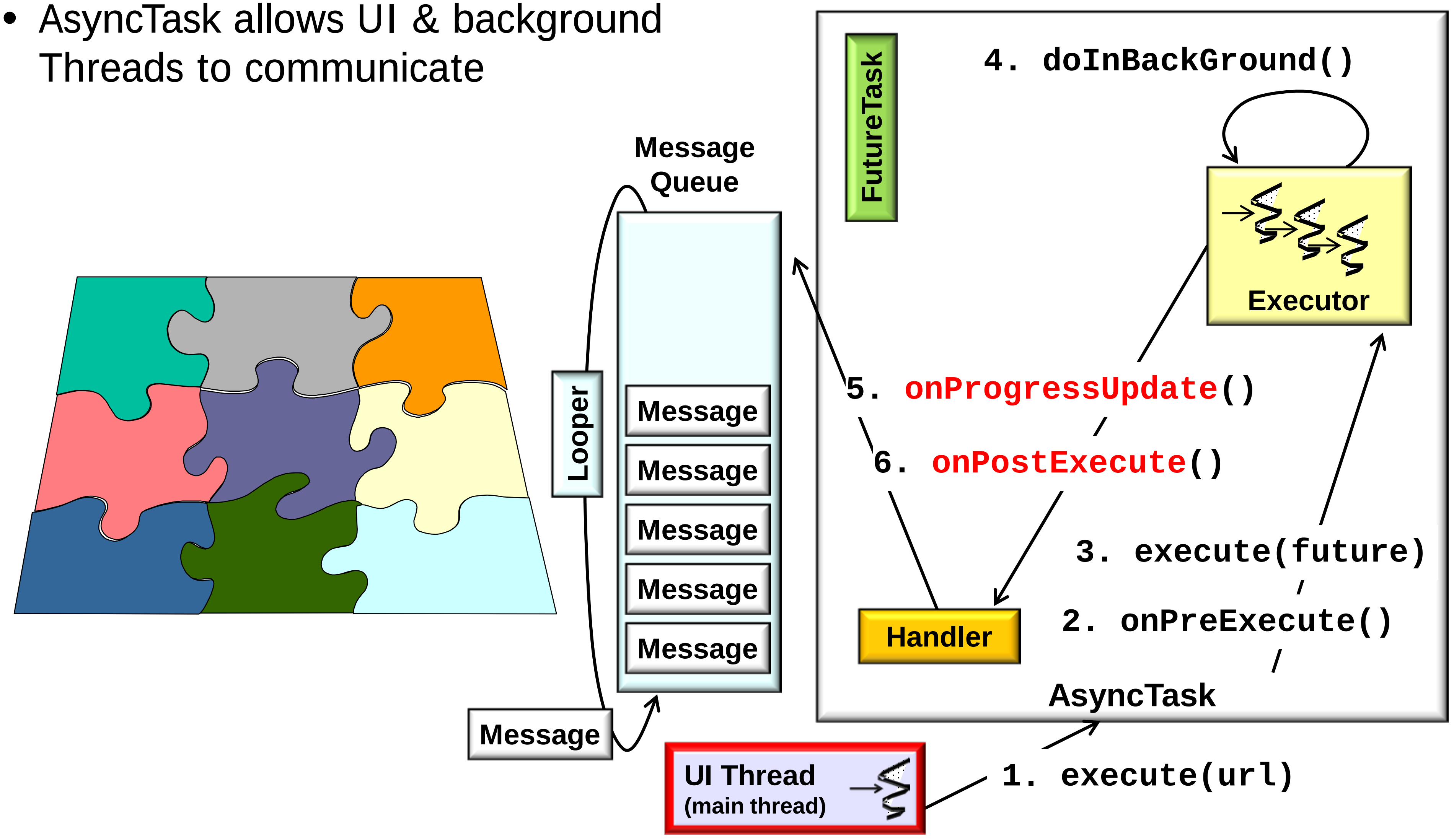
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate



