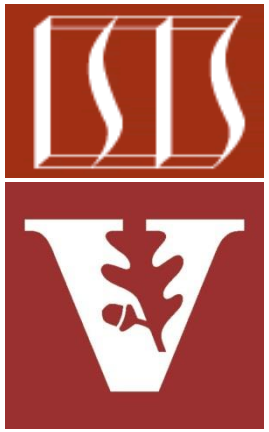
The AsyncTask Framework: Key Methods

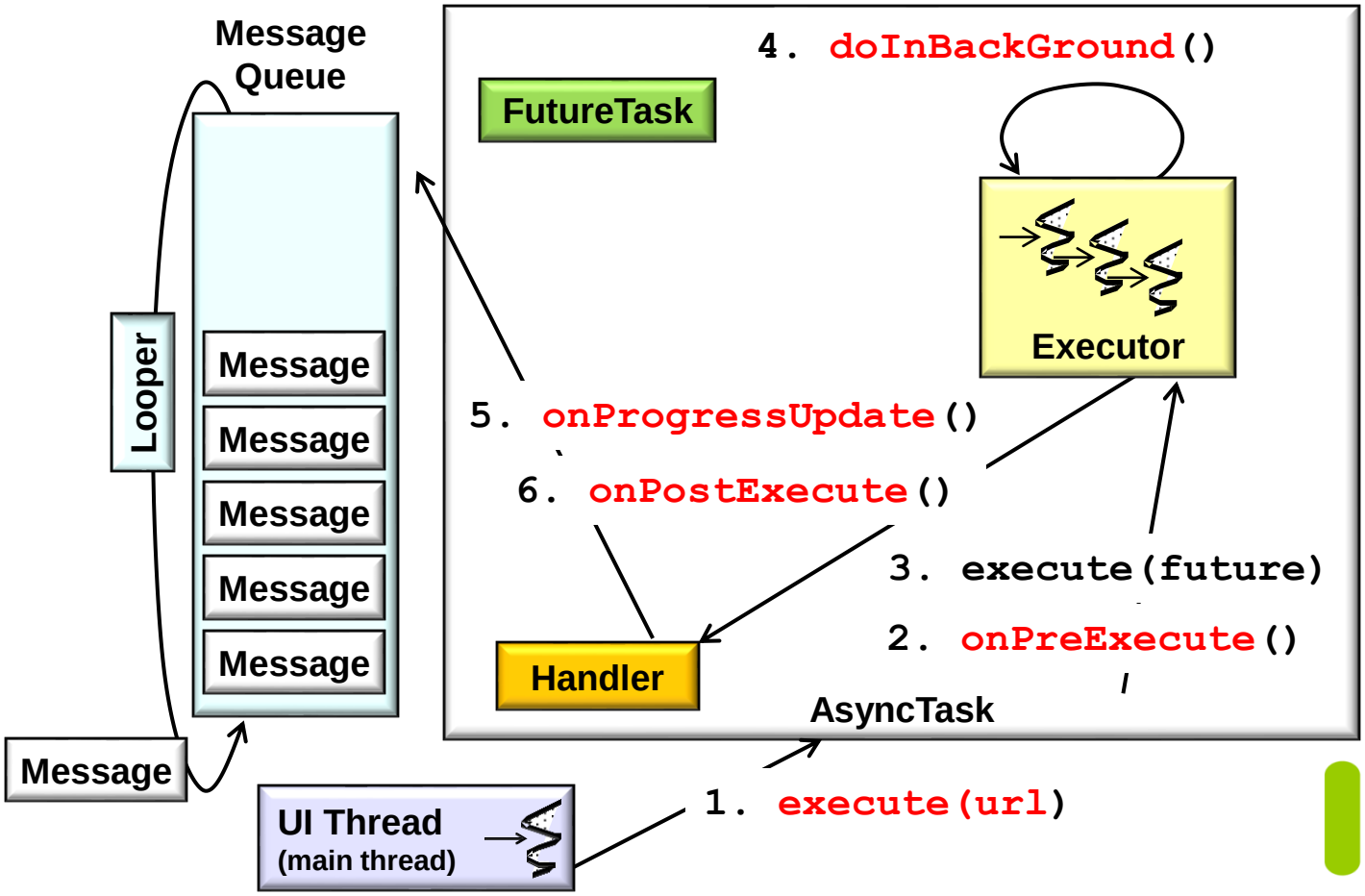


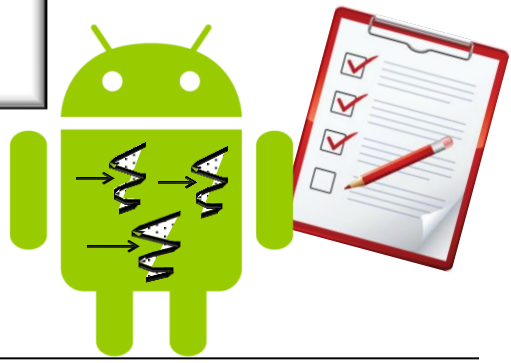
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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

```
public abstract class AsyncTask
extends Object
```

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

AsyncTask enables proper and easy use of the UI thread. This class allows you 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` package 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
    - Typically invoked by clients

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

- Execute task with specified parameters

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

- Execute task with specified parameters on 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

```
boolean isCancelled()
```

- True if task was cancelled before completing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

Run each async task one-at-a-time (serially) in a background thread within a process

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

- Execute task with specified parameters

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

- Execute task with specified parameters on 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

```
boolean isCancelled()
```

- True if task was cancelled before completing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

Runs multiple async tasks concurrently in a pool of threads within a process

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

- Execute task with specified parameters

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

- Execute task with specified parameters on 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

```
boolean isCancelled()
```

- True if task was cancelled before completing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

A simple front-end to the underlying executor

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

- Execute task with specified parameters

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

- Execute task with specified parameters on 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

```
boolean isCancelled()
```

- True if task was cancelled before completing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

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

- Execute task with specified parameters

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

- Execute task with specified parameters on specified Executor

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

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

Requires cooperation by the async task, i.e., it's voluntary

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

- Attempts to cancel execution of this task

```
boolean isCancelled()
```

- True if task was cancelled before completing

See earlier lessons on "*Managing the Thread Lifecycle*"

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
    - Typically invoked by clients

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

- Execute task with specified parameters

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

- Execute task with specified parameters on specified Executor

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

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

Called in doInBackground() to check if task should shutdown

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

- Attempts to cancel execution of this task

```
boolean isCancelled()
```

- True if task was cancelled before completing

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected methods
    - Overridden by subclasses

```
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

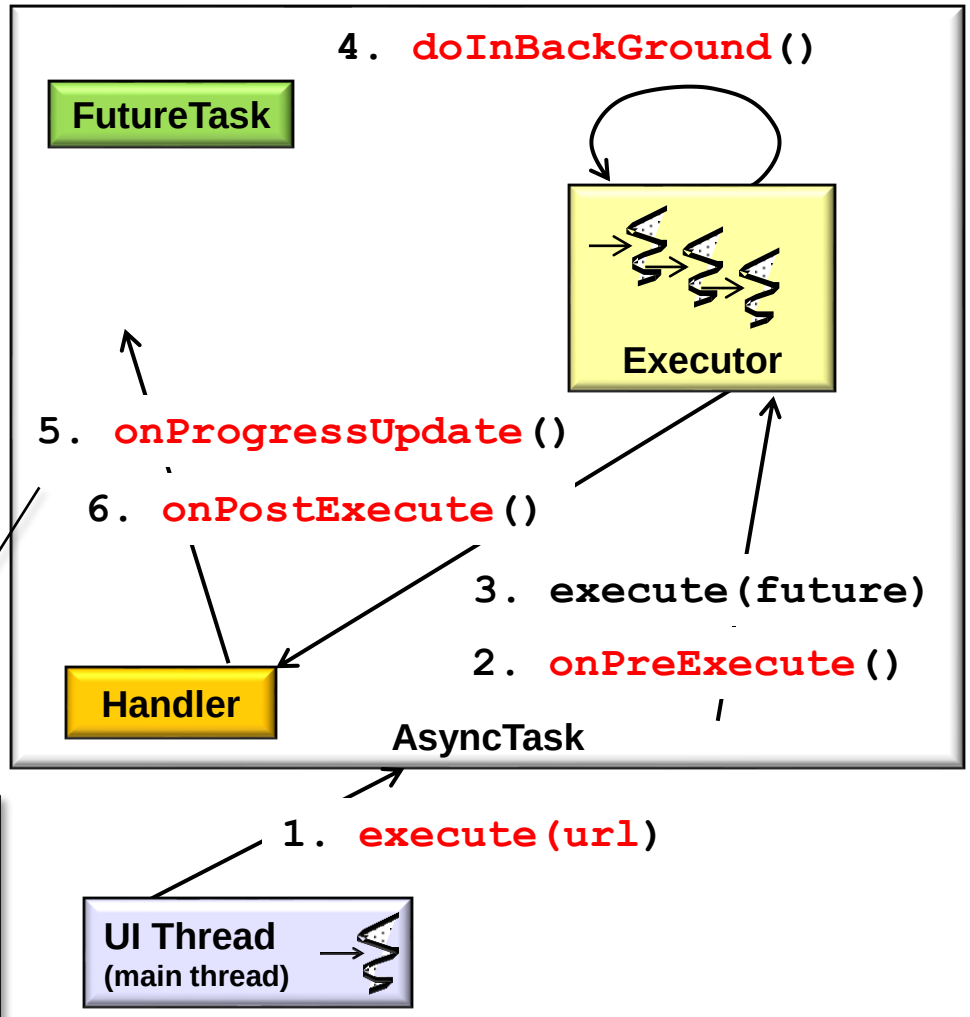
```
void onPostExecute(Result result)
```
- Runs on UI thread after doInBackground()

