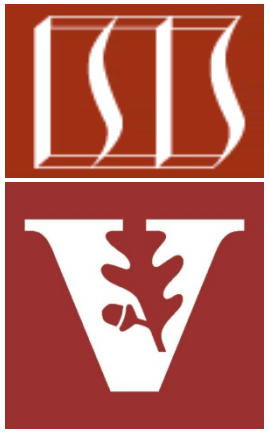
Android Concurrency: The AsyncTask Framework

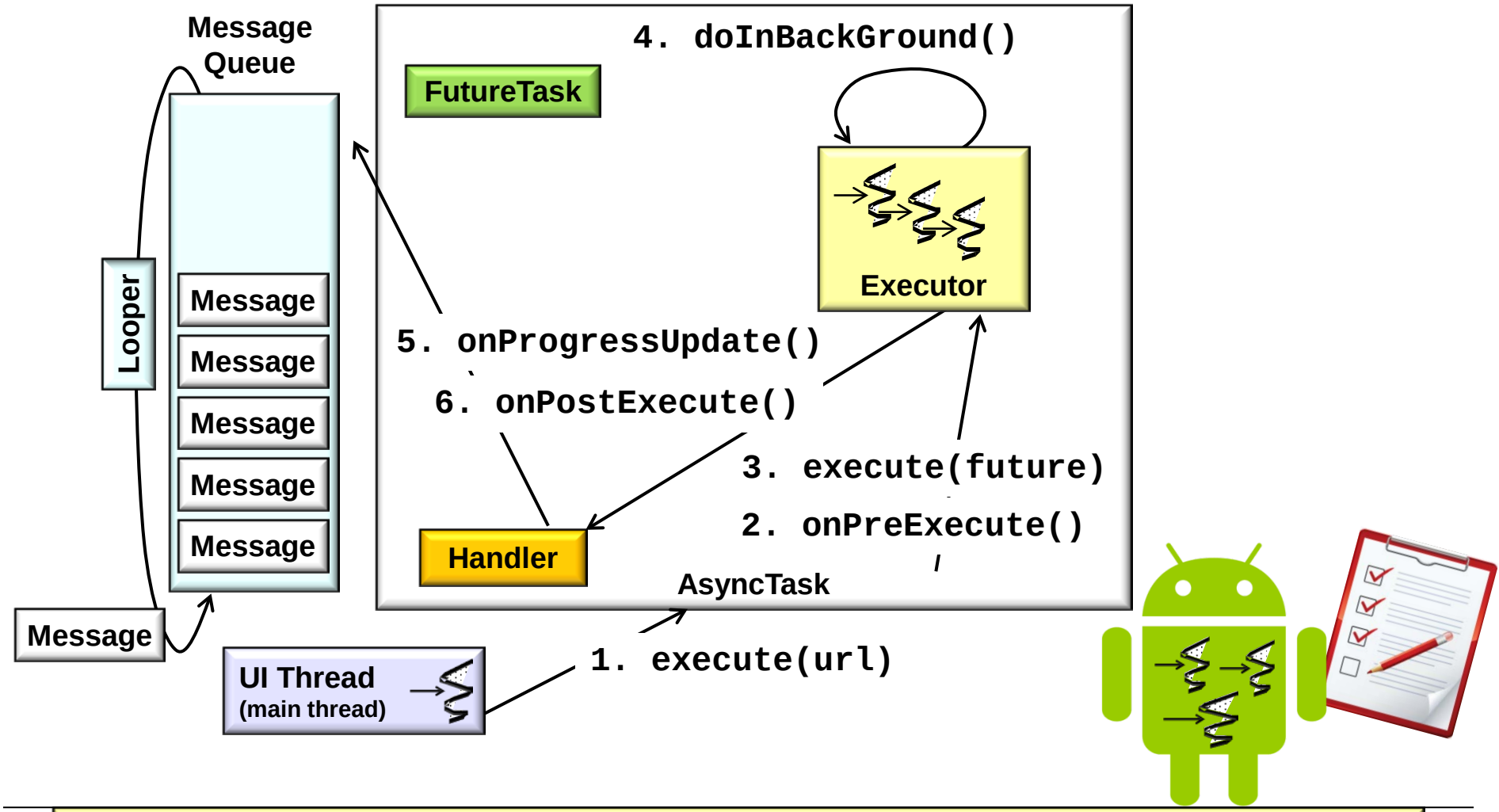


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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

Learning Objectives in this Part of the Module

- Recognize the concurrency idioms & mechanisms associated with programming the Android AsyncTask framework

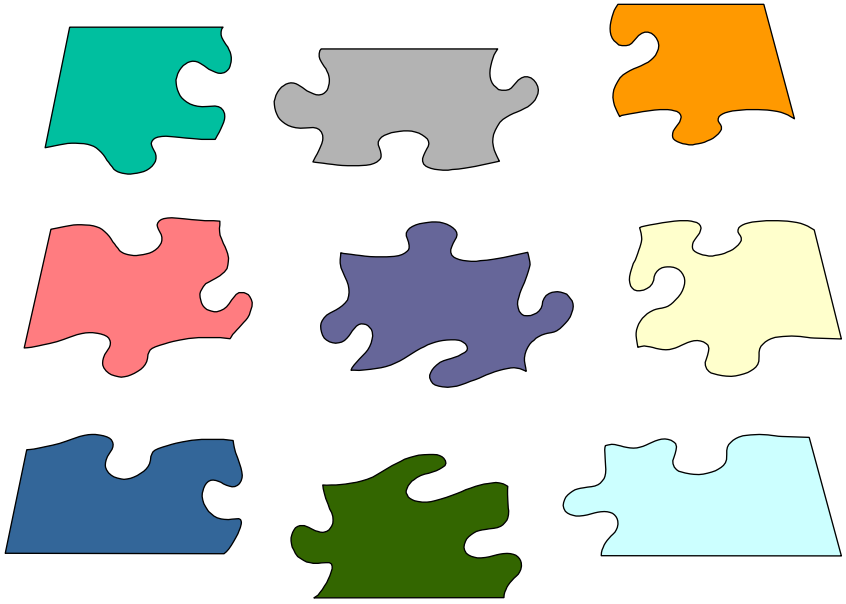


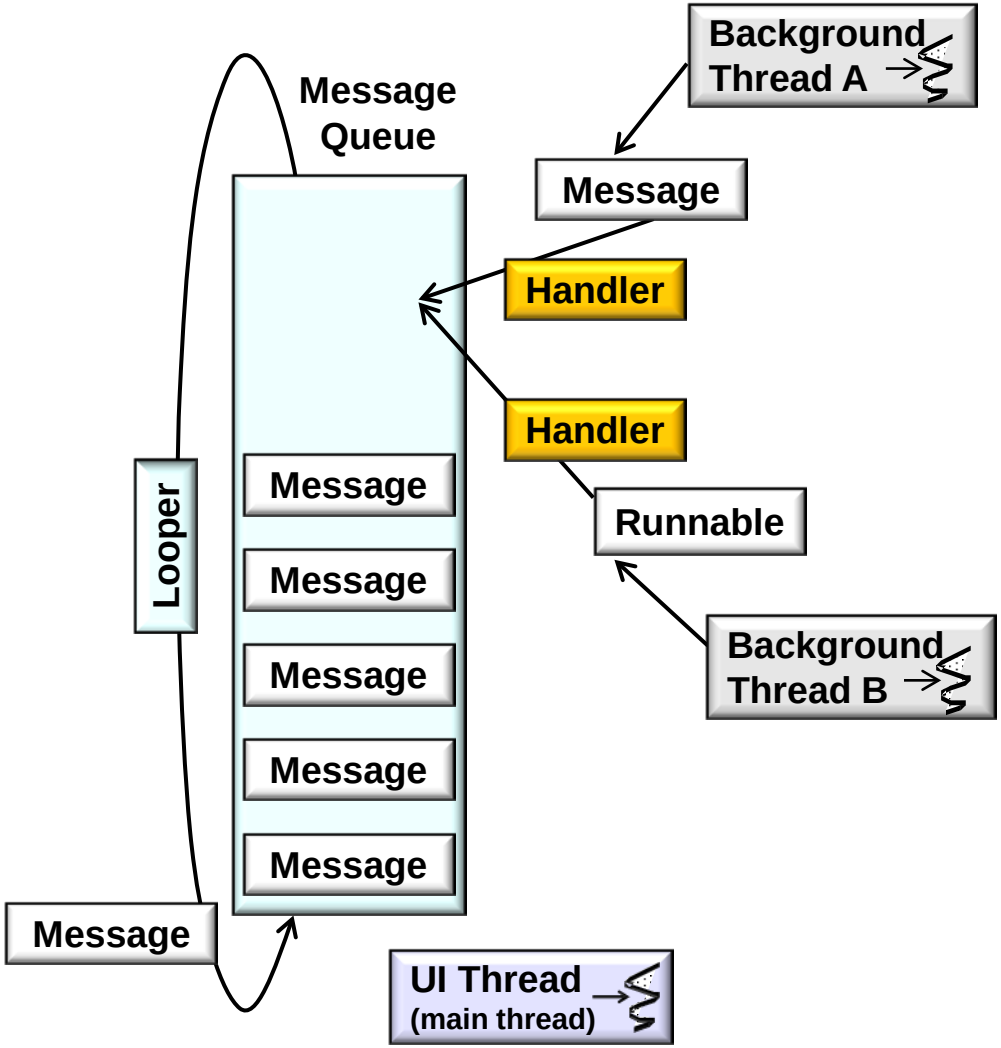
Allows apps to perform background operations & publish results on UI thread *without* manipulating threads, handlers, messages, or runnables

Overview of the AsyncTask Framework

Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected

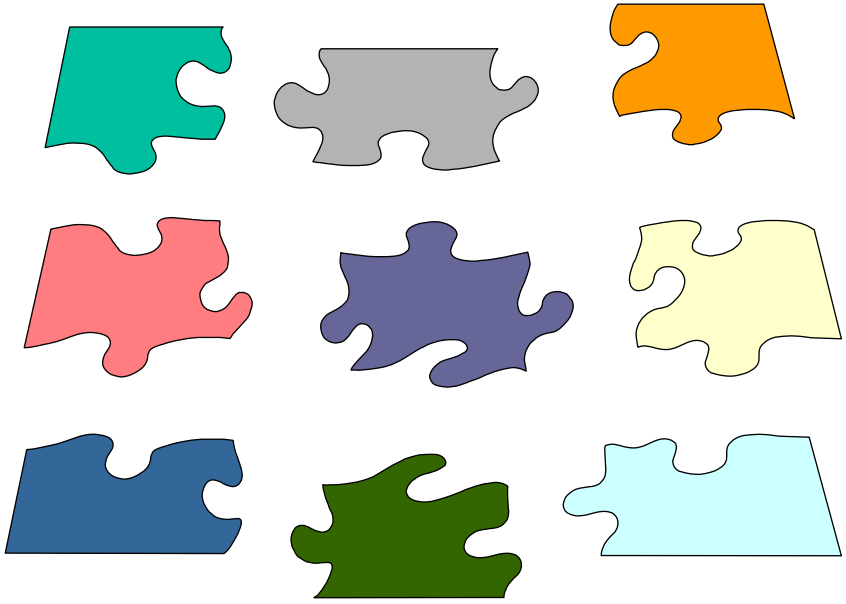


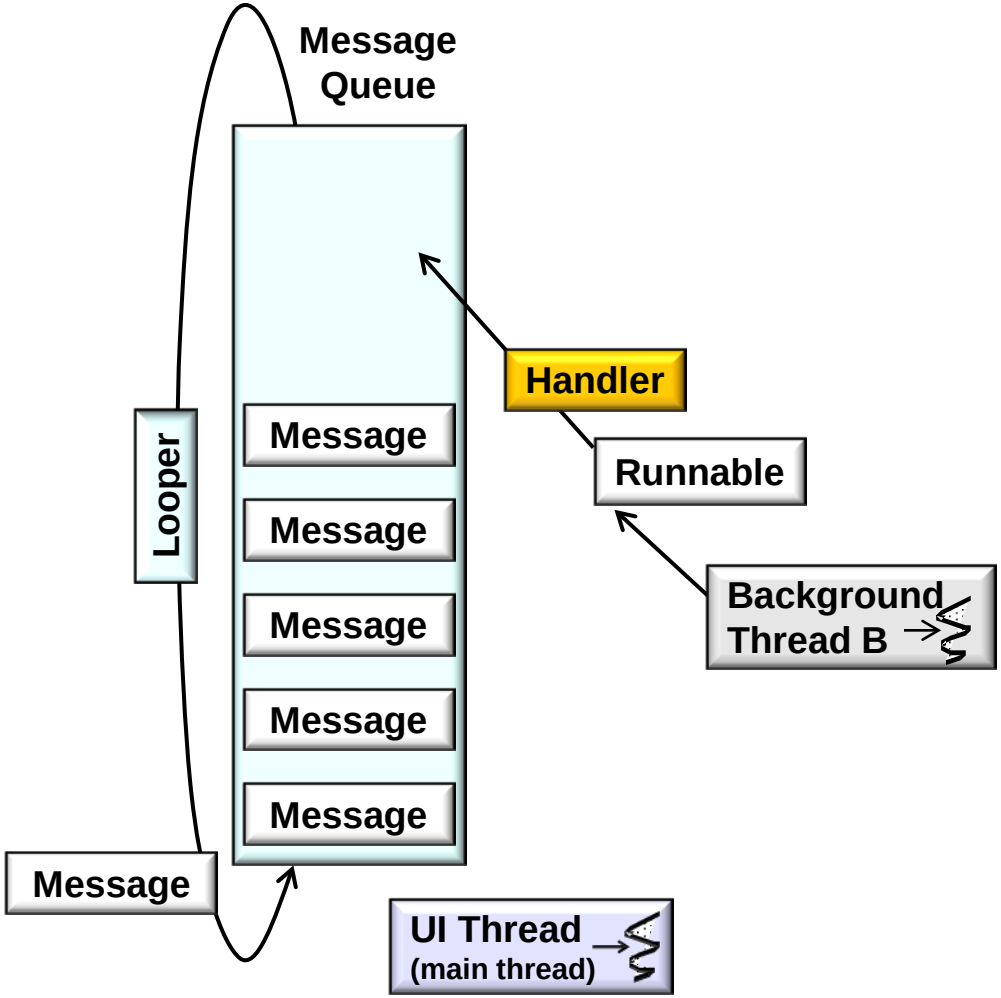


See previous part on the HaMeR framework

Overview of the AsyncTask Framework

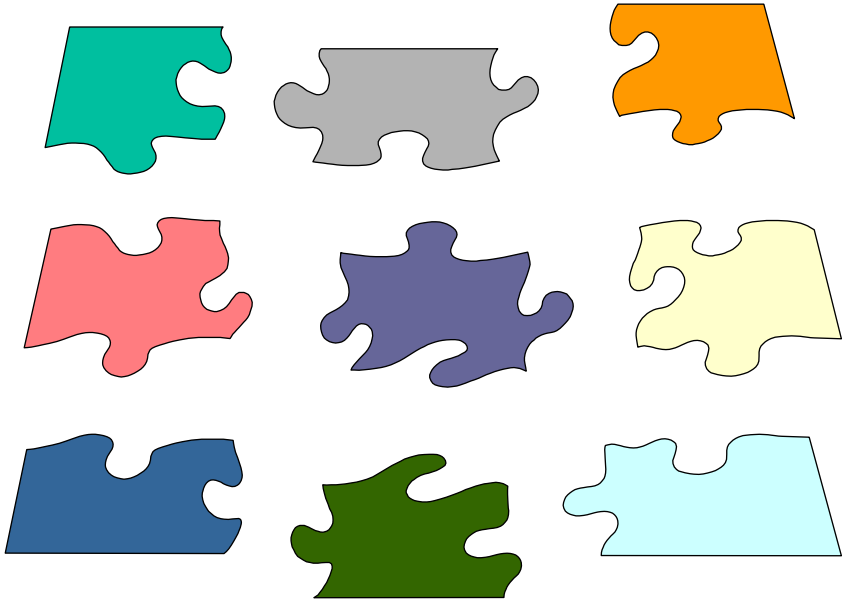
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases

