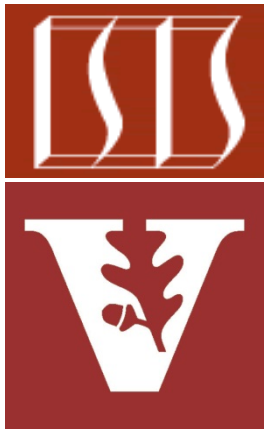
Overview of the AsyncTask Framework (Part 1)

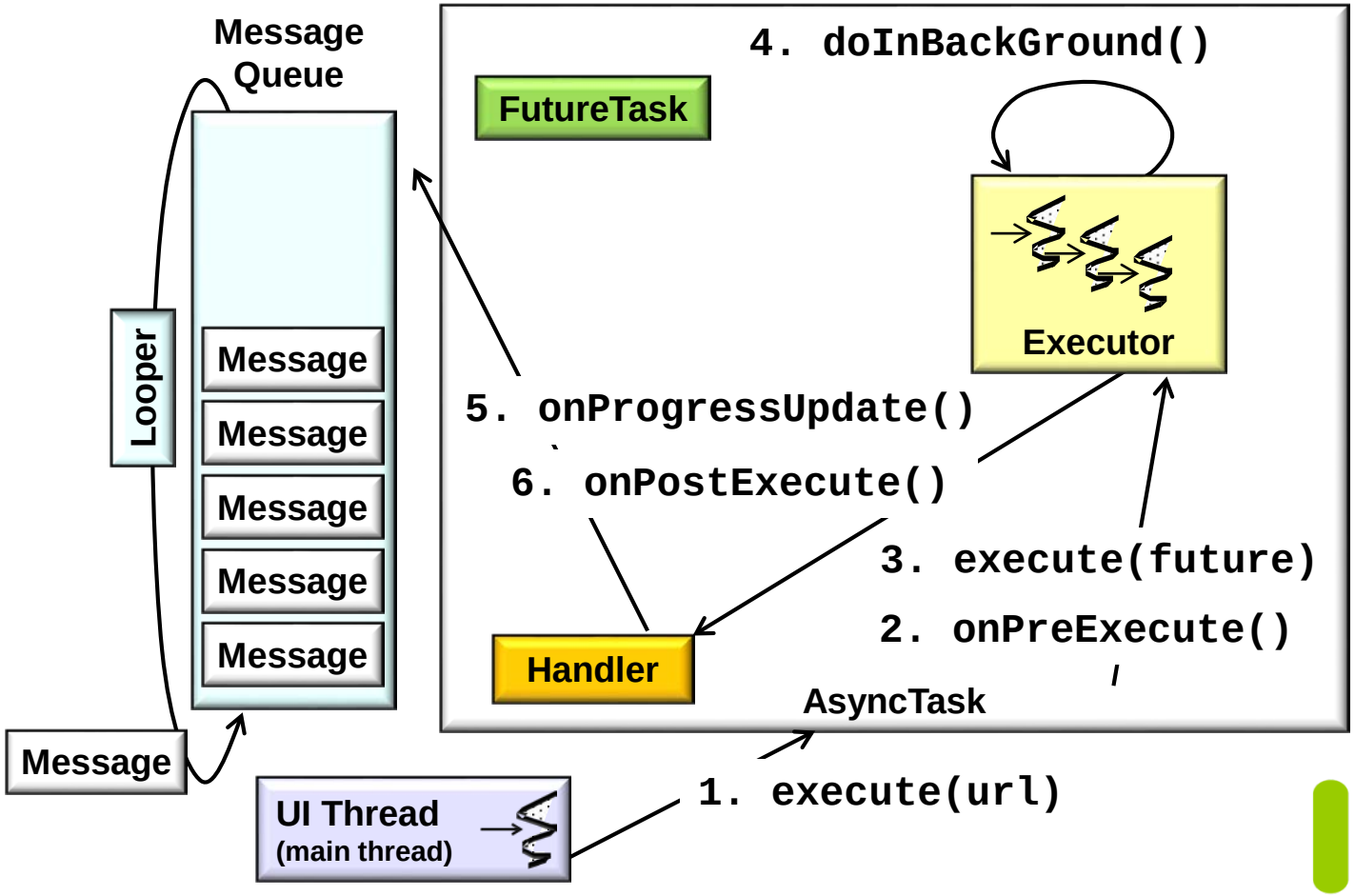


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Recognize the capabilities provided by the Android AsyncTask framework

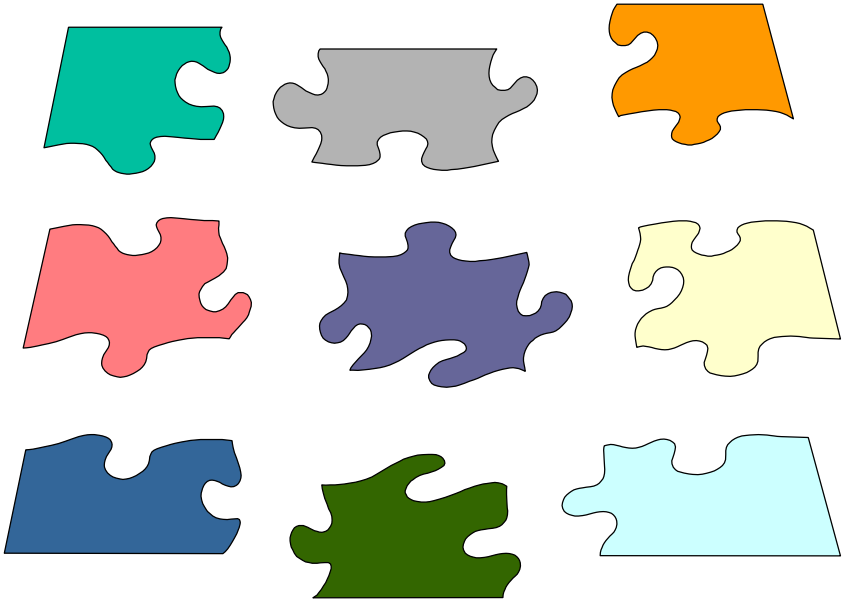


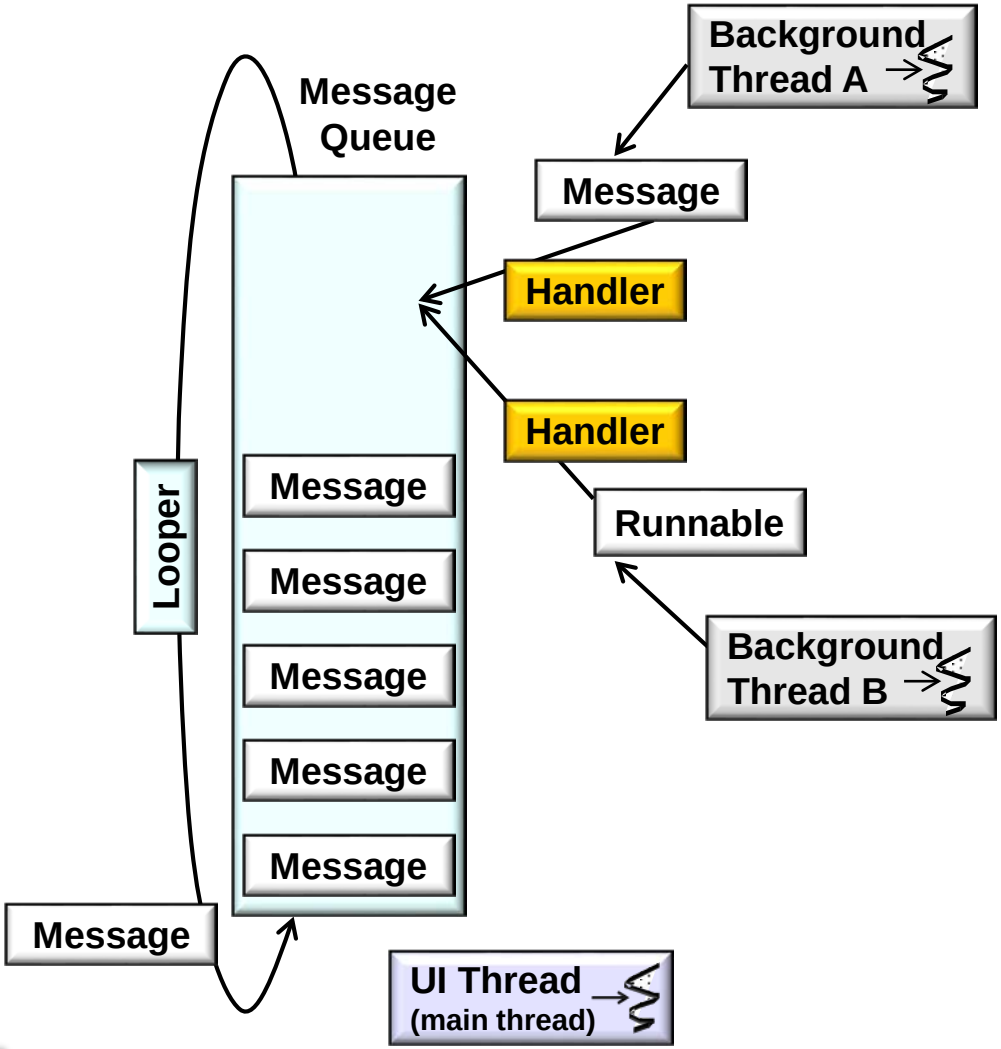
Allows apps to perform background operations & publish results on UI thread *without* manipulating threads, handlers, messages, or runnables

Overview of the AsyncTask Framework

Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected

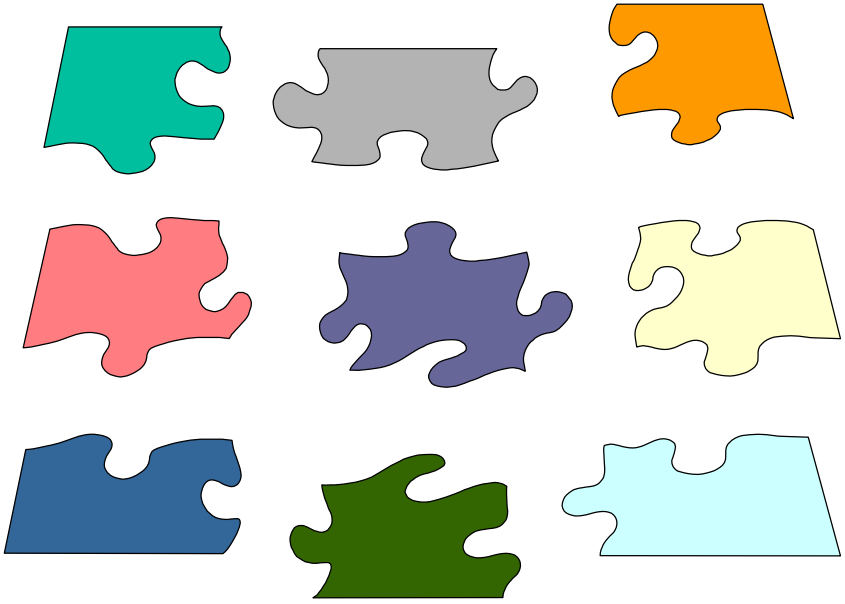


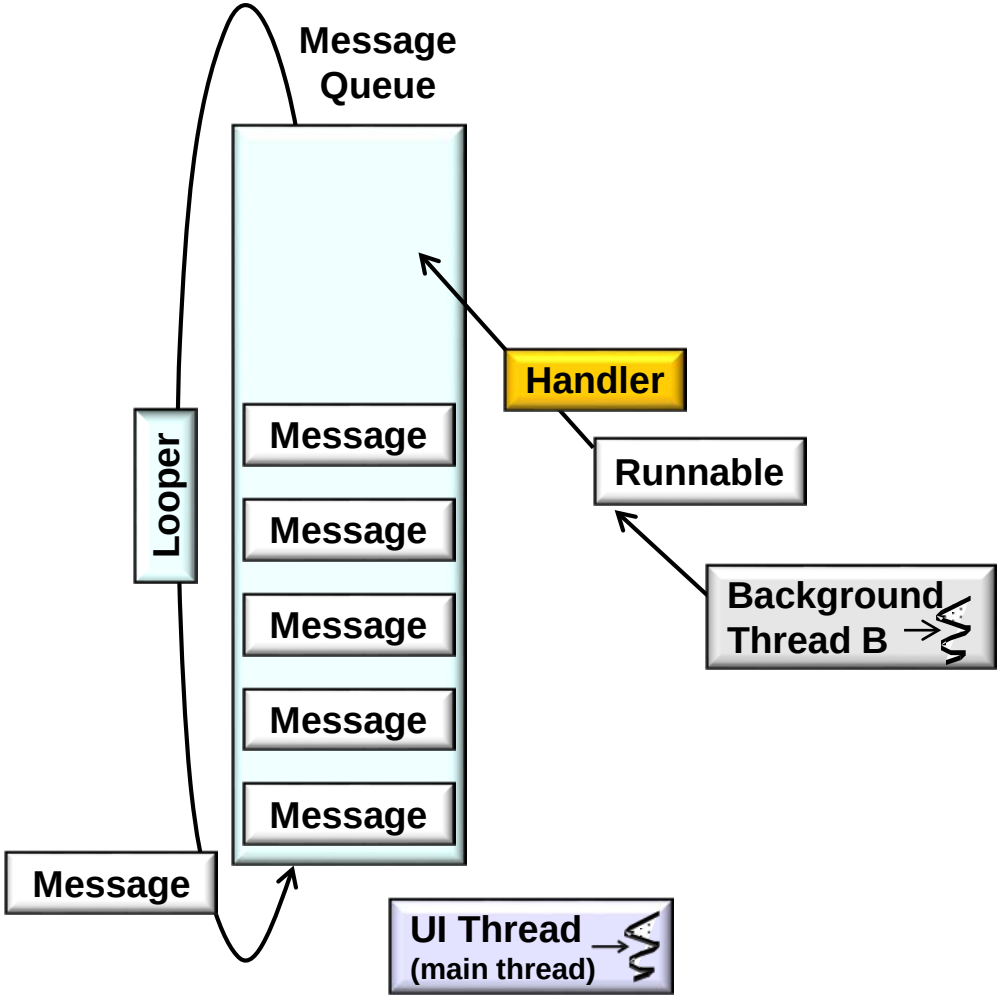


e.g., it's not clear that the classes in the HaMeR framework are related

Overview of the AsyncTask Framework

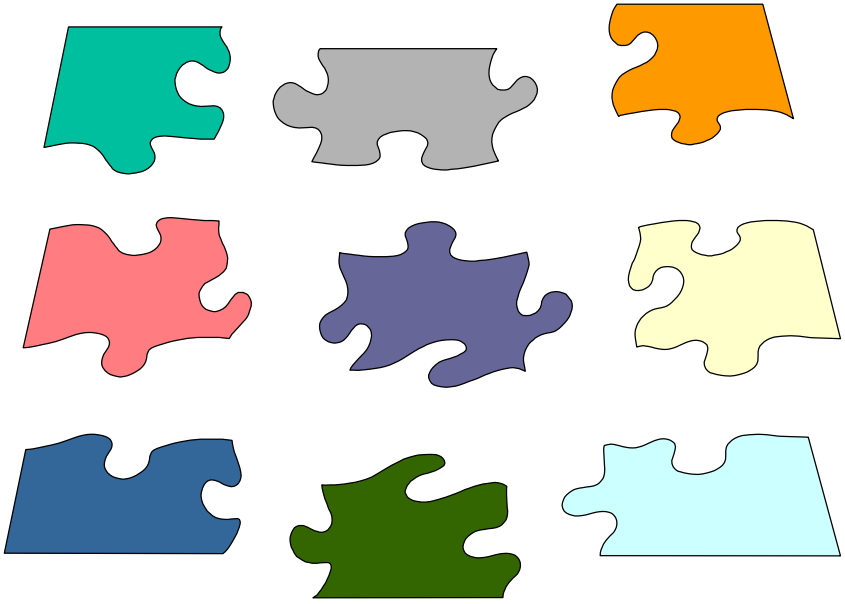
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases

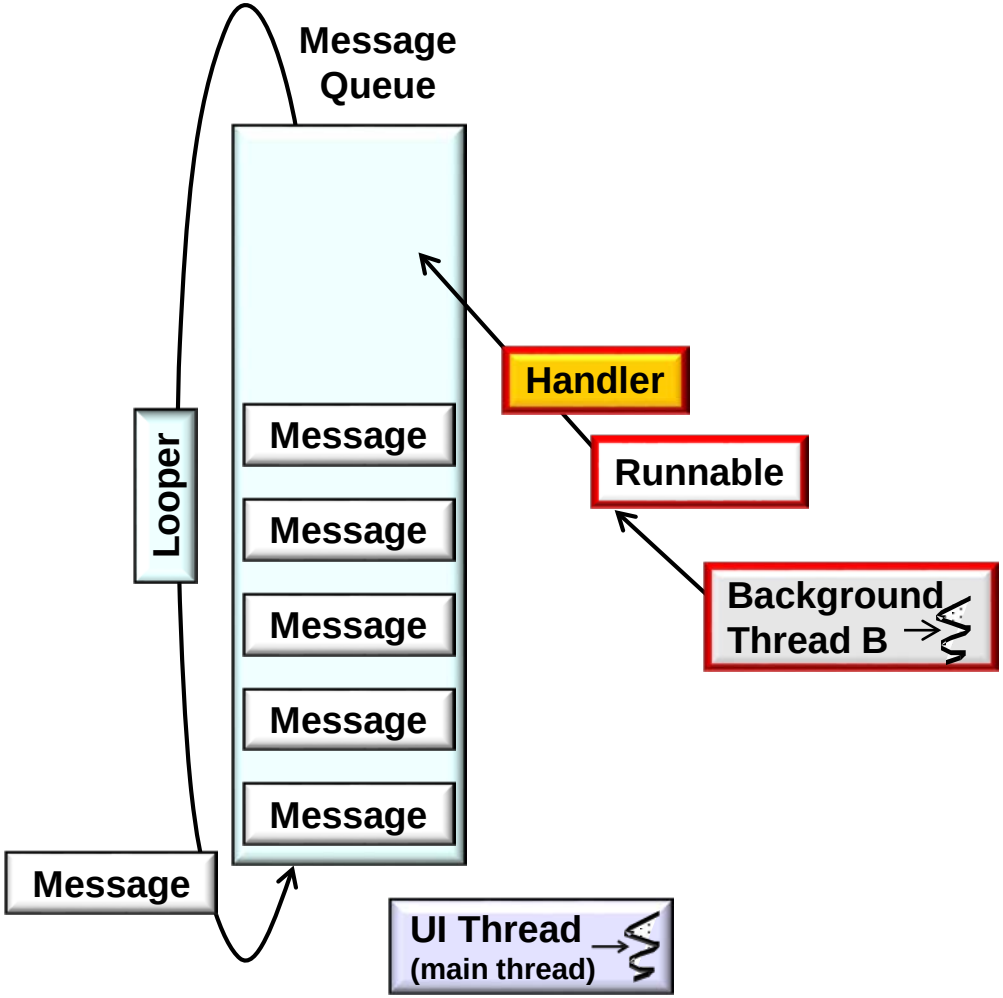




Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases

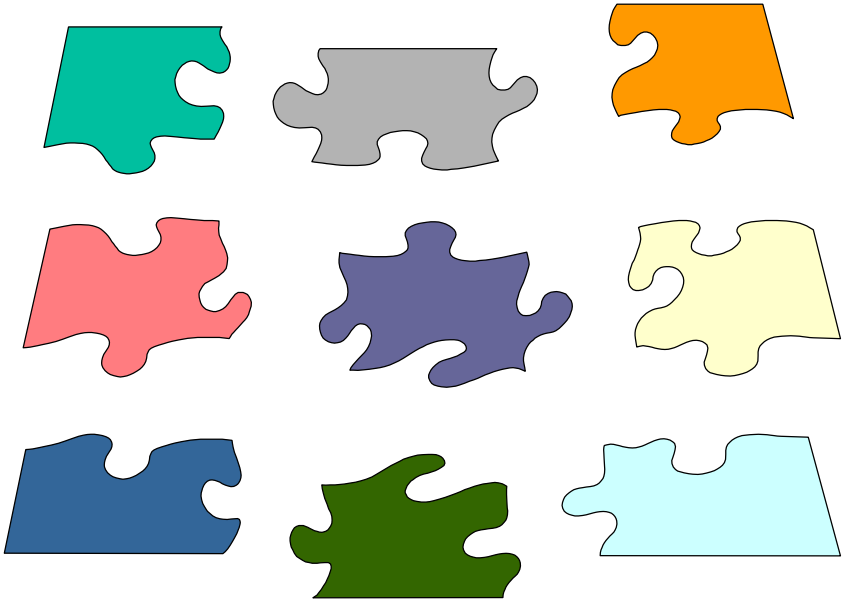


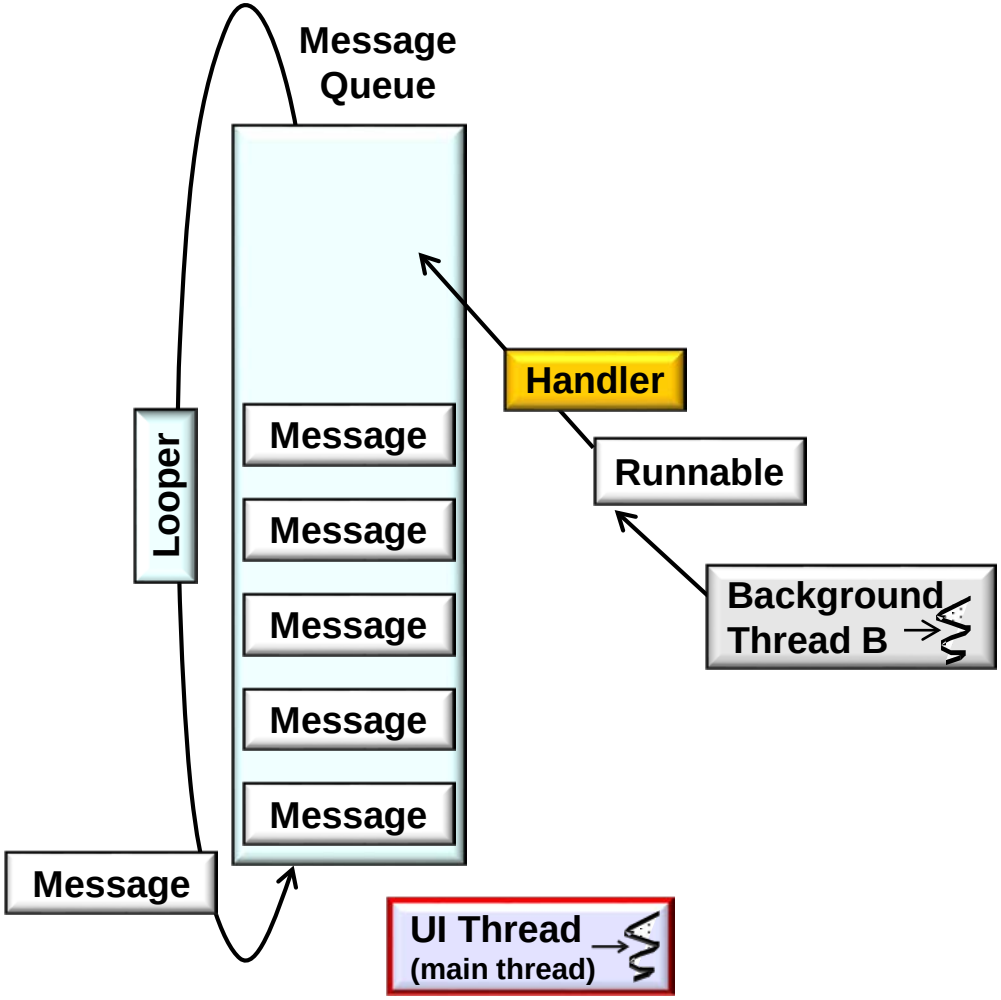


e.g., where a background thread posts a runnable to the UI thread...

Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases

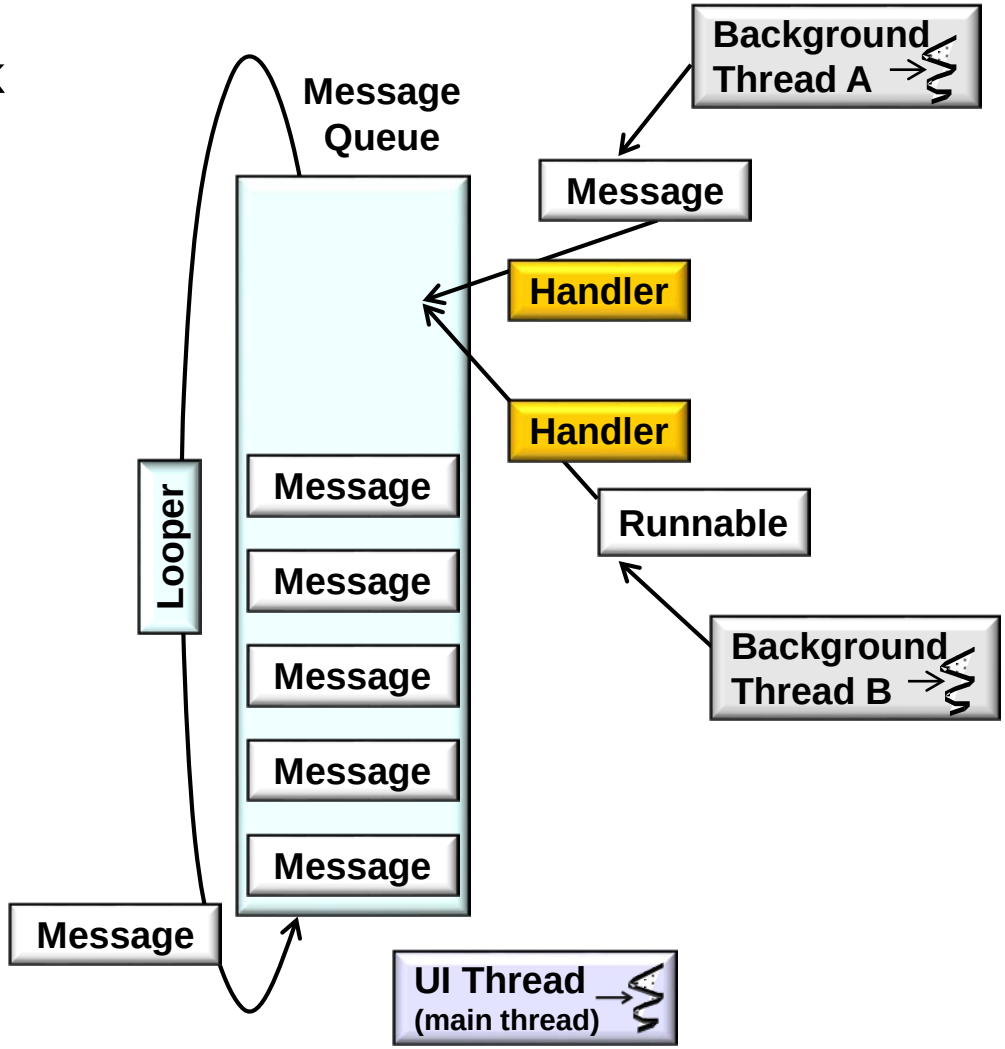




... & the UI thread dispatches the run() hook method of the runnable

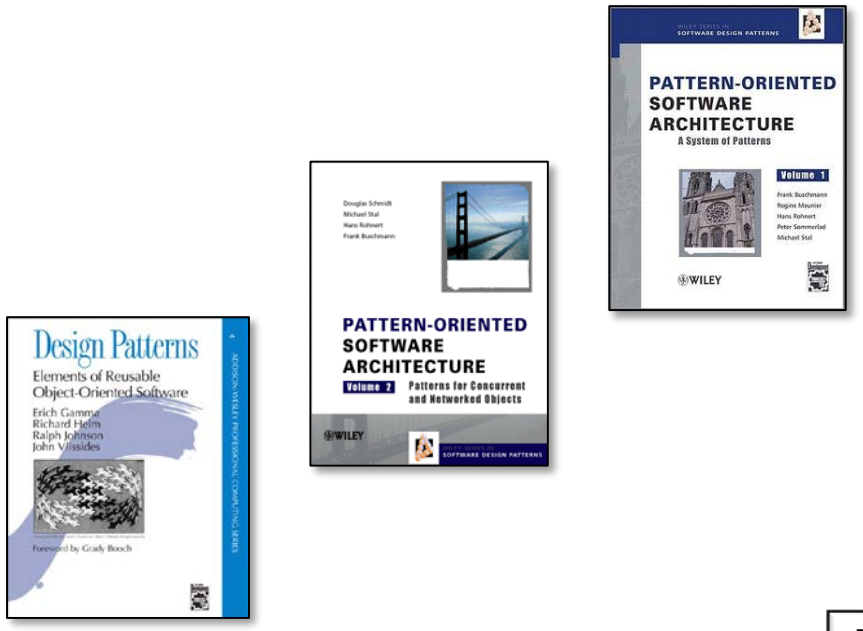
Overview of the AsyncTask Framework

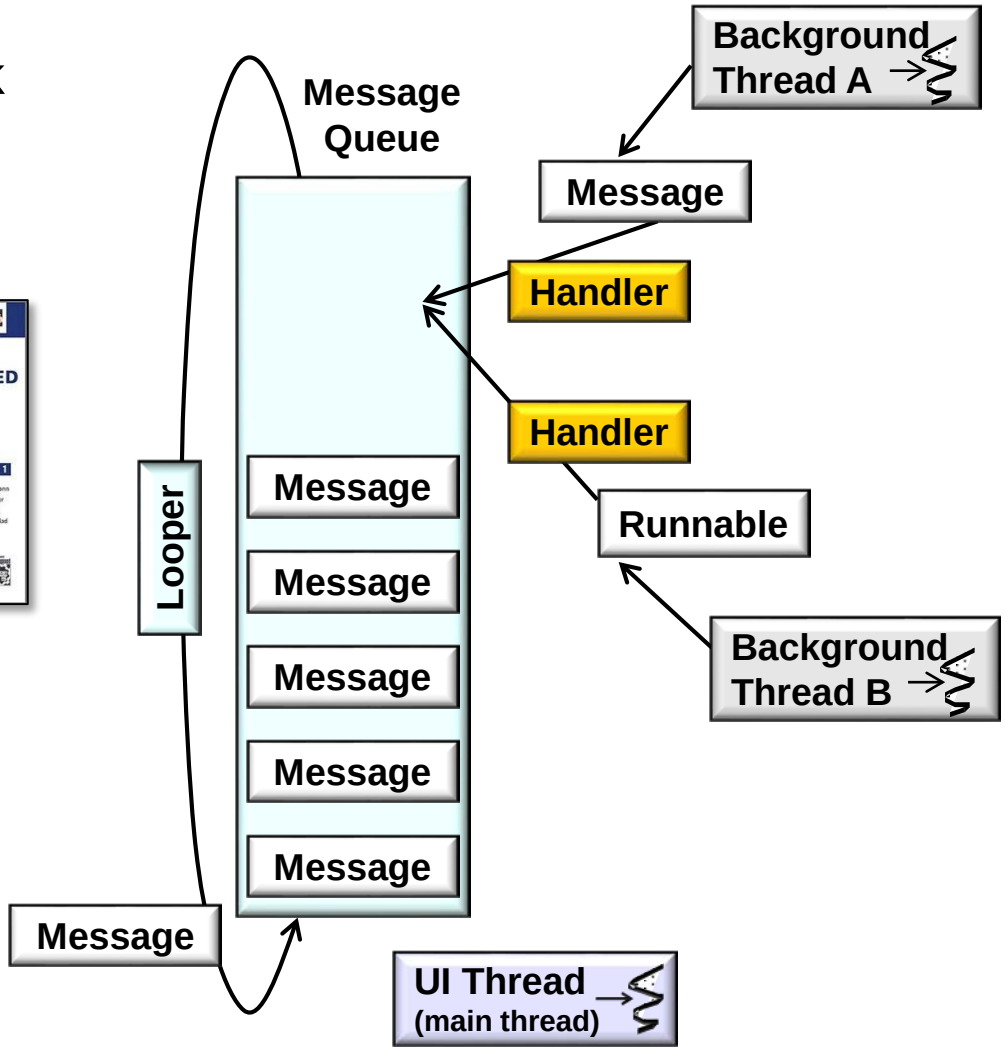
- However, there are drawbacks to the HaMeR concurrency framework



Overview of the AsyncTask Framework

- However, there are drawbacks to the HaMeR concurrency framework
  - Must understand patterns to use this framework effectively

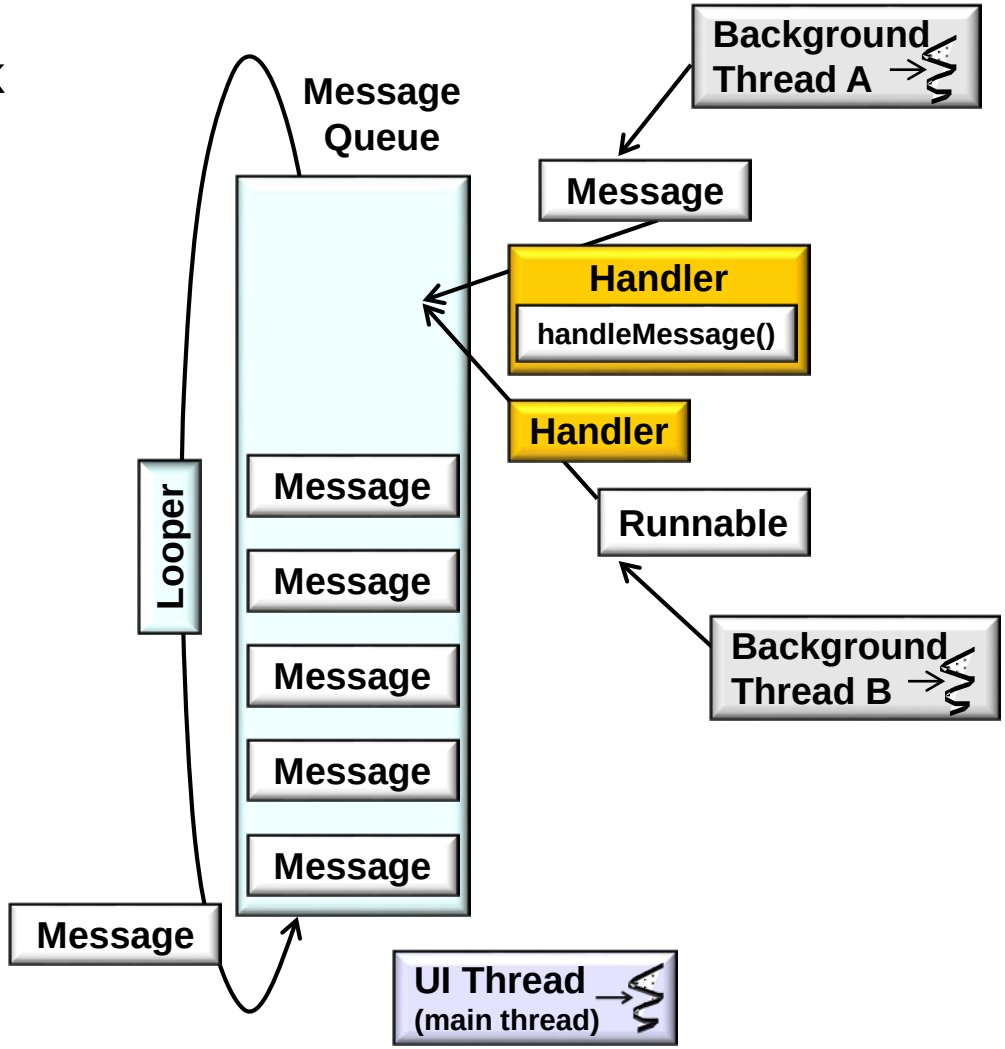




See en.wikipedia.org/wiki/Active_object & www.dre.vanderbilt.edu/~schmidt/CommandProcessor.pdf