```
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 methods
    - Overridden by subclasses



The AsyncTask framework applies the Template Method pattern to call these methods at different points of time & in different thread contexts

See en.wikipedia.org/wiki/Template_method_pattern

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
  - Public methods
  - Protected methods
    - Overridden by subclasses

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

`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 methods
    - Overridden by subclasses

Runs in a background thread to perform the computation

`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 methods
    - Overridden by subclasses

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

`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 methods
    - Overridden by subclasses

`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 methods
    - Overridden by subclasses

Called in UI thread after background processing has been cancelled

`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 methods
    - Overridden by subclasses
    - Final methods

```
void publishProgress(Progress...
                     values)
```

- Invoked from doInBackground() to publish updates on UI thread while the background computation is still running

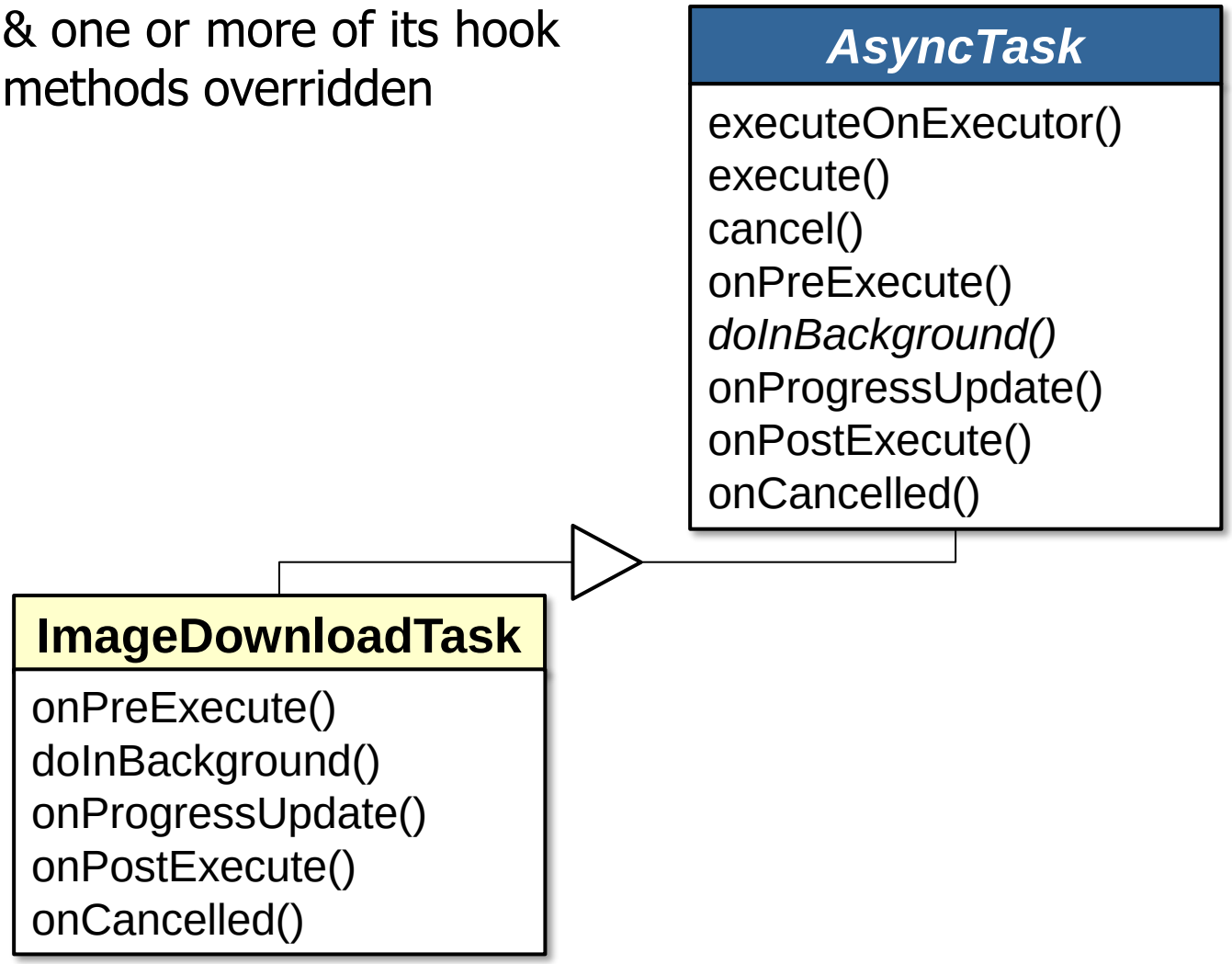
. . .

Each call to this method triggers execution of onProgressUpdate() in the UI thread

Overriding Hook Methods in the AsyncTask Class

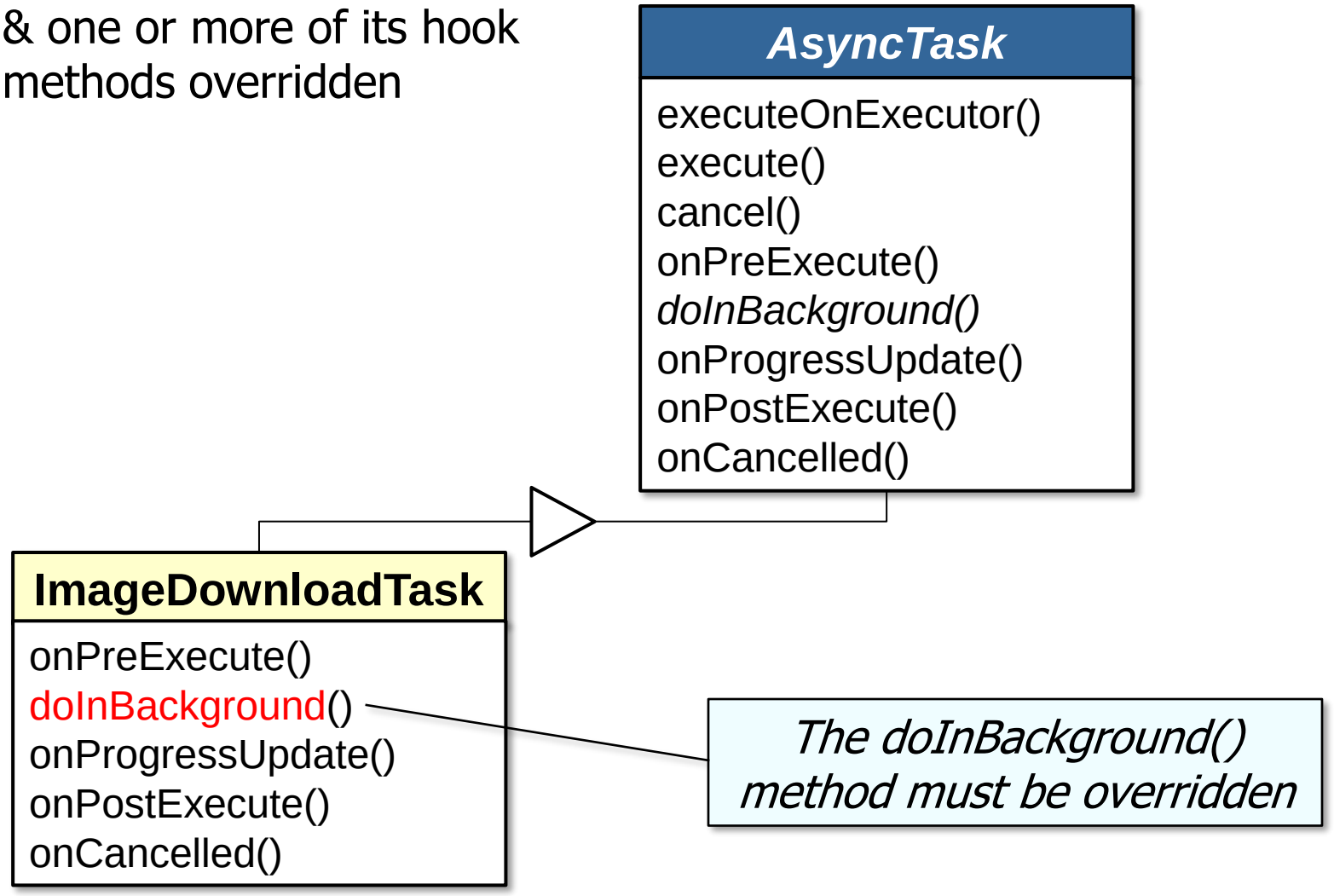
Overriding Hook Methods in the AsyncTask Class