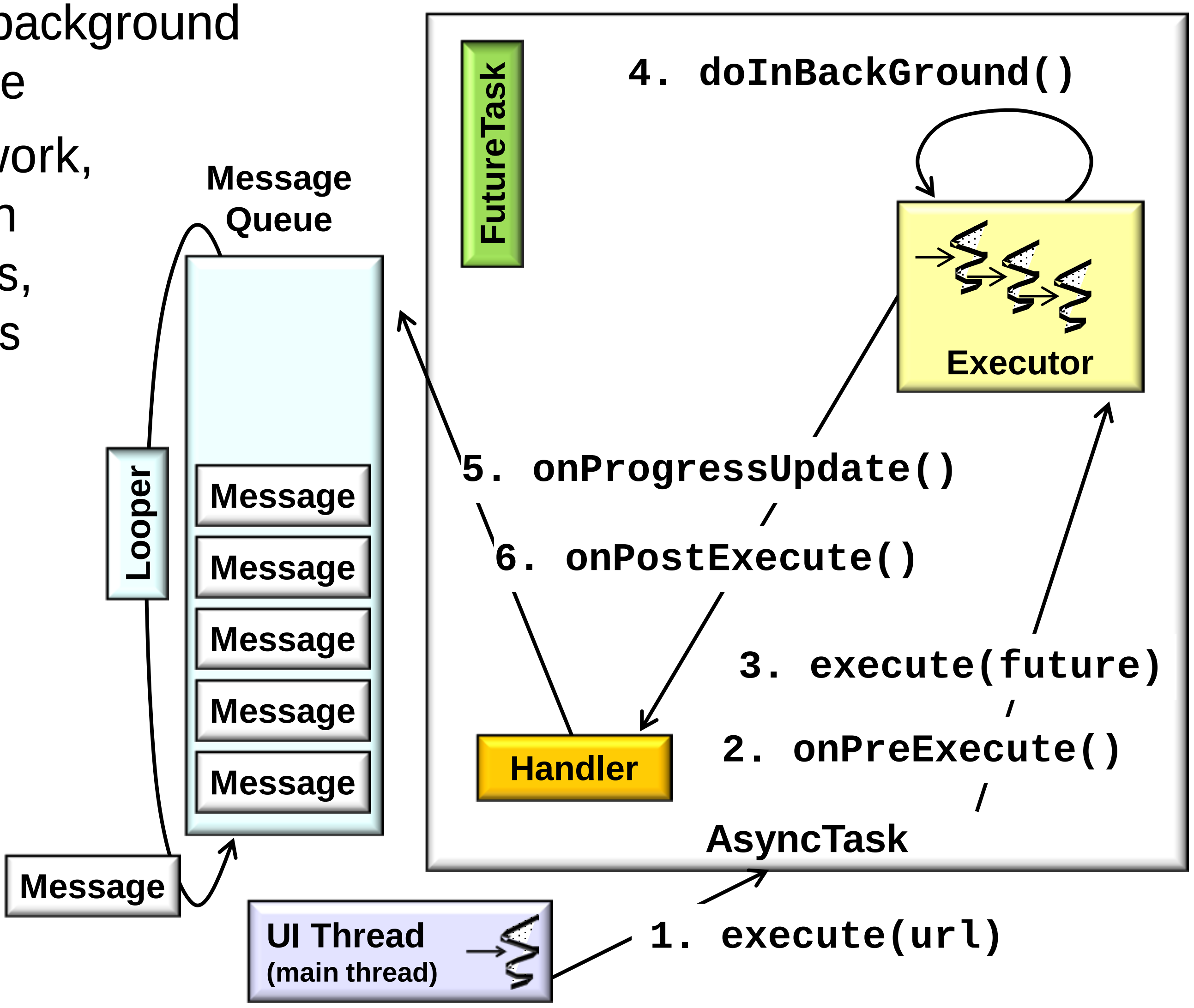
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate



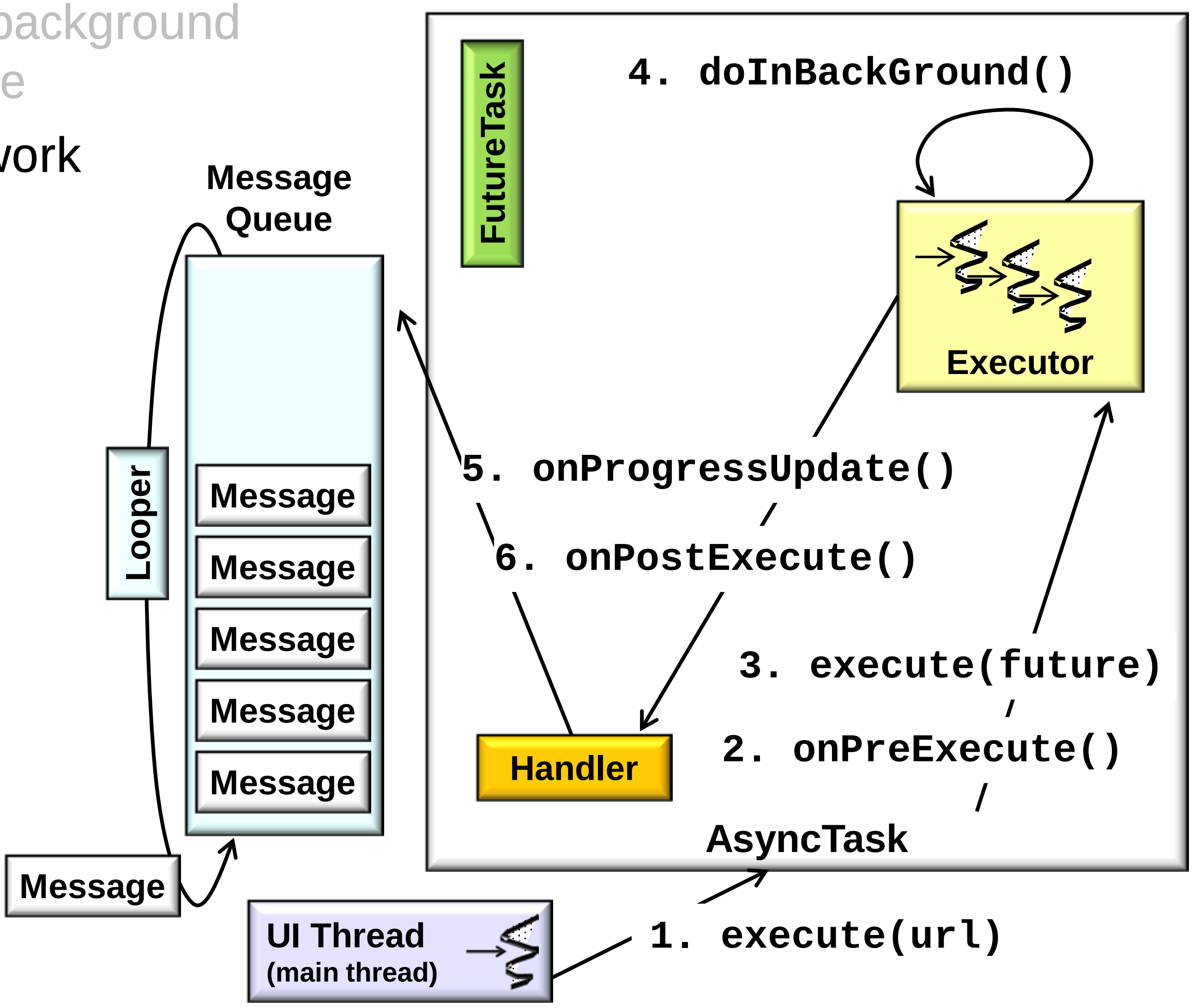
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
  - Unlike HaMeR framework, no direct manipulation of Handlers, Messages, Runnables, or Threads