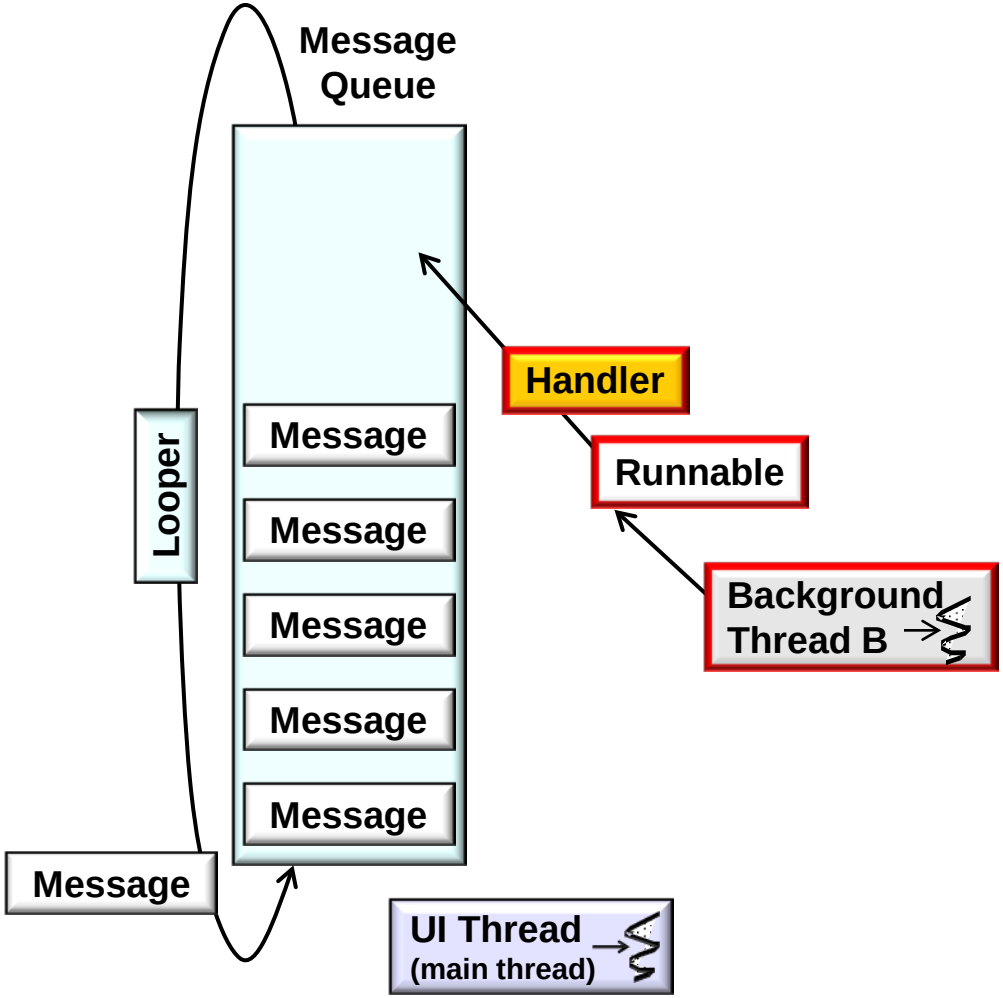




Overview of the AsyncTask Framework

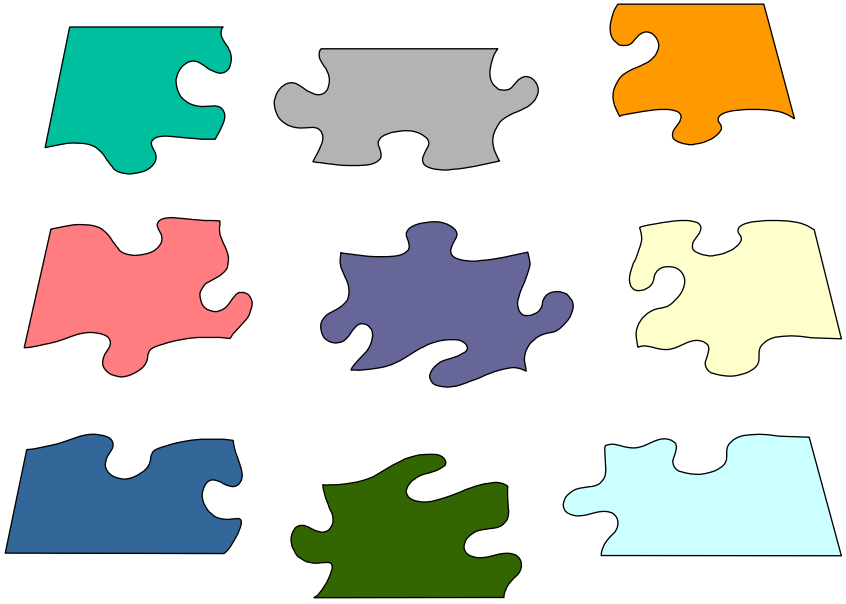
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases

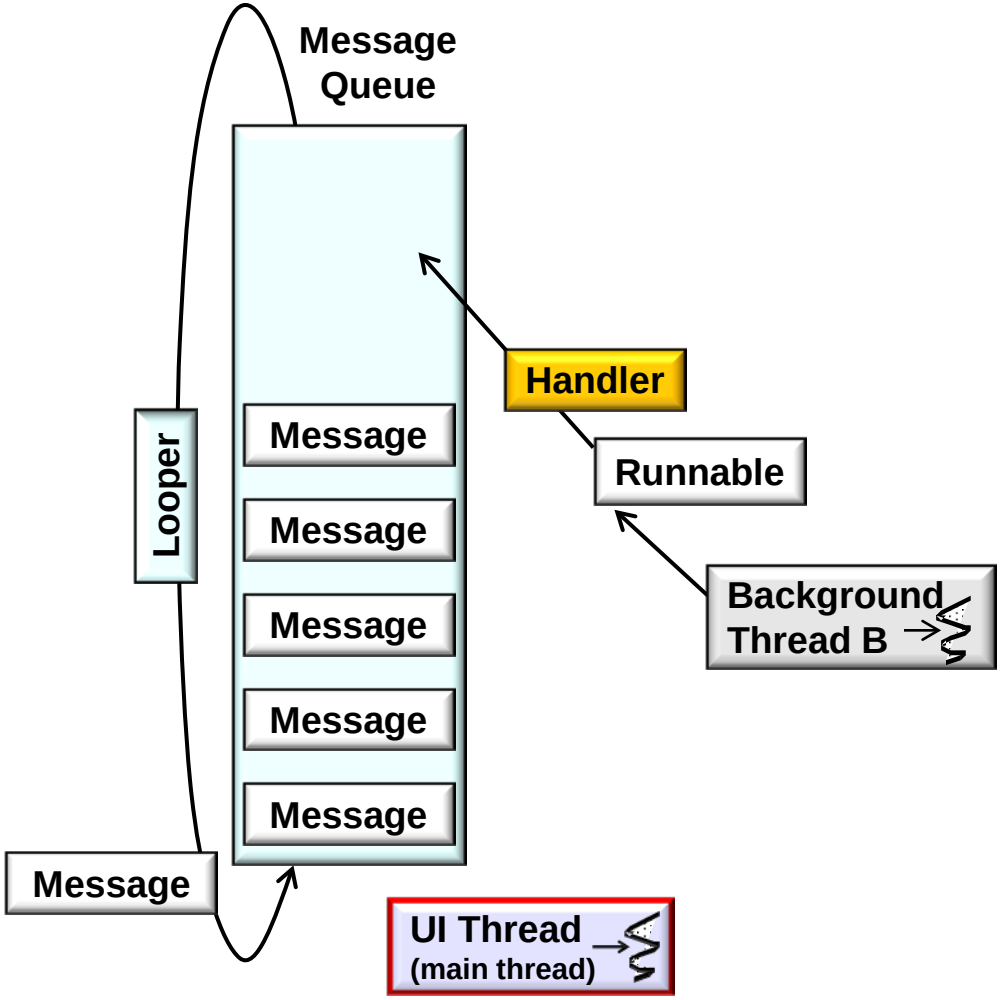




Overview of the AsyncTask Framework

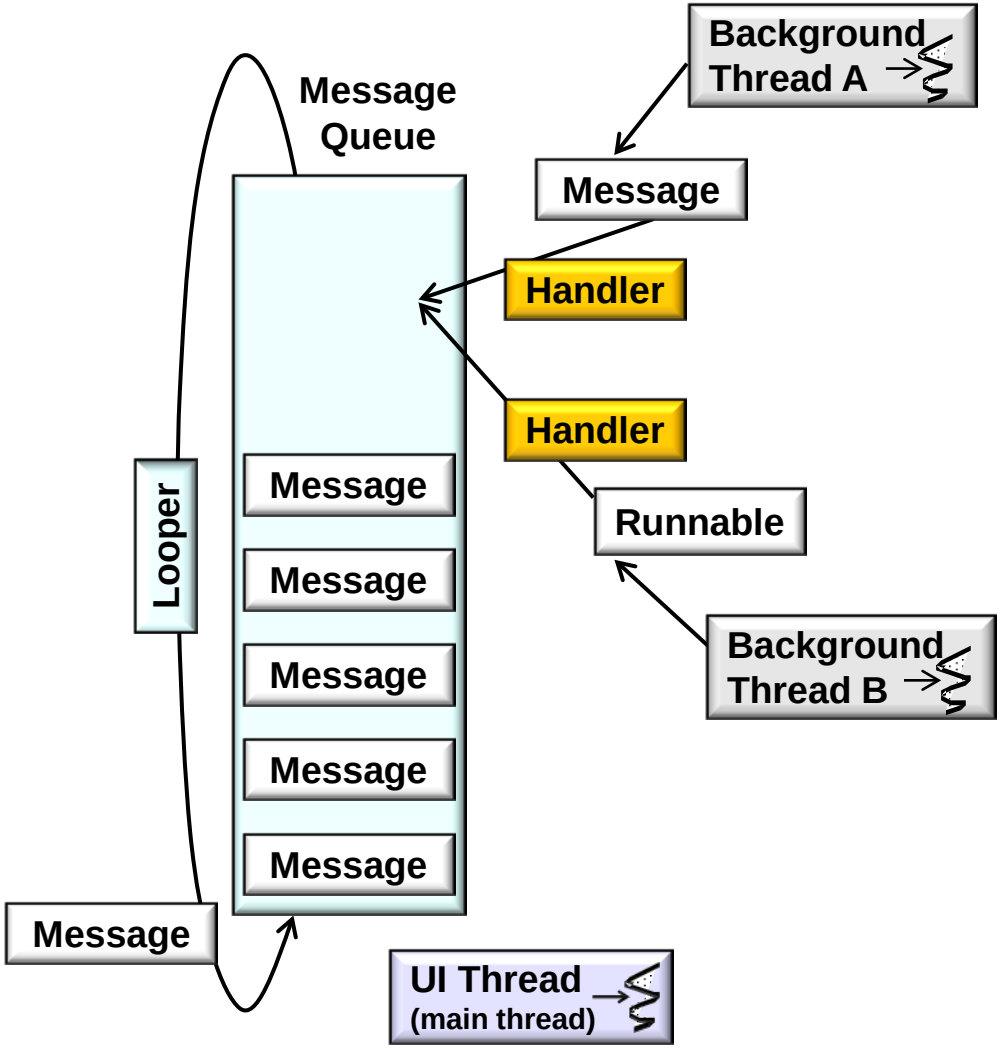
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases