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



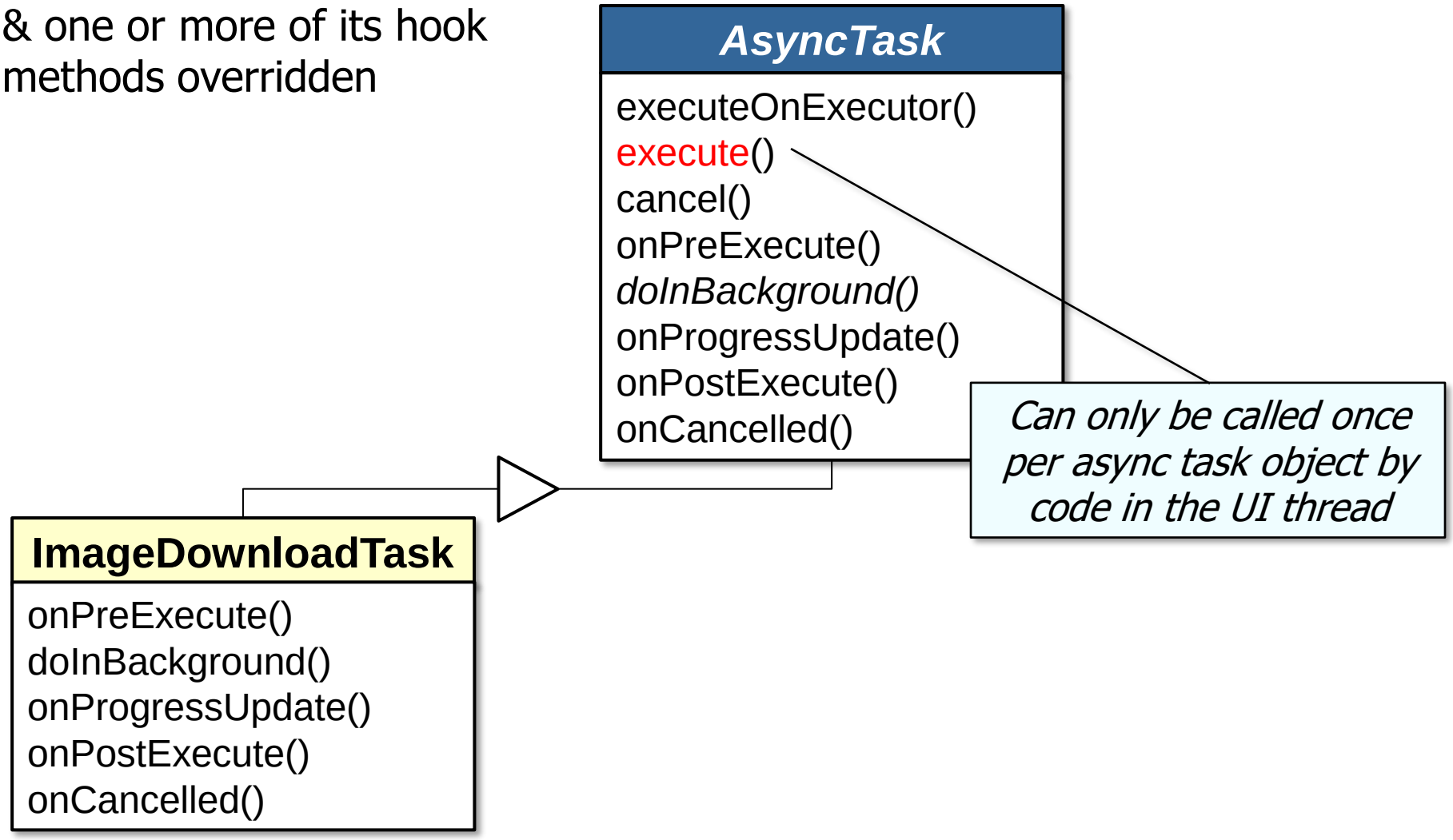
Overriding Hook Methods in the AsyncTask Class

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



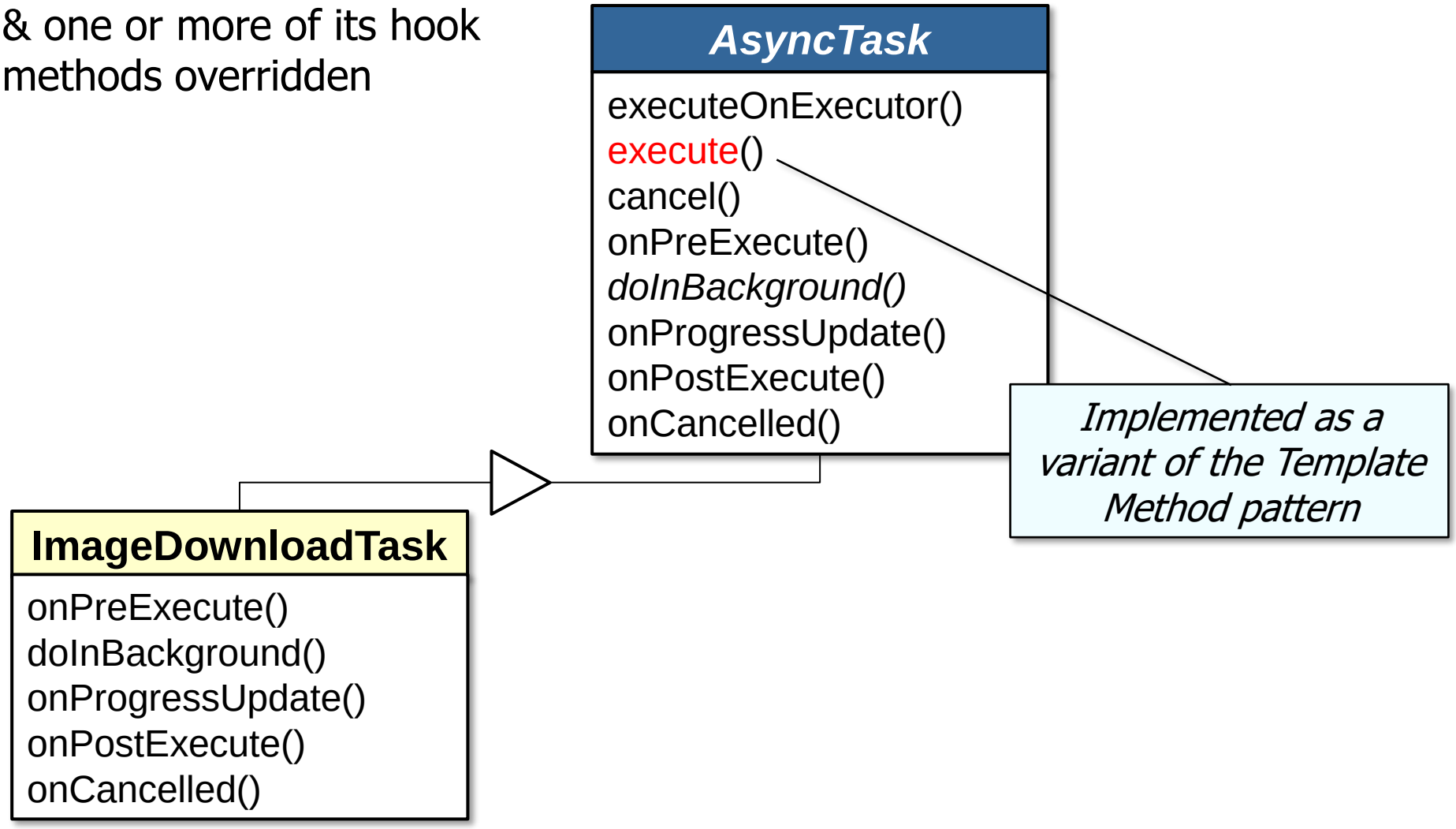
Overriding Hook Methods in the AsyncTask Class