Overview of the AsyncTask Framework

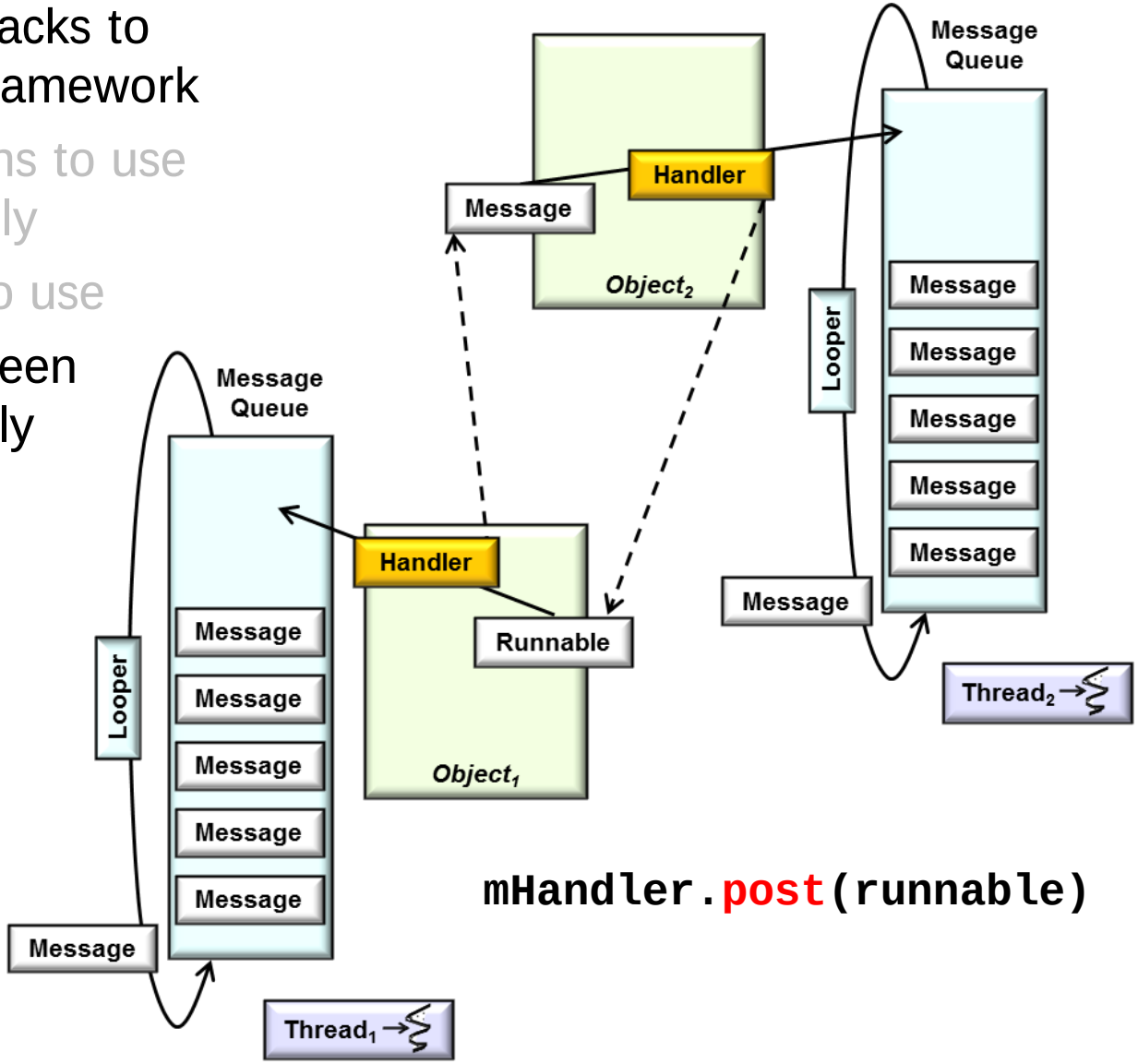
- However, there are drawbacks to the HaMeR concurrency framework
  - Must understand patterns to use this framework effectively
  - Tedious & error-prone to use



e.g., apps must understand how to manage the lifecycle of messages

Overview of the AsyncTask Framework

- However, there are drawbacks to the HaMeR concurrency framework
  - Must understand patterns to use this framework effectively
  - Tedious & error-prone to use
  - All communication between threads must be explicitly programmed

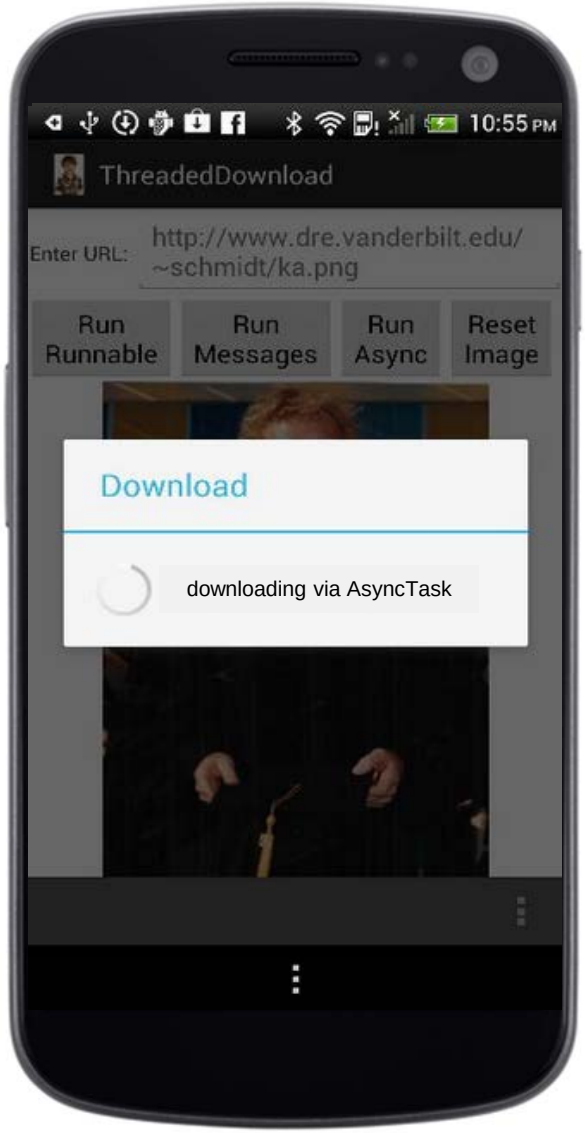



```
mHandler.sendMessage
        (message)
```

```
mHandler.post(runnable)
```

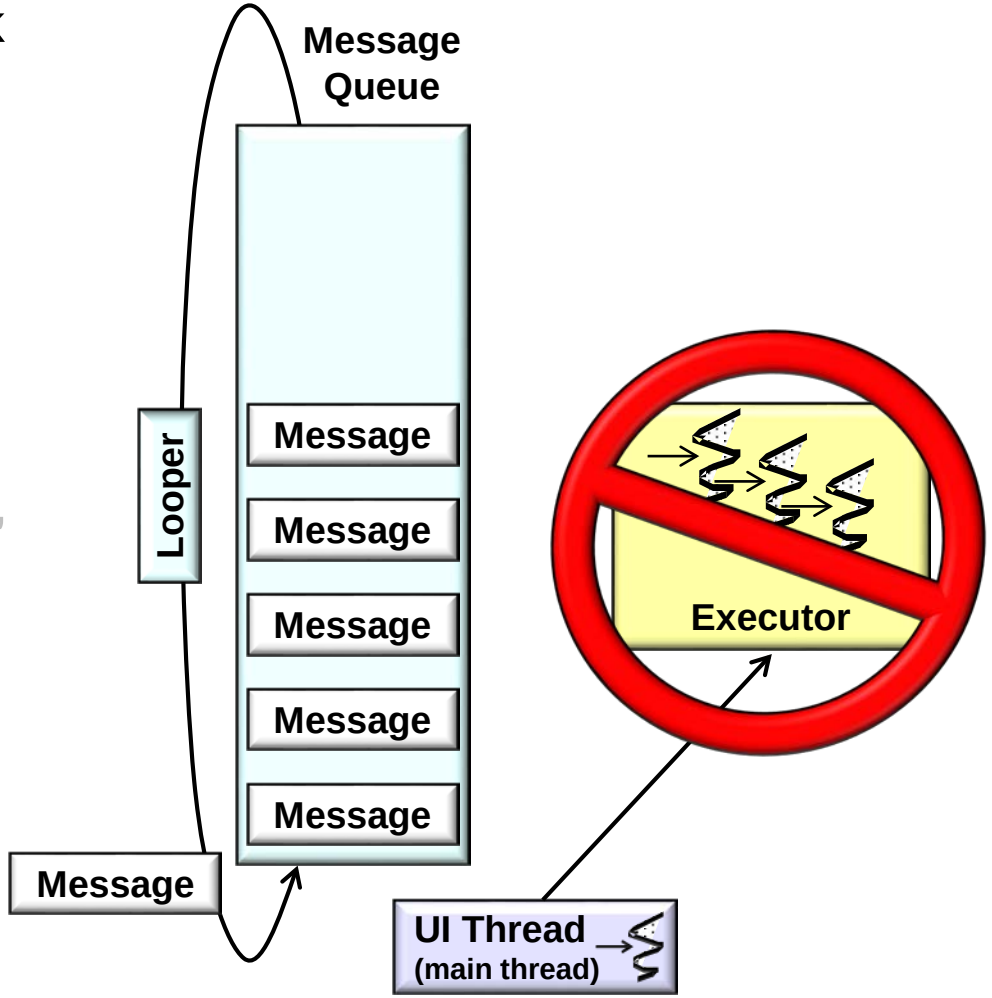
Overview of the AsyncTask Framework

- However, there are drawbacks to the HaMeR concurrency framework
  - Must understand patterns to use this framework effectively
  - Tedious & error-prone to use
  - All communication between threads must be explicitly programmed
  - Likewise, any “pre” and/or “post” processing must be explicitly programmed
    - e.g., starting & stopping a progress dialog box



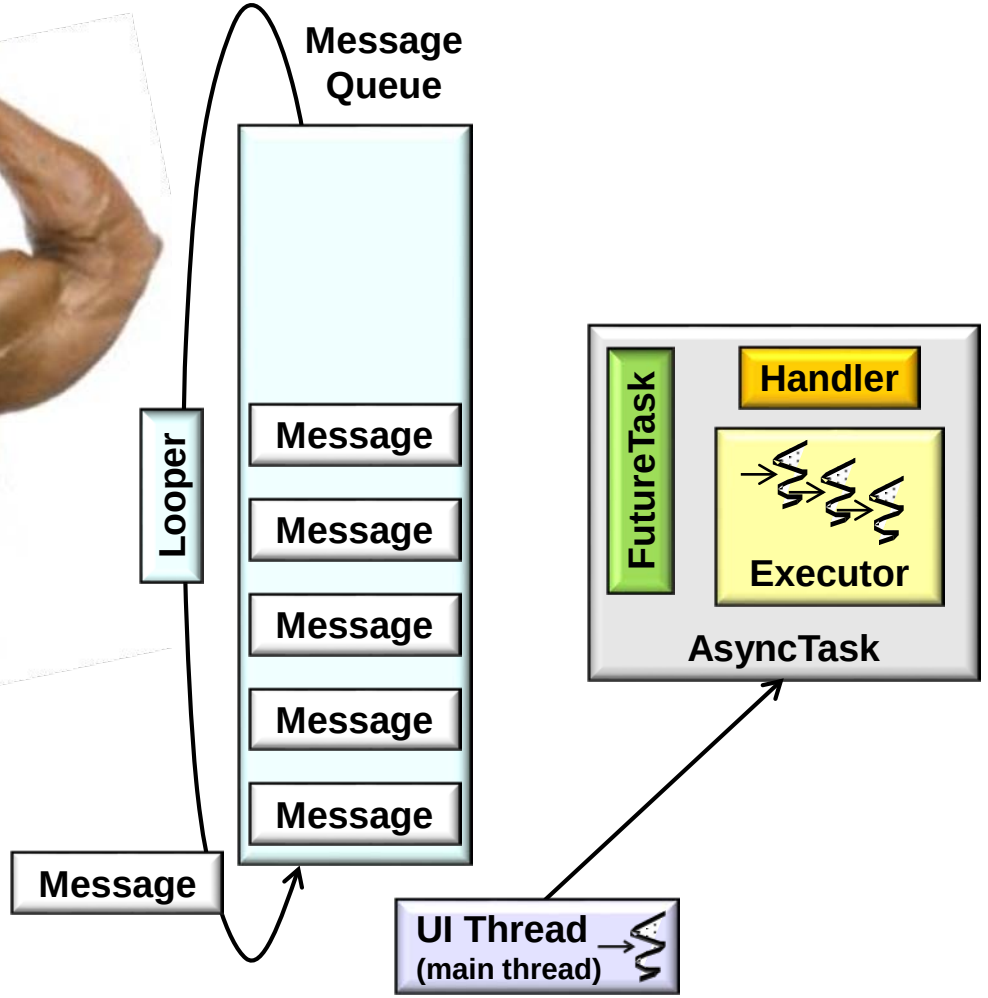
Overview of the AsyncTask Framework

- However, there are drawbacks to the HaMeR concurrency framework
  - Must understand patterns to use this framework effectively
  - Tedious & error-prone to use
  - All communication between threads must be explicitly programmed
  - Likewise, any “pre” and/or “post” processing must be explicitly programmed
  - Performance can’t be scaled up transparently



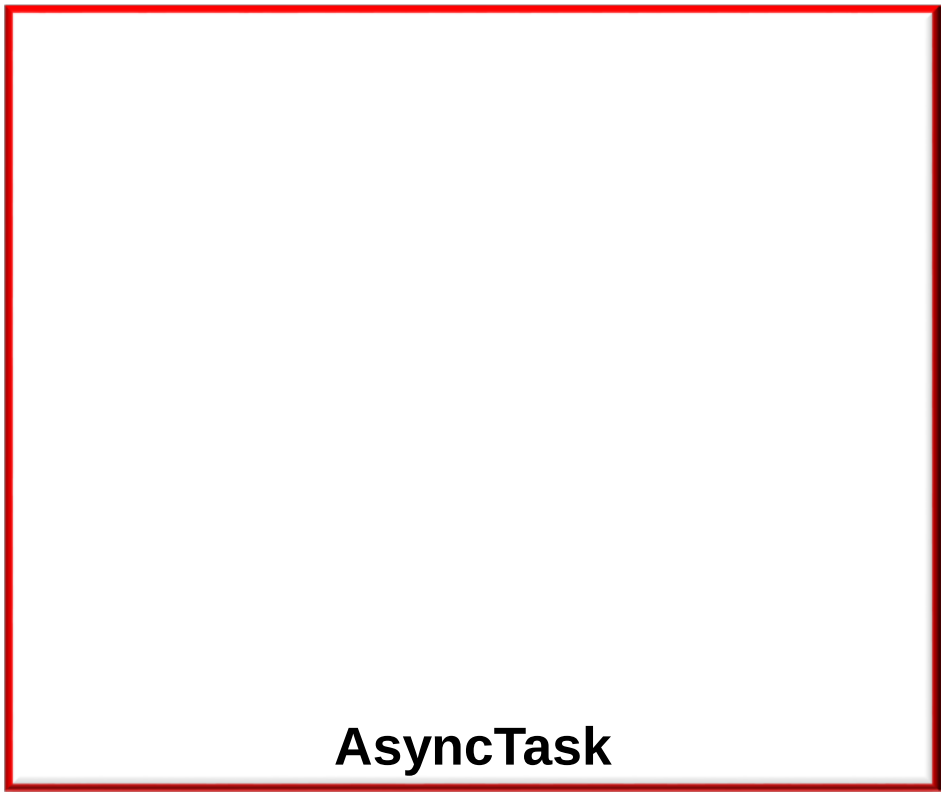
Overview of the AsyncTask Framework

- In contrast, AsyncTask framework classes are more strongly connected



Overview of the AsyncTask Framework

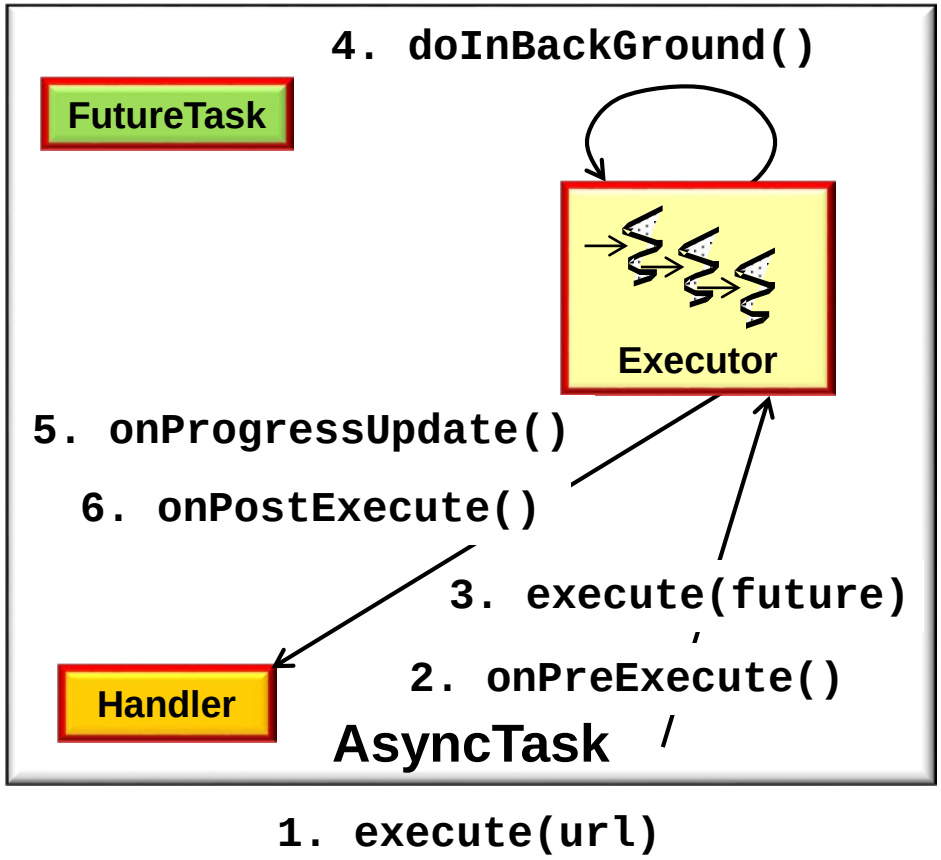
- In contrast, AsyncTask framework classes are more strongly connected
  - Complex framework details hidden via *Façade* pattern



See en.wikipedia.org/wiki/Facade_pattern

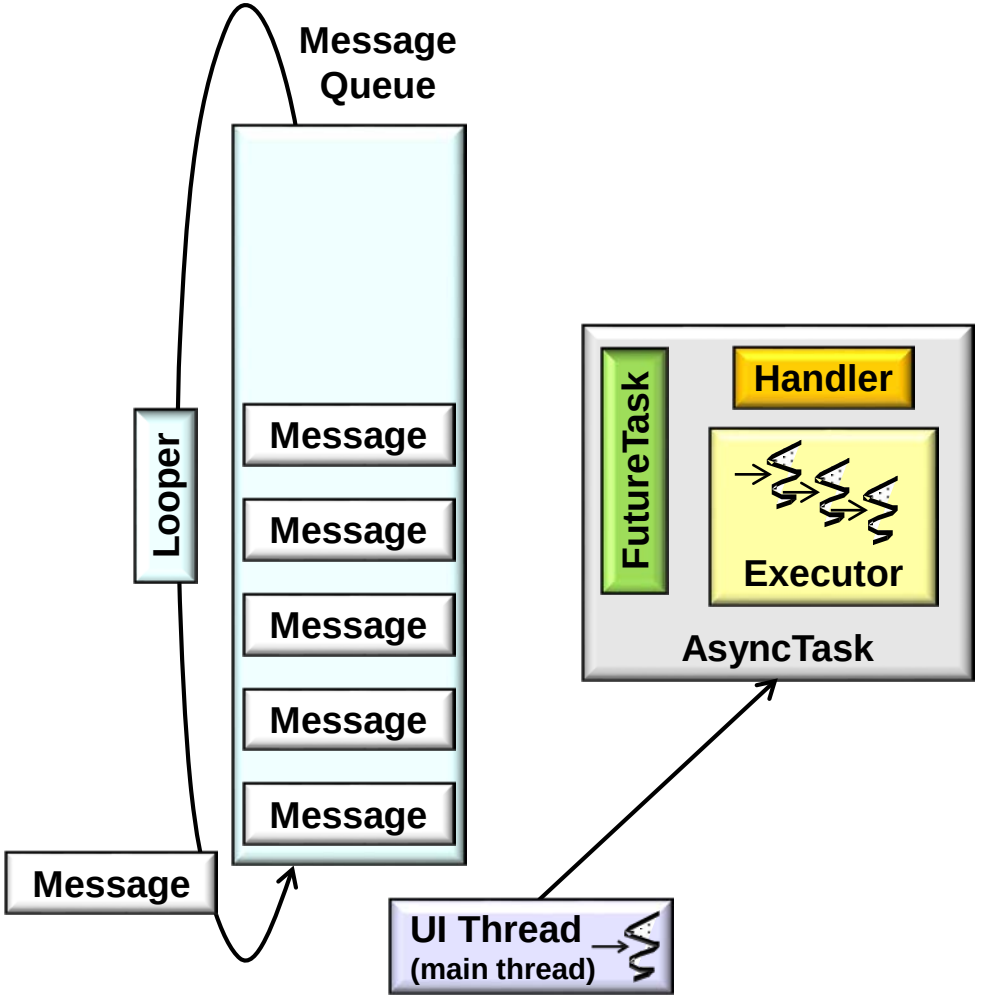
Overview of the AsyncTask Framework

- In contrast, AsyncTask framework classes are more strongly connected
  - Complex framework details hidden via *Façade* pattern
    - Encapsulates a complicated subsystem or framework with a simpler interface