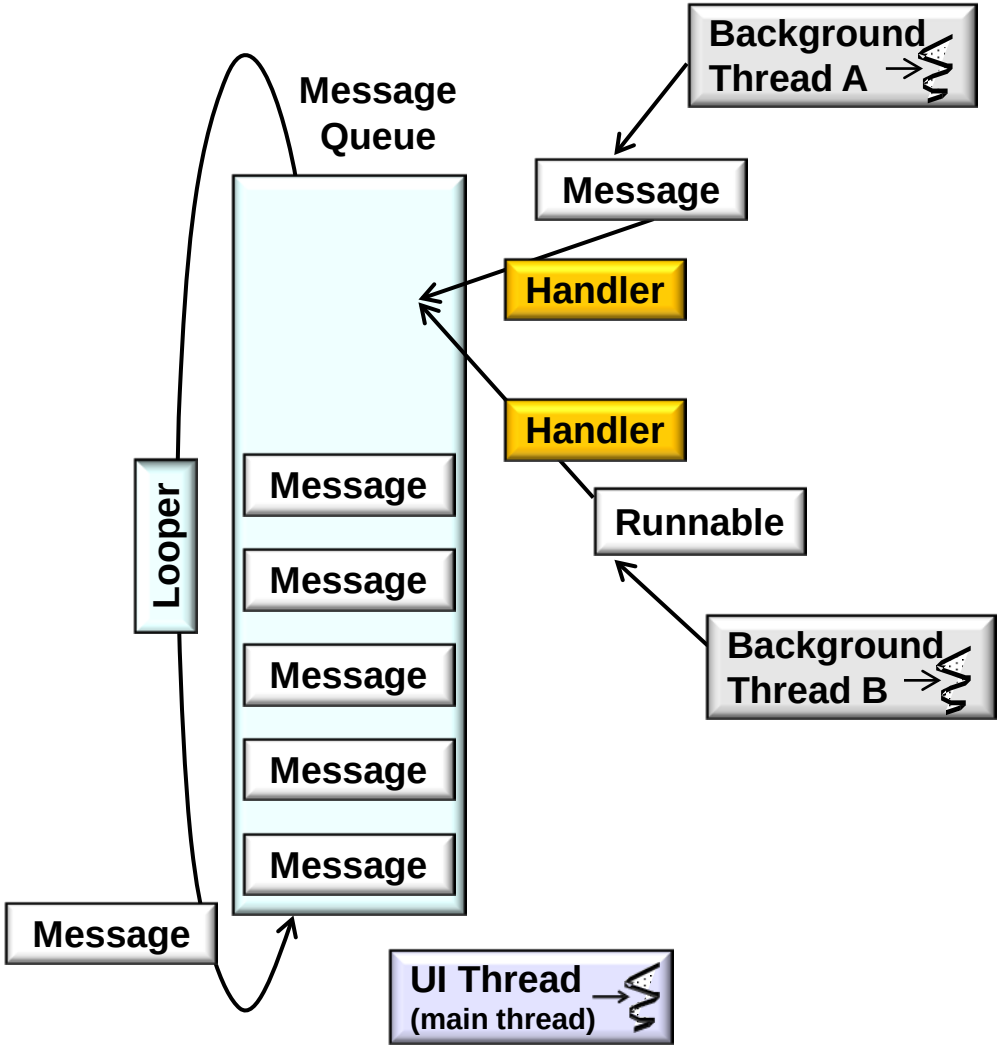
Overview of the AsyncTask Framework

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



Overview of the AsyncTask Framework

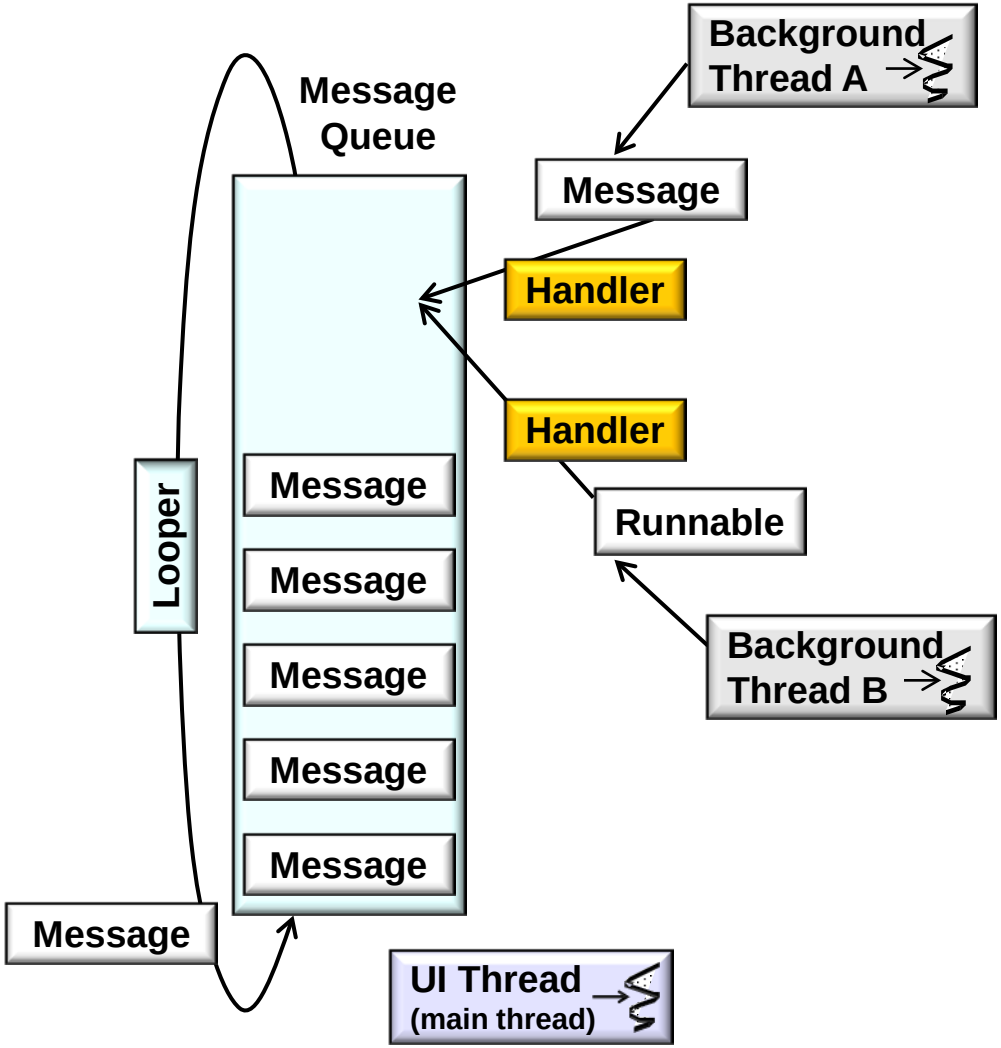
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases
  - However, there are drawbacks
    - Must understand patterns



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

Overview of the AsyncTask Framework

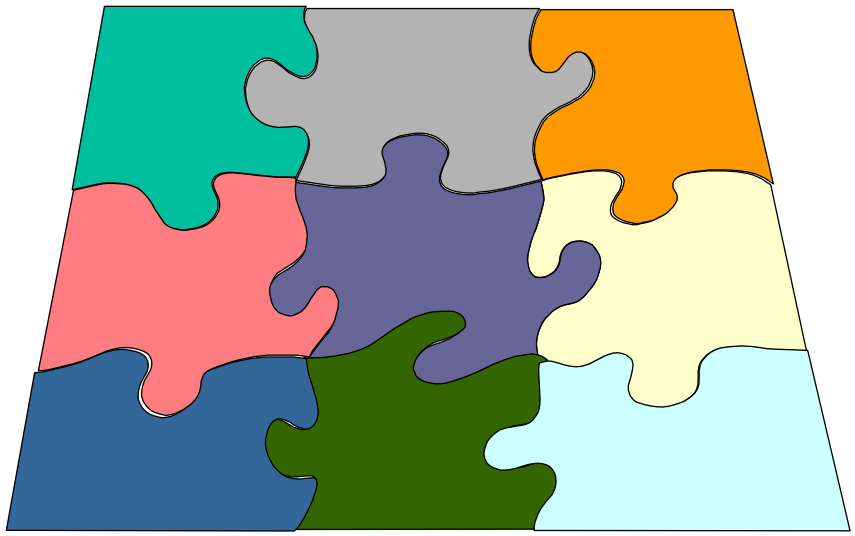
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple concurrency use cases
  - However, there are drawbacks
    - Must understand patterns
    - Tedious & error-prone

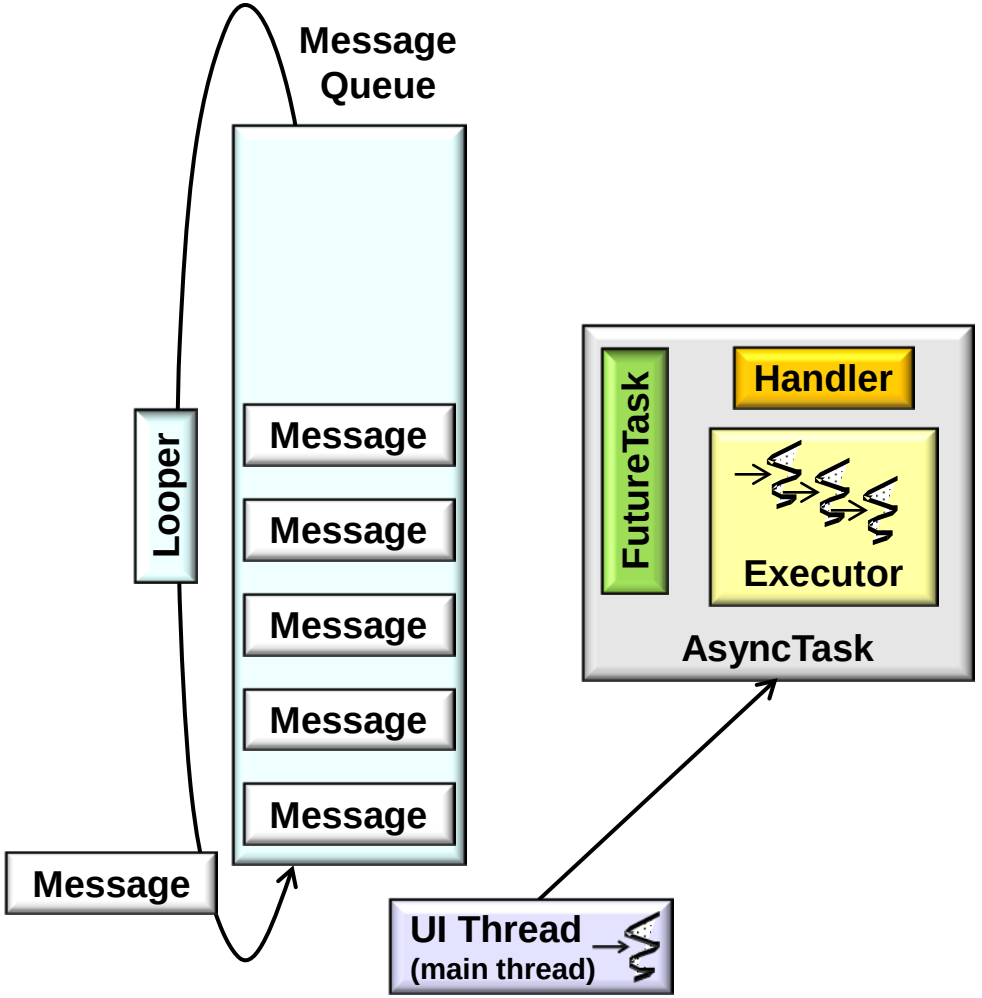


e.g., apps must understand how to manage the lifecycle of Messages

Overview of the AsyncTask Framework

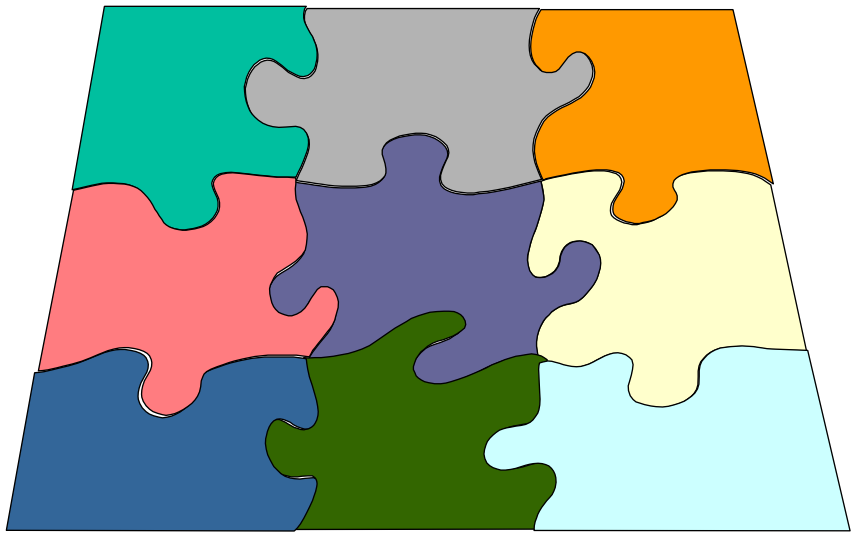
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

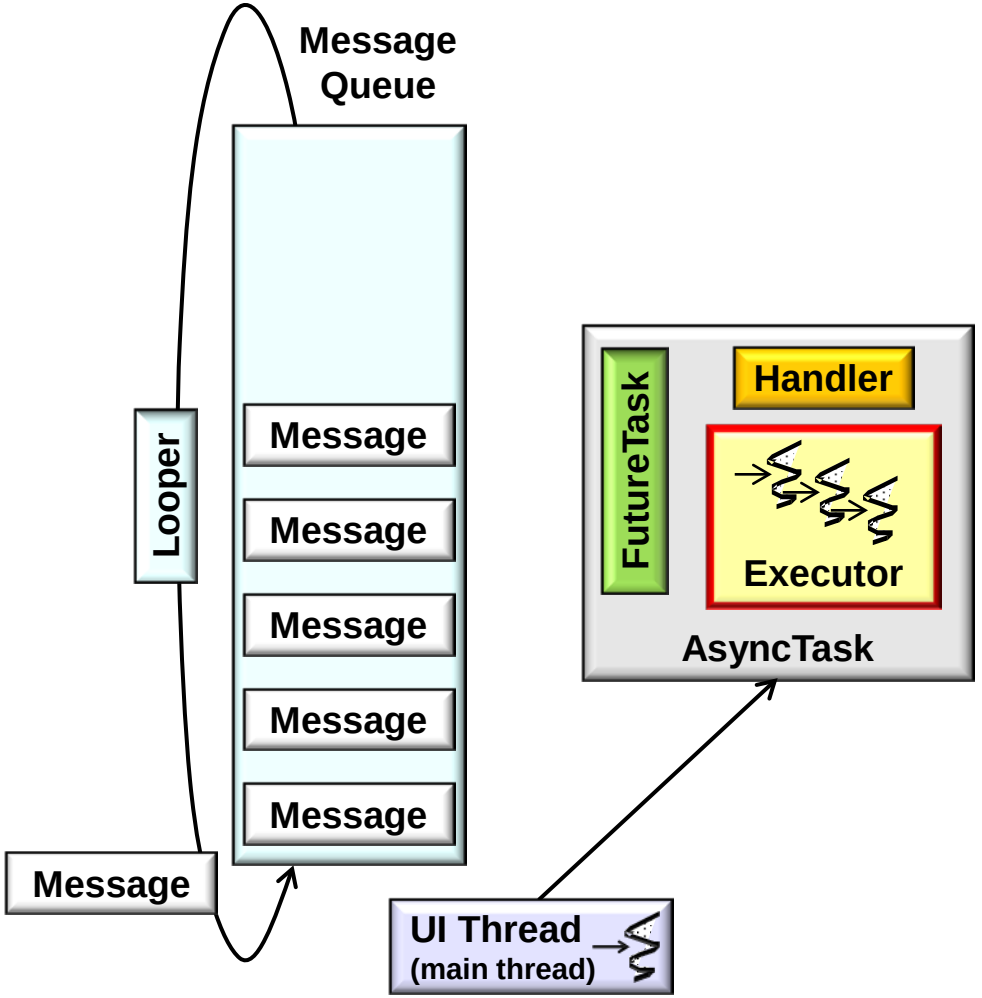




Overview of the AsyncTask Framework

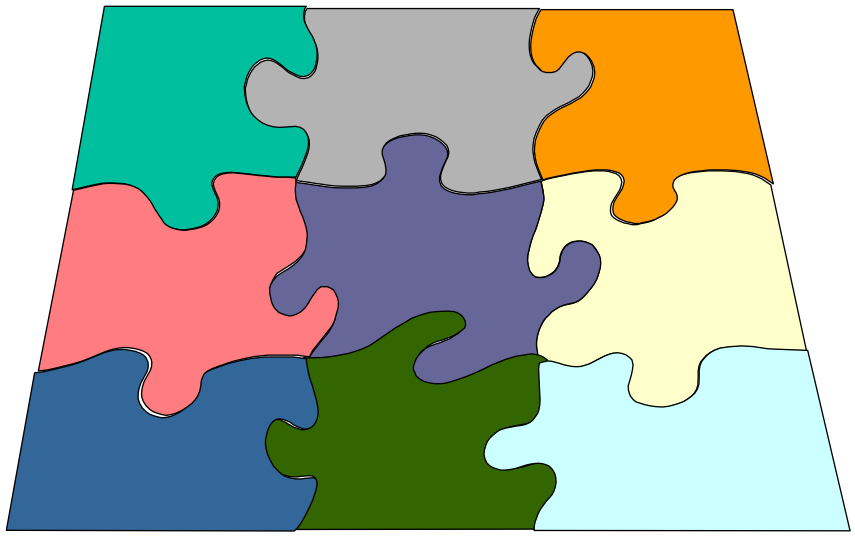
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