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



Overriding Hook Methods in the AsyncTask Class

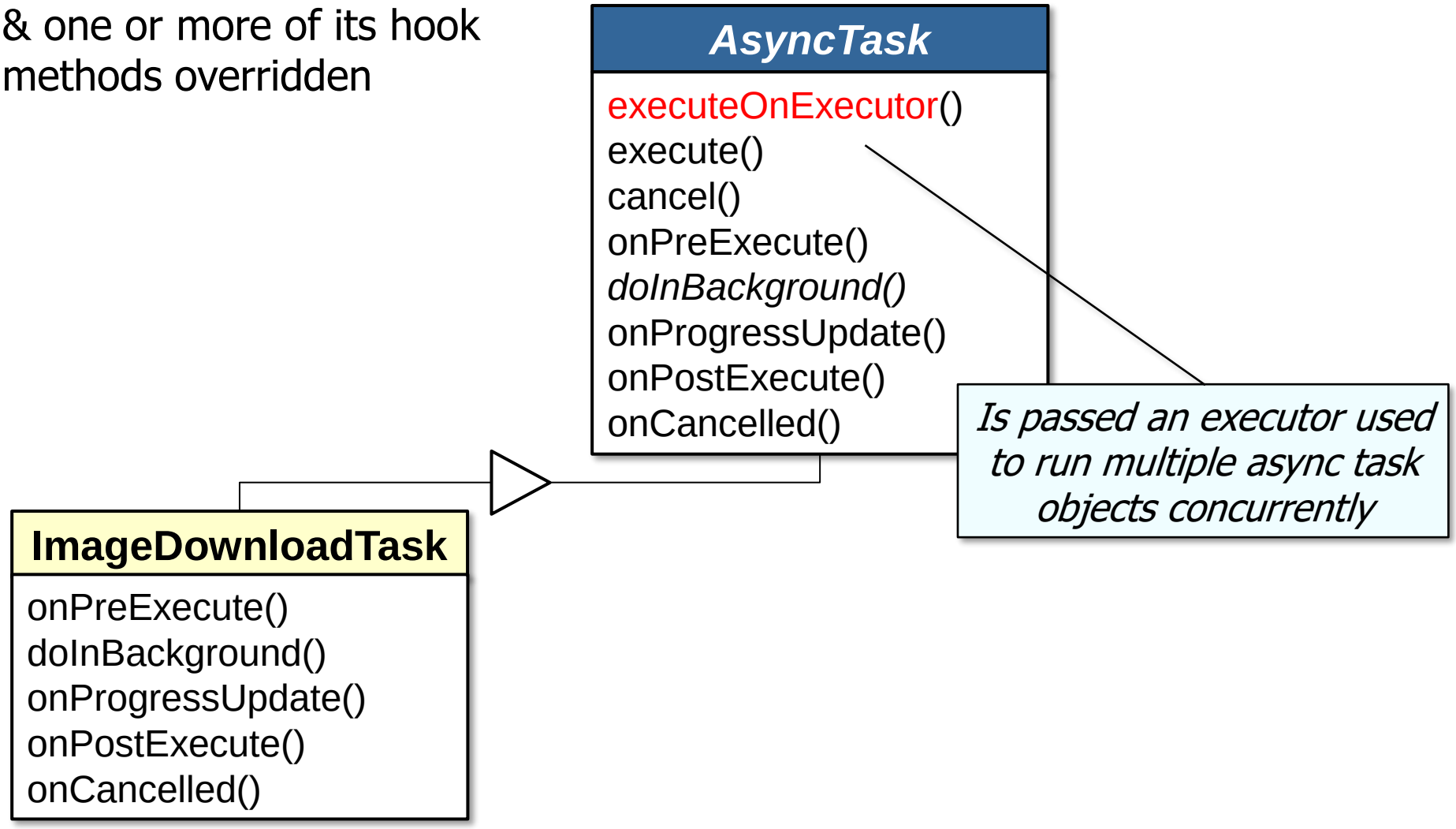
- AsyncTask must be extended & one or more of its hook methods overridden



See en.wikipedia.org/wiki/Template_method_pattern

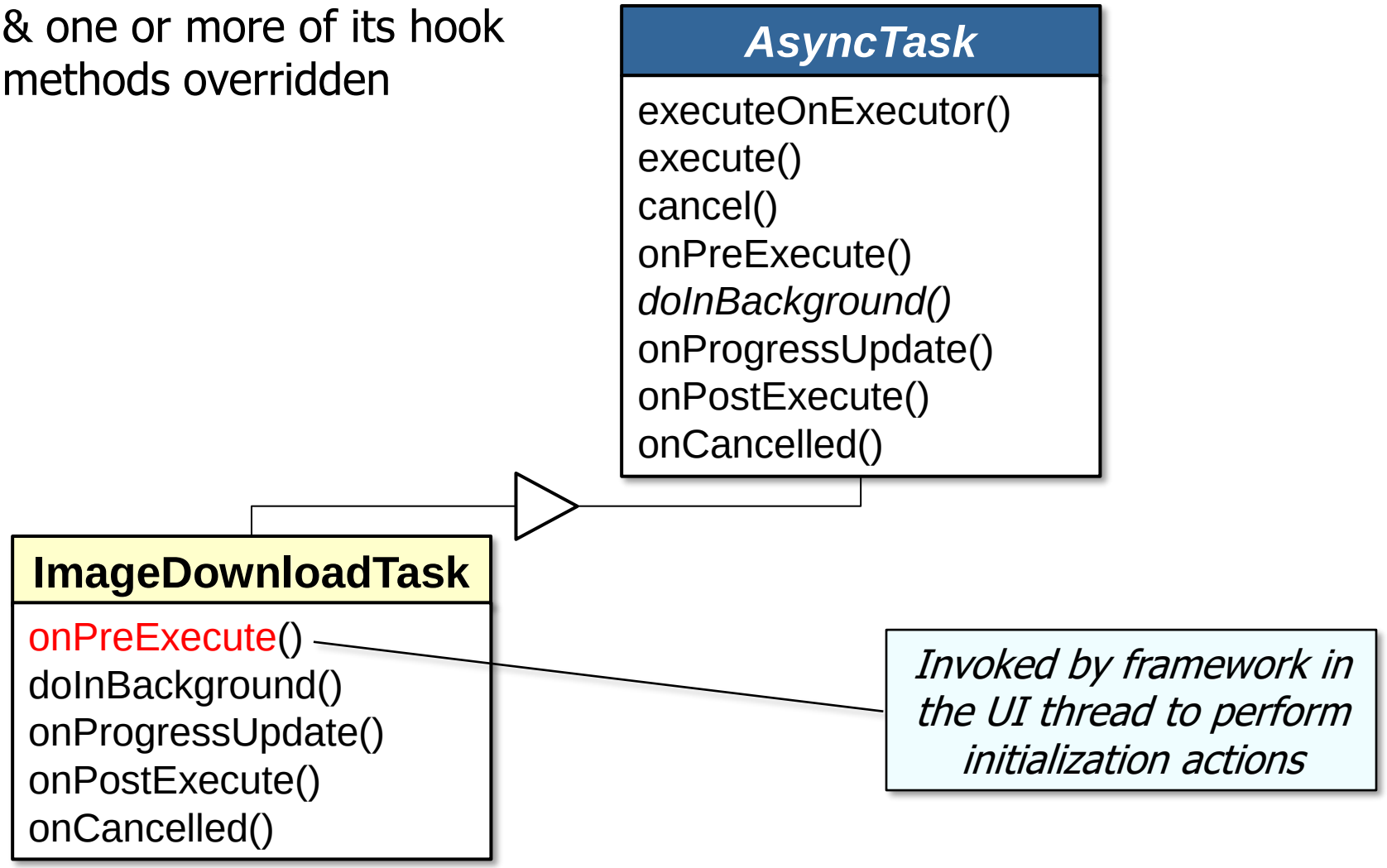
Overriding Hook Methods in the AsyncTask Class

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



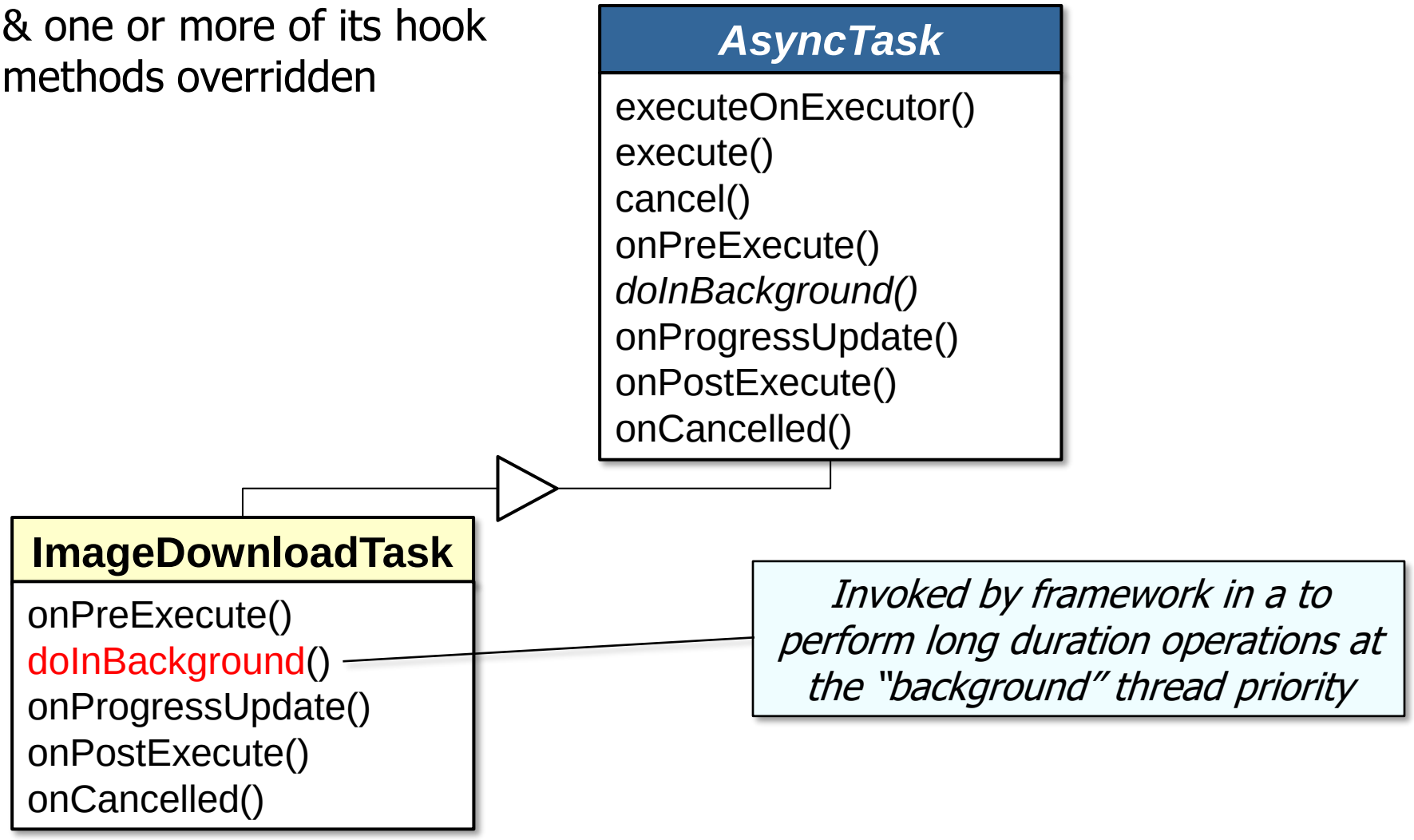
Overriding Hook Methods in the AsyncTask Class

