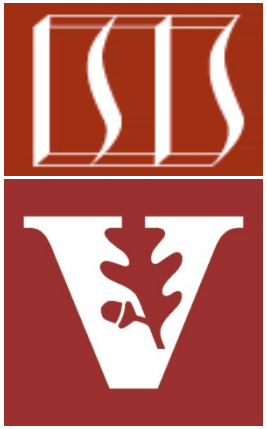
Android Concurrency: The AsyncTask Framework (Part 1)

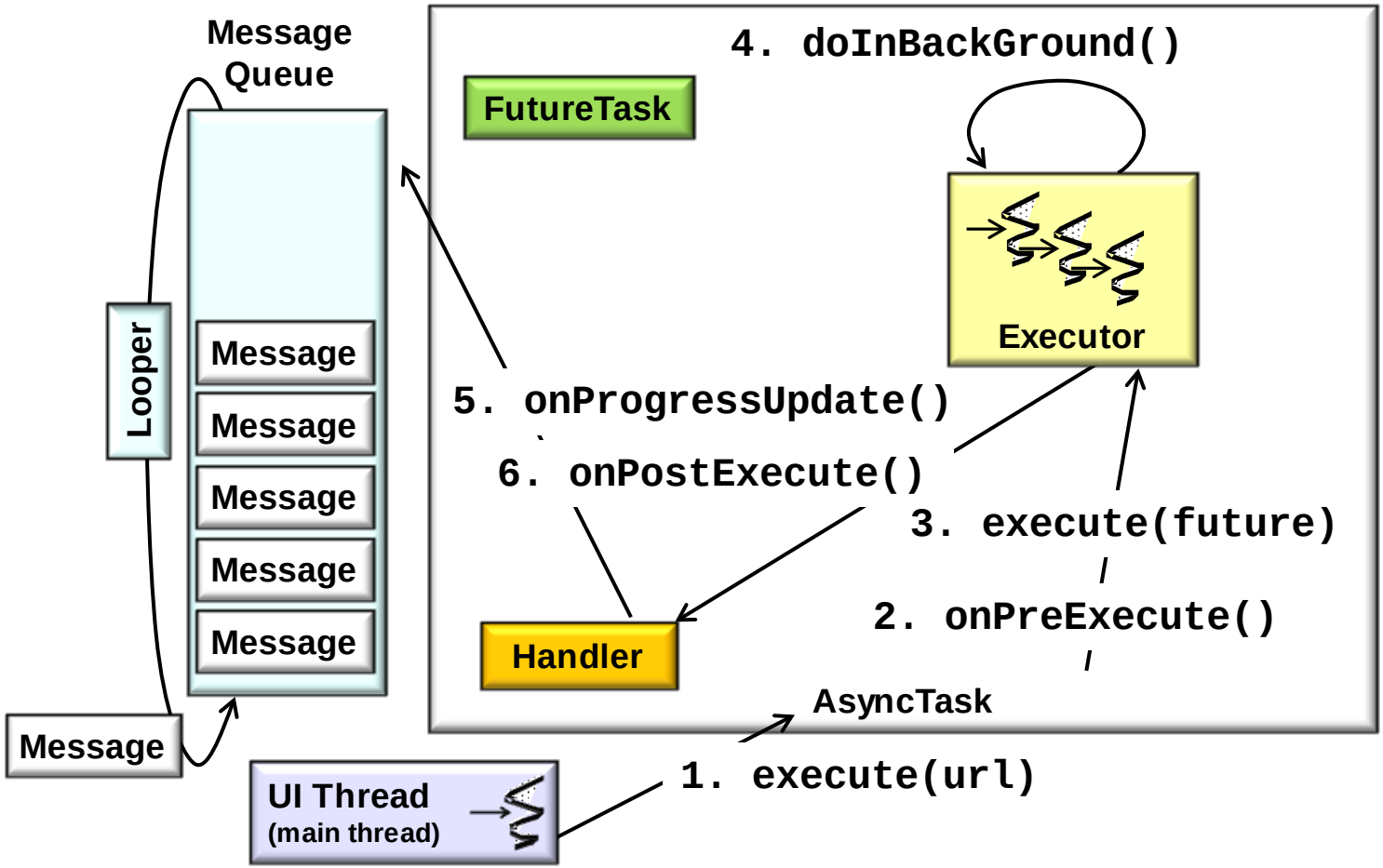


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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

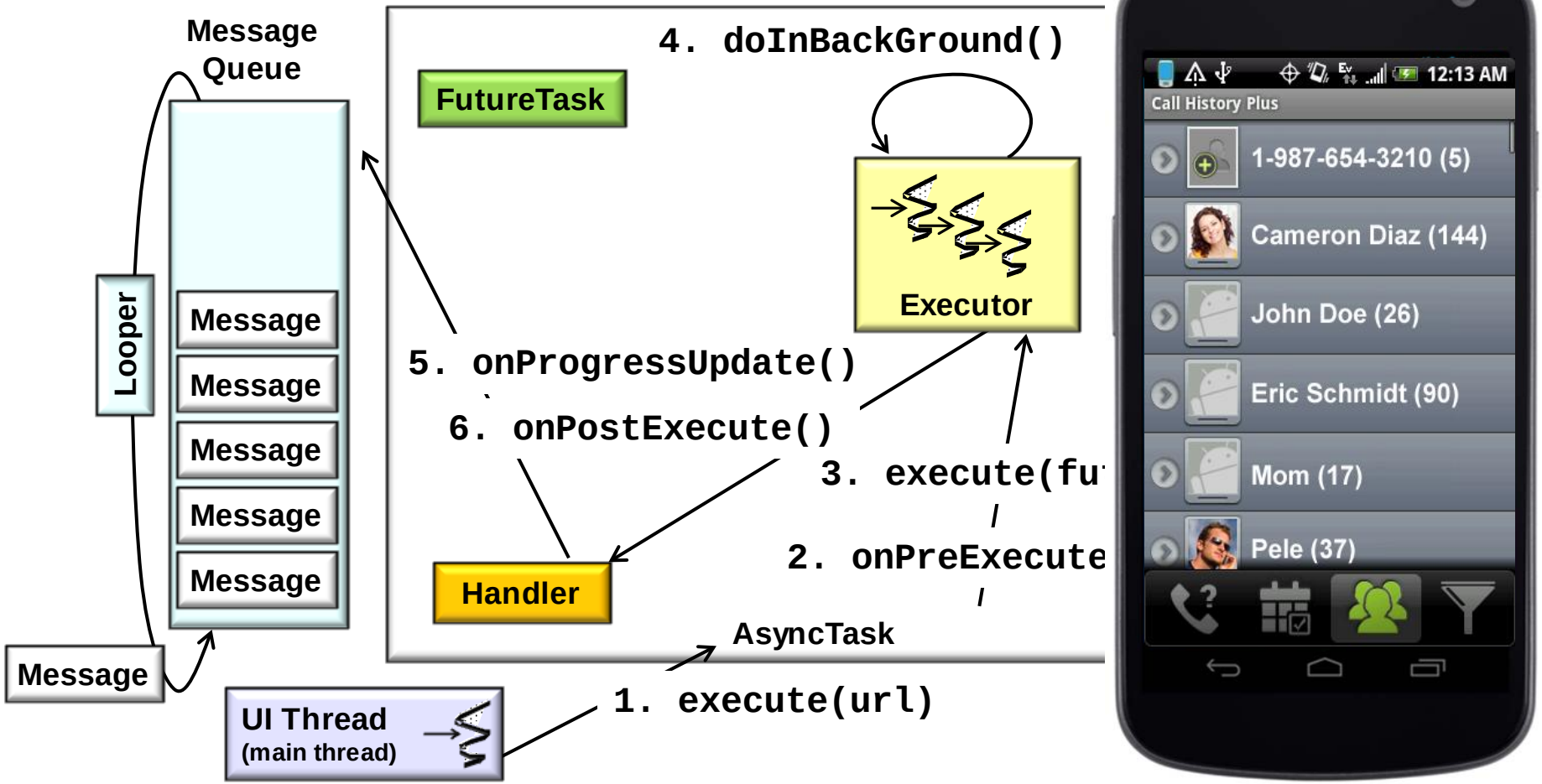
Learning Objectives in this Part of the Module