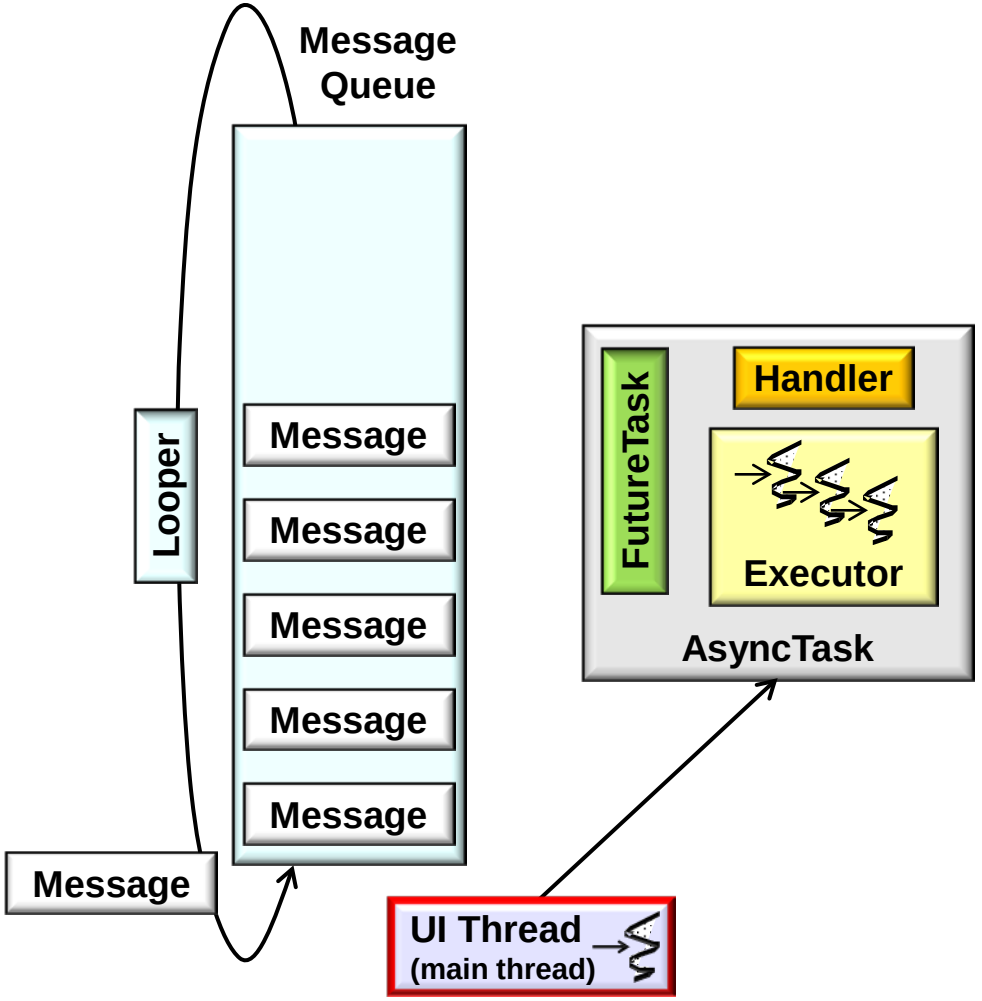




Overview of the AsyncTask Framework

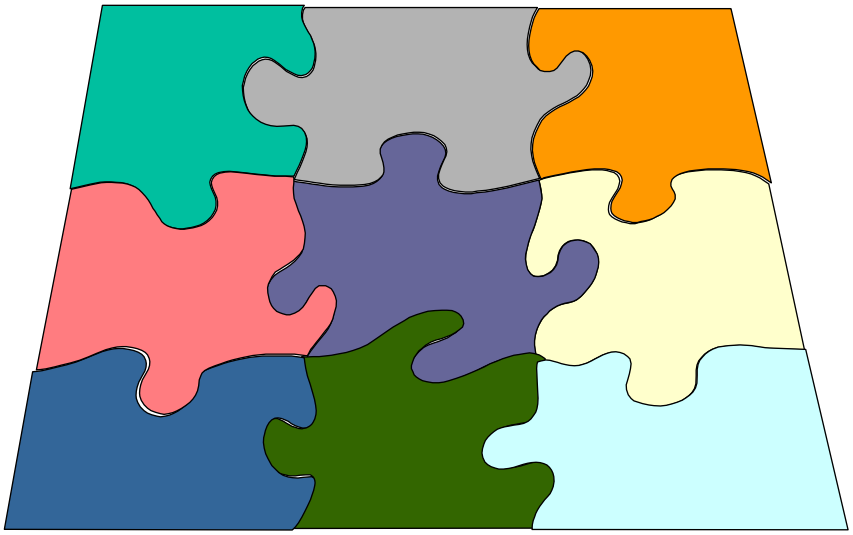
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

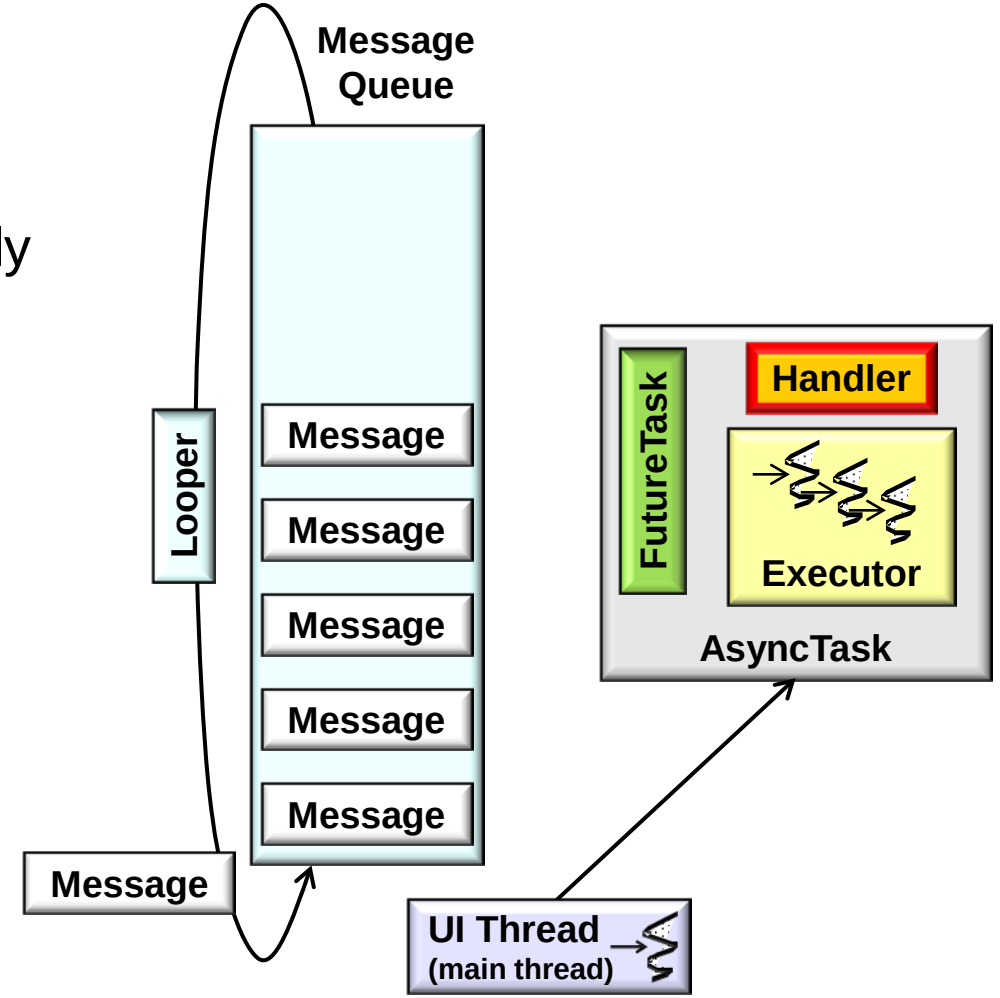




Overview of the AsyncTask Framework

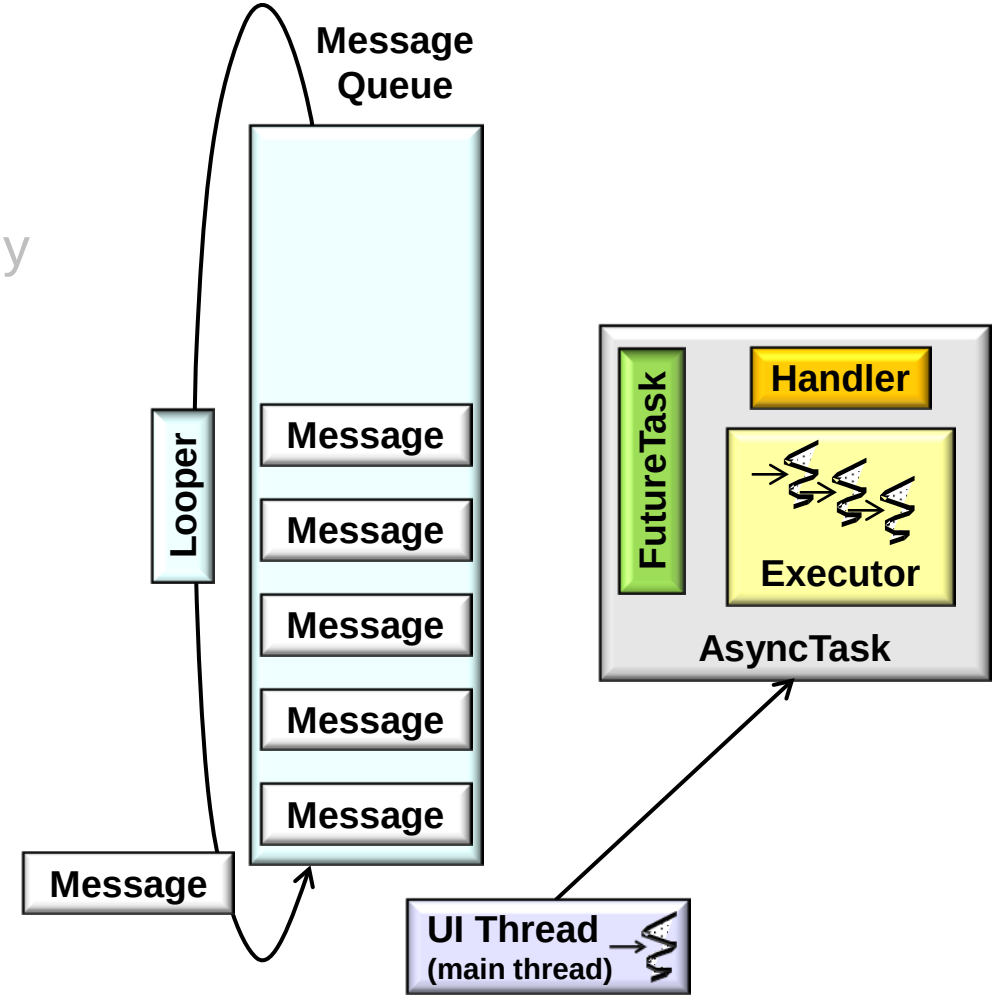
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - Run concurrently, *without* directly manipulating Threads, Handlers, Messages, or Runnables





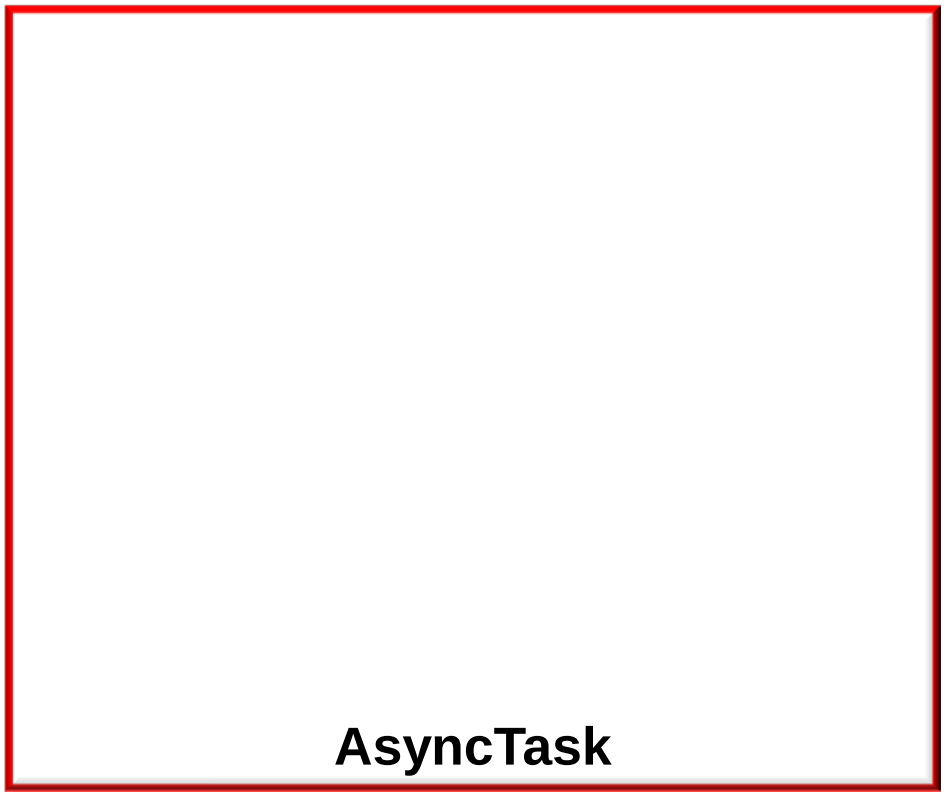
Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - Run concurrently, *without* directly manipulating Threads, Handlers, Messages, or Runnables
  - Smaller “surface area”