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



Overriding Hook Methods in the AsyncTask Class

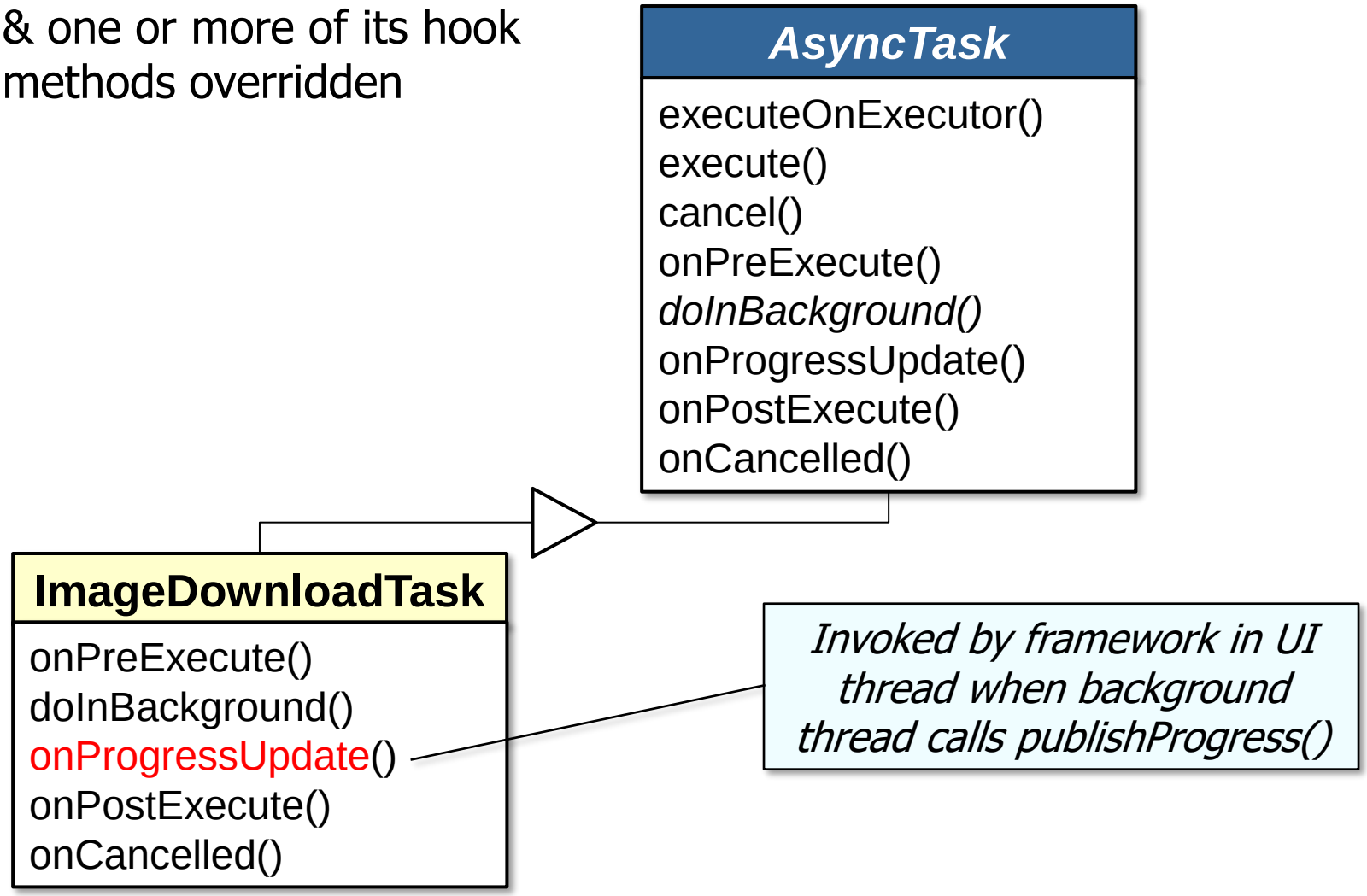
- AsyncTask must be extended & one or more of its hook methods overridden



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

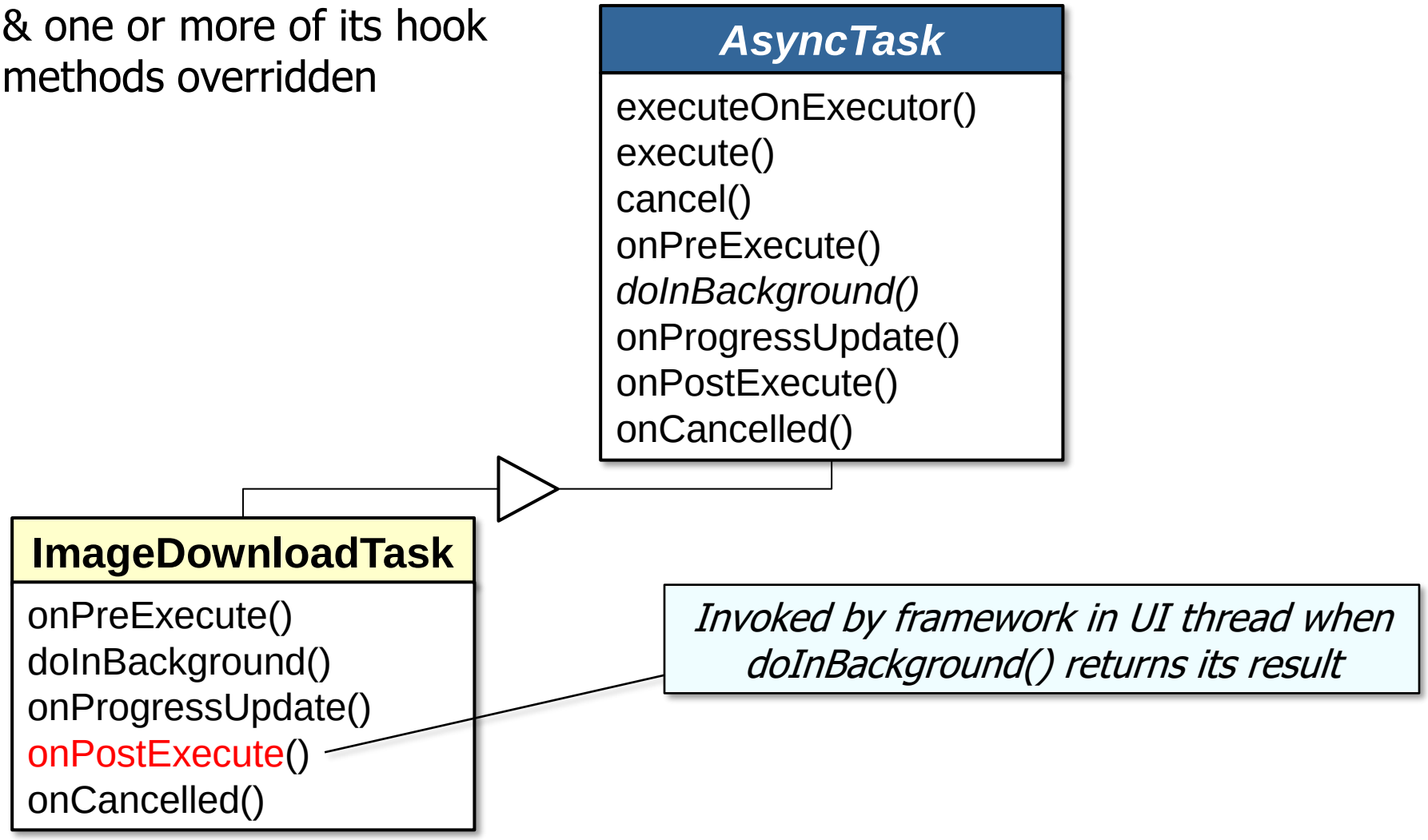
Overriding Hook Methods in the AsyncTask Class