- Understand Android concurrency idioms & programming mechanisms related to the AsyncTask framework



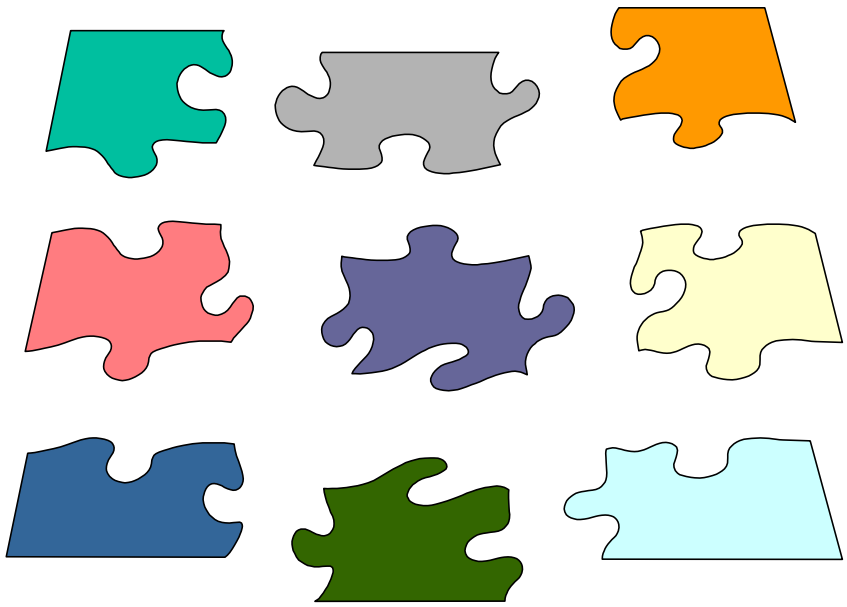
Learning Objectives in this Part of the Module

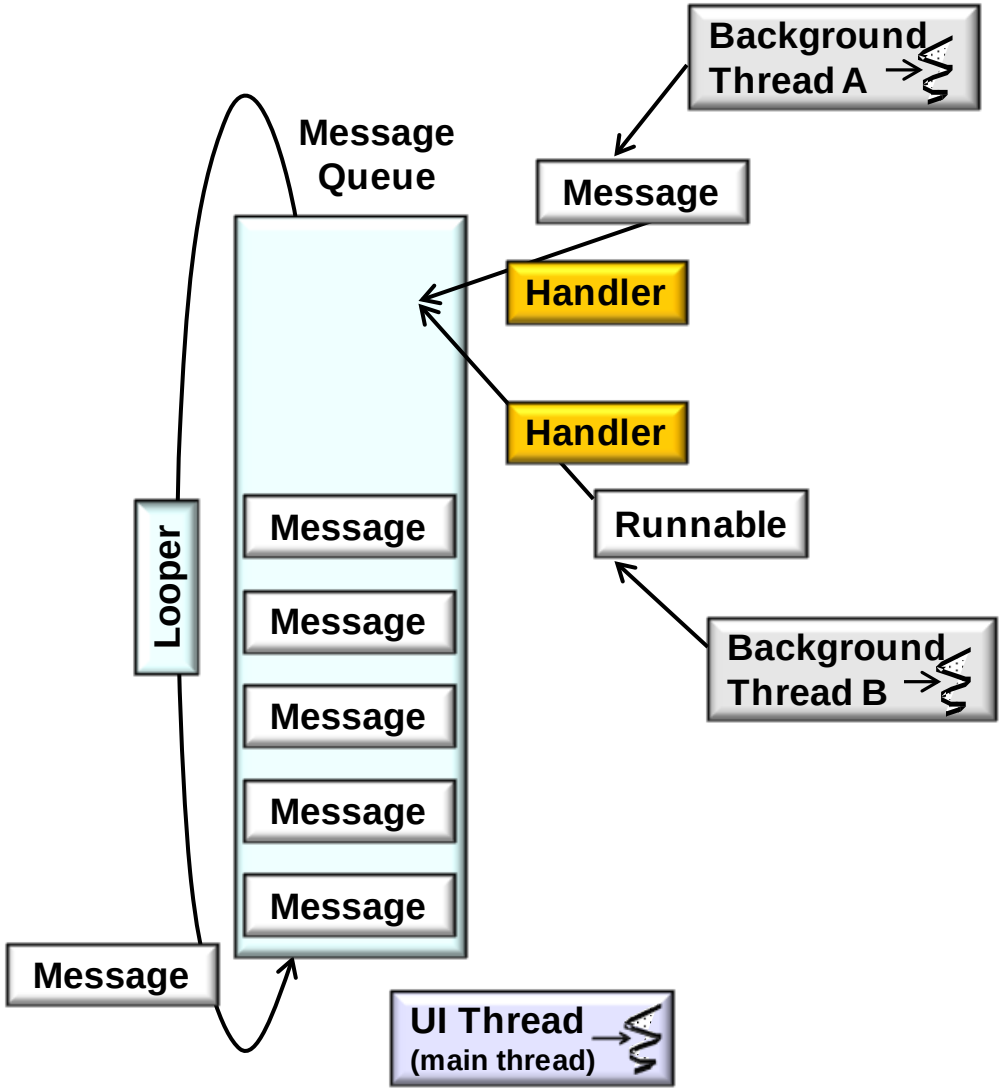
- Understand Android concurrency idioms & programming mechanisms related to the AsyncTask framework



Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected

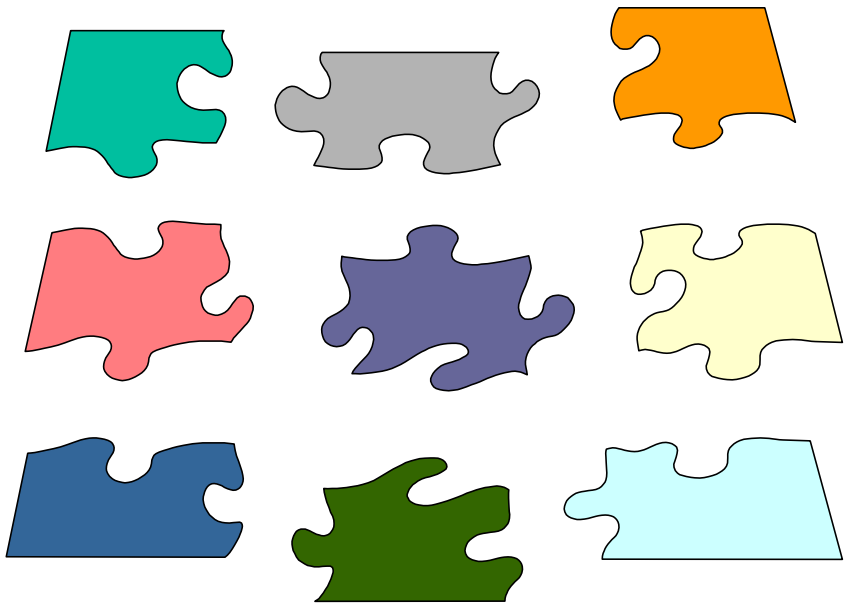


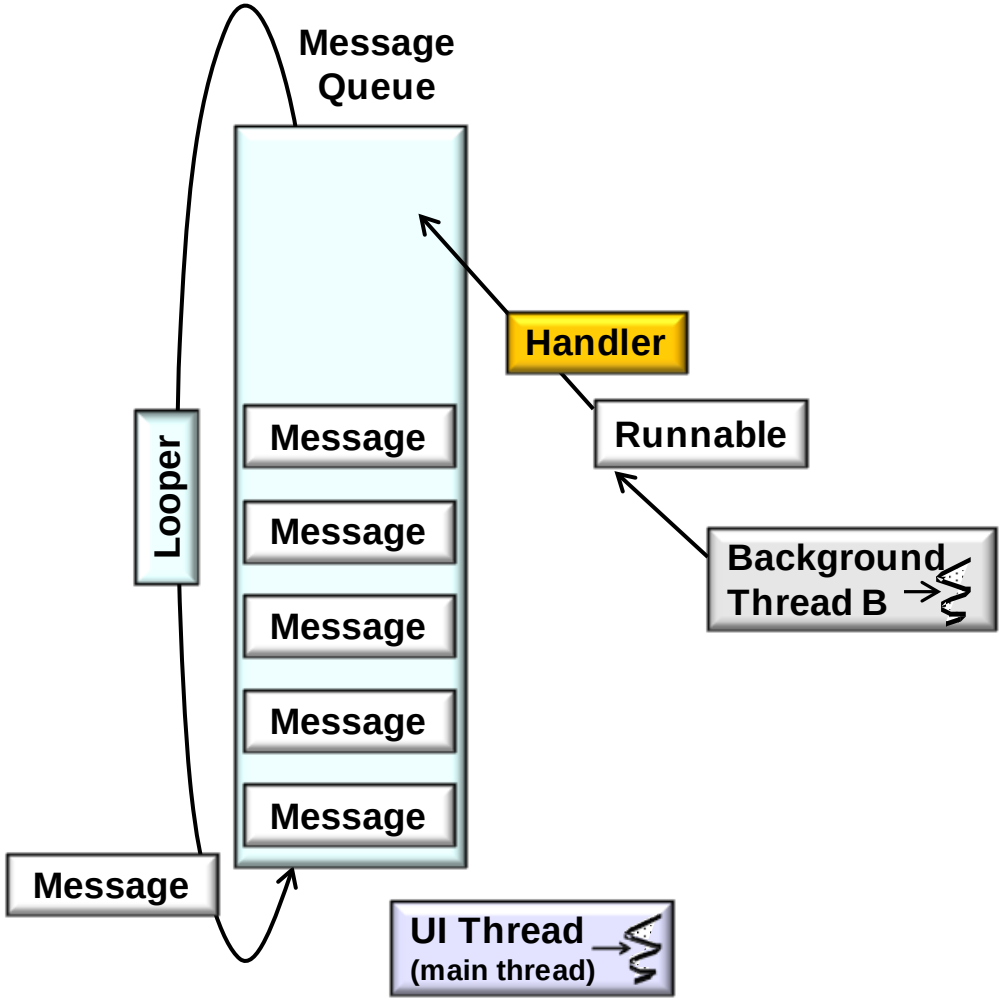


See previous parts on Posting Runnables & Sending Messages in HaMeR

Overview of the AsyncTask Framework

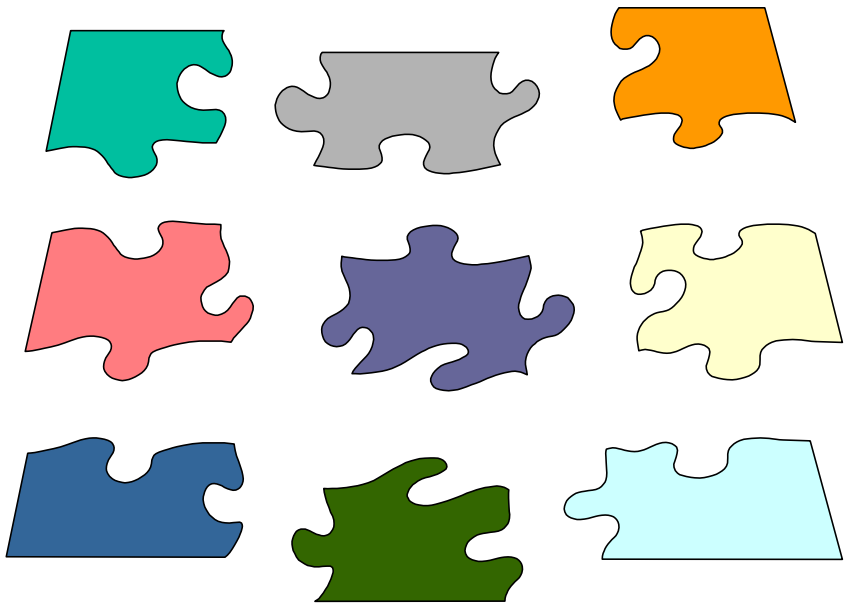
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple use cases