AsyncTask Usage Considerations

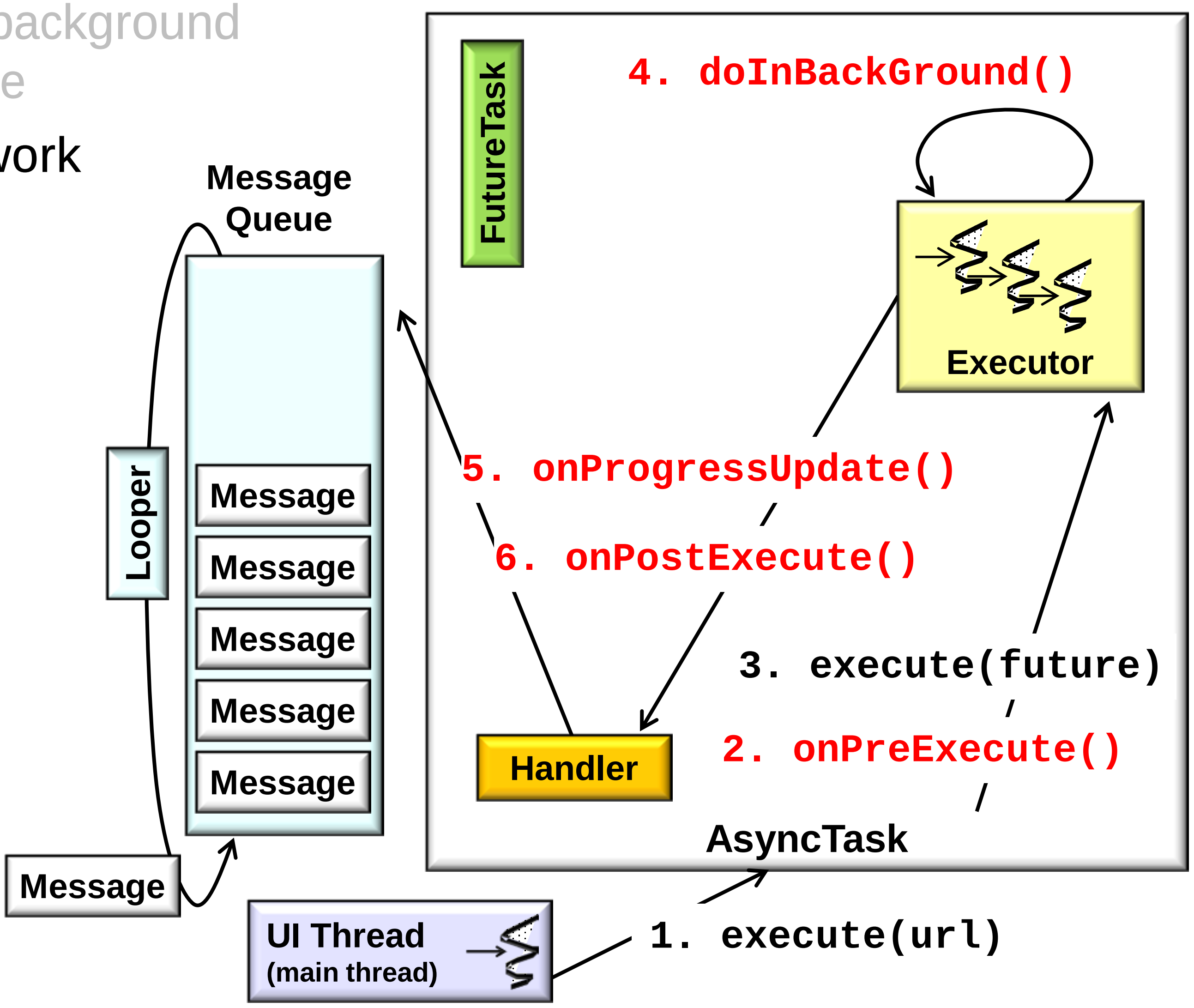
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics