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



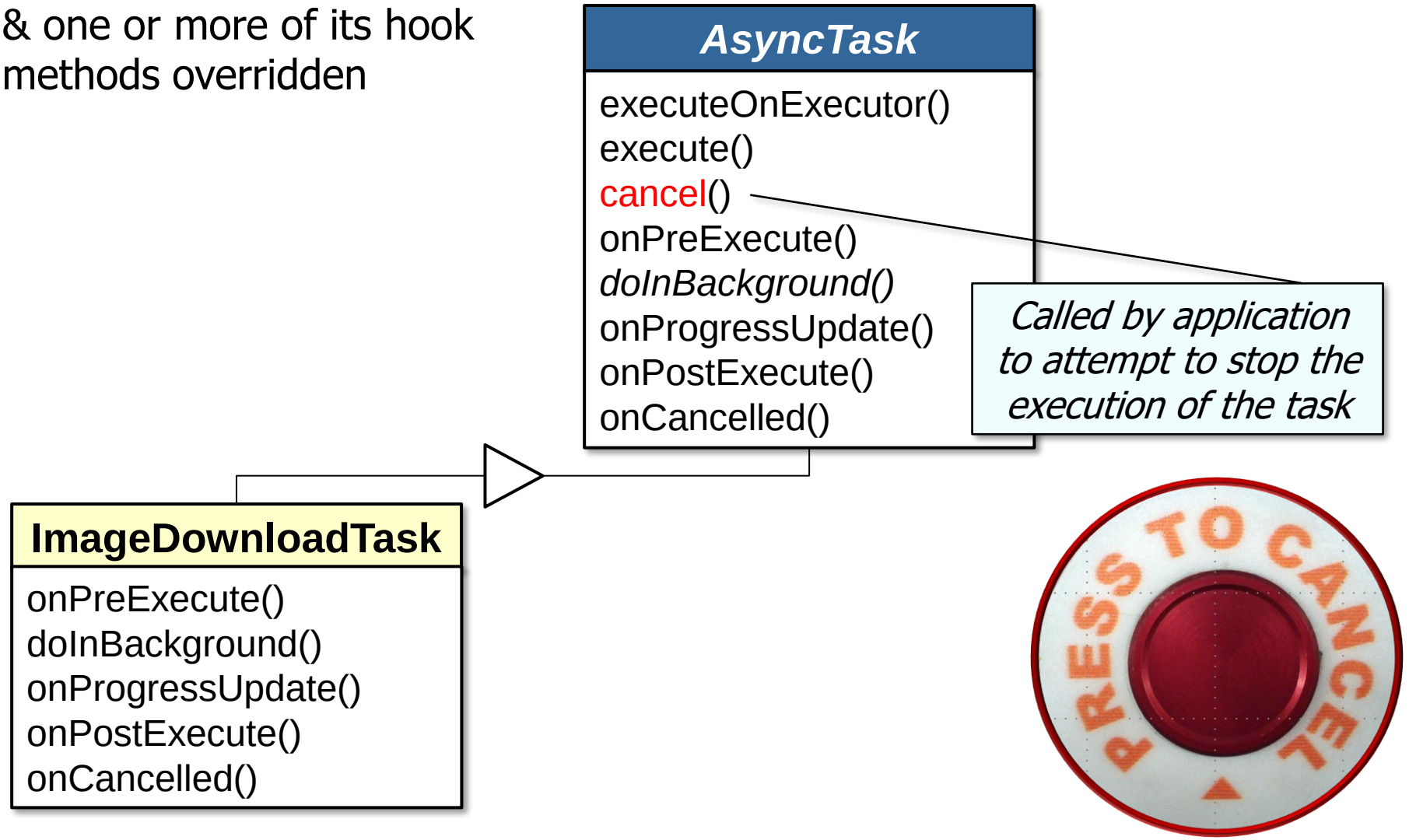
Overriding Hook Methods in the AsyncTask Class

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



Overriding Hook Methods in the AsyncTask Class

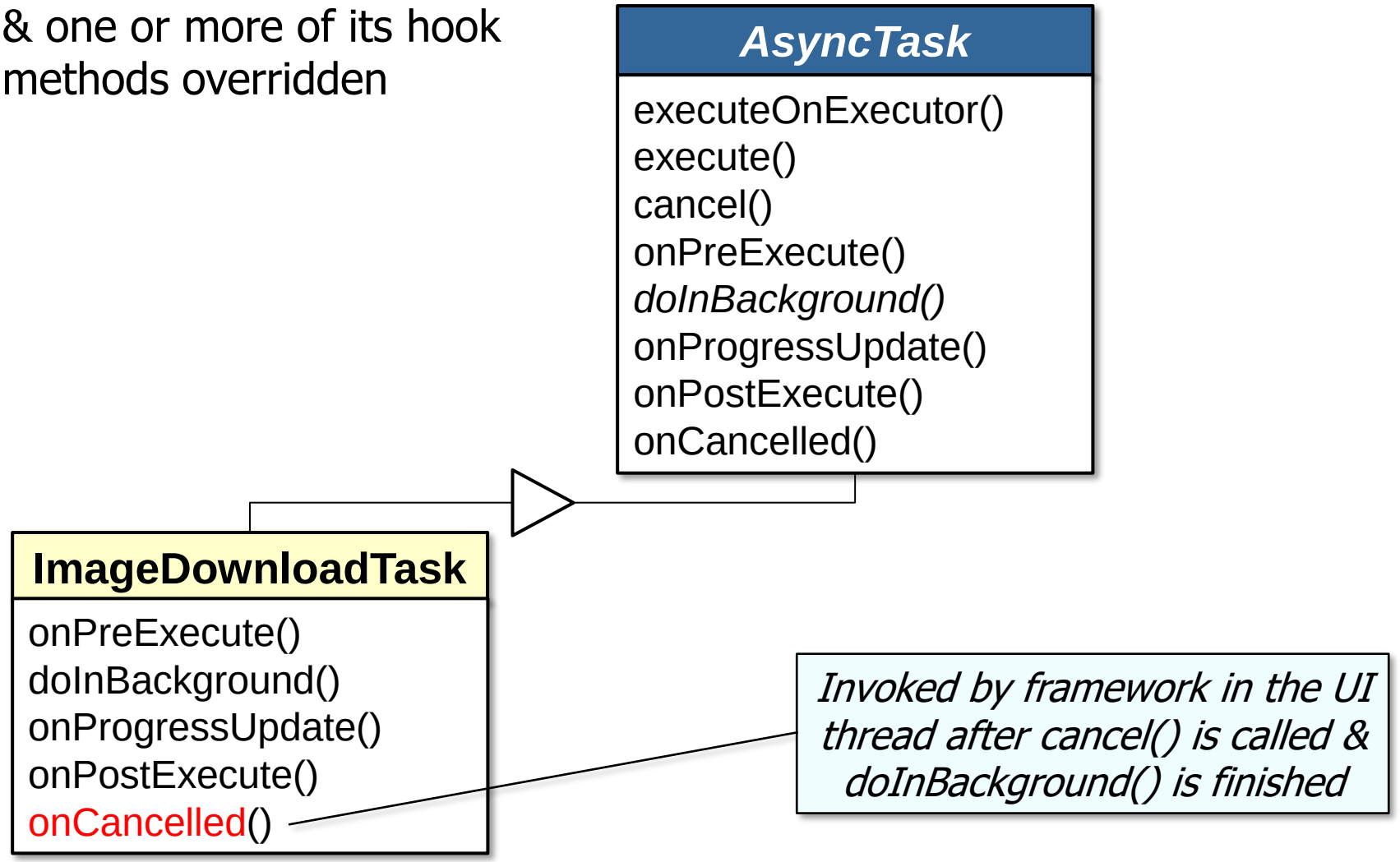
- AsyncTask must be extended & one or more of its hook methods overridden



See upcoming lessons on "*Managing the Thread Lifecycle*"