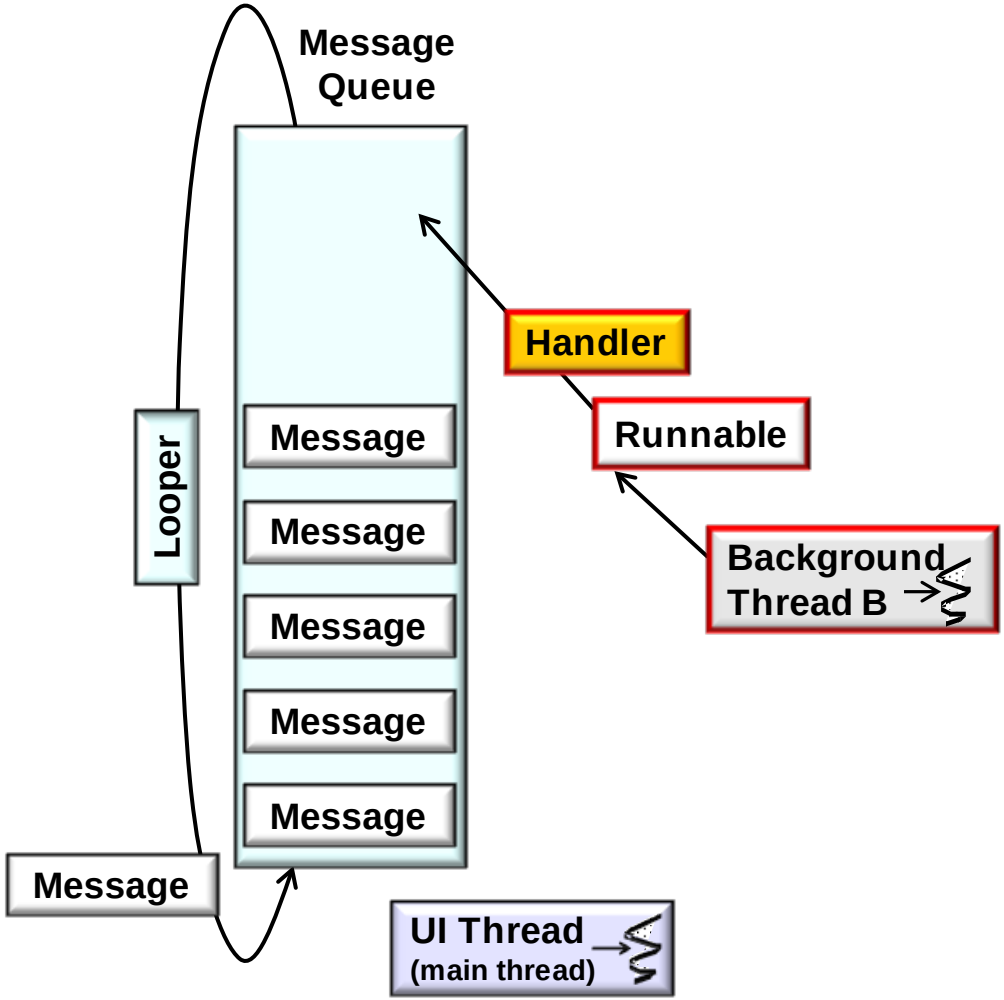




Overview of the AsyncTask Framework

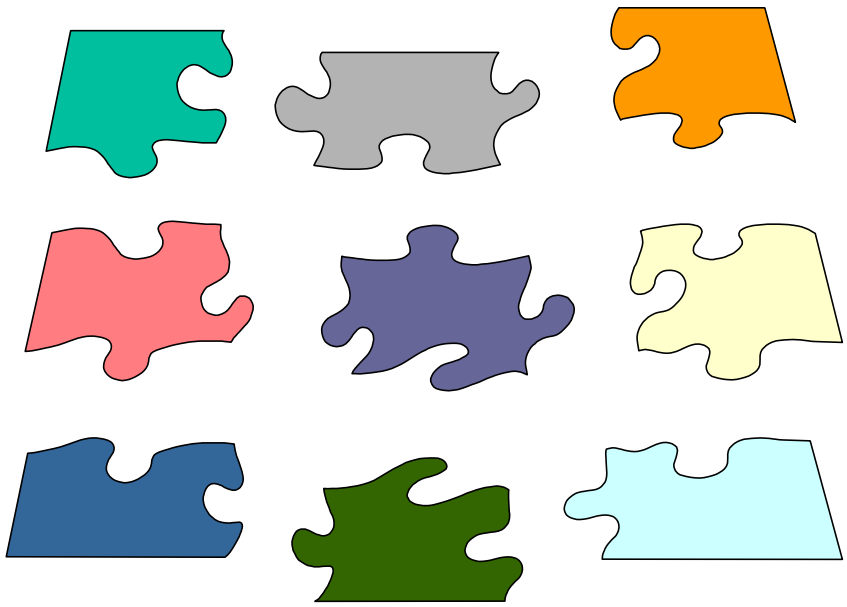
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple use cases