See earlier discussions on "Android of Concurrency Patterns & Frameworks"

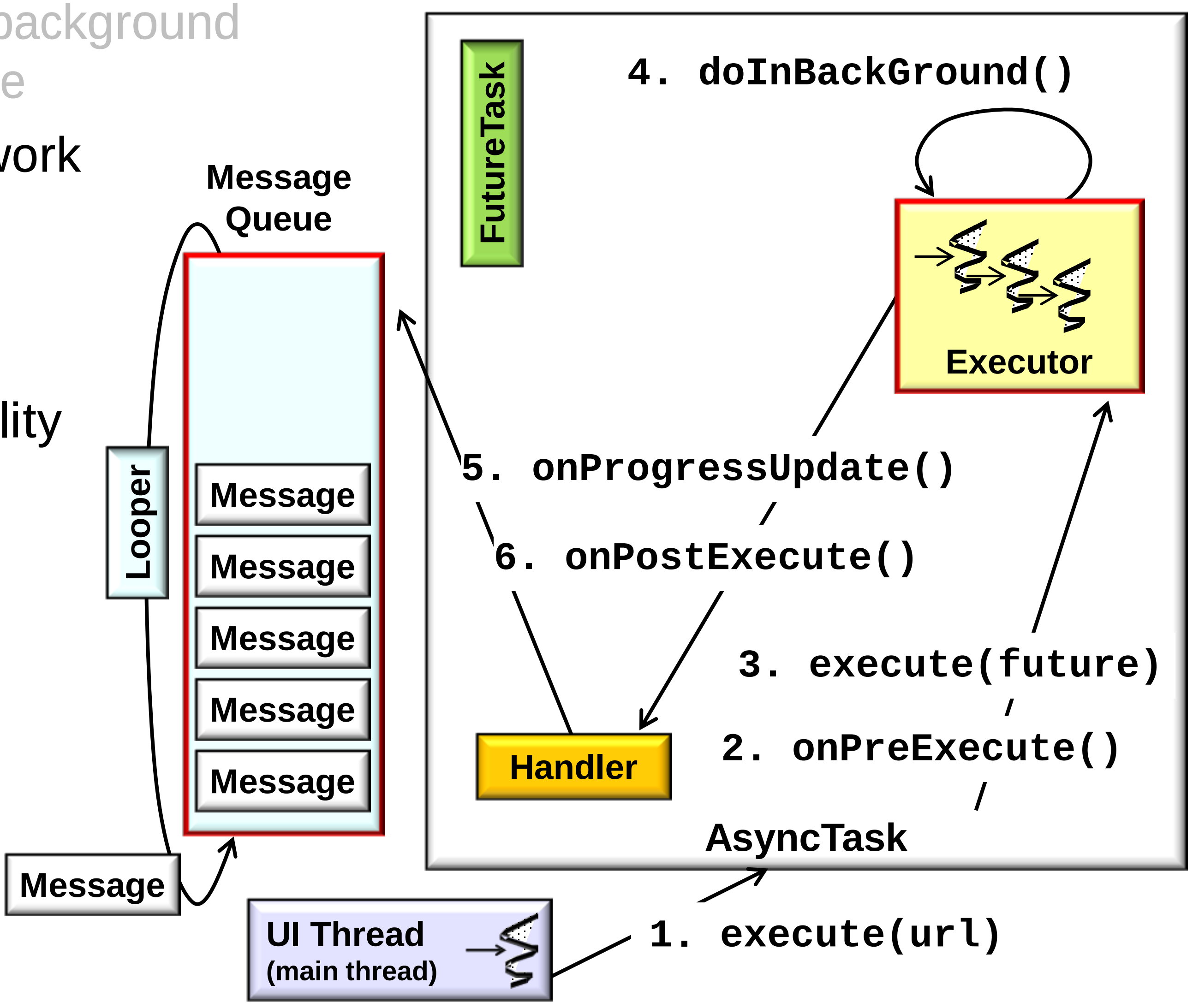
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics, e.g.
  - Inversion of control