Overriding Hook Methods in the AsyncTask Class

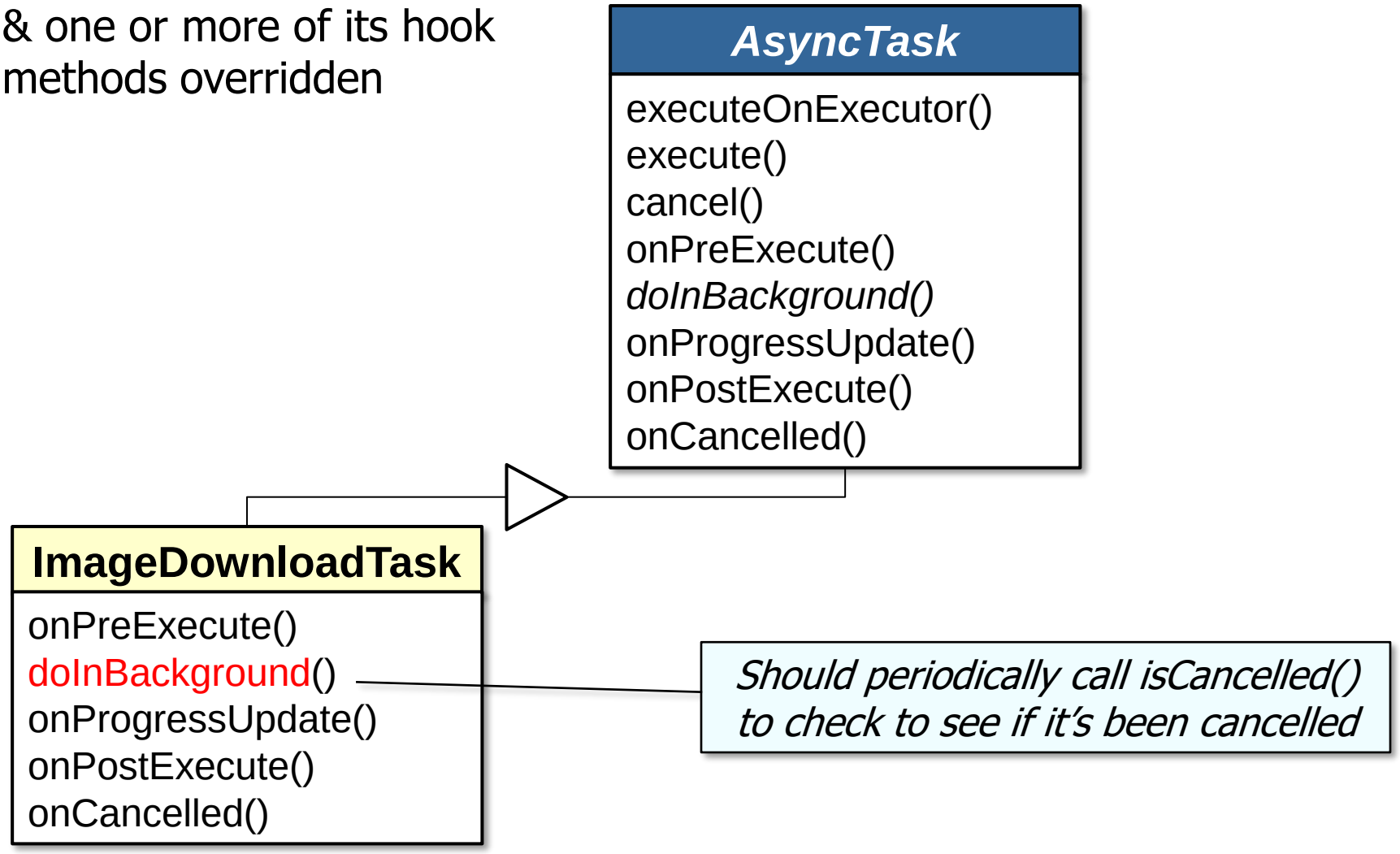
- AsyncTask must be extended & one or more of its hook methods overridden



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

Overriding Hook Methods in the AsyncTask Class

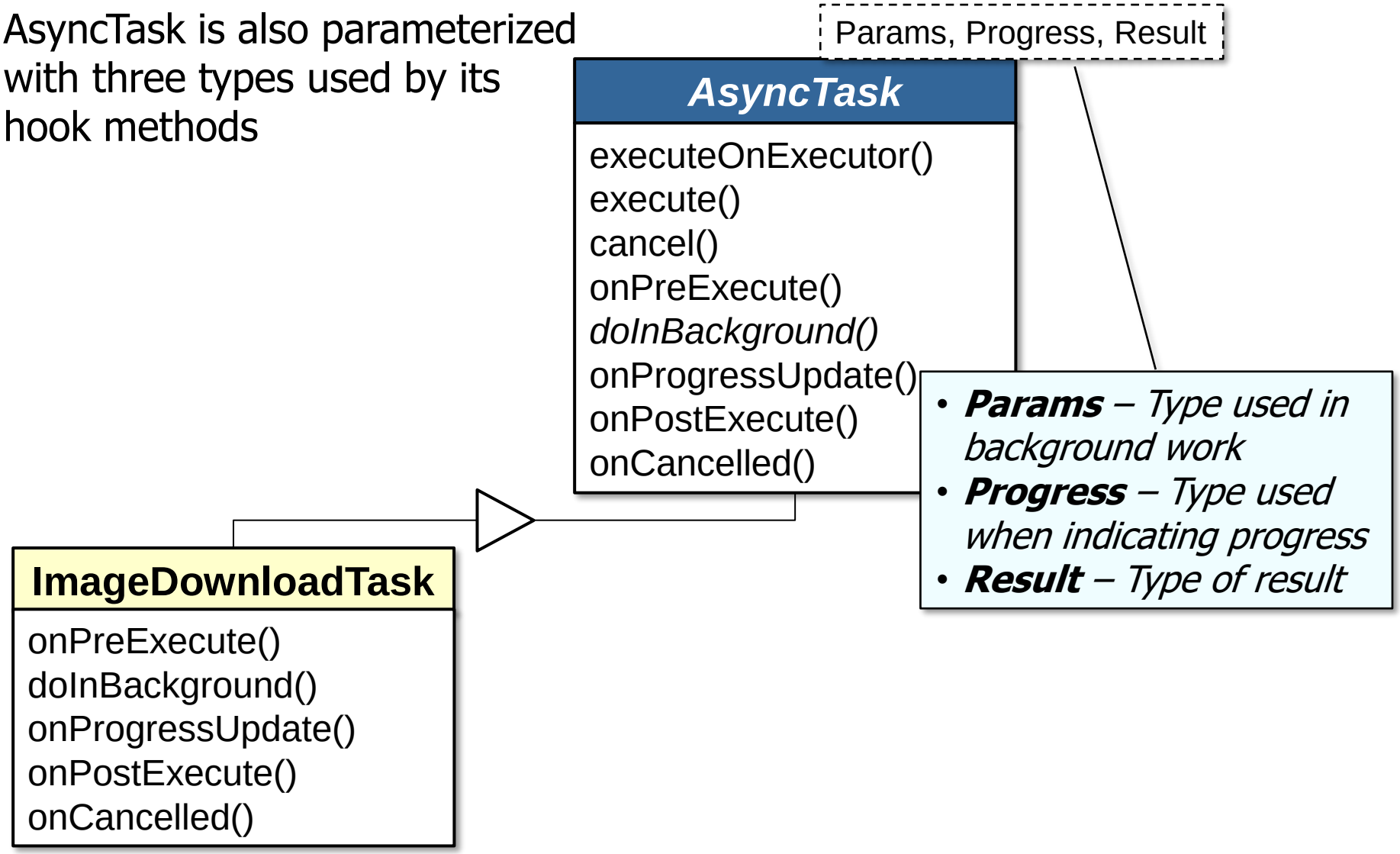
- AsyncTask must be extended & one or more of its hook methods overridden



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

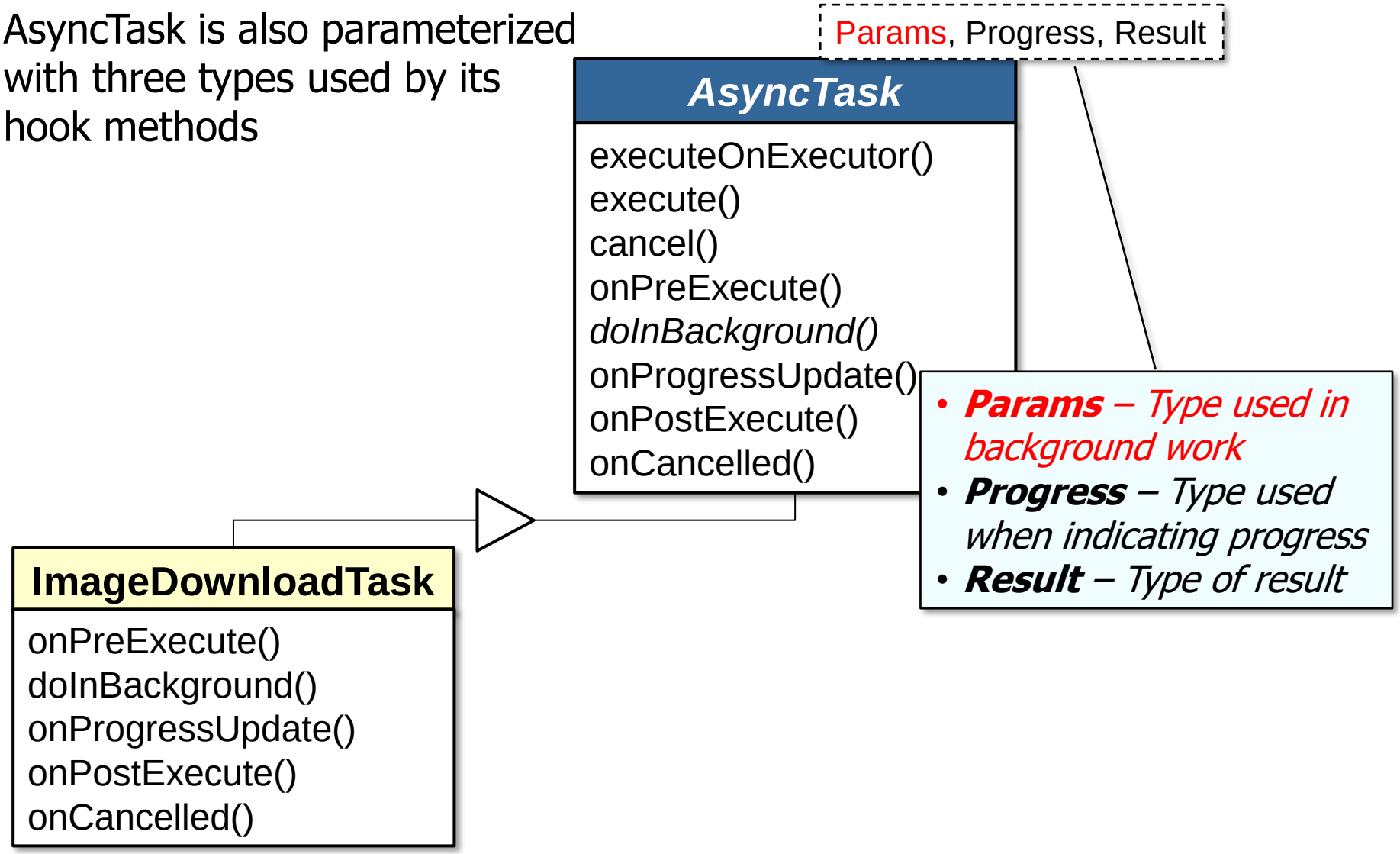
Overriding Hook Methods in the AsyncTask Class