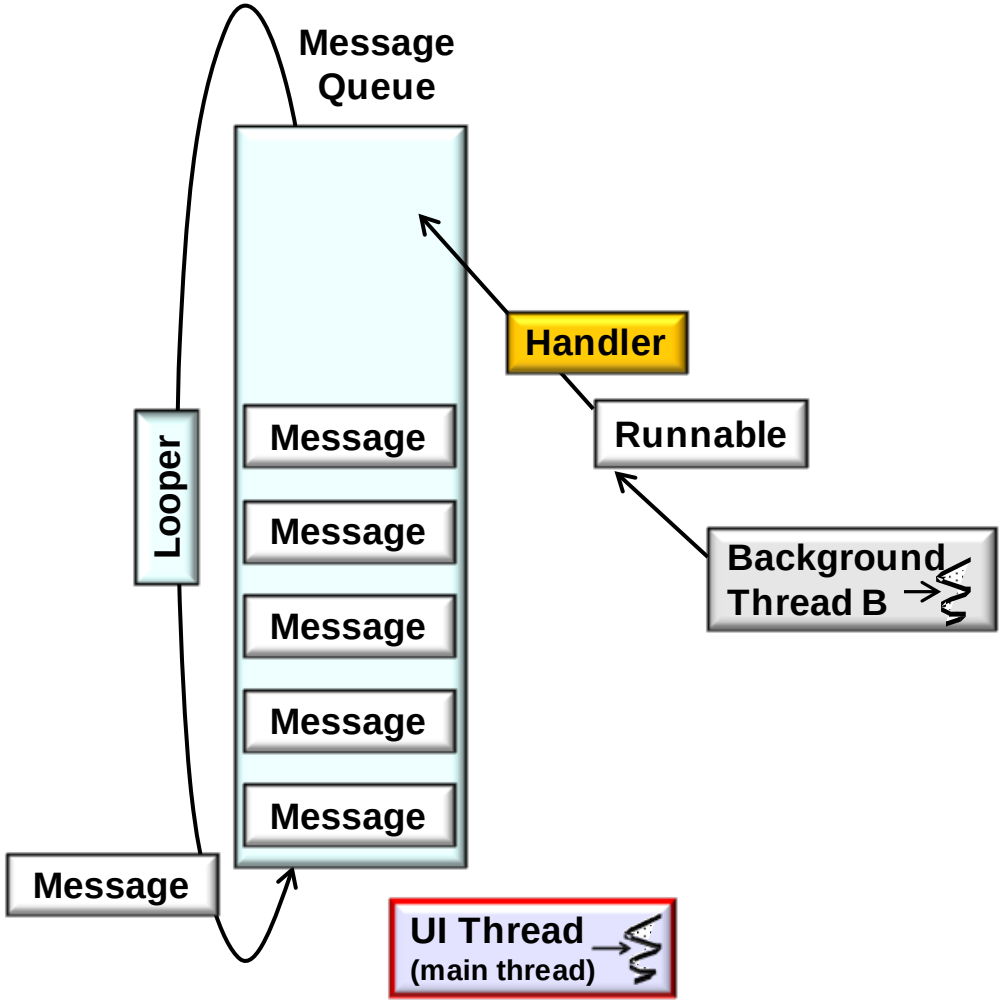




Overview of the AsyncTask Framework

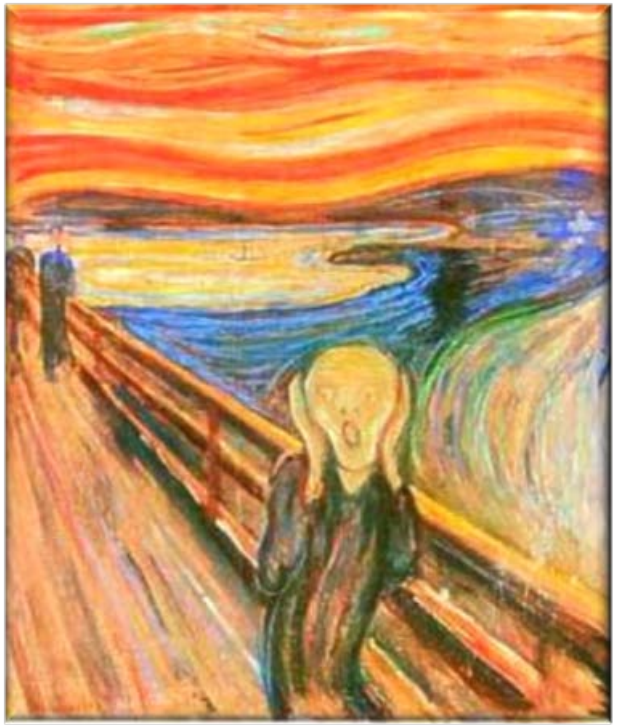
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple use cases