Overview of the AsyncTask Framework

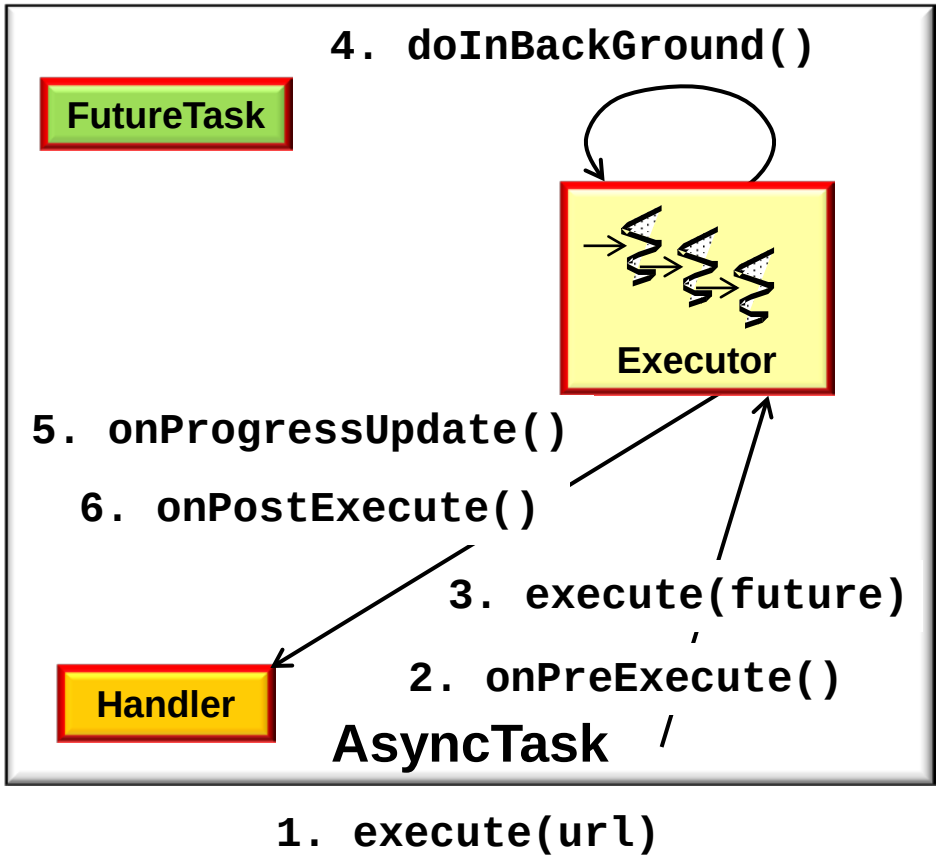
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - Run concurrently, *without* directly manipulating Threads, Handlers, Messages, or Runnables
  - Smaller "surface area"
  - Complex framework details accessed via *Façade* pattern



See en.wikipedia.org/wiki/Facade_pattern

Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - Run concurrently, *without* directly manipulating Threads, Handlers, Messages, or Runnables
  - Smaller "surface area"
  - Complex framework details accessed via *Façade* pattern
    - Wraps complicated subsystem or framework with simpler interface



Categories of Methods in AsyncTask (Part 1)

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

`...`

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

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

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

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

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

`void onCancelled()`
- 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

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

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

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

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

`void onCancelled()`
- 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 framework
      - At different points of time &
      - In different threading contexts

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

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

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

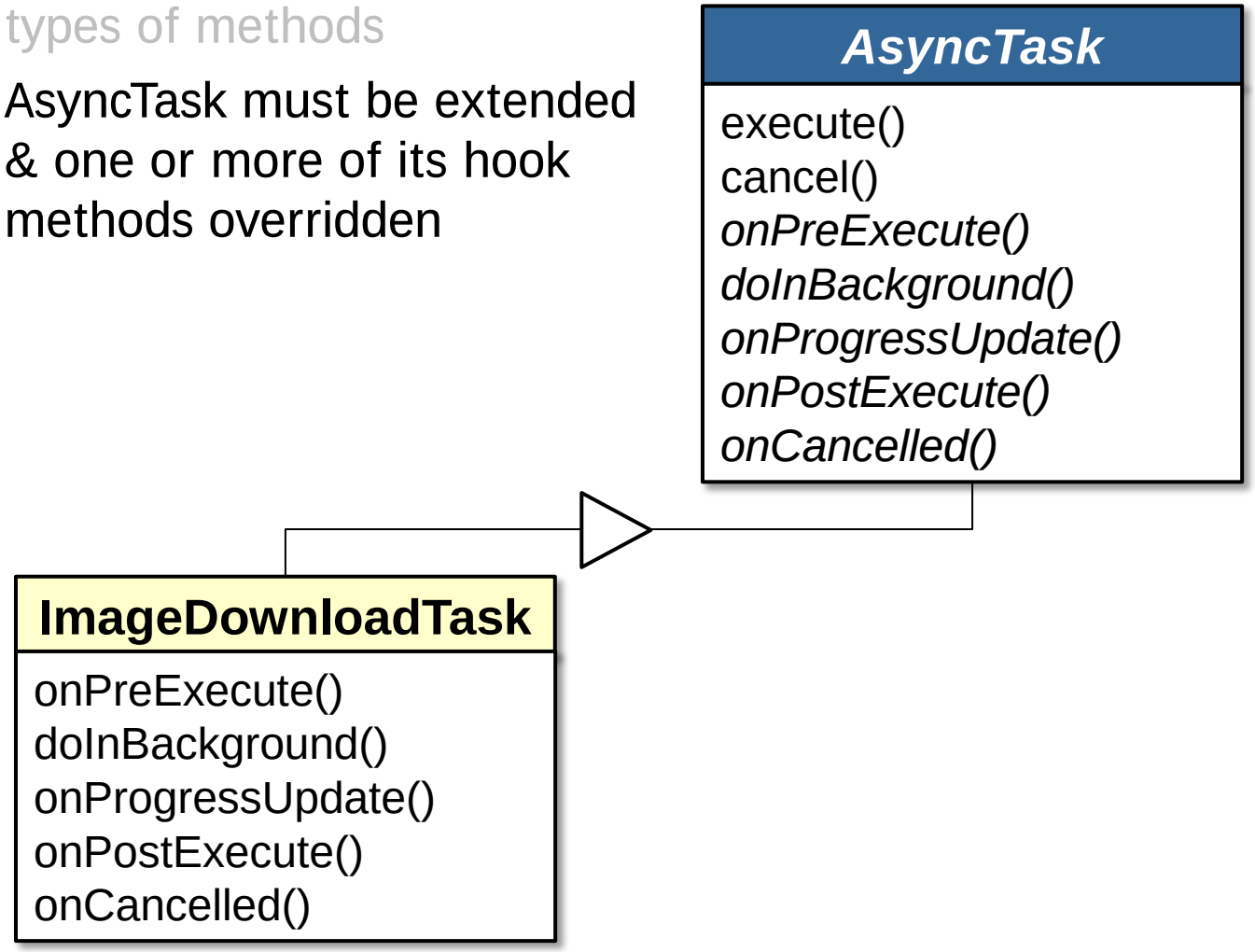
`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

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

`...`

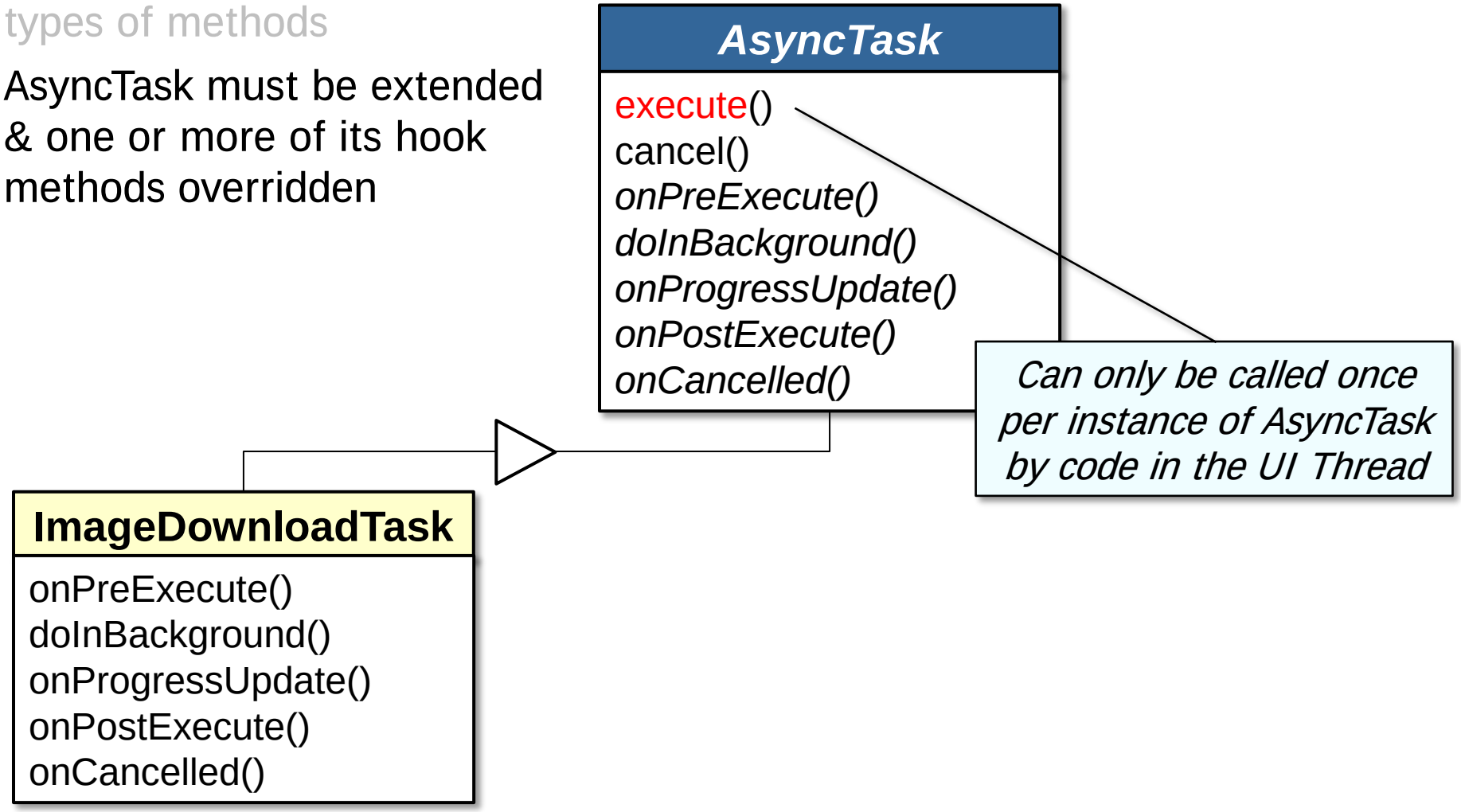
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



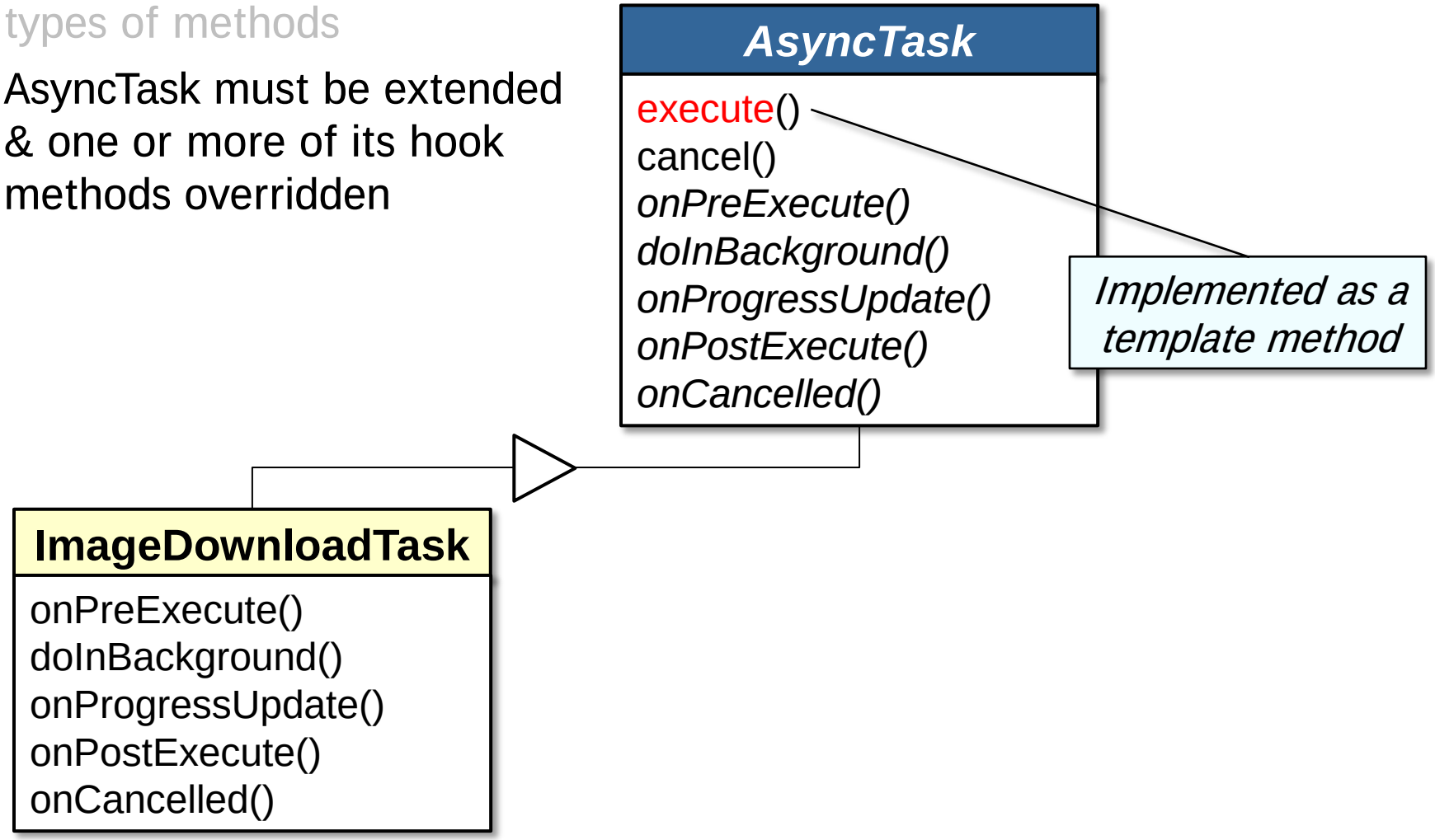
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in the AsyncTask Class