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



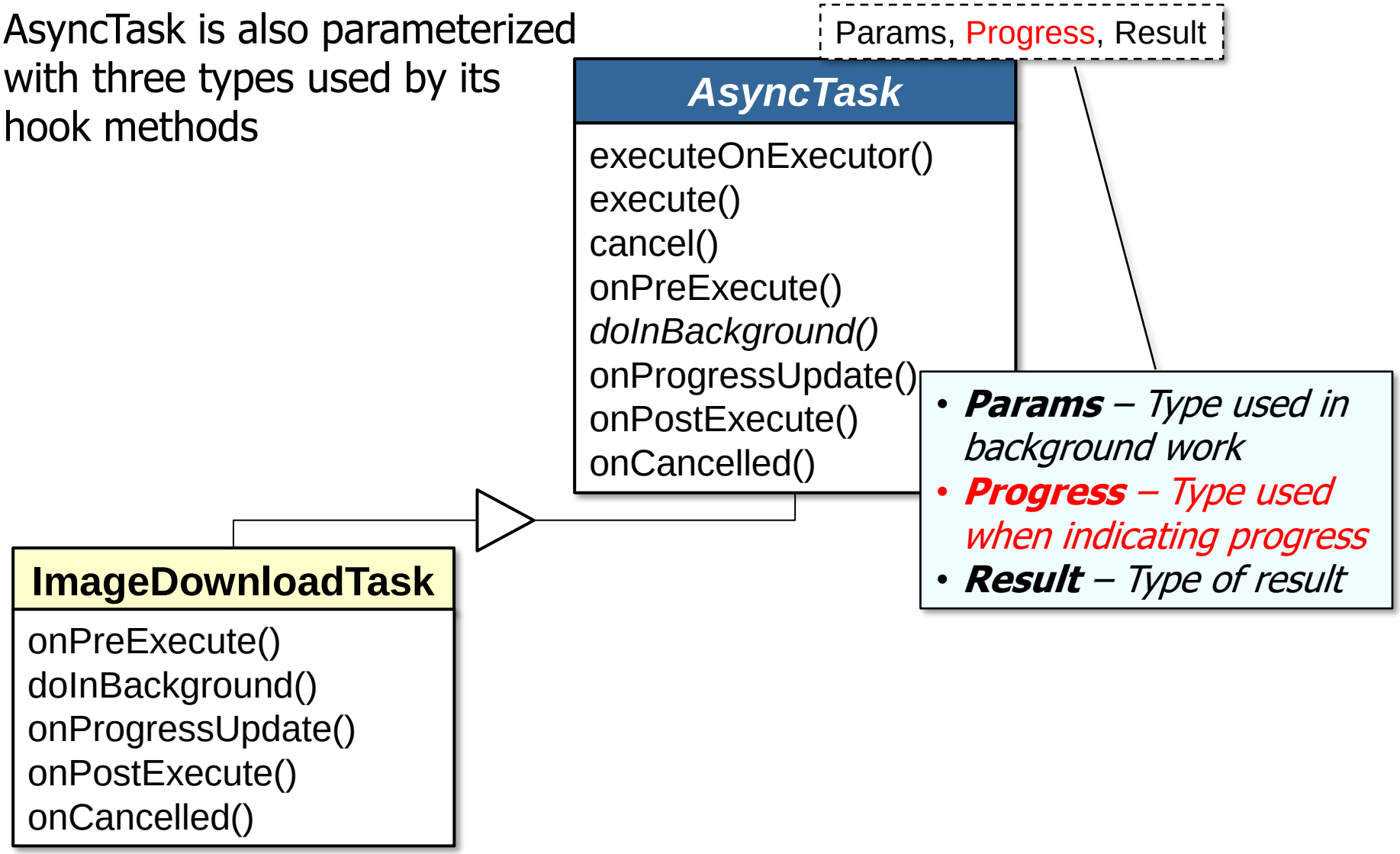
Overriding Hook Methods in the AsyncTask Class

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



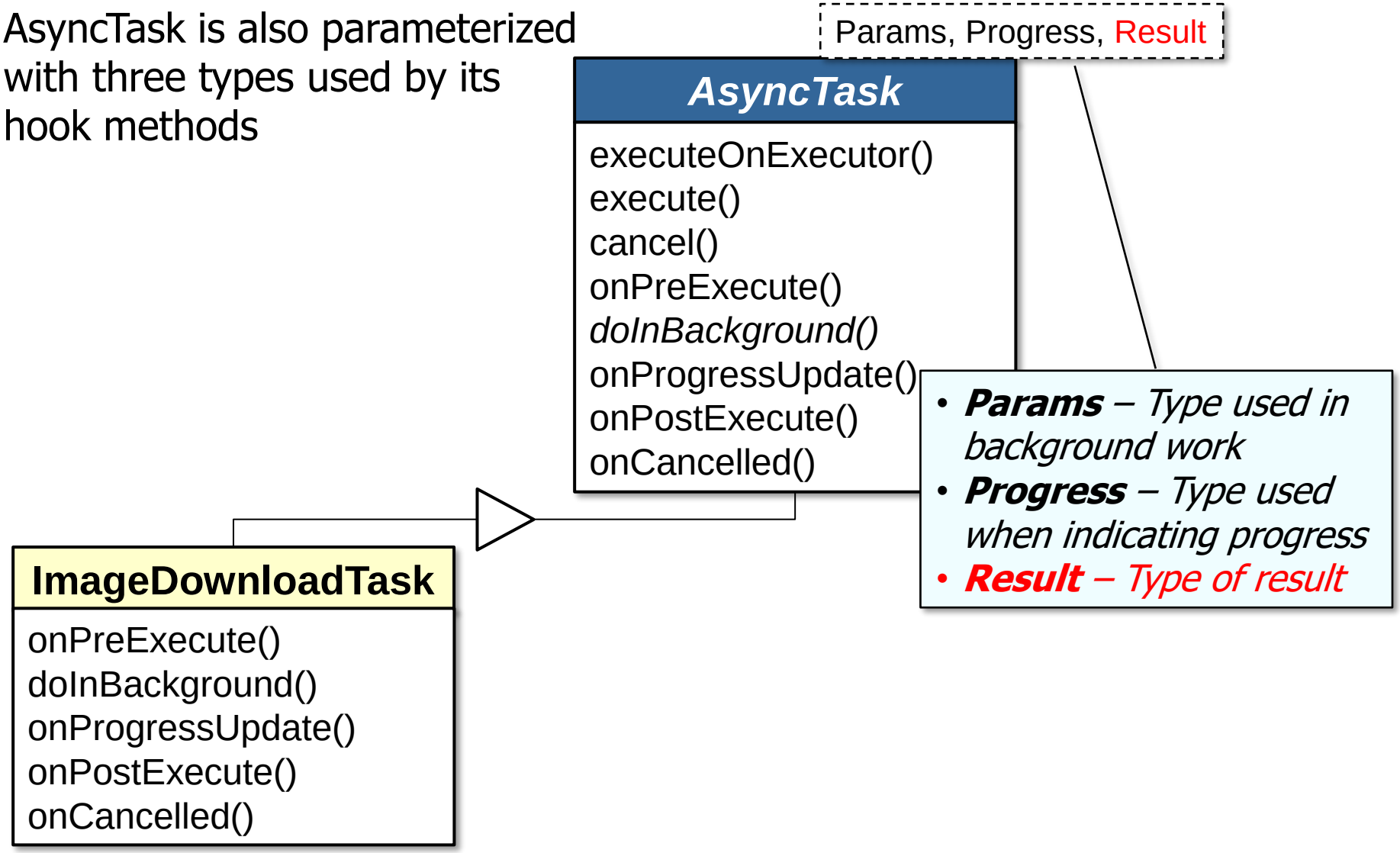
Overriding Hook Methods in the AsyncTask Class

- 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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " bytes"); }
}

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

Variant of 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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " 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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " 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

Hook method called by framework in UI thread

```
class DownloadTask extends
      AsyncTask<Uri, Integer, Long> {

  protected void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " 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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " bytes"); }
}

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

Hook method called by framework in background

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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " bytes"); }
}

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

Hook method called by framework in UI thread

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 void onPreExecute()
  { startDialog("Downloading file"); }

  protected Long doInBackground
                          (Uri... urls)
  { /* Download url & publish process */ }

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

  protected void onPostExecute(Long res)
  { stopDialog("Got " + res + " bytes"); }
}

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

Hook method called by framework in UI thread

End of the AsyncTask Framework: Key Methods