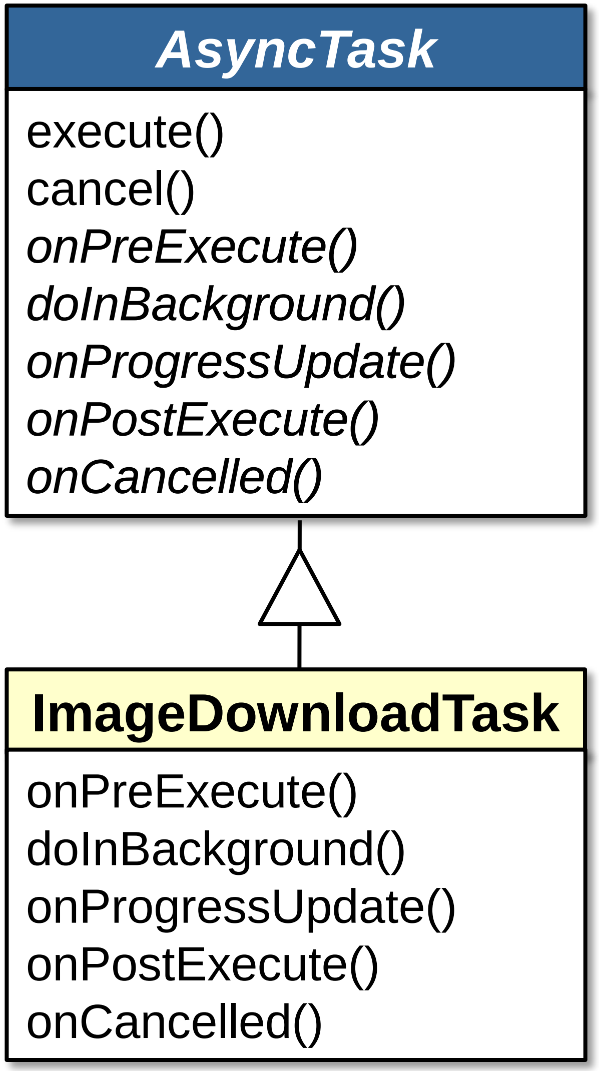
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics, e.g.
  - Inversion of control
  - Domain-specific structure & functionality



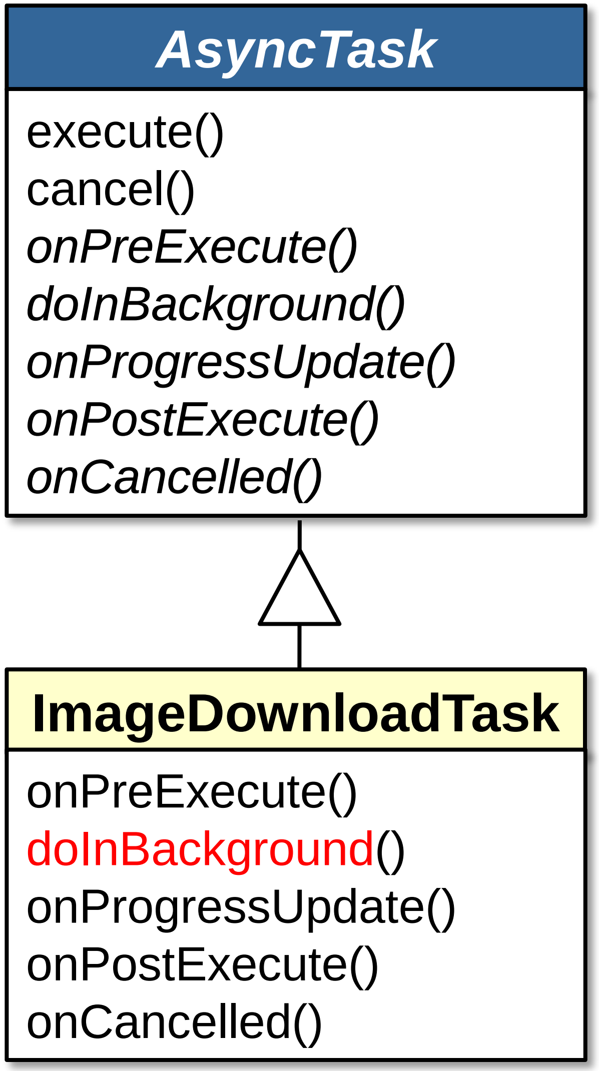
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics, e.g.
  - Inversion of control
  - Domain-specific structure & functionality
  - Semi-complete portions of applications



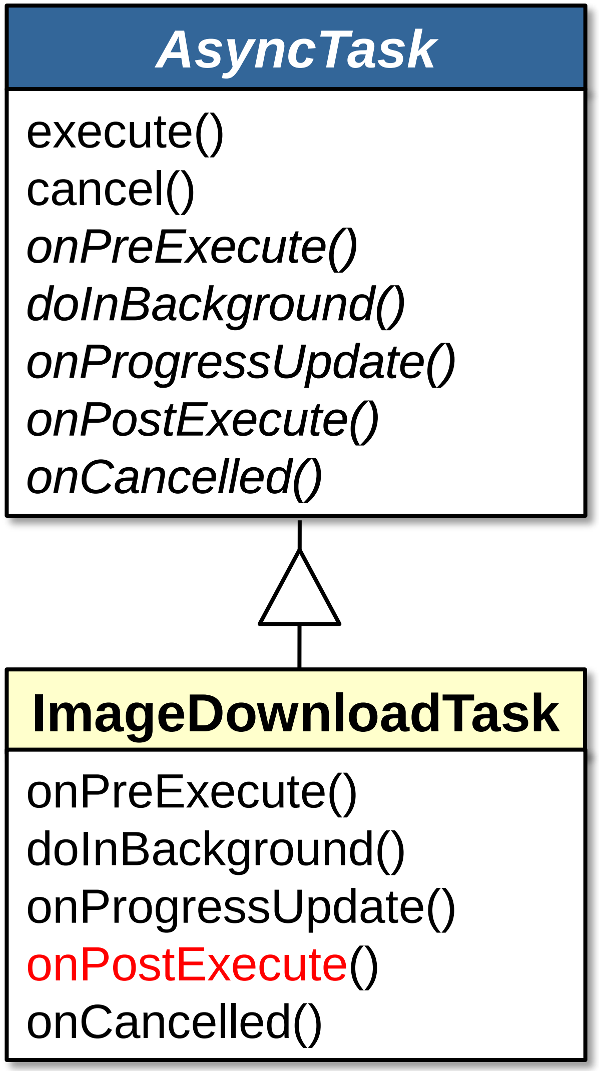
AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics, e.g.
  - Inversion of control
  - Domain-specific structure & functionality
  - Semi-complete portions of apps



AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics, e.g.
    - Inversion of control
    - Domain-specific structure & functionality
    - Semi-complete portions of apps



AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls

See bon-app-etit.blogspot.com/2013/04/the-dark-side-of-asynctask.html

AsyncTask Usage Considerations

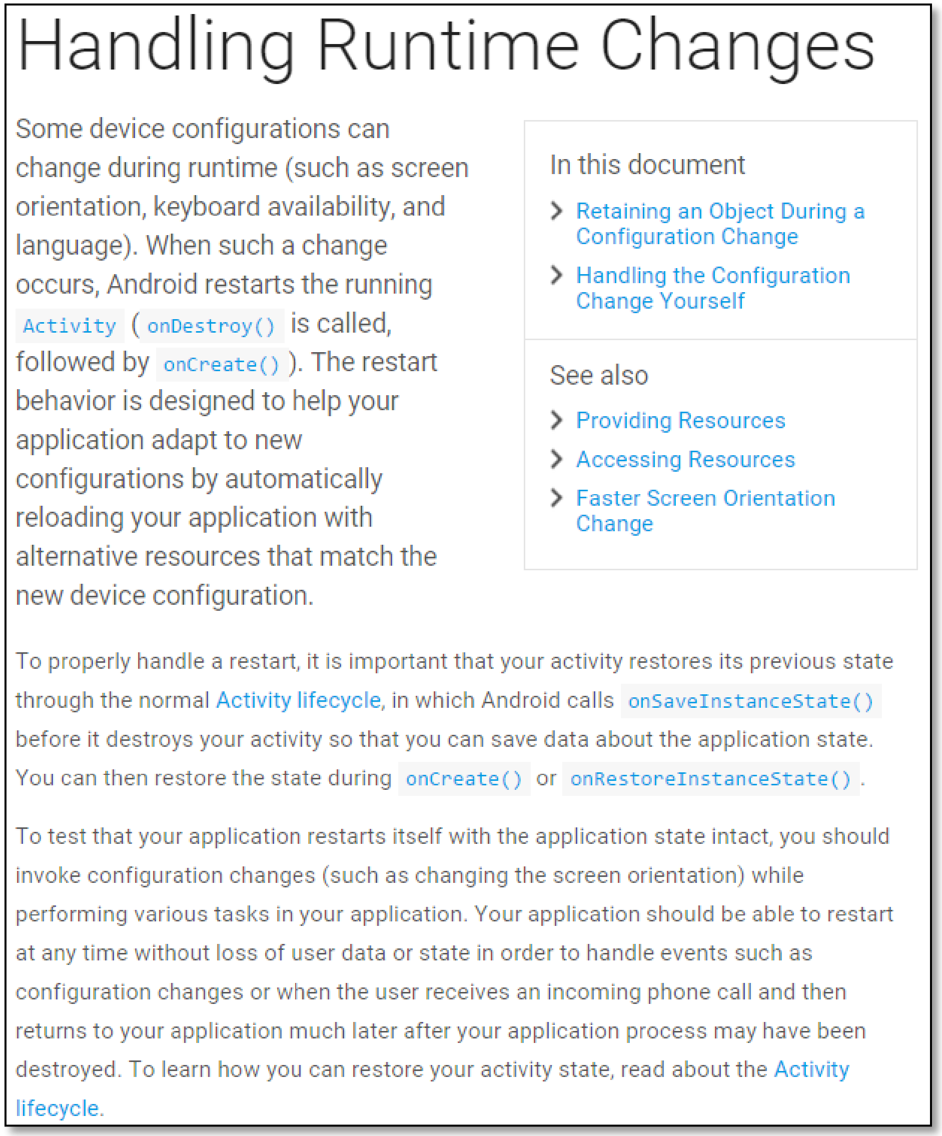
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
    - Cancellation is voluntary, just like Thread.interrupt()

AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
  - Dependency on Activity
    - Memory leaks occur if there’s a strong references to enclosing Activity

AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
  - Dependency on Activity
  - Losing results if/when runtime configurations change
    - e.g., Activity associated with an AsyncTask may be destroyed



Handling Runtime Changes

Some device configurations can change during runtime (such as screen orientation, keyboard availability, and language). When such a change occurs, Android restarts the running `Activity` (`onDestroy()` is called, followed by `onCreate()`). The restart behavior is designed to help your application adapt to new configurations by automatically reloading your application with alternative resources that match the new device configuration.