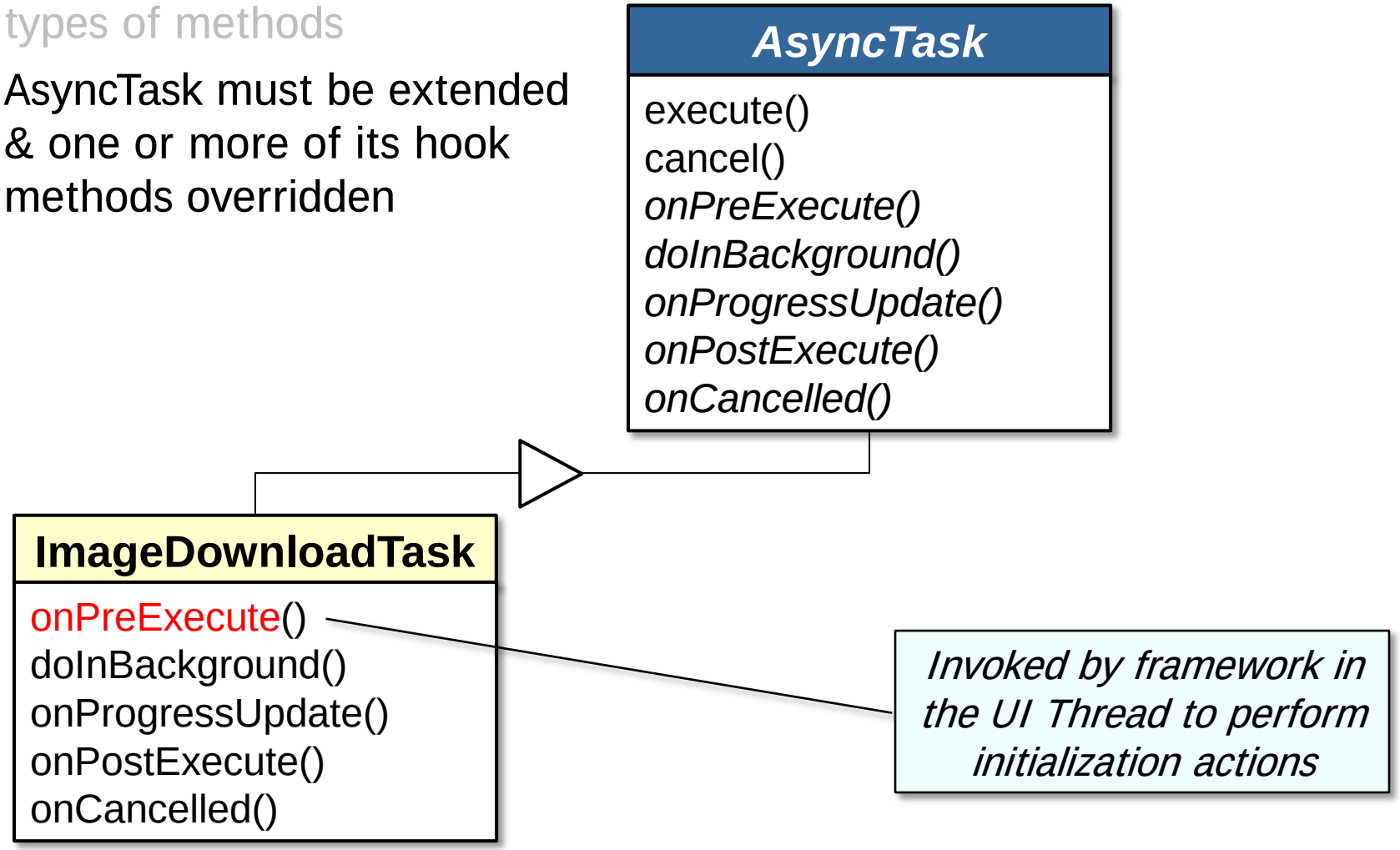
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



See en.wikipedia.org/wiki/Template_method_pattern

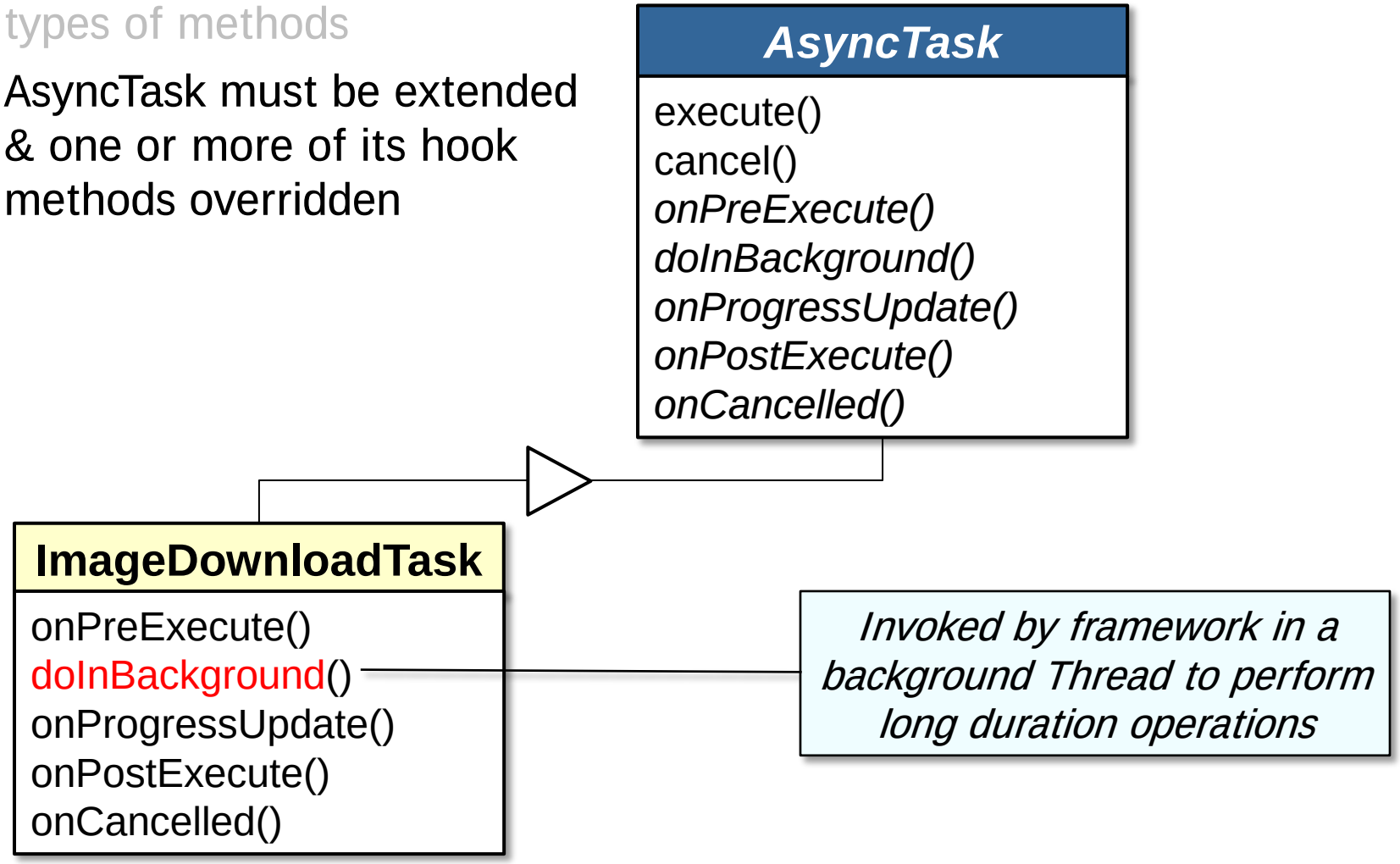
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in the AsyncTask Class

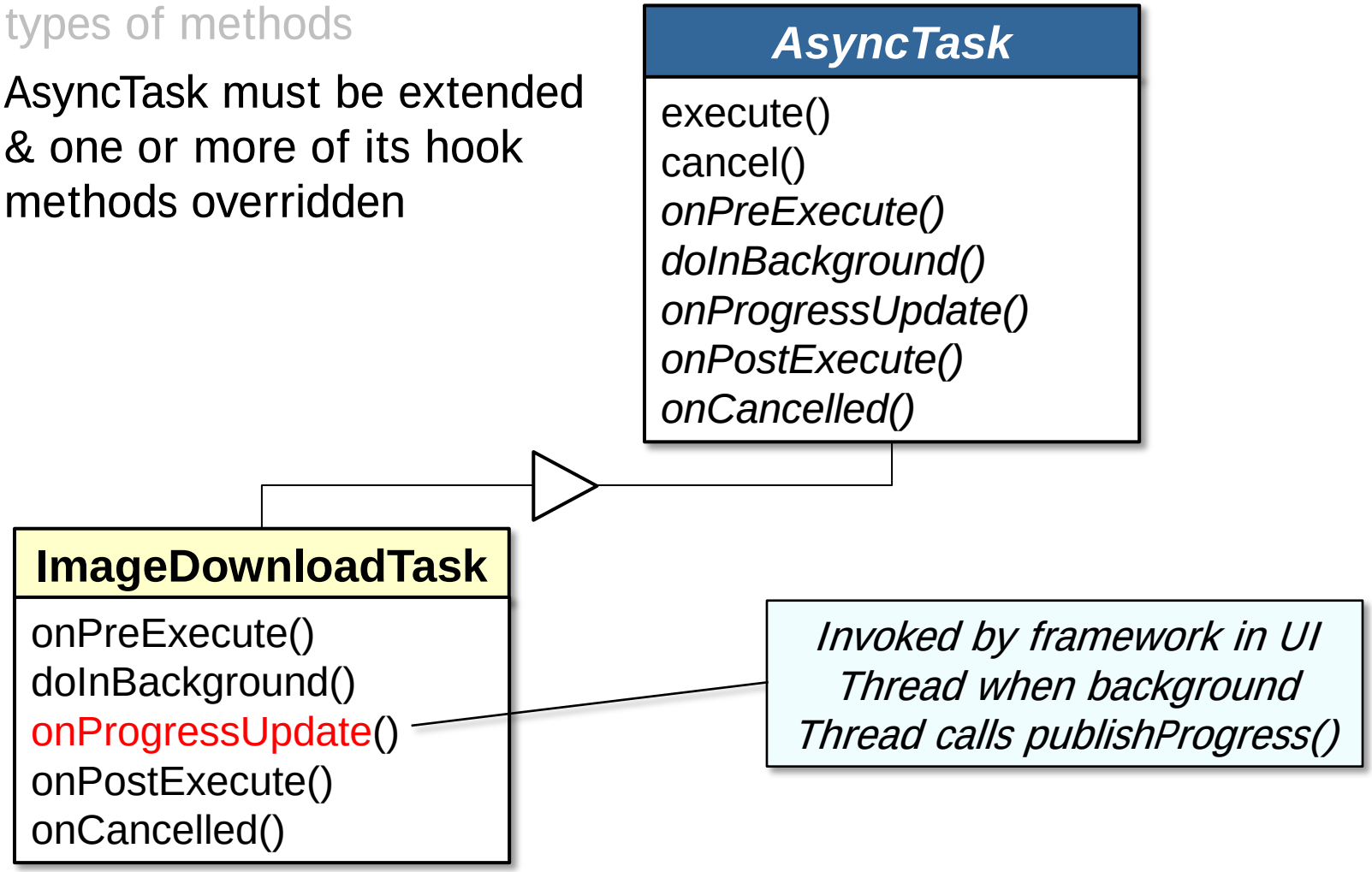
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



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

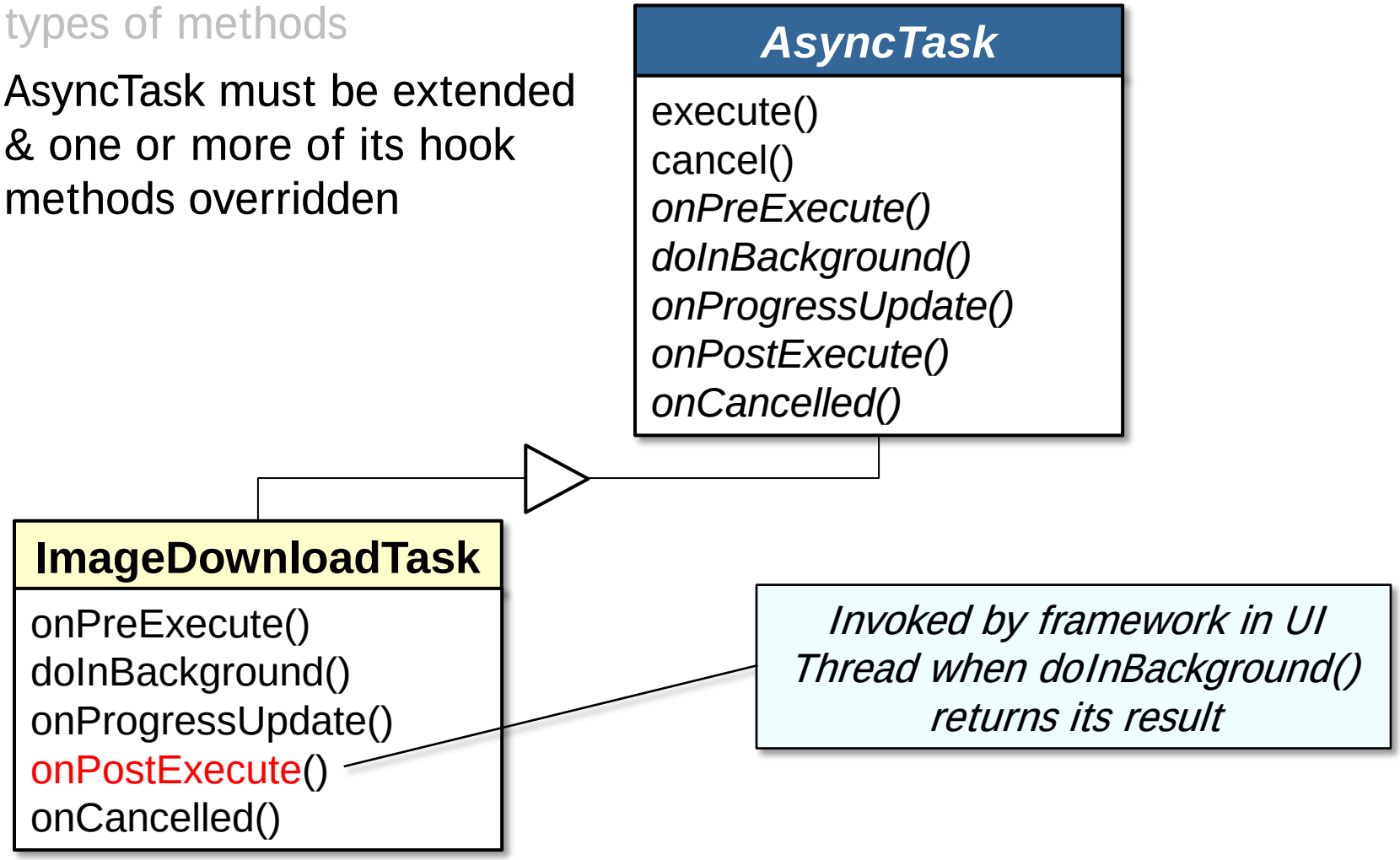
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in the AsyncTask Class