Overview of the AsyncTask Framework

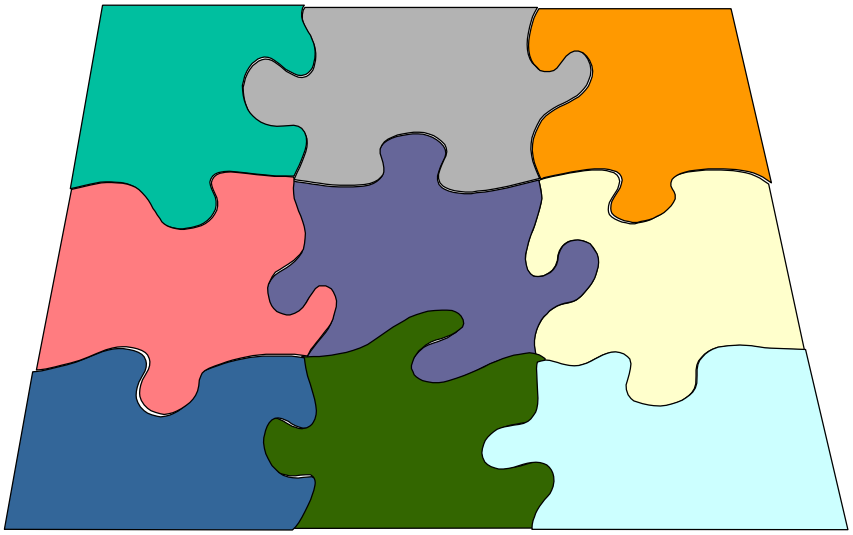
- In contrast, AsyncTask framework classes are more strongly connected
  - Complex framework details hidden via *Façade* pattern
  - Yields a smaller "surface area"

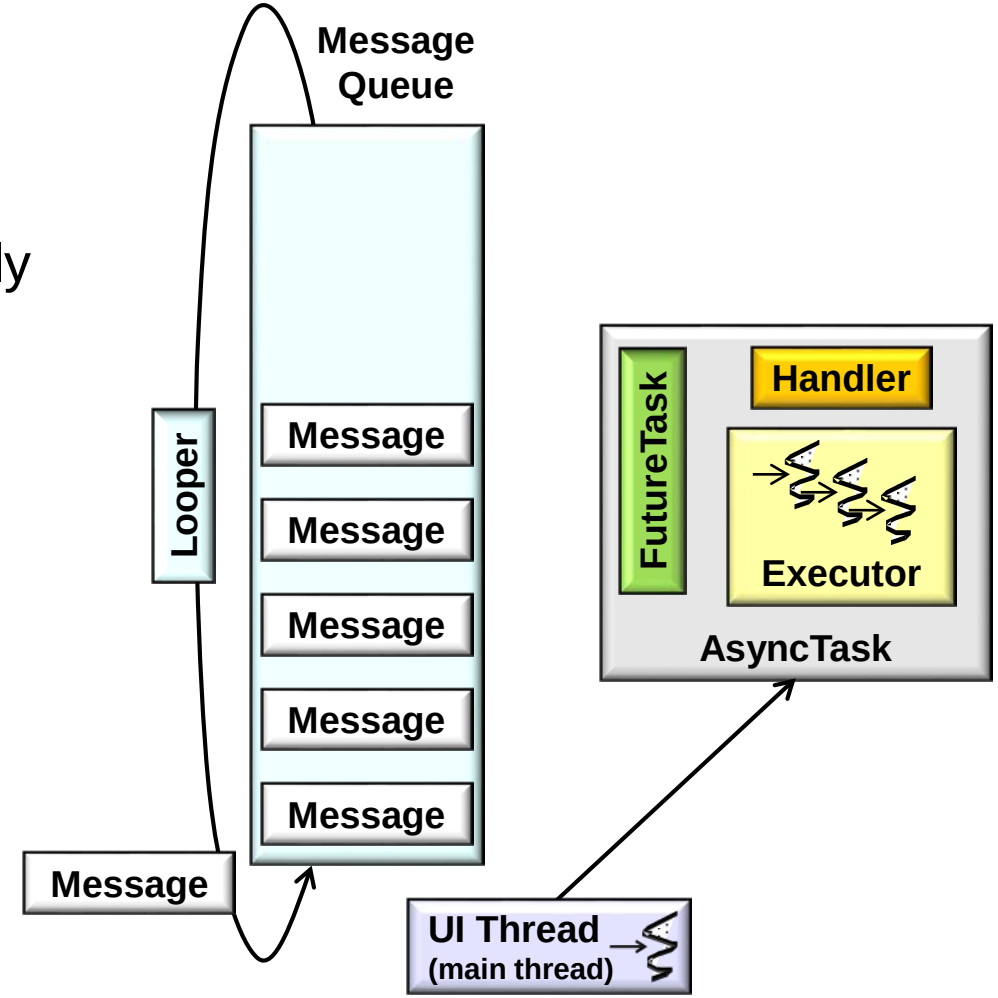


i.e., programmers can focus on the "what" not the "how"

Overview of the AsyncTask Framework

- In contrast, AsyncTask framework classes are more strongly connected
  - Complex framework details hidden via *Façade* pattern
  - Yields a smaller "surface area"
  - Run concurrently, *without* directly manipulating threads, handlers, messages, or runnables

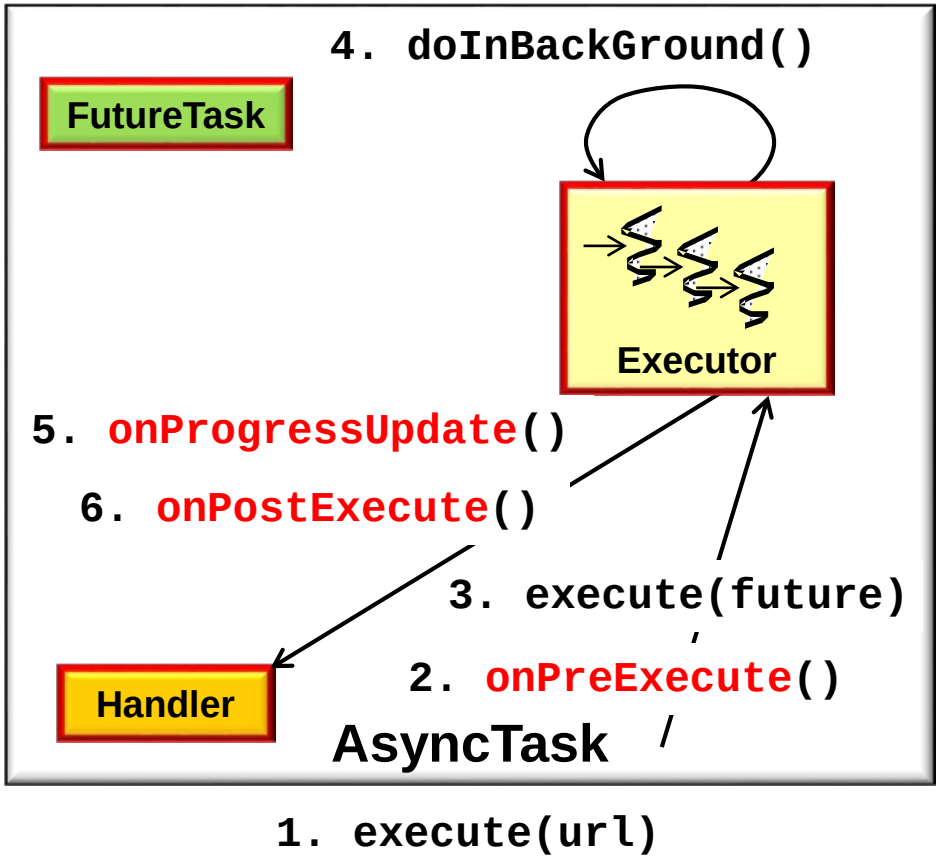




See en.wikipedia.org/wiki/Template_Method_pattern

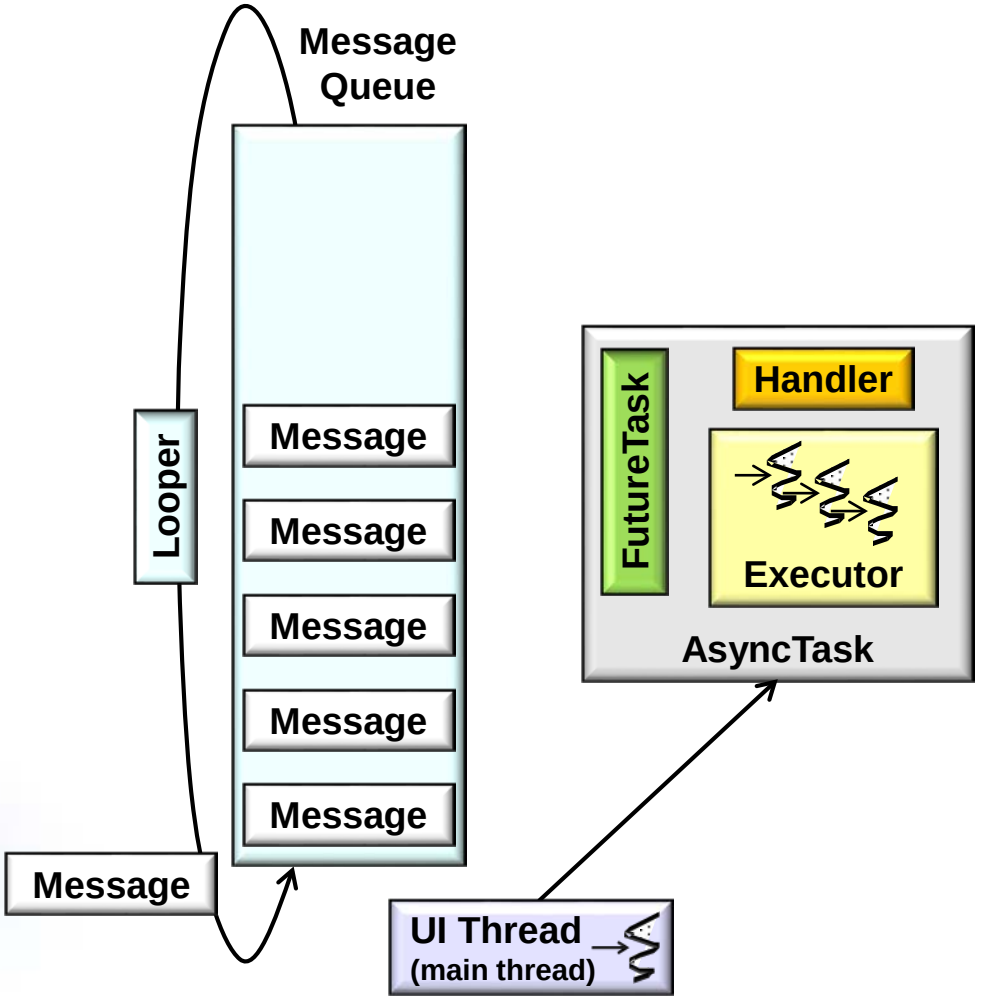
Overview of the AsyncTask Framework

- In contrast, AsyncTask framework classes are more strongly connected
  - Complex framework details hidden via *Façade* pattern
  - Yields a smaller "surface area"
  - Run concurrently, *without* directly manipulating threads, handlers, messages, or runnables
  - Hook methods can perform pre-processing, post-processing, & communication between the background & UI threads



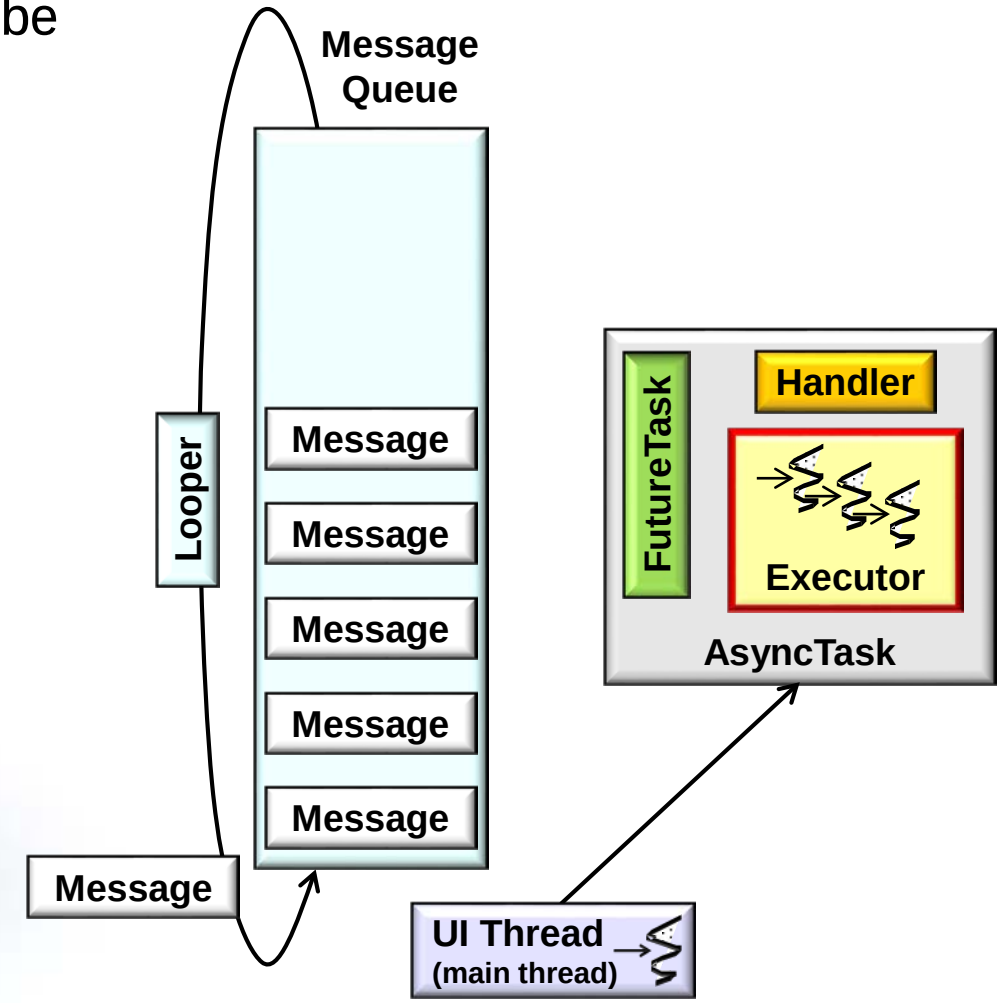
Overview of the AsyncTask Framework

- Likewise, AsyncTask performance can be scaled up transparently



Overview of the AsyncTask Framework

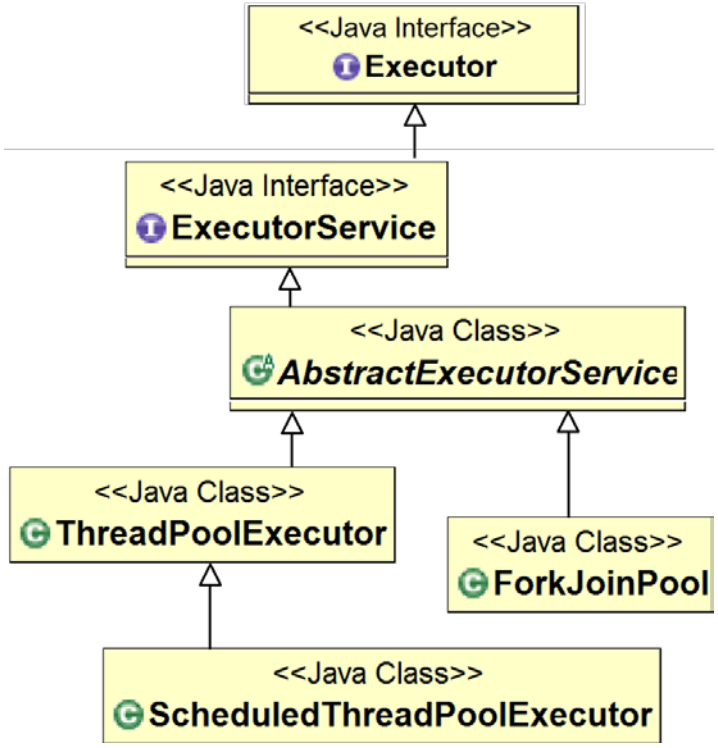
- Likewise, AsyncTask performance can be scaled up transparently, e.g.
  - Contains a thread pool that can be specified by programmers

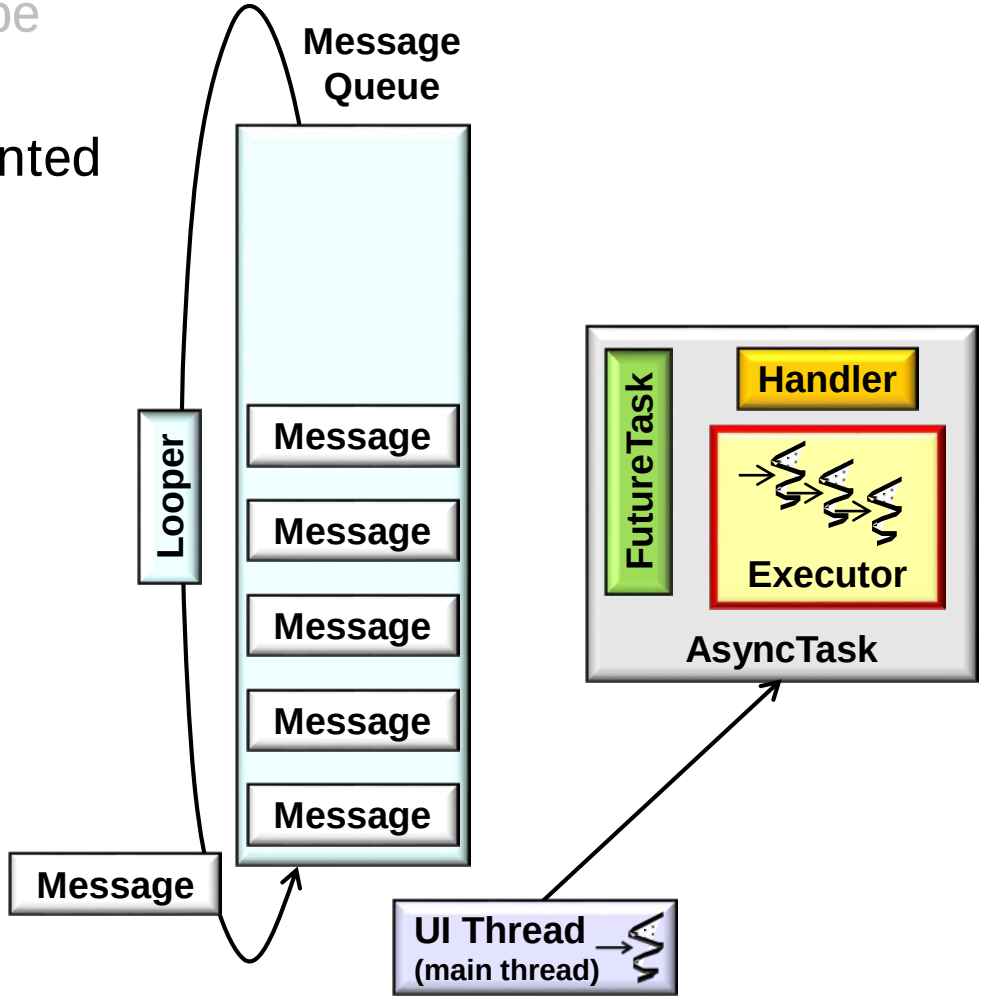


See en.wikipedia.org/wiki/Thread_pool

Overview of the AsyncTask Framework

- Likewise, AsyncTask performance can be scaled up transparently, e.g.
  - Contains a thread pool that can be specified by programmers
  - This thread pool can be implemented via the Java Executor framework