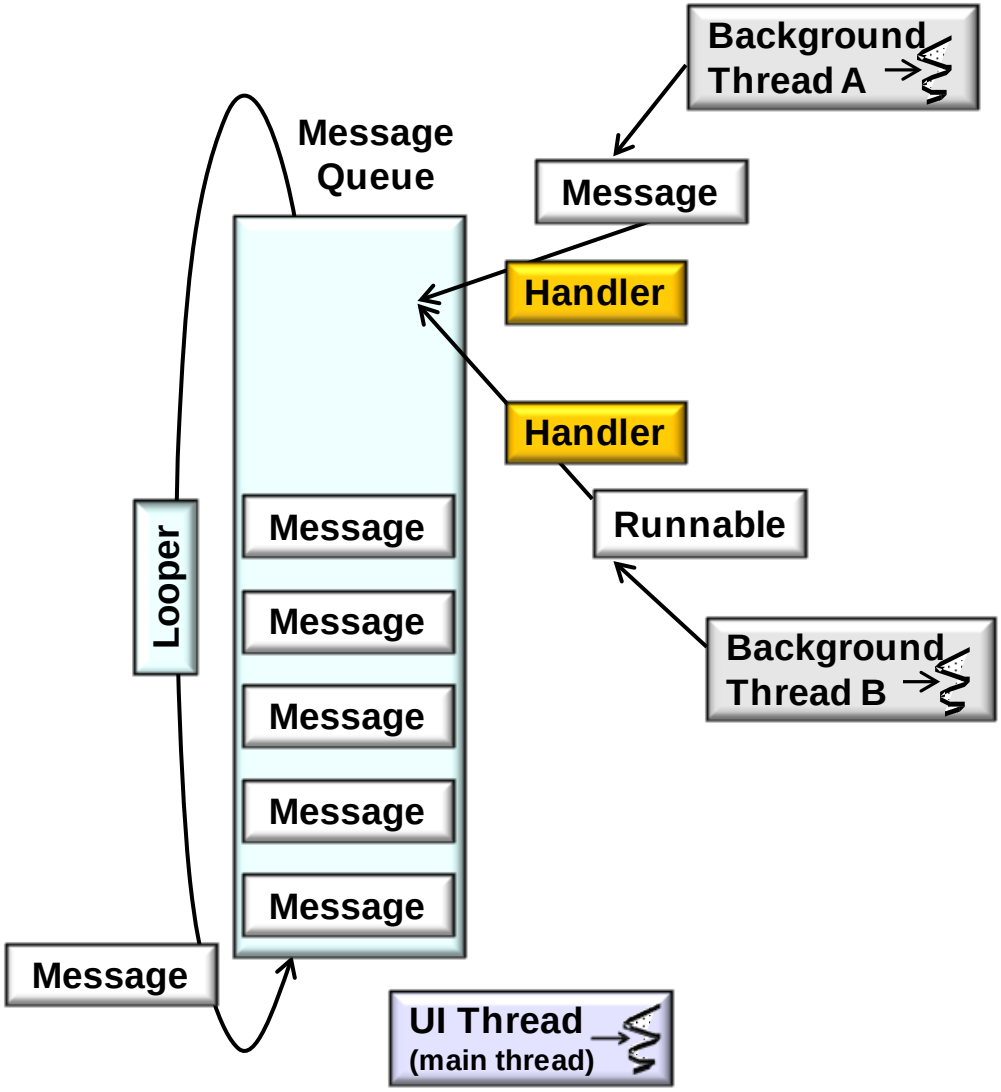




Overview of the AsyncTask Framework

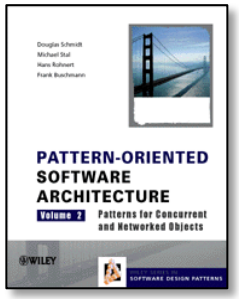
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple use cases
  - However, there are drawbacks