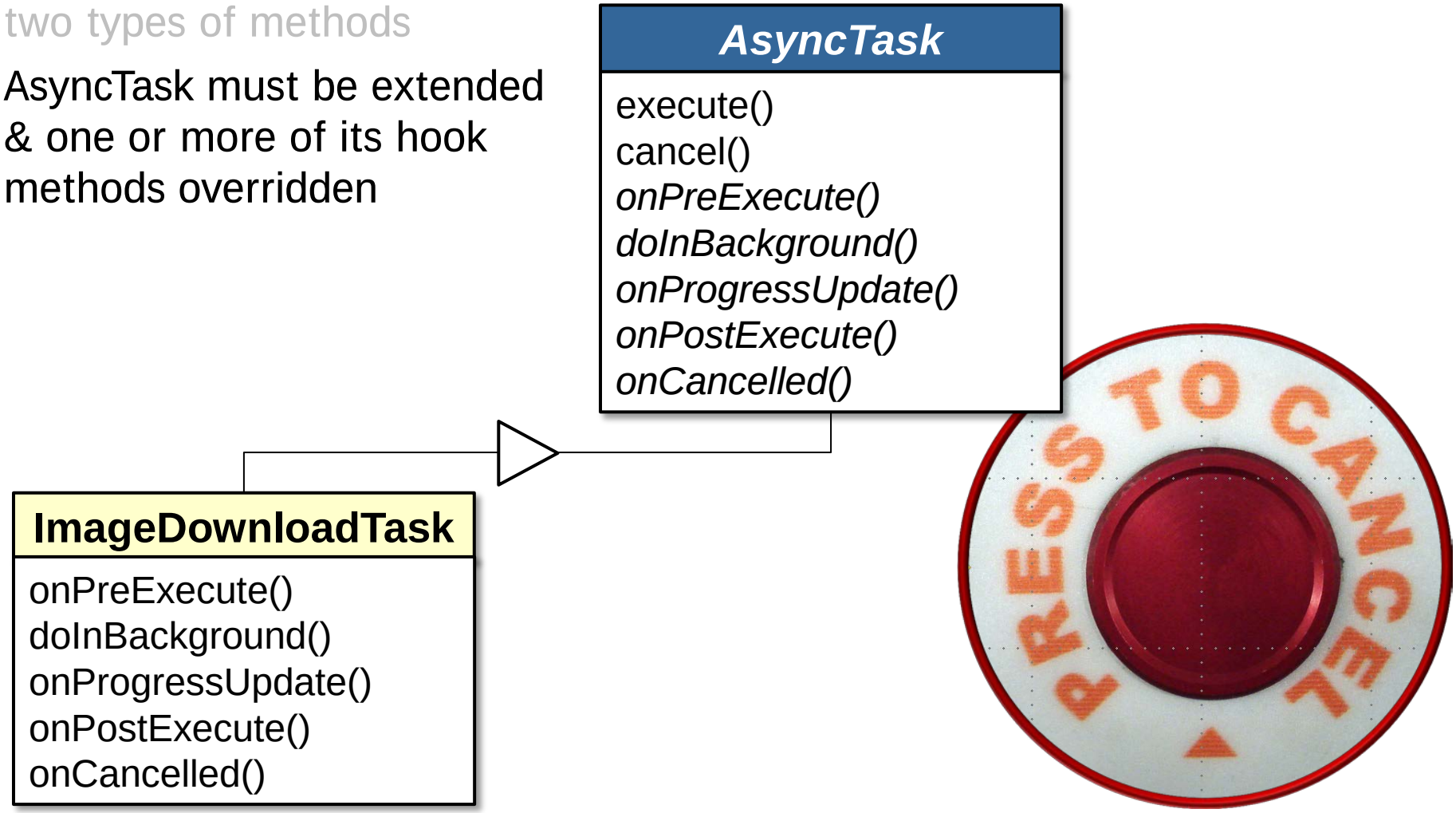
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in AsyncTask (Part 2)

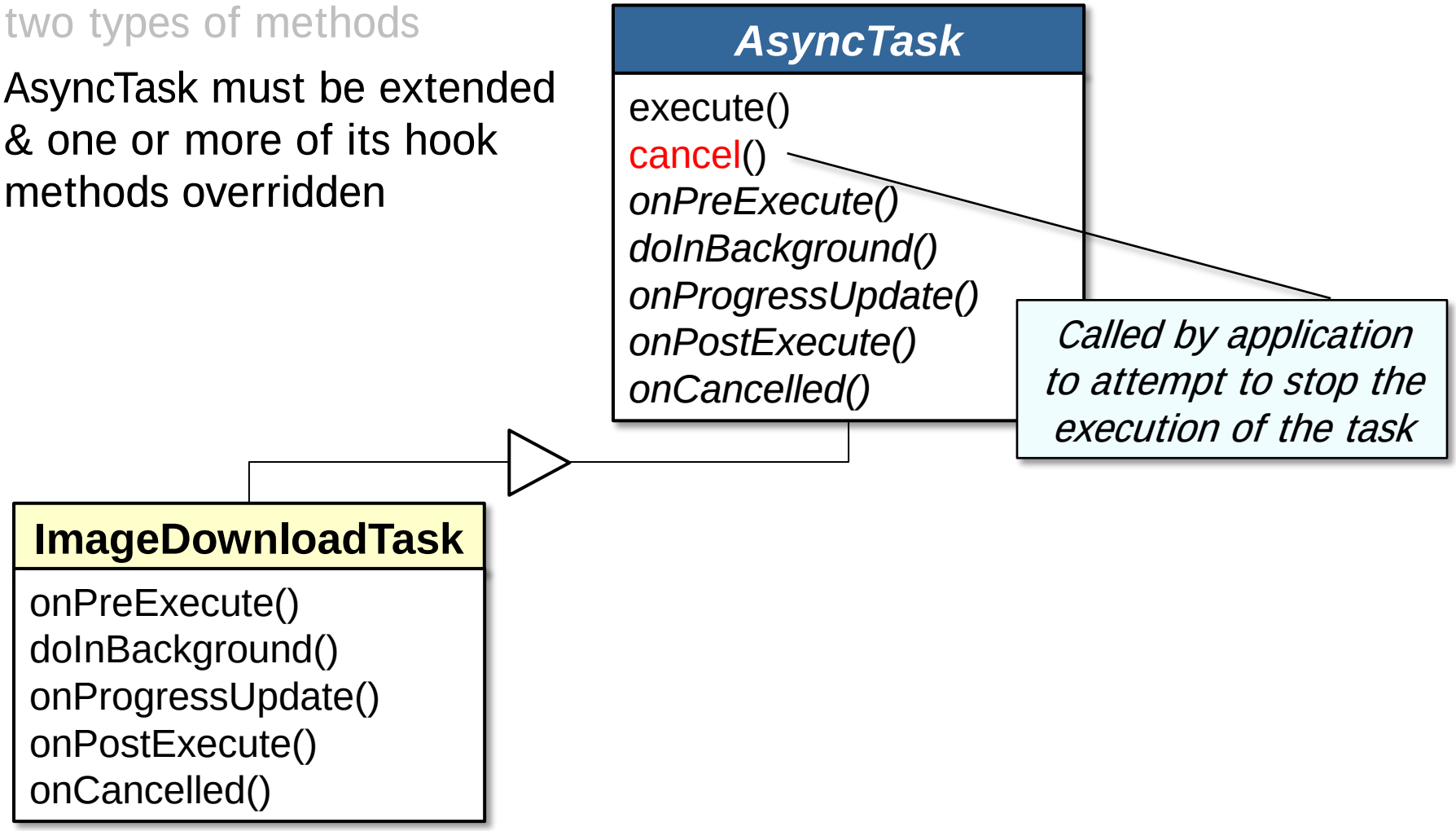
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



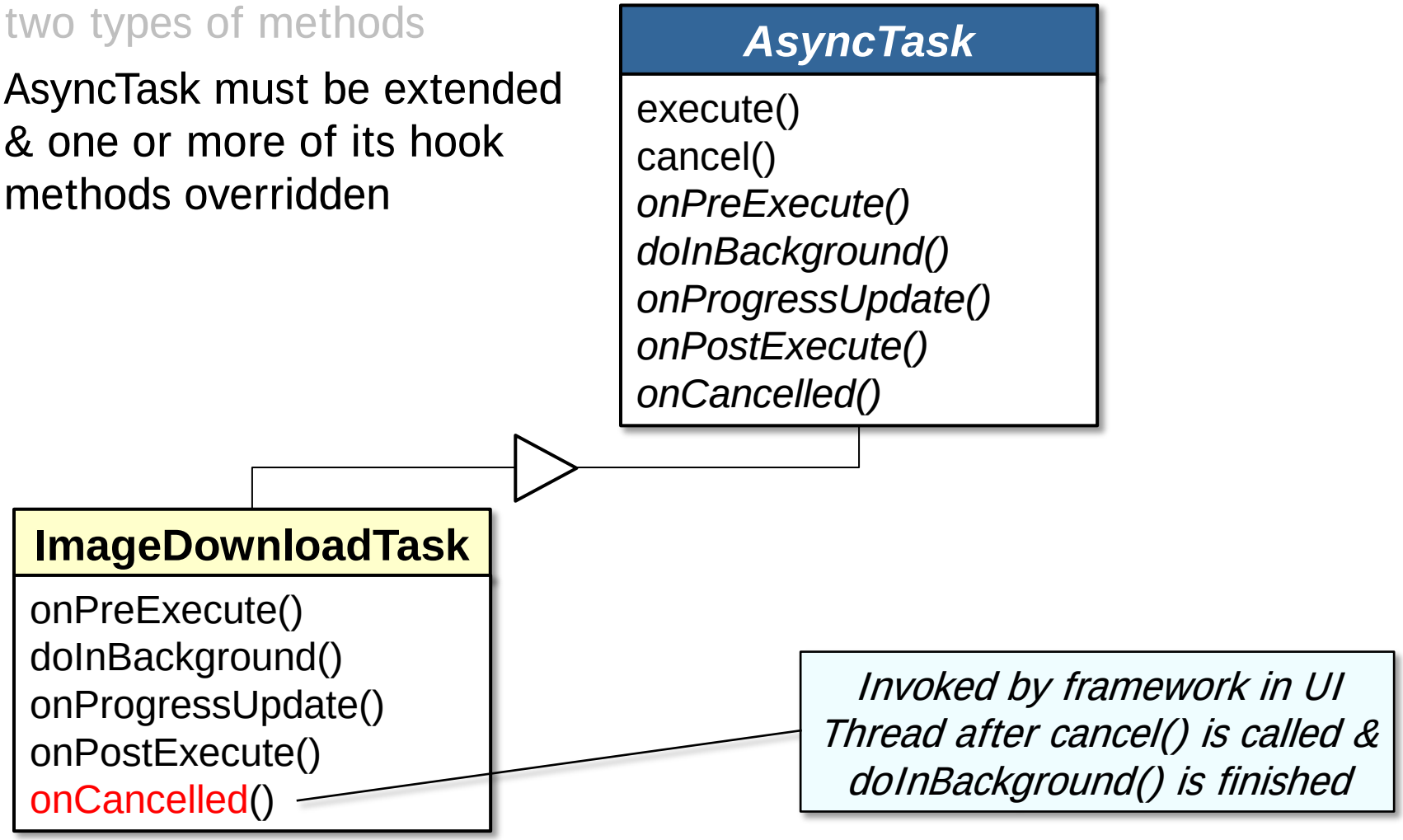
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in the AsyncTask Class

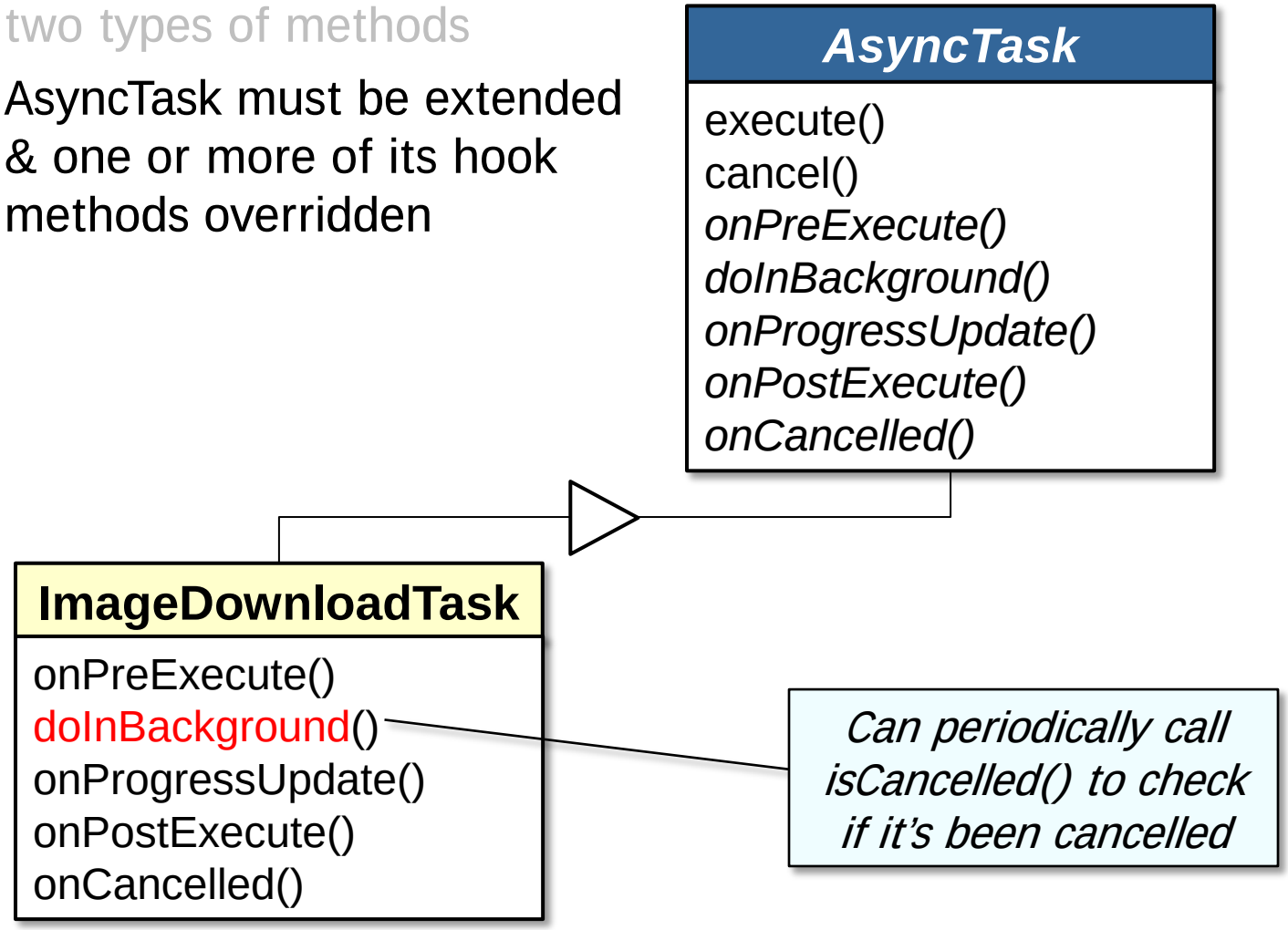
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