In this document

- Retaining an Object During a Configuration Change
- Handling the Configuration Change Yourself

See also

- Providing Resources
- Accessing Resources
- Faster Screen Orientation Change

To properly handle a restart, it is important that your activity restores its previous state through the normal Activity lifecycle, in which Android calls `onSaveInstanceState()` before it destroys your activity so that you can save data about the application state. You can then restore the state during `onCreate()` or `onRestoreInstanceState()`.

To test that your application restarts itself with the application state intact, you should invoke configuration changes (such as changing the screen orientation) while performing various tasks in your application. Your application should be able to restart at any time without loss of user data or state in order to handle events such as configuration changes or when the user receives an incoming phone call and then returns to your application much later after your application process may have been destroyed. To learn how you can restore your activity state, read about the Activity lifecycle.

AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
  - Dependency on Activity
  - Losing results if/when runtime configurations change
  - Portability
    - Concurrency semantics of AsyncTask execute() have changed over time

Before API 1.6 (Donut):

- In the first version of AsyncTask, the tasks were executed serially, so a task won't start before a previous task is finished. This caused quite some performance problems. One task had to wait on another one to finish.

API 1.6 to API 2.3 (Gingerbread):

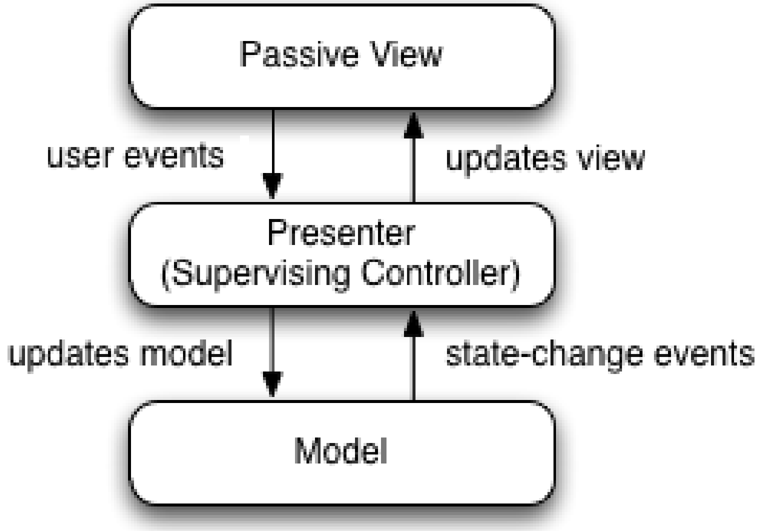
- The Android developers team decided to change this so that AsyncTasks could run parallel on a separate worker thread. There was one problem. Many developers relied on the sequential behavior and suddenly they were having a lot of concurrency issues.

API 3.0 (Honeycomb) until now

- "Hmmm, developers don't seem to get it? Let's just switch it back." The AsyncTasks where executed serially again. However, they can run parallel via executeOnExecutor(Executor).

AsyncTask Usage Considerations

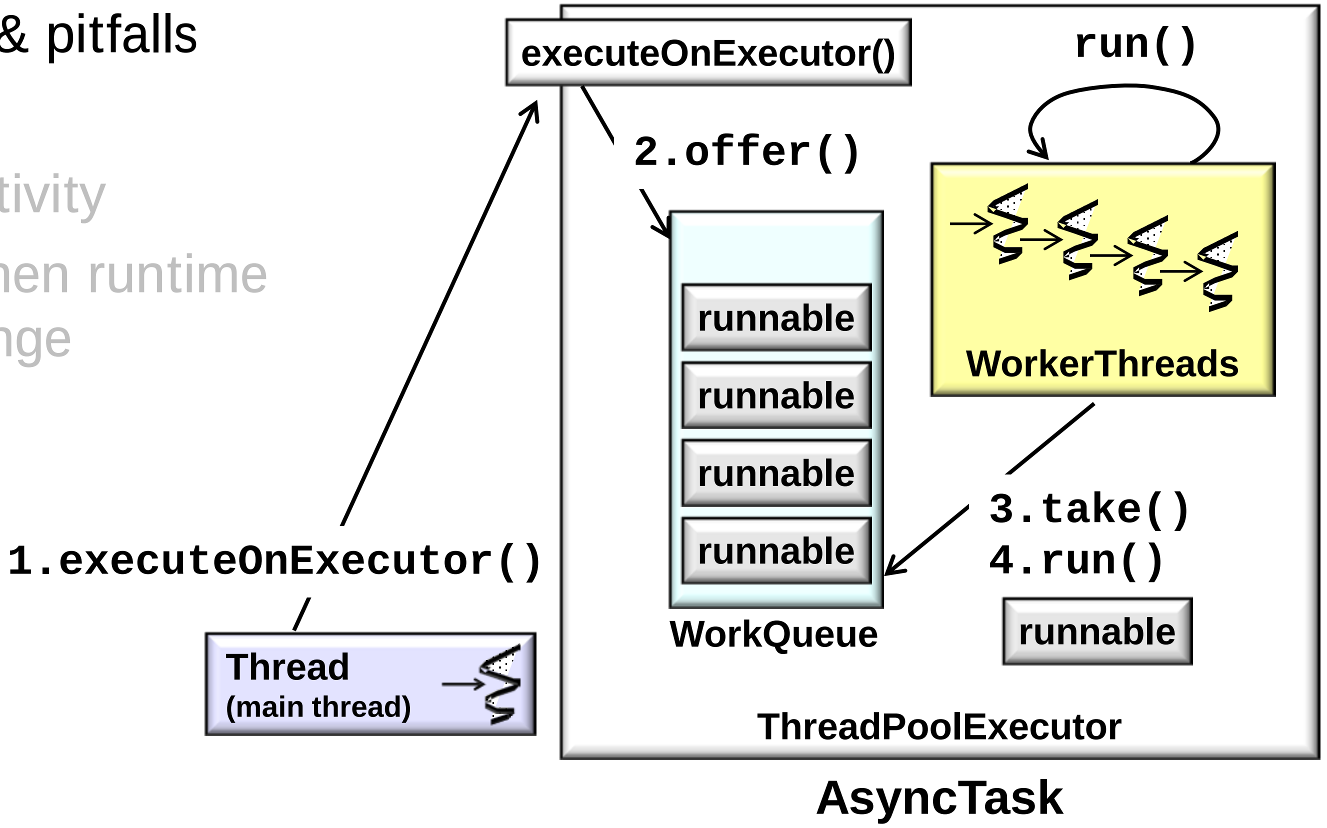
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
  - Dependency on Activity
  - Losing results if/when runtime configurations change
  - Portability