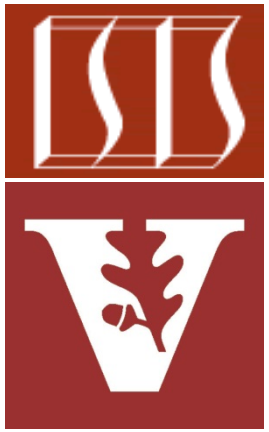




See docs.oracle.com/javase/tutorial/essential/concurrency/executors.html

End of Overview of the AsyncTask Framework (Part 1)

Overview of the AsyncTask Framework (Part 2)

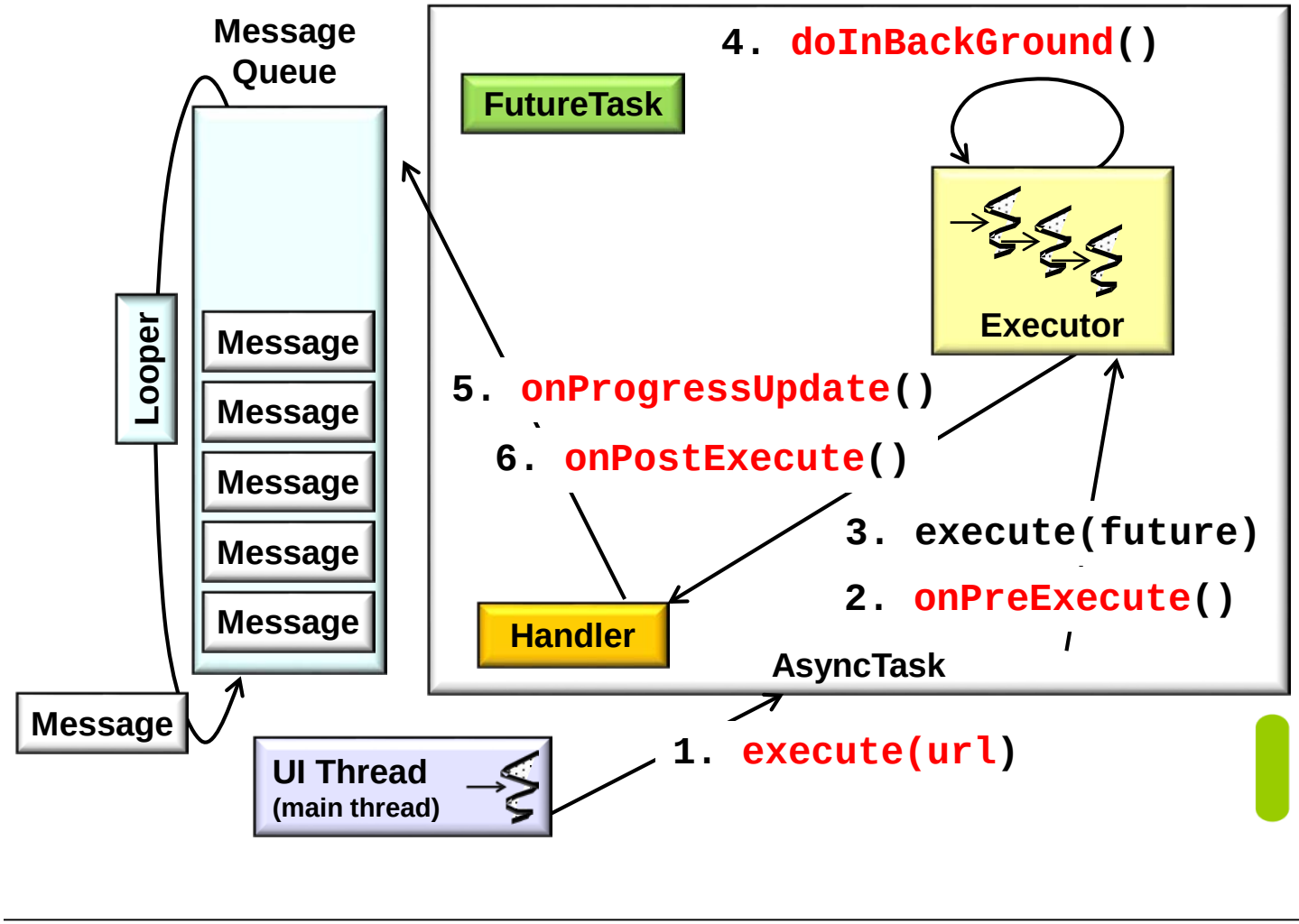


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Recognize the capabilities provided by the Android AsyncTask framework
- Know which methods are provided by AsyncTask class



Categories of Methods in AsyncTask

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods

AsyncTask

Added in API level 3

extends Object

java.lang.Object
↳android.os.AsyncTask<Params, Progress, Result>

Class Overview

AsyncTask enables proper and easy use of the UI thread. This class allows 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` pacakge such as `Executor`, `ThreadPoolExecutor` and `FutureTask`.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

See developer.android.com/reference/android/os/AsyncTask.html

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Invoked by apps

```
AsyncTask<Params, Progress, Result>
        execute(Params... params)
```

- Executes the task with the specified parameters

```
AsyncTask<Params, Progress, Result>
   executeOnExecutor(Executor exec,
                     Params... params)
```

- Executes the task with the specified parameters on the specified Executor

```
static void execute(Runnable
                    runnable)
```

- Convenience version of execute(Object) for use with a simple Runnable object

```
boolean cancel
   (boolean mayInterruptIfRunning)
```

- Attempts to cancel execution of this task

...

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Invoked by apps

`AsyncTask<Params, Progress, Result> execute(Params... params)`

- Executes the task with the specified parameters

`AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, Params... params)`

- Executes the task with the specified parameters on the specified Executor

`static void execute(Runnable runnable)`

- Convenience version of execute(Object) for use with a simple Runnable object

`boolean cancel (boolean mayInterruptIfRunning)`

- Attempts to cancel execution of this task

`...`

execute() runs each AsyncTask one-at-a-time (serially) in a background thread within a process

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Invoked by apps

`AsyncTask<Params, Progress, Result> execute(Params... params)`

- Executes the task with the specified parameters

`AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, Params... params)`

- Executes the task with the specified parameters on the specified Executor

`static void execute(Runnable runnable)`

- Convenience version of execute(Object) for use with a simple Runnable object

`boolean cancel (boolean mayInterruptIfRunning)`

- Attempts to cancel execution of this task

`...`

executeOnExecutor() can run multiple AsyncTasks concurrently in a pool of threads within a process

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Invoked by apps

`AsyncTask<Params, Progress, Result> execute(Params... params)`

- Executes the task with the specified parameters

`AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, Params... params)`

- Executes the task with the specified parameters on the specified Executor

`static void execute(Runnable runnable)`

- Convenience version of execute(Object) for use with a simple Runnable object

`boolean cancel (boolean mayInterruptIfRunning)`

- Attempts to cancel execution of this task

`...`

This method is simply a front-end to the underlying Executor

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Invoked by apps

`AsyncTask<Params, Progress, Result> execute(Params... params)`

- Executes the task with the specified parameters

`AsyncTask<Params, Progress, Result> executeOnExecutor(Executor exec, Params... params)`

- Executes the task with the specified parameters on the specified Executor

`static void execute(Runnable runnable)`

- Convenience version of execute(Object) for use with a simple Runnable object

`boolean cancel (boolean mayInterruptIfRunning)`

- Attempts to cancel execution of this task

`...`

cancel() requires cooperation by the AsyncTask, i.e., it's voluntary

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

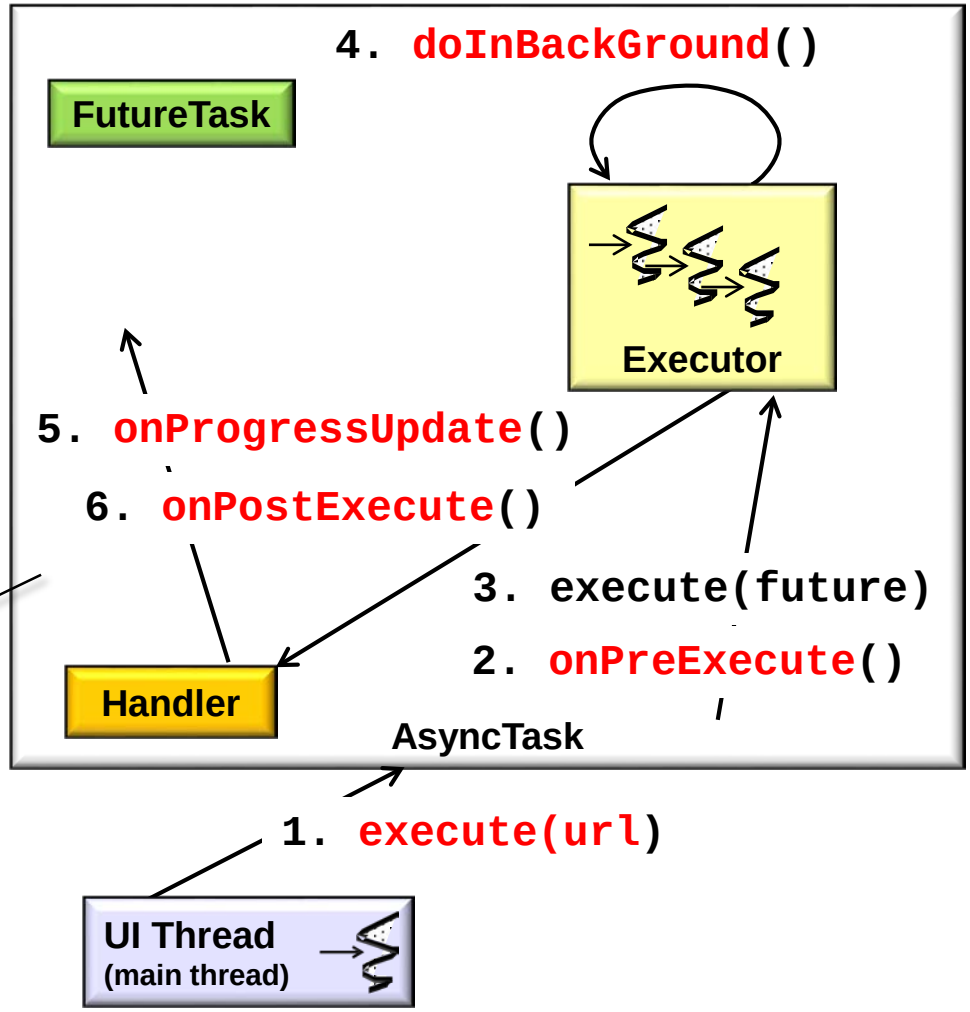
`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

`...`

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework



The AsyncTask framework invokes these methods at different points of time & in different thread contexts

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

`...`

Called in UI thread after execute() called, i.e., prior to any other processing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

`...`

Runs in the background thread to perform the computation

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

`...`

Called in UI thread to convey incremental results sent from a background thread

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

`...`

Called in UI thread after all background processing is finished successfully

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected hook methods
    - Overridden by apps & invoked by the Async Task framework

`void onPreExecute()`
- Runs on UI thread before doInBackground()

`abstract Result doInBackground (Params... params)`
- Override this method to perform a computation in a background thread

`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

`void onPostExecute(Result result)`
- Runs on UI thread after doInBackground()

`void onCancelled(Result result)`
- Runs on UI thread after cancel() is invoked & doInBackground() has finished

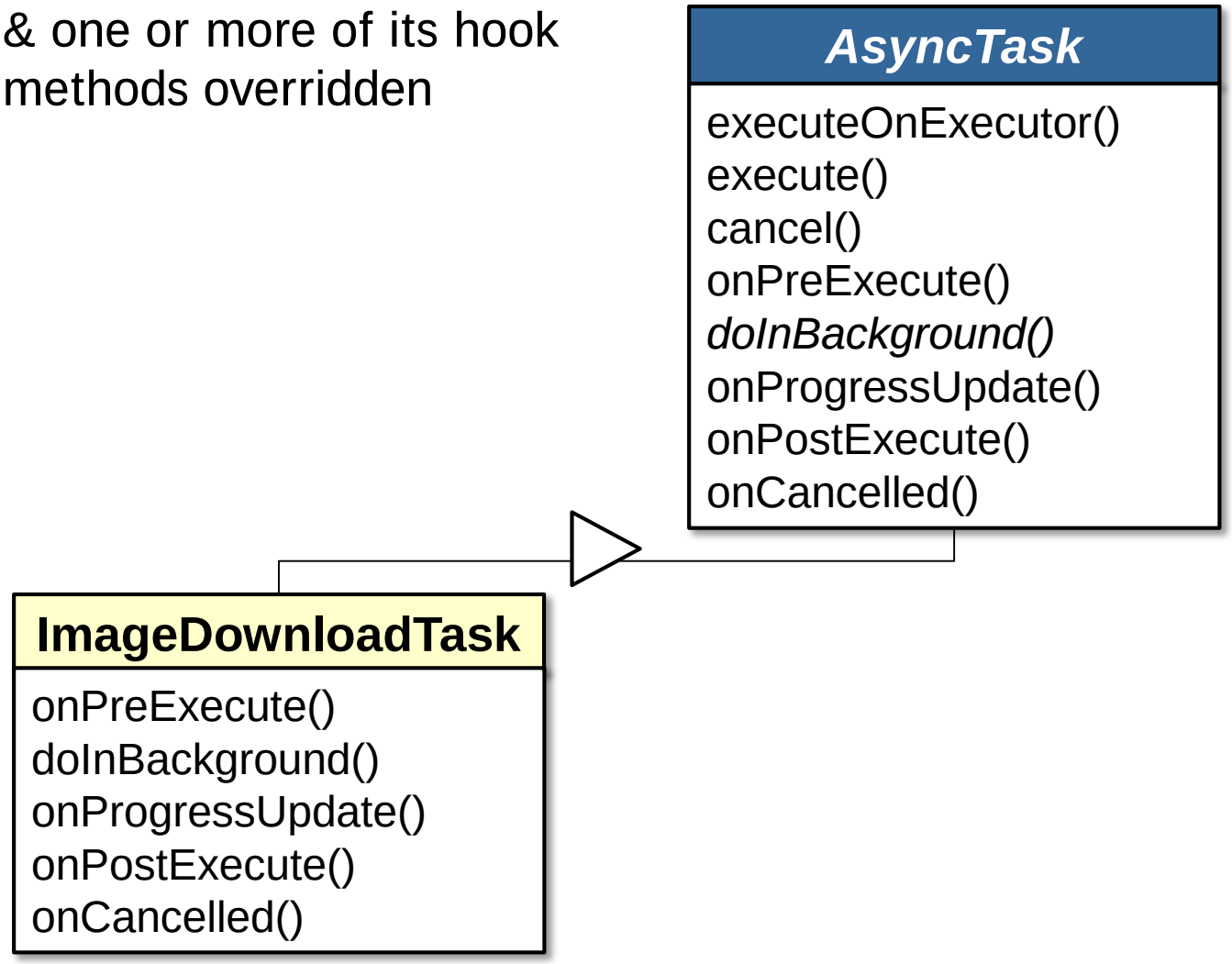
`...`

Called in UI thread after background processing has been cancelled

Overriding Hook Methods in the AsyncTask Class

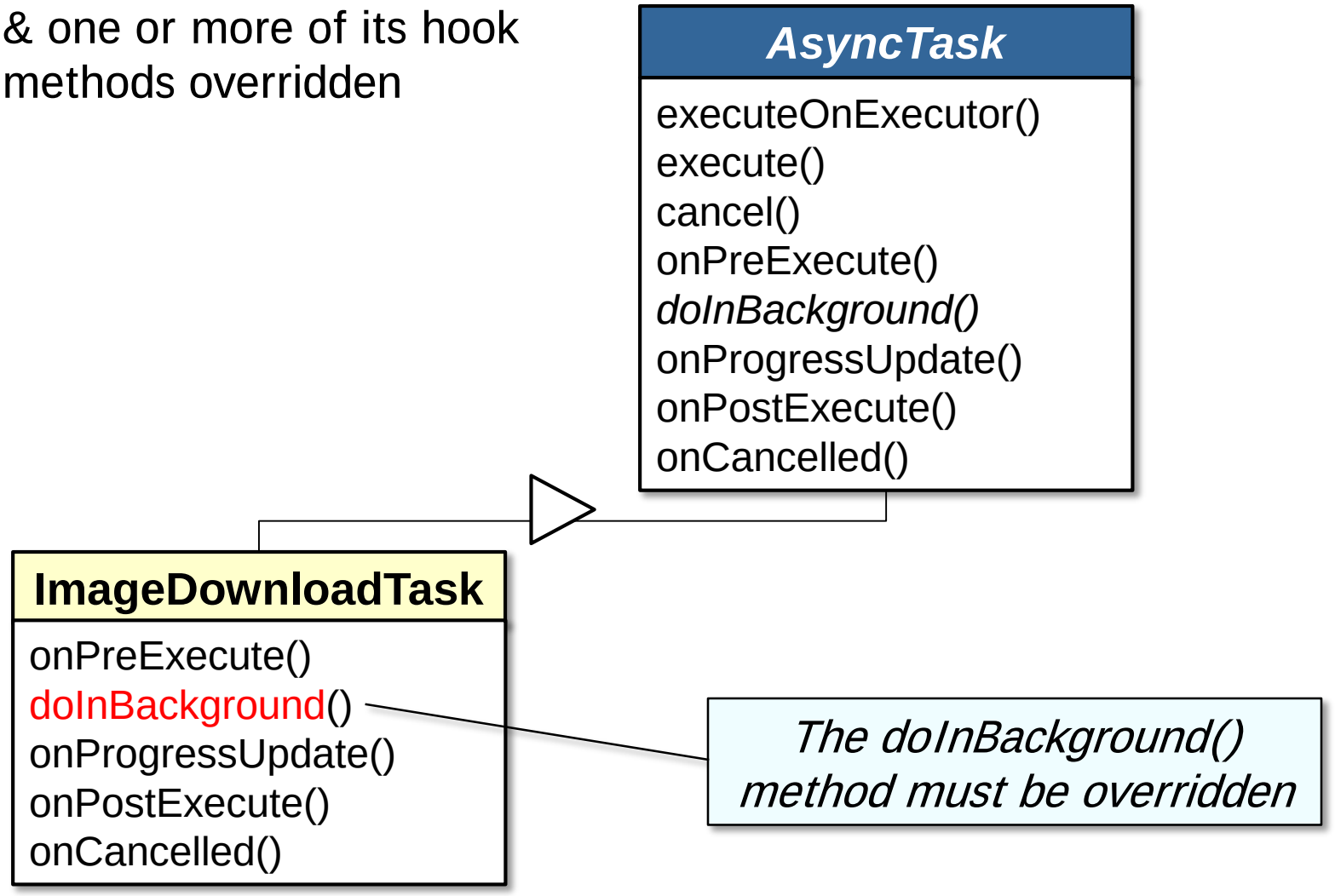
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