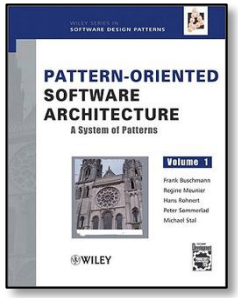


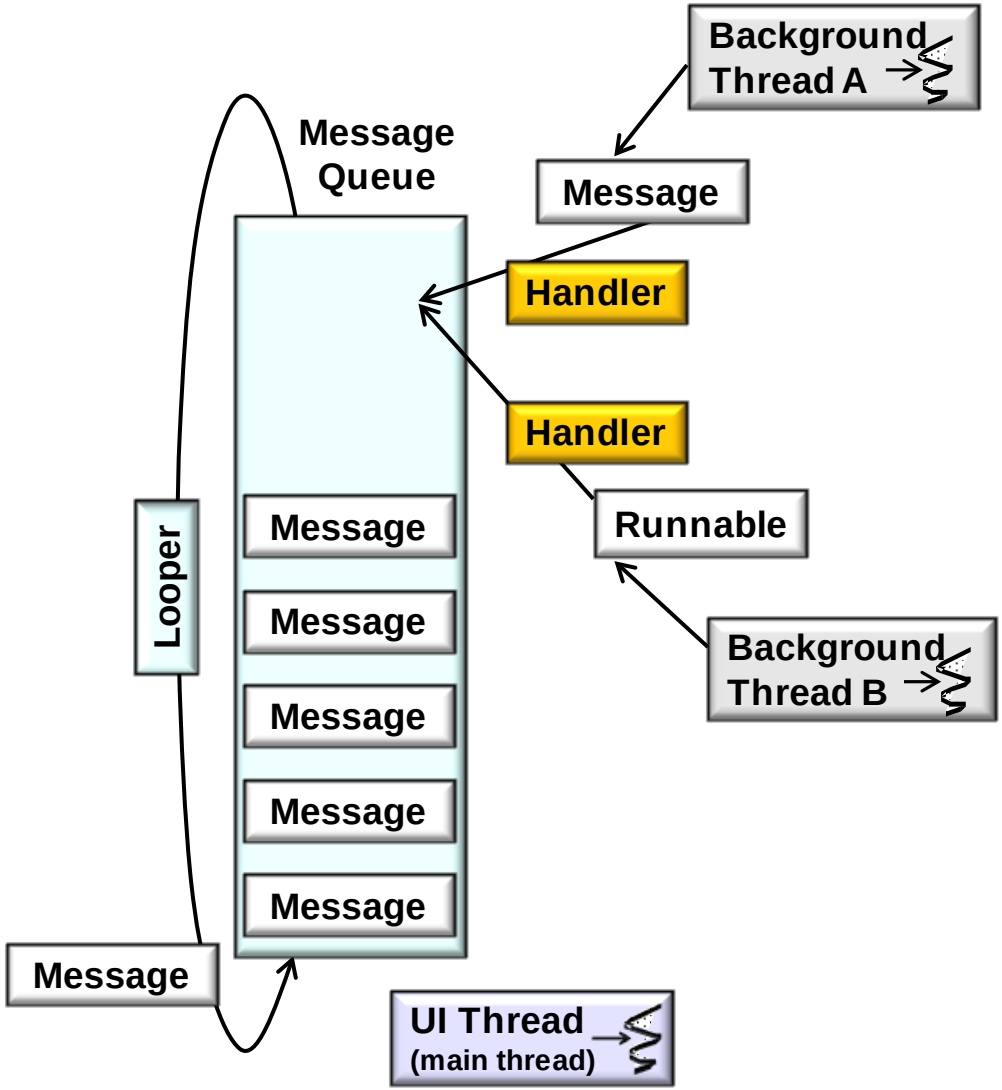


Overview of the AsyncTask Framework

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

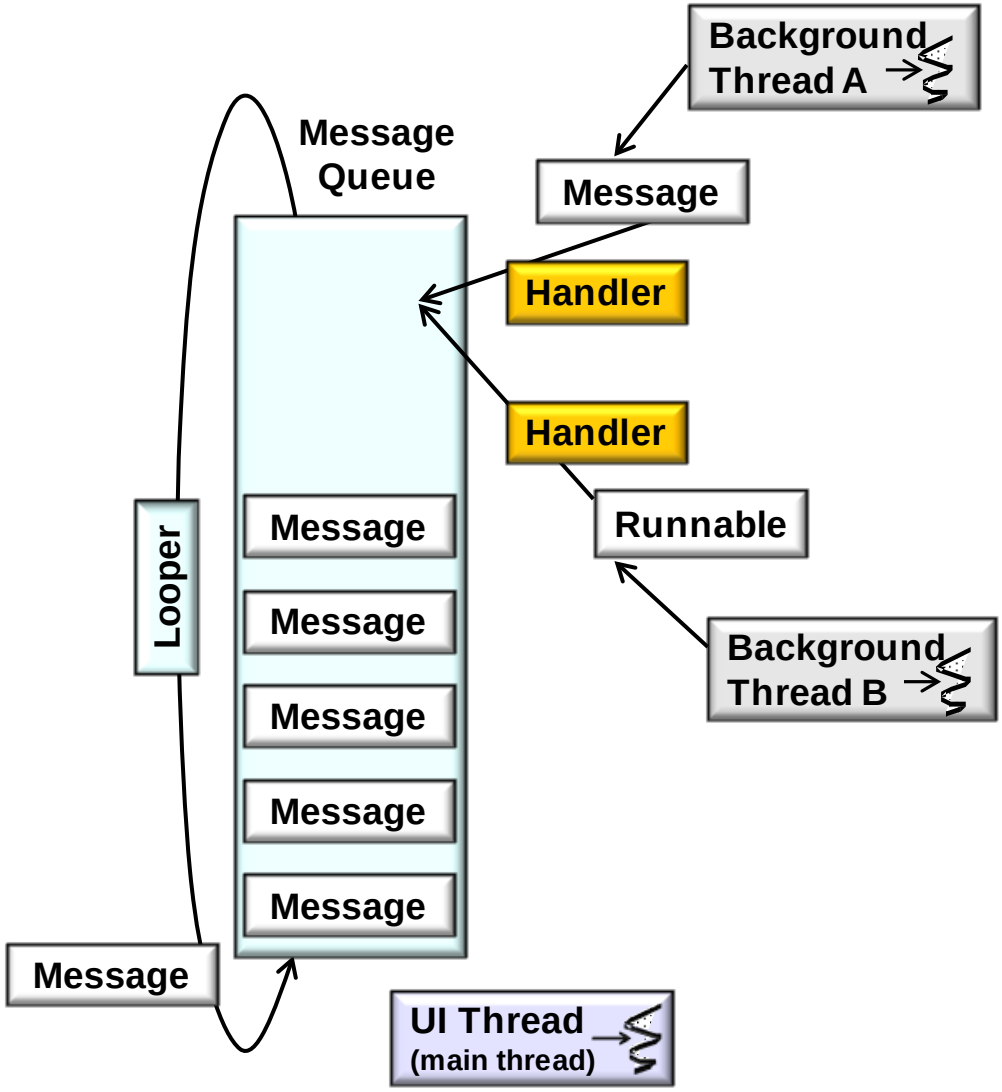






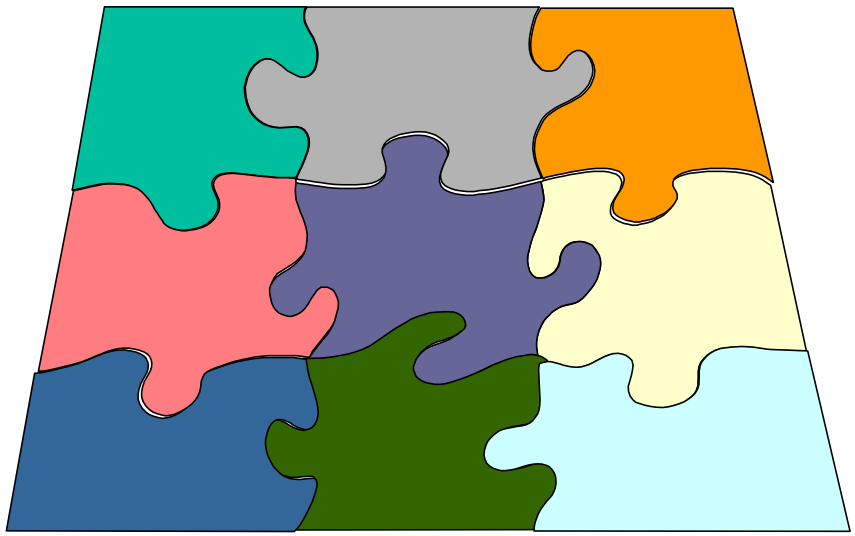
Overview of the AsyncTask Framework

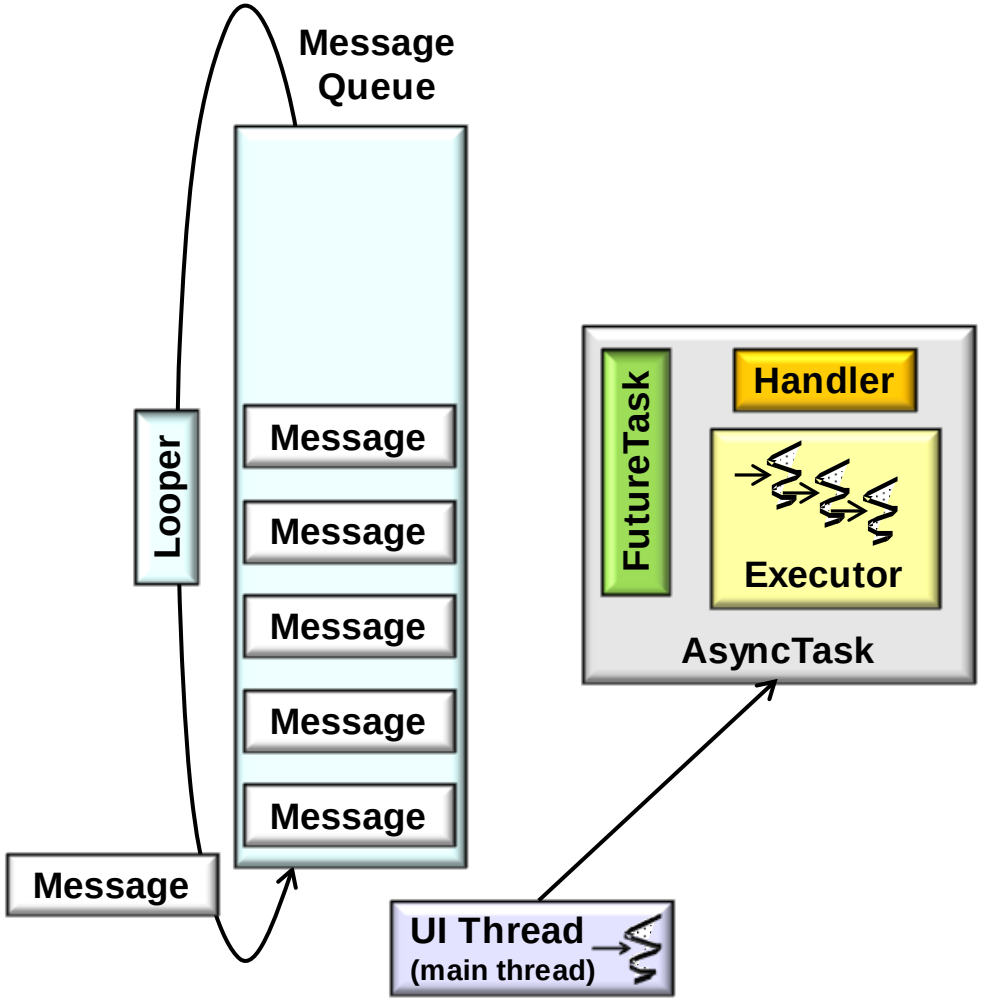
- Classes in HaMeR framework are loosely connected
  - This flexibility works well for simple use cases
  - However, there are drawbacks
    - Must understand patterns
    - Tedious & error-prone