If onCancelled() is called then onPostExecute() is *not* called

Categories of Methods in the AsyncTask Class

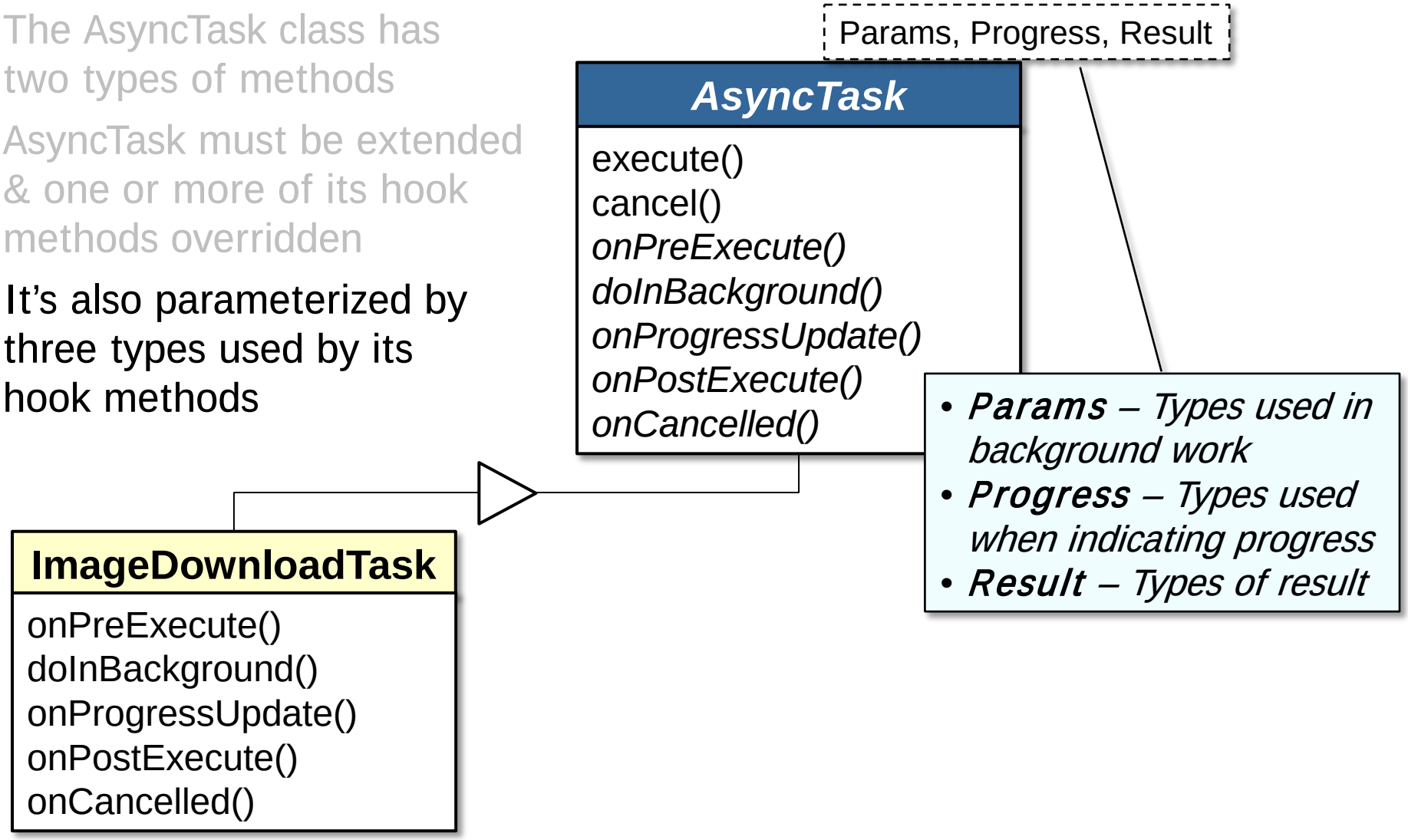
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Similar to using Java Thread interrupt requests to voluntarily shutdown Threads

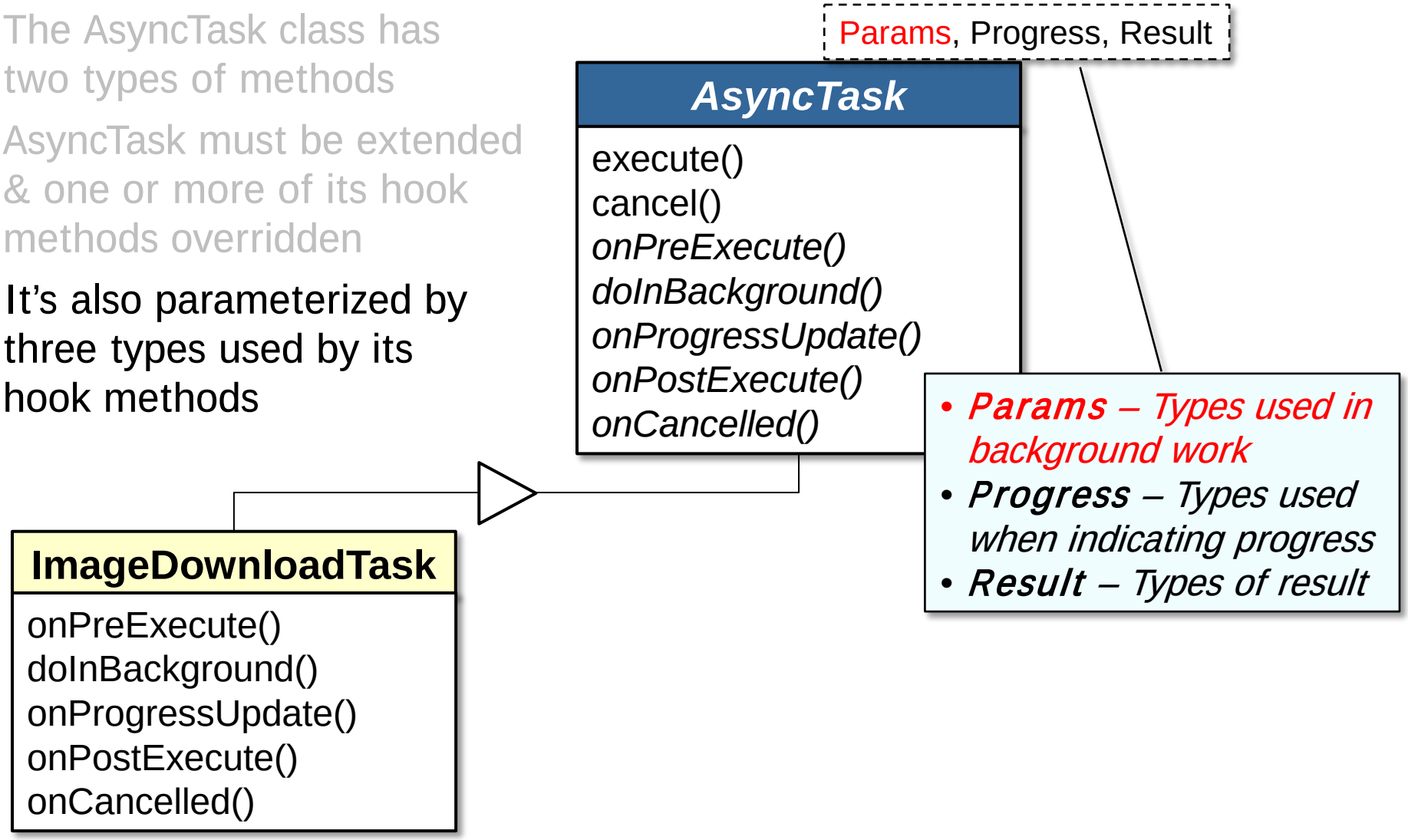
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods



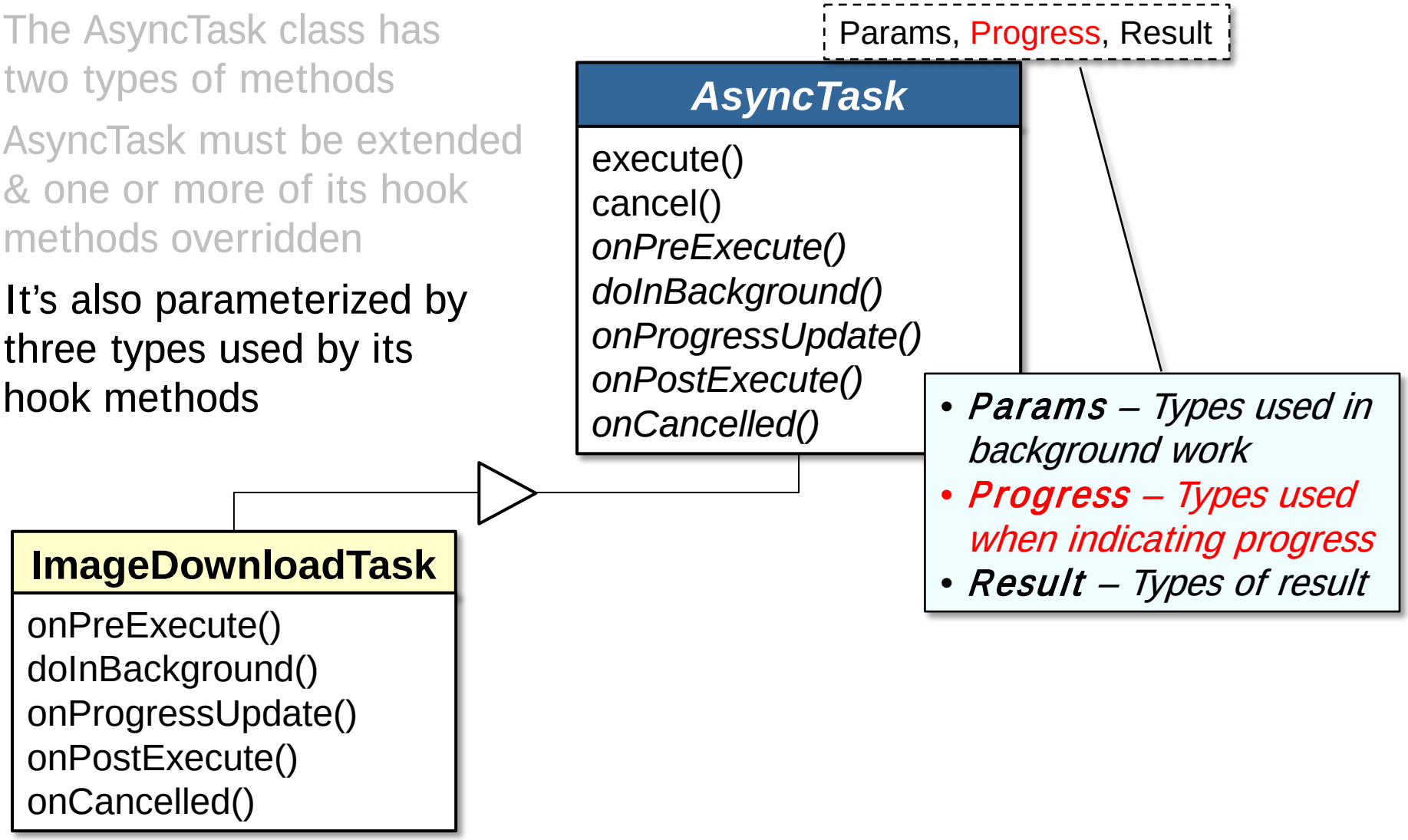
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods



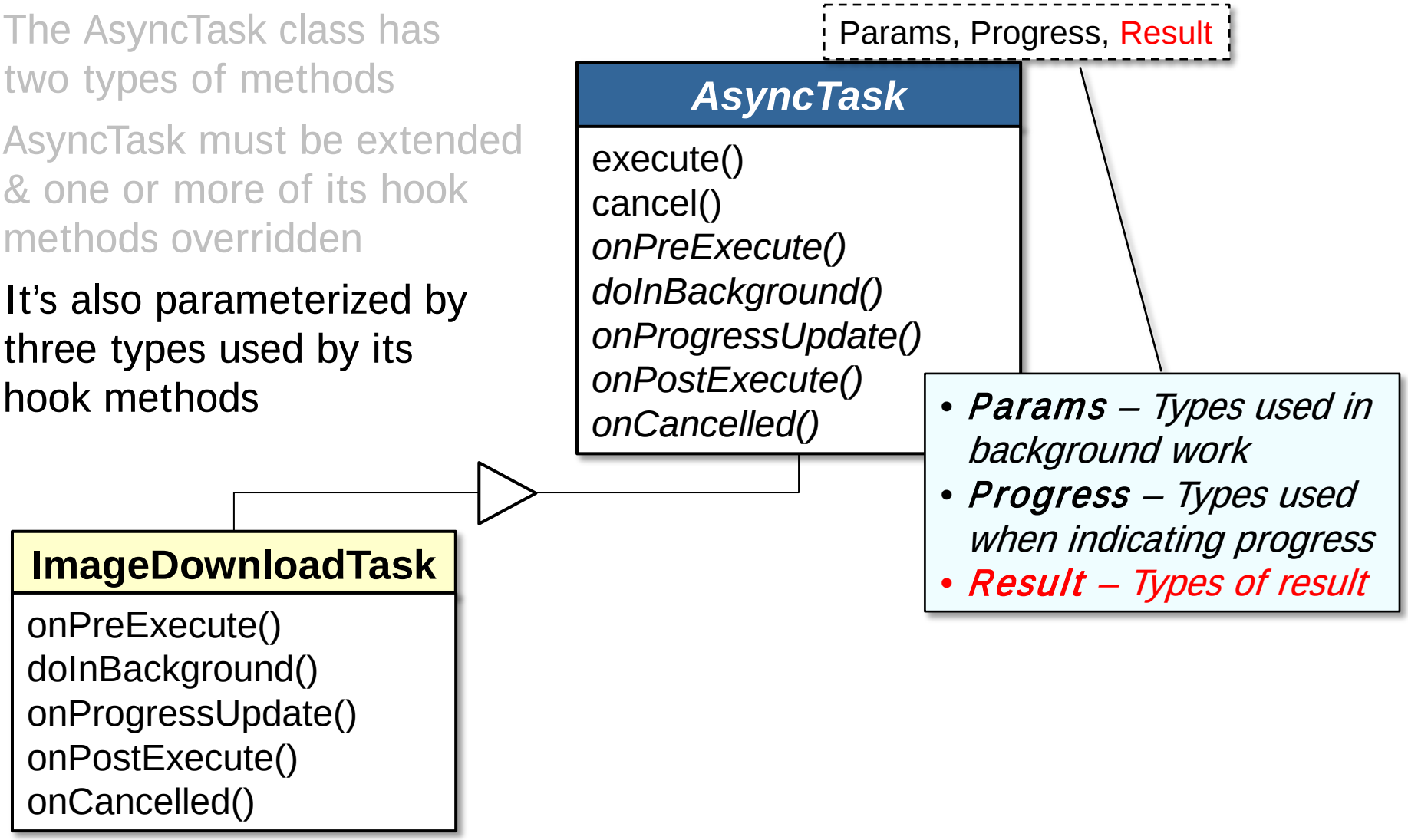
Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods



Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods



Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods
- Apps customize AsyncTask 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);
```

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods
- Apps customize AsyncTask 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);
```

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods
- Apps customize AsyncTask 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);
```

Categories of Methods in the AsyncTask Class

- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden
- It's also parameterized by three types used by its hook methods
- Apps customize AsyncTask 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