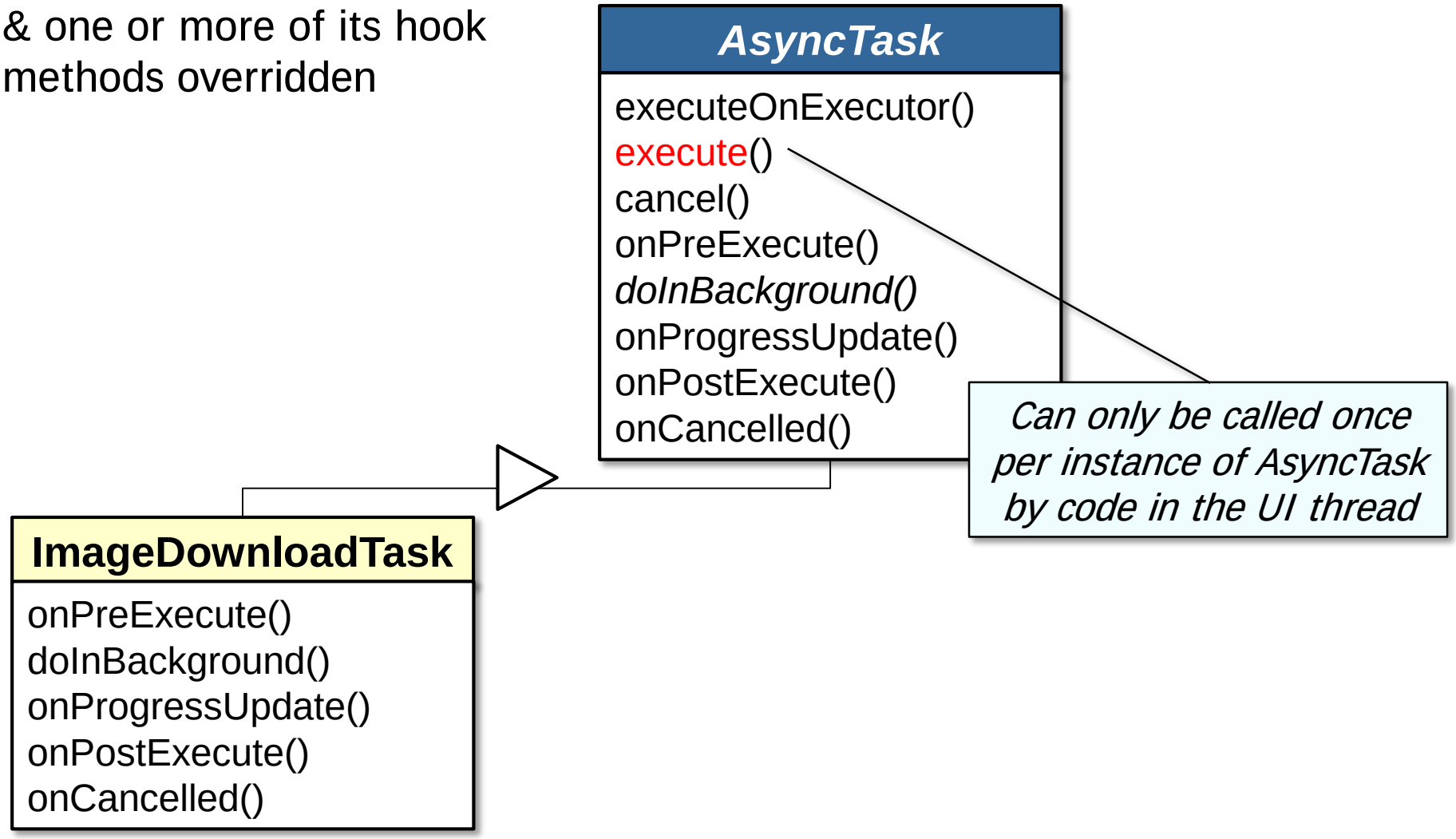
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



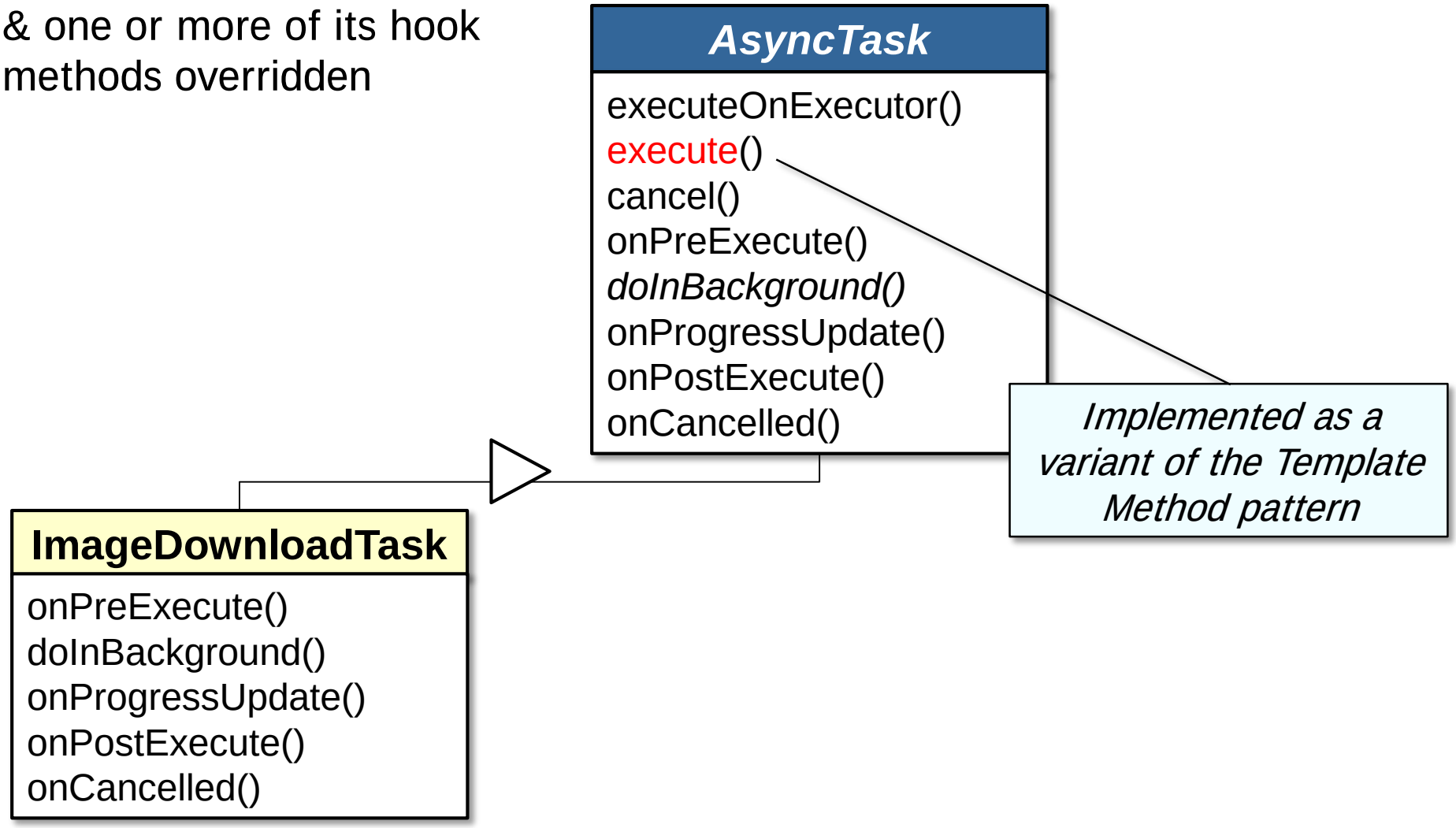
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



Overriding Hook Methods in the AsyncTask Class

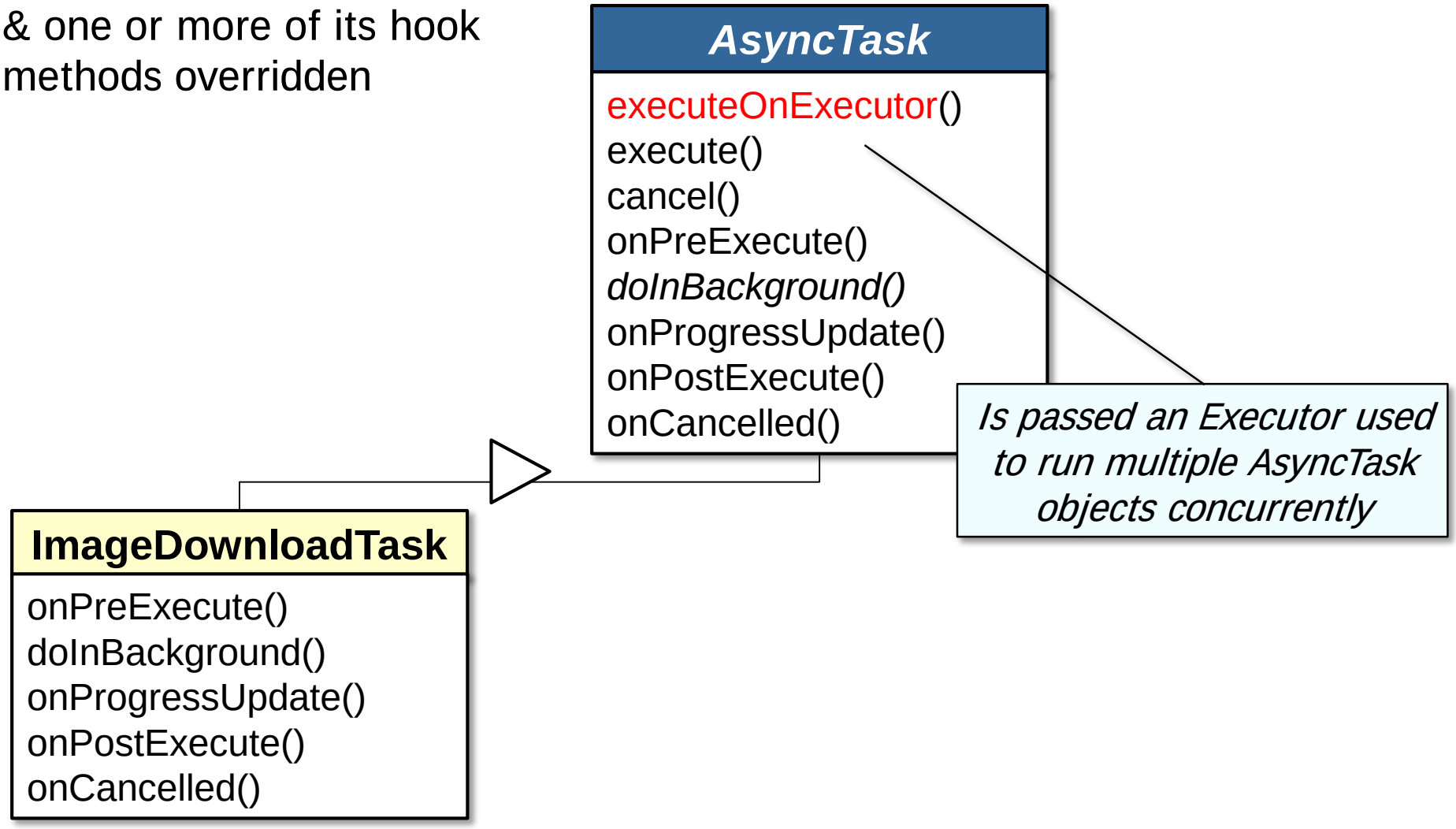
- AsyncTask must be extended & one or more of its hook methods overridden



See en.wikipedia.org/wiki/Template_method_pattern

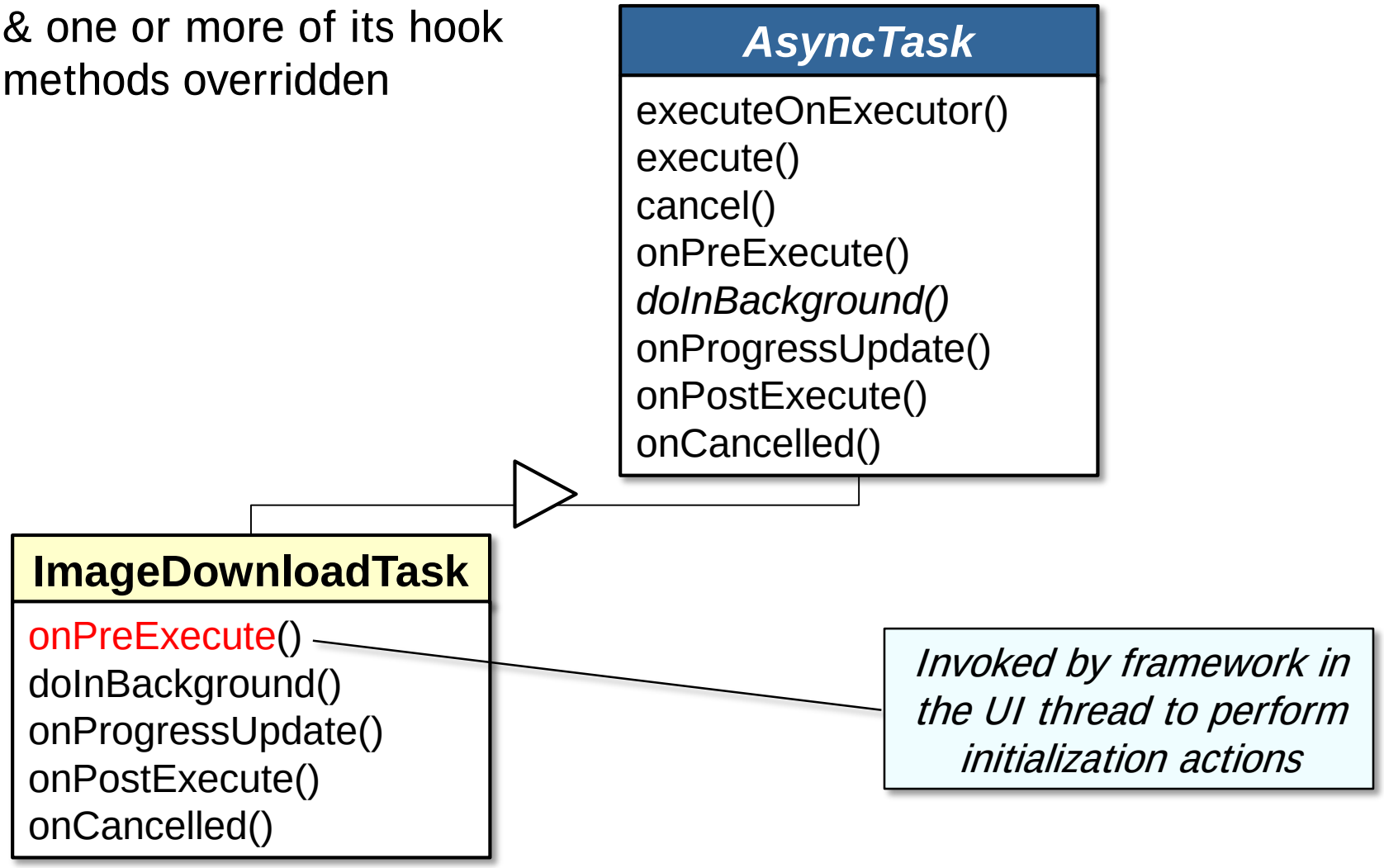
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



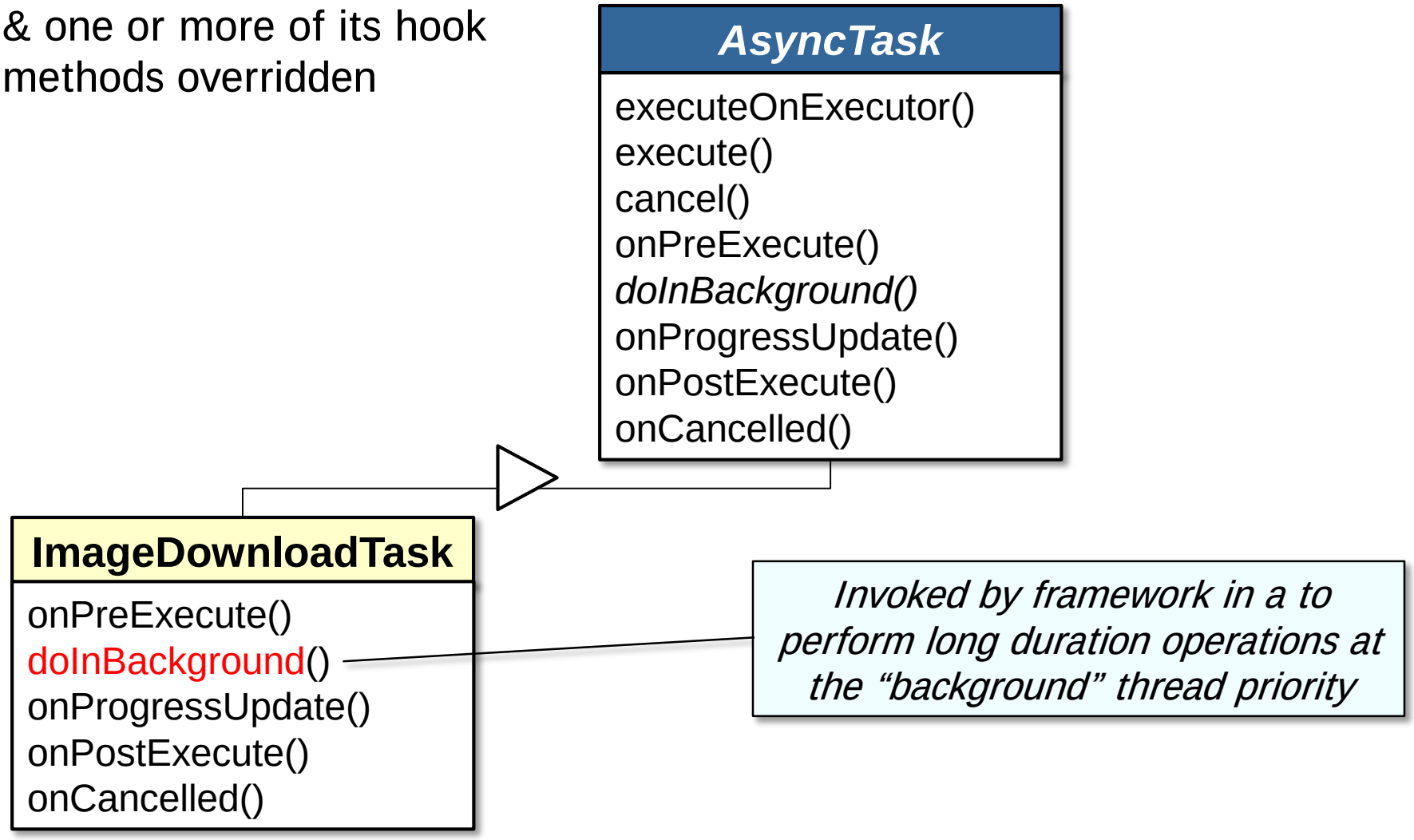
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