The *Model-View Presenter* (MVP) pattern addresses some of these issues

AsyncTask Usage Considerations

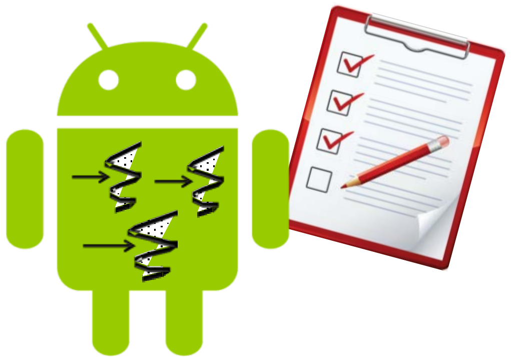
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
  - Cancellation
  - Dependency on Activity
  - Losing results if/when runtime configurations change
  - Portability



Other issues can be addressed by understanding Android patterns & APIs

AsyncTask Usage Considerations

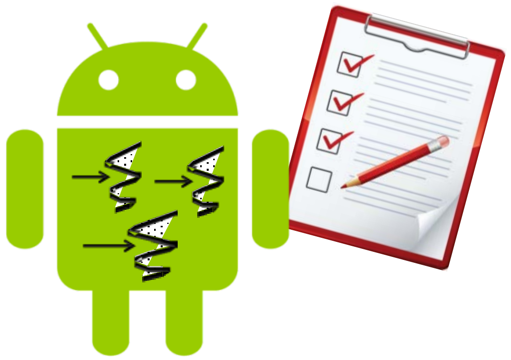
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
- AsyncTask used throughout Android



```
frameworks/base/core/java/android/content/AsyncTaskLoader.java
frameworks/base/core/java/android/content/CursorLoader.java
frameworks/base/core/java/android/os/AsyncTask.java
packages/apps/Browser/src/com/android/browser/UrlHandler.java
packages/apps/Calendar/src/com/android/calendar/CalendarController.java
packages/apps/Gallery/src/com/android/camera/ReverseGeocoderTask.java
packages/apps/Nfc/src/com/android/nfc/NfcService.java
packages/apps/Mms/src/com/android/mms/transaction/PushReceiver.java
packages/apps/Phone/src/com/android/phone/CallLogAsync.java
packages/apps/VideoEditor/src/com/android/videoeditor/BaseAdapterWithImages.java
...
```

AsyncTask Usage Considerations

- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics
- AsyncTask has traps & pitfalls
- AsyncTask used throughout Android
- onProgressUpdate() is not widely used



frameworks/base/media/java/android/media/videoeditor/MediaArtistNativeHelper.java
frameworks/base/packages/SystemUI/src/com/android/systemui/recent/
    RecentTasksLoader.java
packages/apps/Email/emailcommon/src/com/android/emailcommon/utility/
    EmailAsyncTask.java
packages/apps/Email/src/com/android/email/activity/setup/
    AccountCheckSettingsFragment.java
packages/apps/Gallery2/src/com/android/gallery3d/app/ManageCachePage.java
packages/apps/Gallery2/src/com/android/gallery3d/ui/ImportCompleteListener.java
packages/apps/Gallery2/src/com/android/gallery3d/ui/MenuExecutor.java
packages/apps/Settings/src/com/android/settings/TrustedCredentialsSettings.java