Overriding Hook Methods in the AsyncTask Class

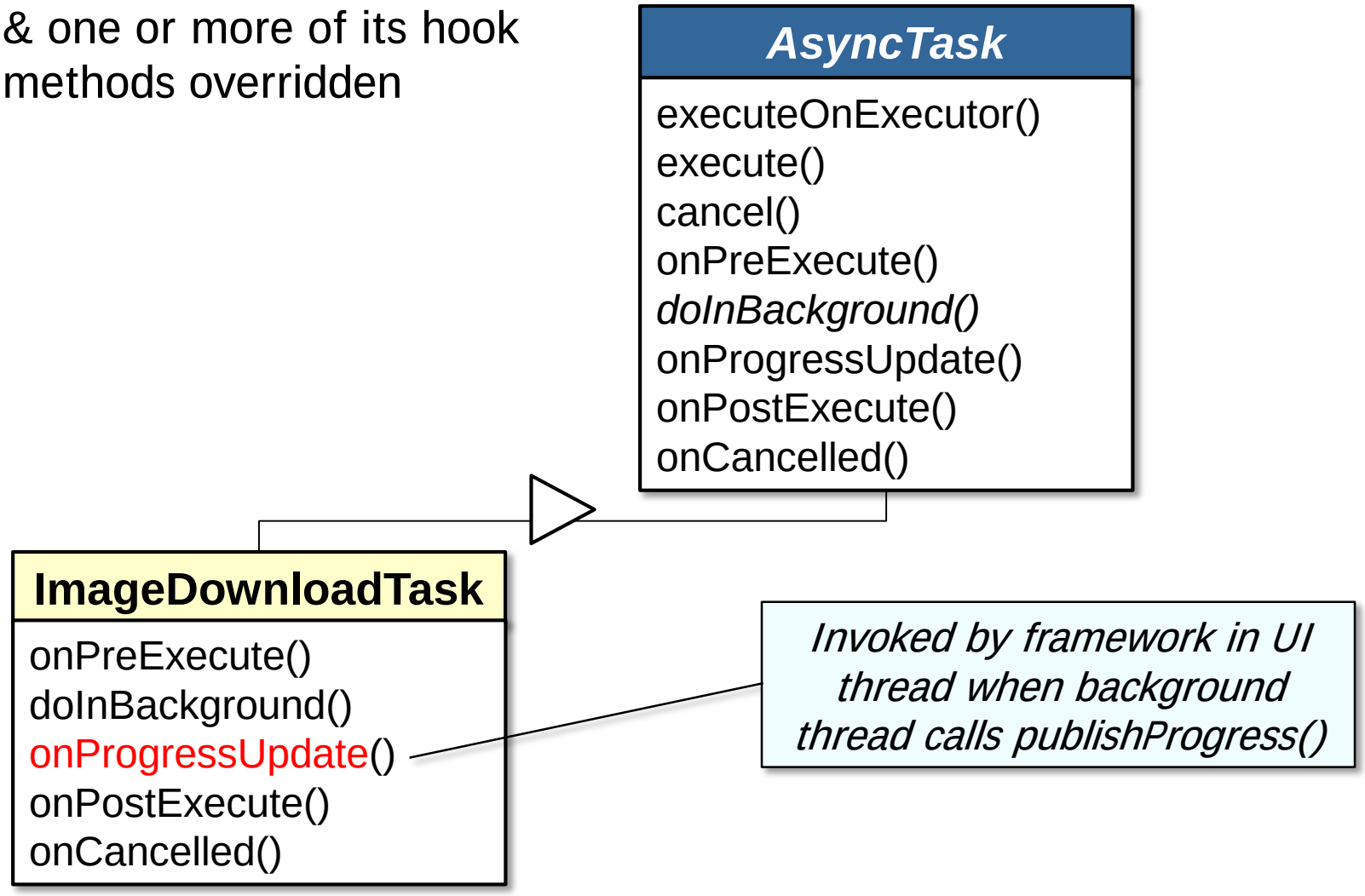
- AsyncTask must be extended & one or more of its hook methods overridden



See www.androiddesignpatterns.com/2014/01/thread-scheduling-in-android.html

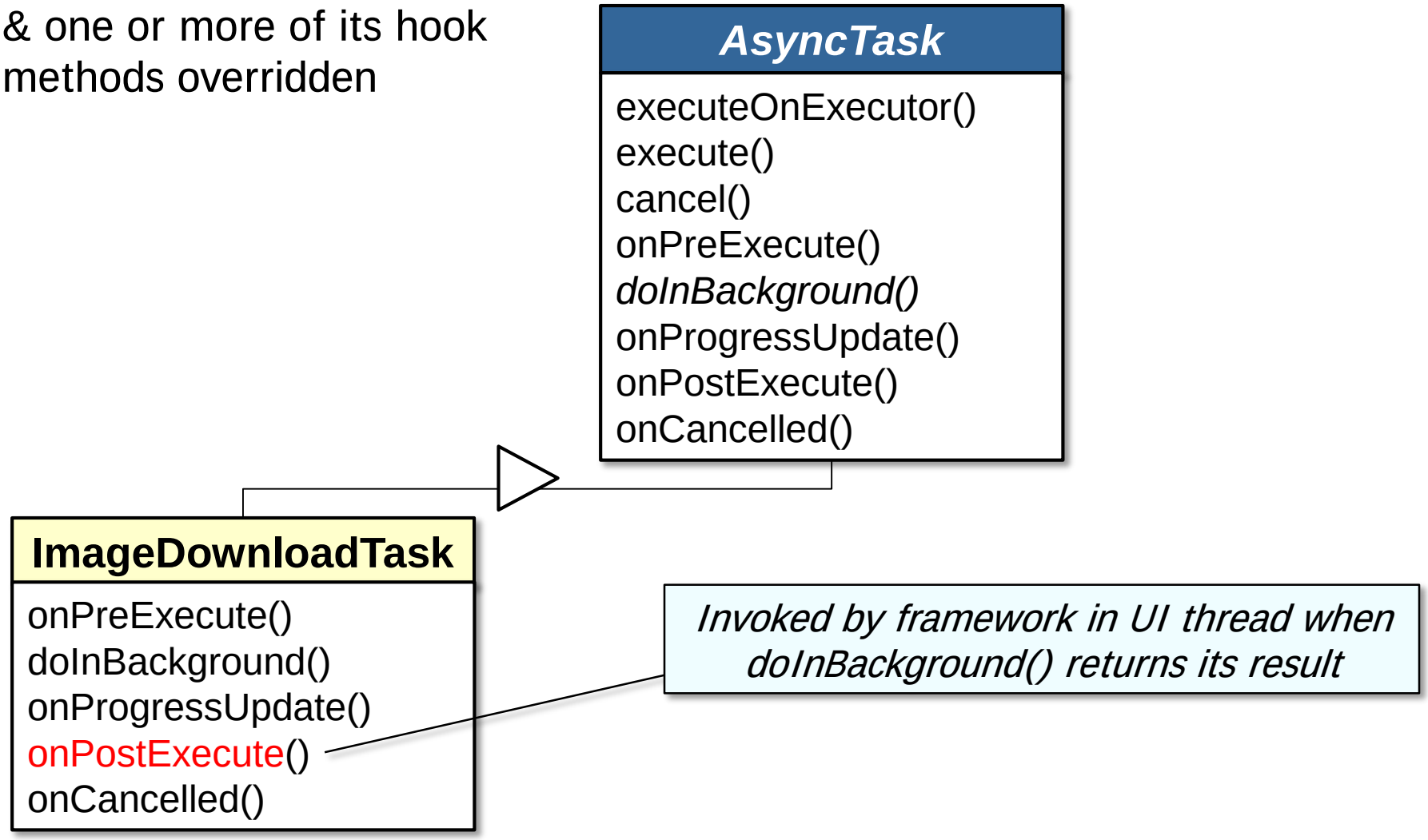
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



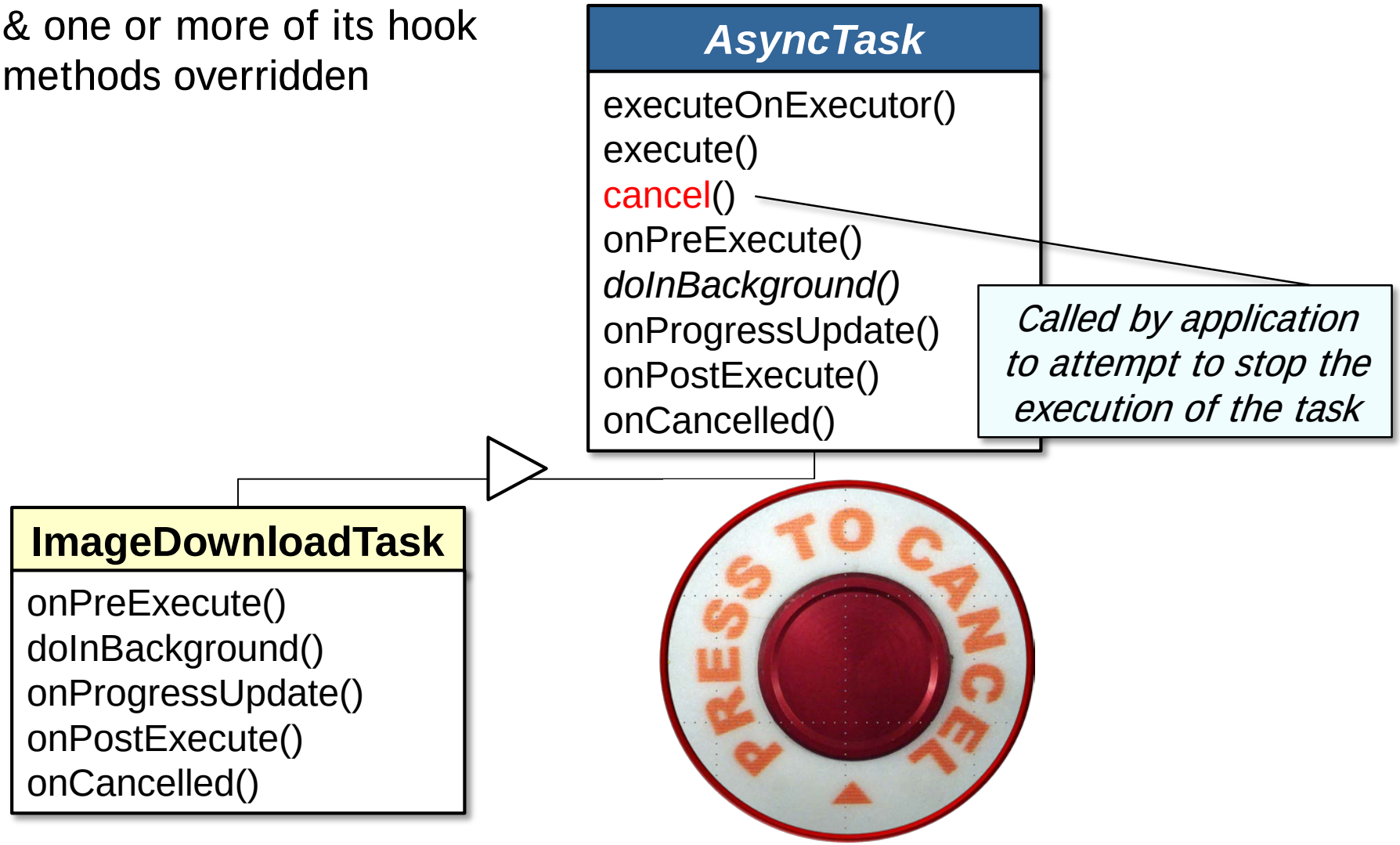
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



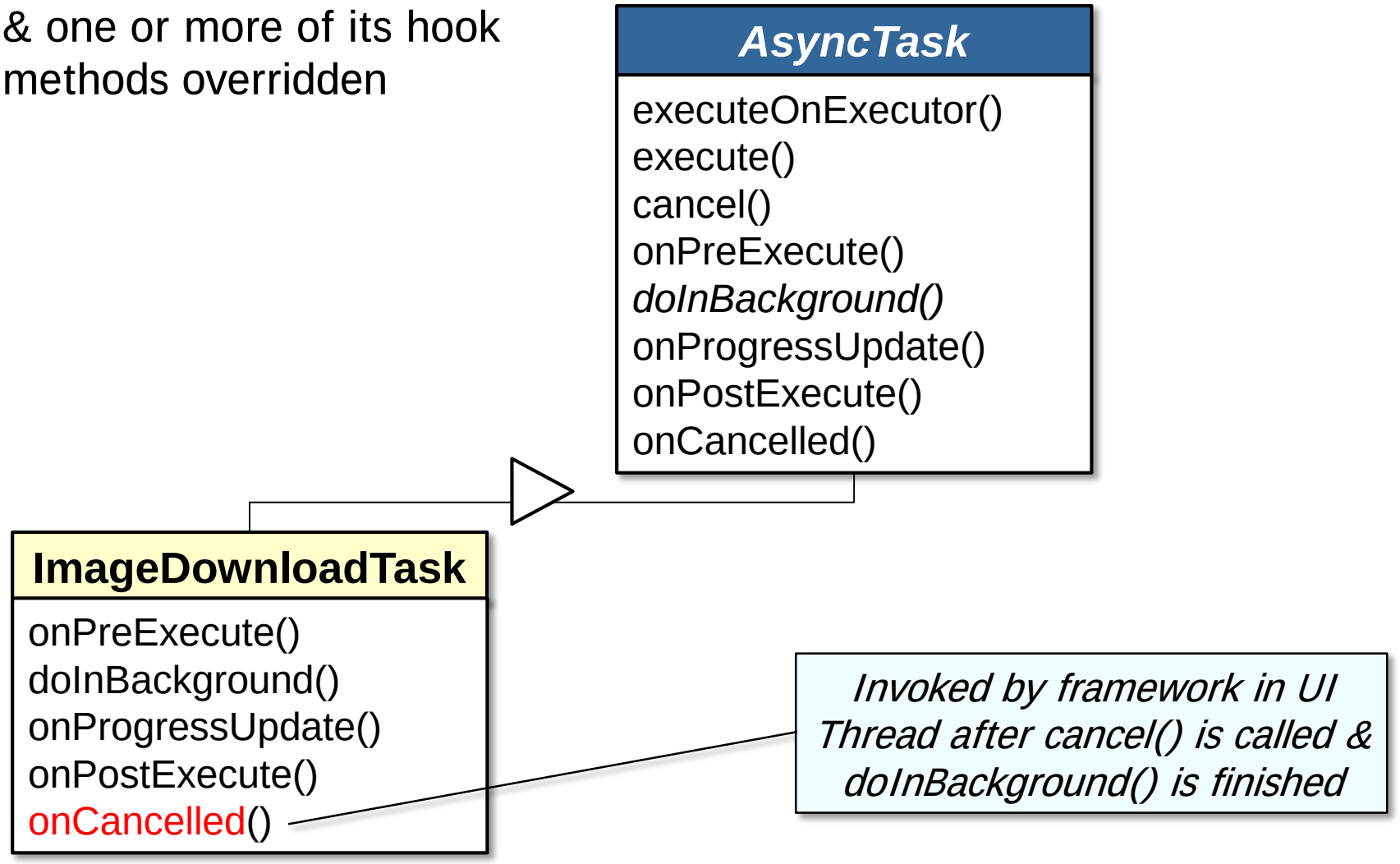
Overriding Hook Methods in the AsyncTask Class

- AsyncTask must be extended & one or more of its hook methods overridden



Overriding Hook Methods in the AsyncTask Class

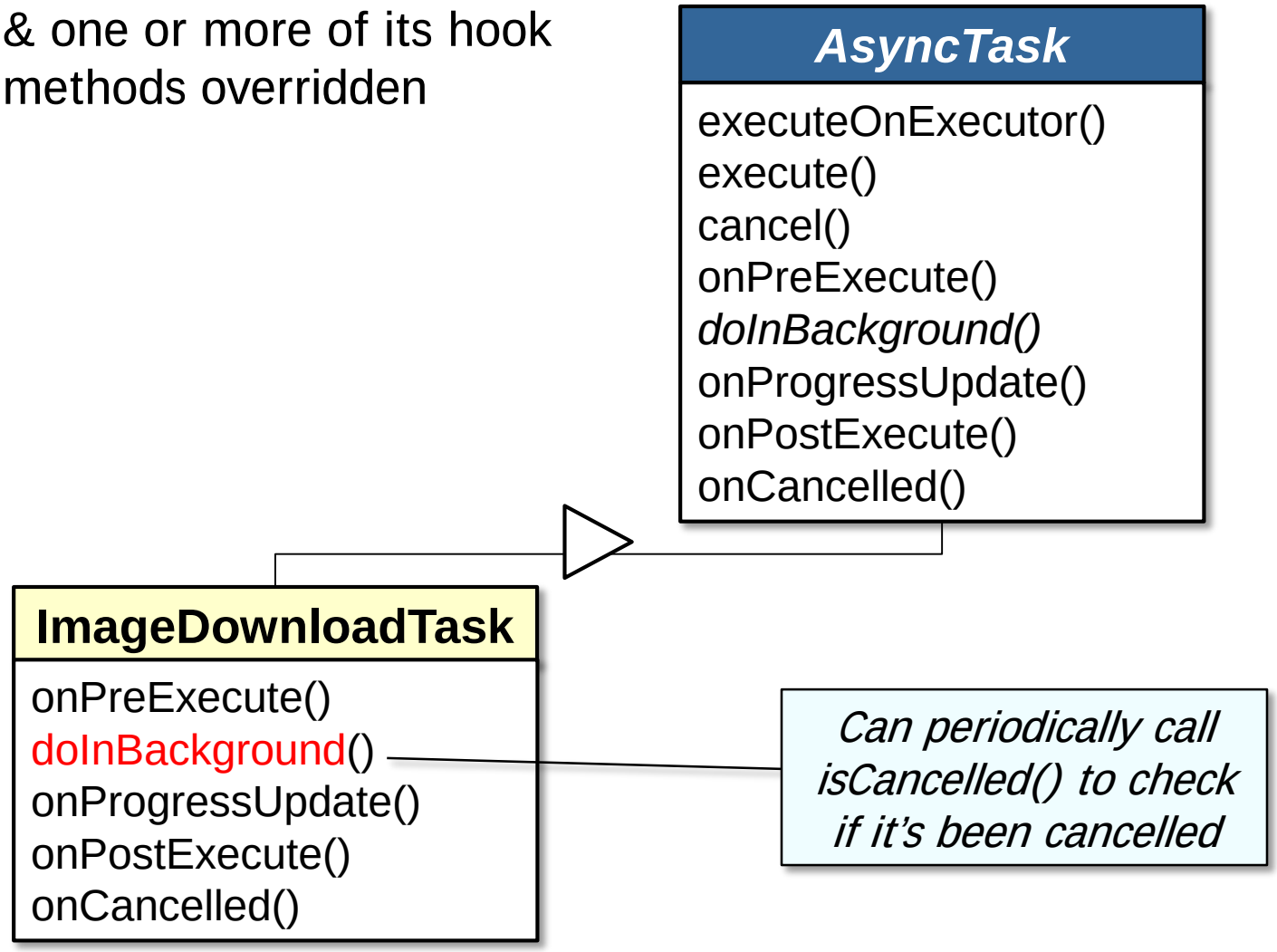
- AsyncTask must be extended & one or more of its hook methods overridden



If onCancelled() is called then onPostExecute() is *not* called & vice versa

Overriding Hook Methods in the AsyncTask Class

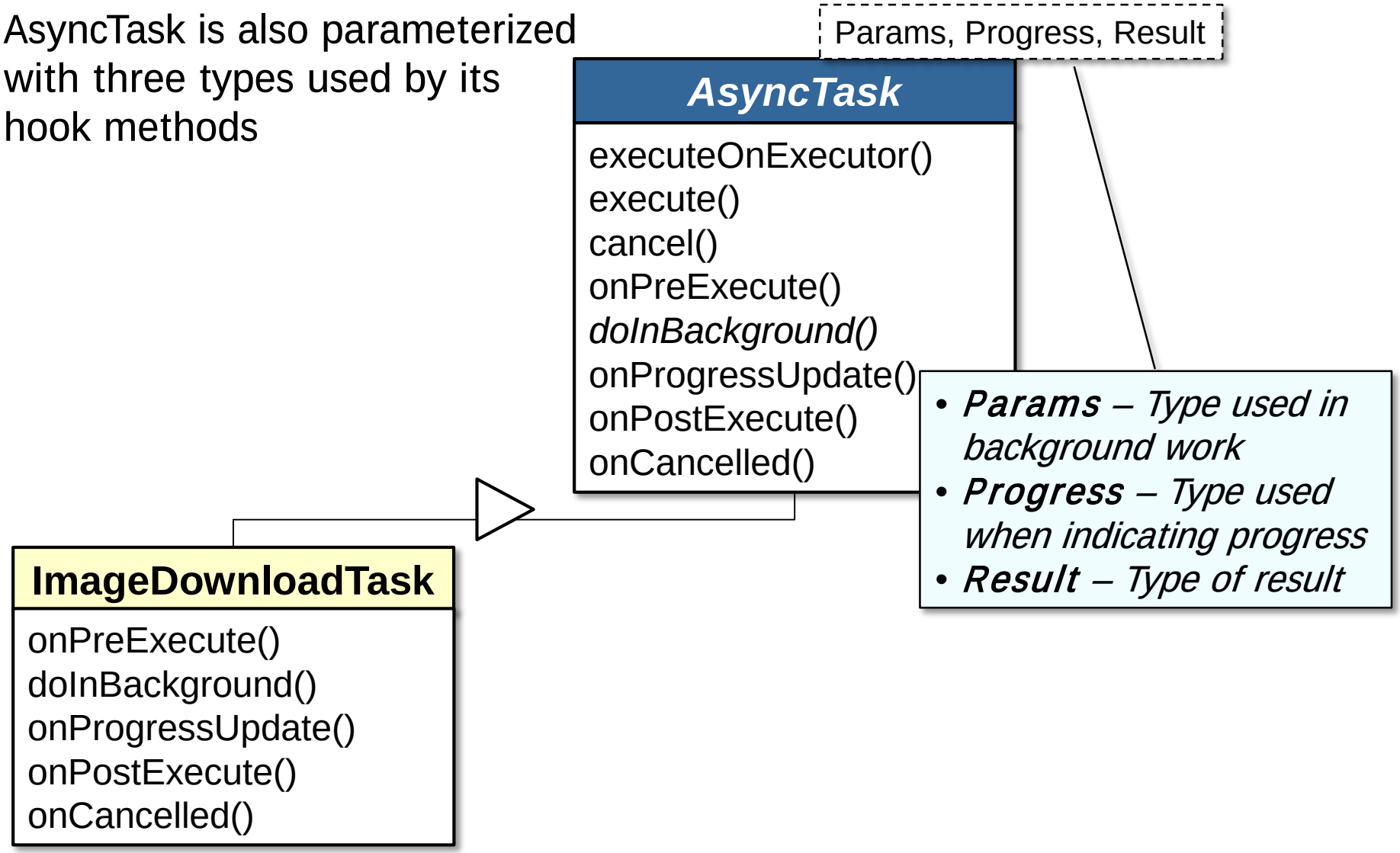
- AsyncTask must be extended & one or more of its hook methods overridden



Similar to using the Java interrupt() method to voluntarily shutdown threads

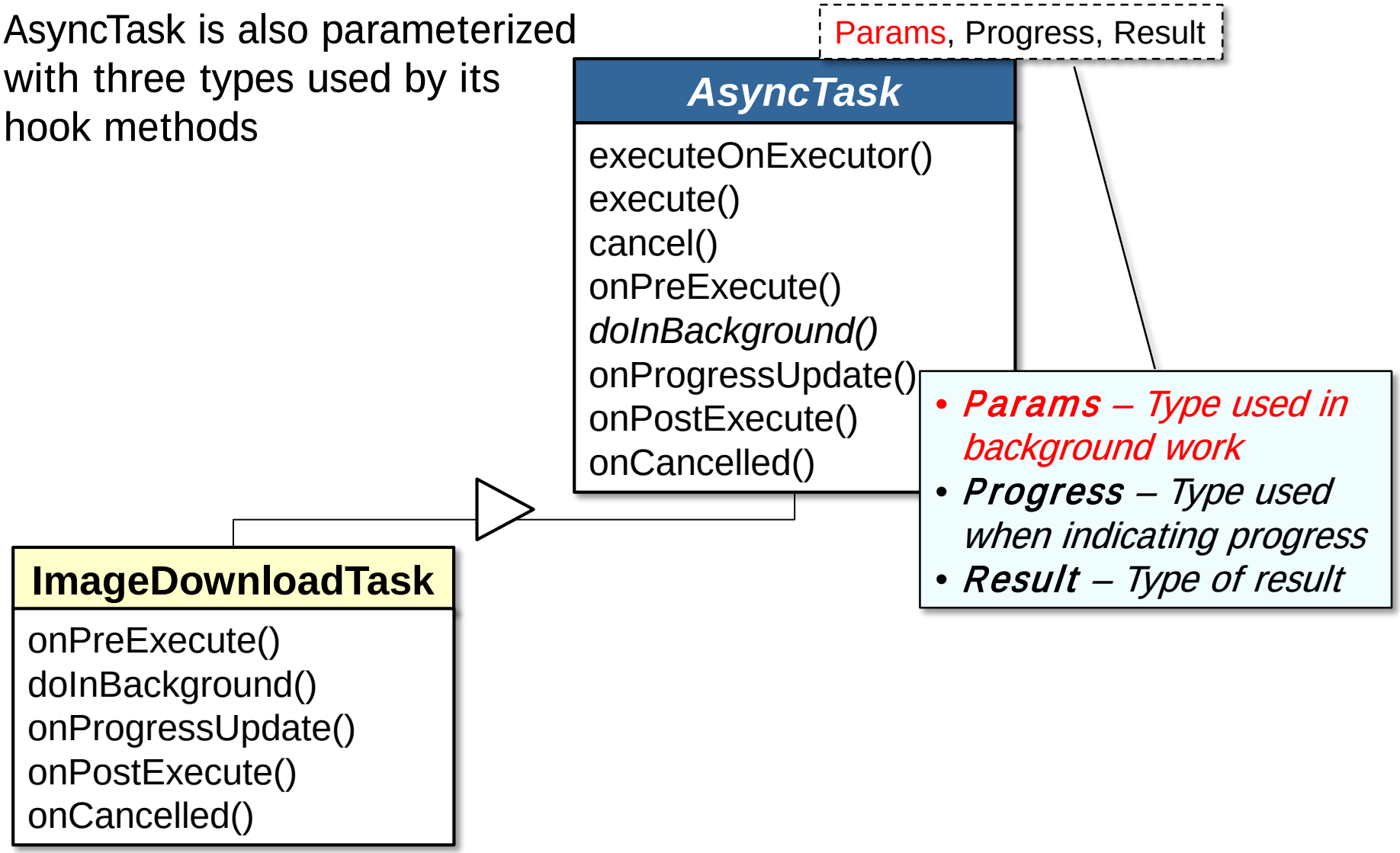
Overriding Hook Methods in the AsyncTask Class

- AsyncTask is also parameterized with three types used by its hook methods



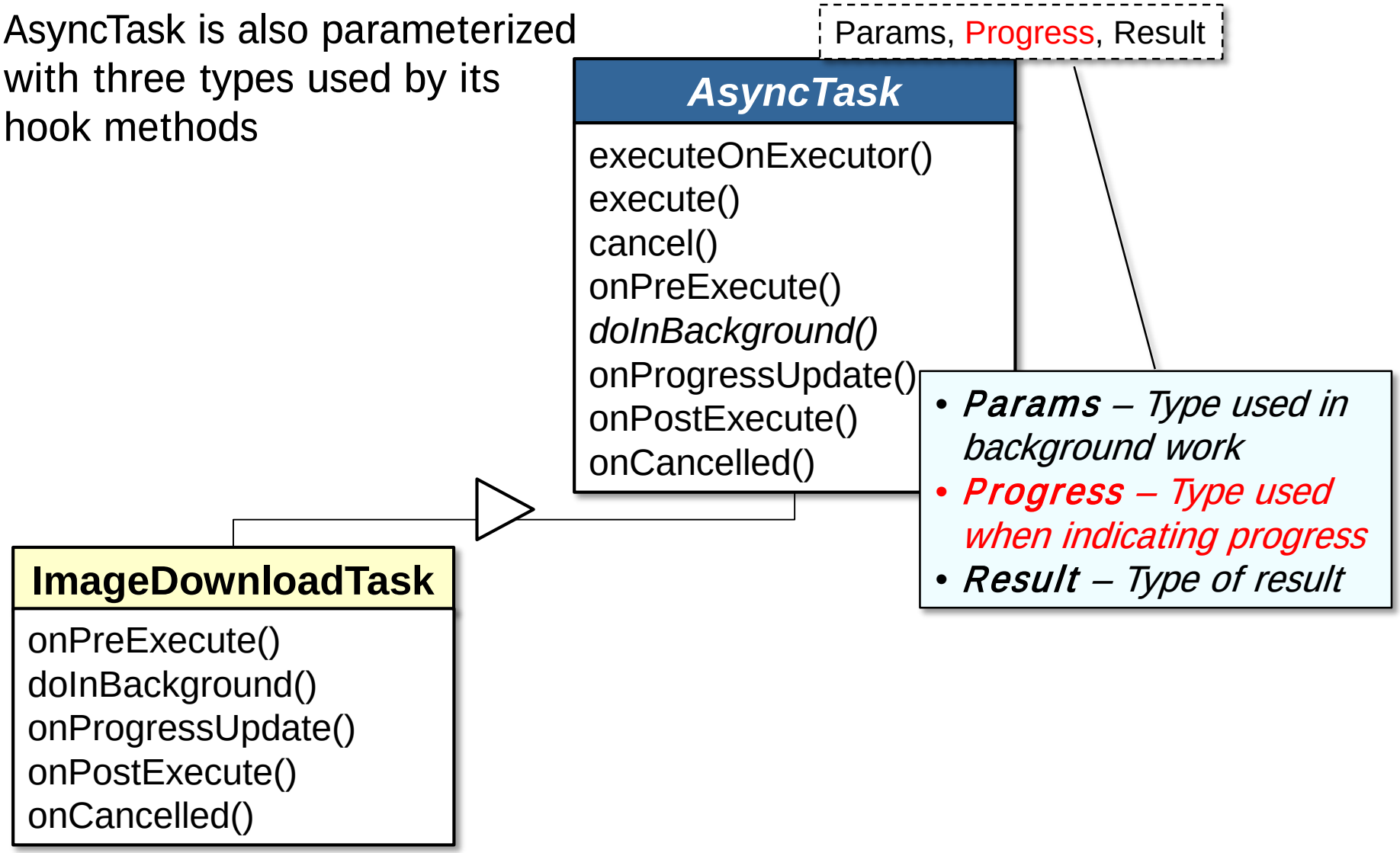
Overriding Hook Methods in the AsyncTask Class

- AsyncTask is also parameterized with three types used by its hook methods



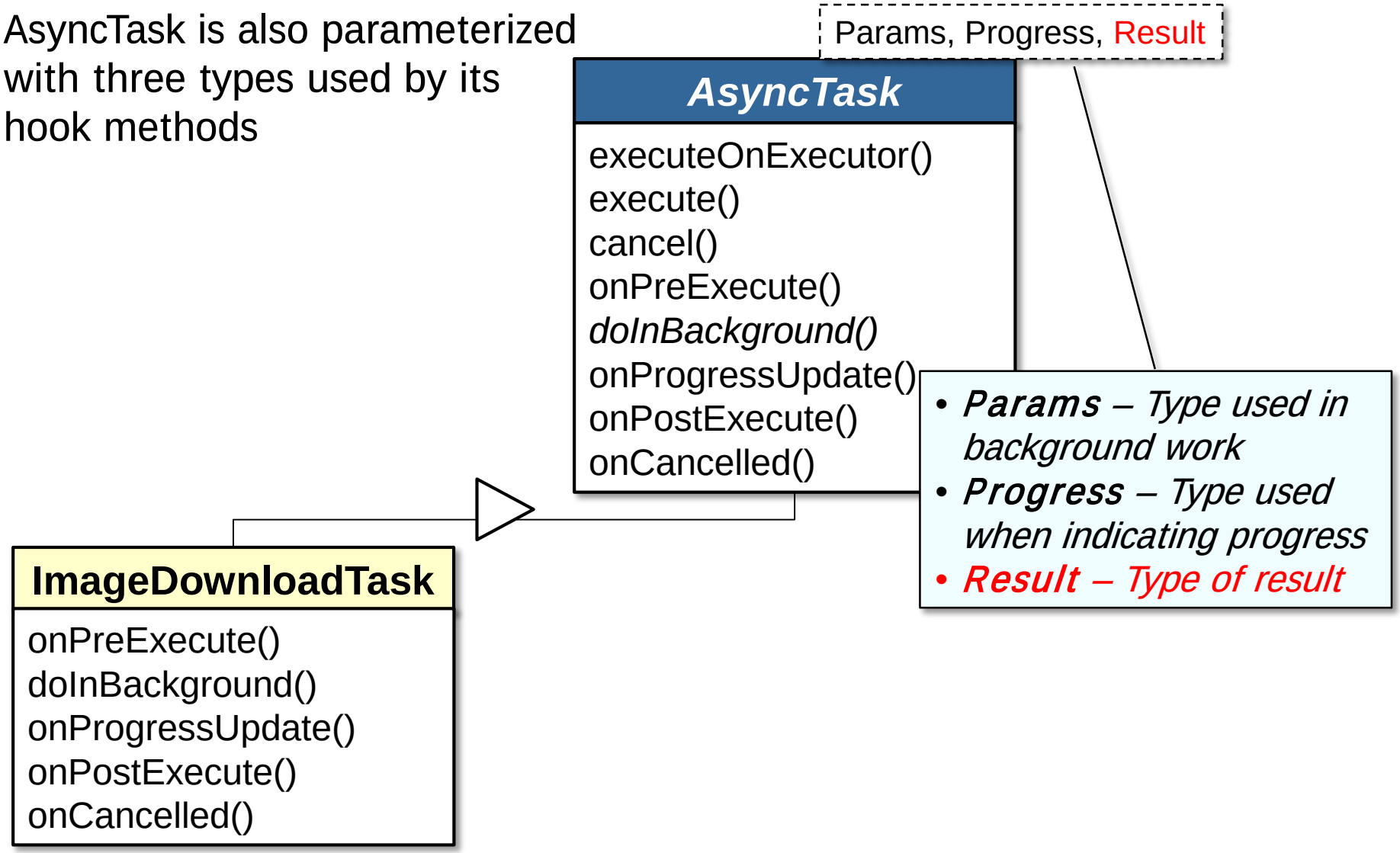
Overriding Hook Methods in the AsyncTask Class

- AsyncTask is also parameterized with three types used by its hook methods



Overriding Hook Methods in the AsyncTask Class

- AsyncTask is also parameterized with three types used by its hook methods



Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs

```
class DownloadTask extends
      AsyncTask<Uri, Integer, Long> {
  protected Long doInBackground
                          (Uri... urls)
  { /* Download files */ }

  protected void onProgressUpdate
                (Integer... progress)
  { setProgressPercent(progress[0]); }

  protected void onPostExecute
                           (Long result)
  { showDialog("Downloaded "
               + result
               + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```

See developer.android.com/reference/android/os/AsyncTask.html

Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs

Extend AsyncTask & fill in generic parameters

```
class DownloadTask extends
      AsyncTask<Uri, Integer, Long> {
  protected Long doInBackground
                          (Uri... urls)
  { /* Download files */ }

  protected void onProgressUpdate
                (Integer... progress)
  { setProgressPercent(progress[0]); }

  protected void onPostExecute
                           (Long result)
  { showDialog("Downloaded "
               + result
               + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```

Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs

```
class DownloadTask extends
      AsyncTask<Uri, Integer, Long> {
  protected Long doInBackground
                          (Uri... urls)
  { /* Download files */ }

  protected void onProgressUpdate
                (Integer... progress)
  { setProgressPercent(progress[0]); }

  protected void onPostExecute
                           (Long result)
  { showDialog("Downloaded "
                + result
                + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```

This template method initiates async processing

Overriding Hook Methods in the AsyncTask Class

- Apps must customize the AsyncTask class to meet their concurrency needs

These hook methods are dispatched by the Async Task framework in several different thread contexts

```
class DownloadTask extends
      AsyncTask<Uri, Integer, Long> {
  protected Long doInBackground
                          (Uri... urls)
  { /* Download files */ }

  protected void onProgressUpdate
                  (Integer... progress)
  { setProgressPercent(progress[0]); }

  protected void onPostExecute
                           (Long result)
  { showDialog("Downloaded "
                + result
                + " bytes"); }
}

new DownloadTask().execute(downloadURL);
```

End of Overview of the AsyncTask Framework (Part 2)