Overview of the AsyncTask Framework

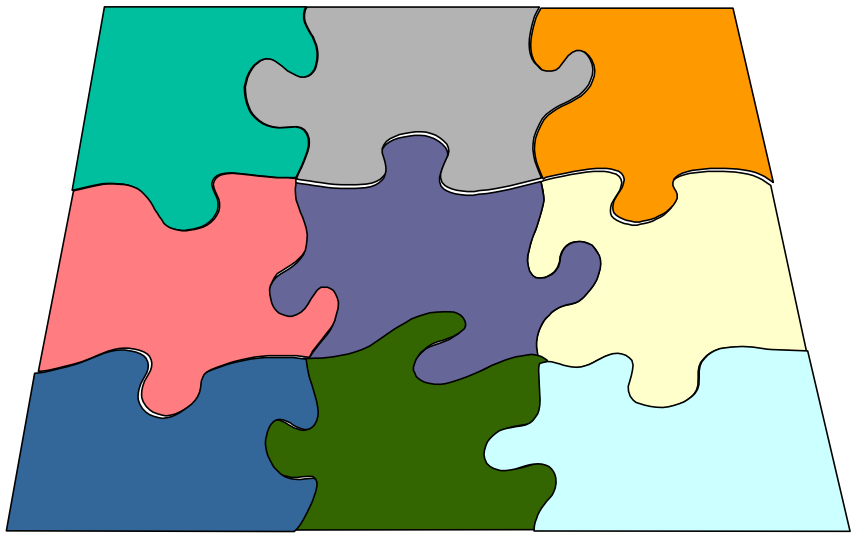
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

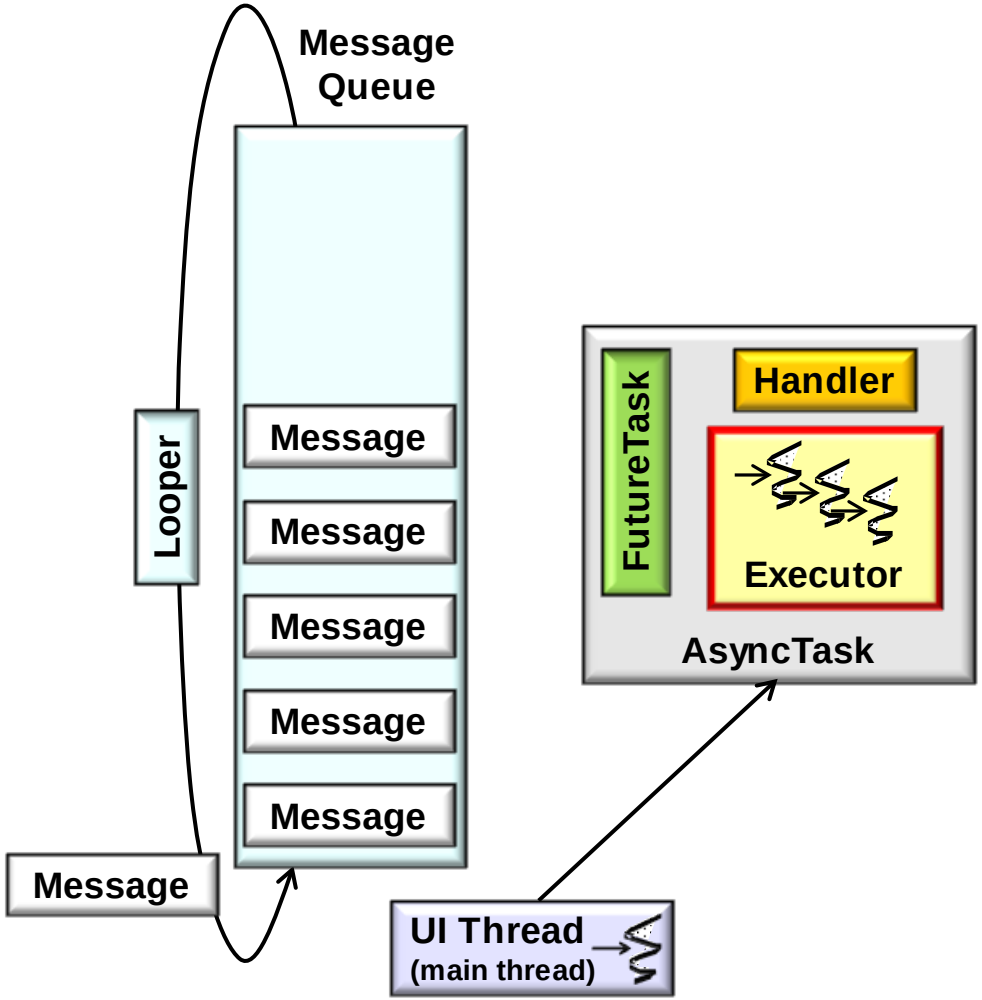




Overview of the AsyncTask Framework

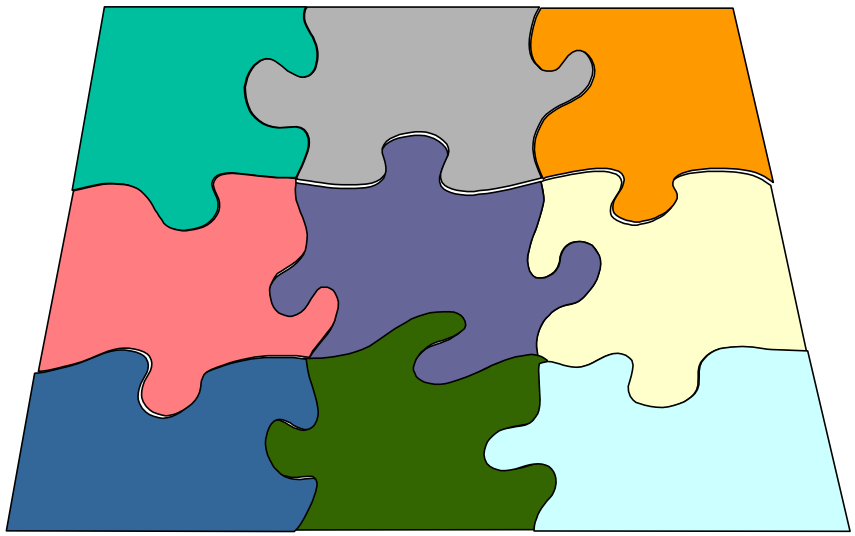
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

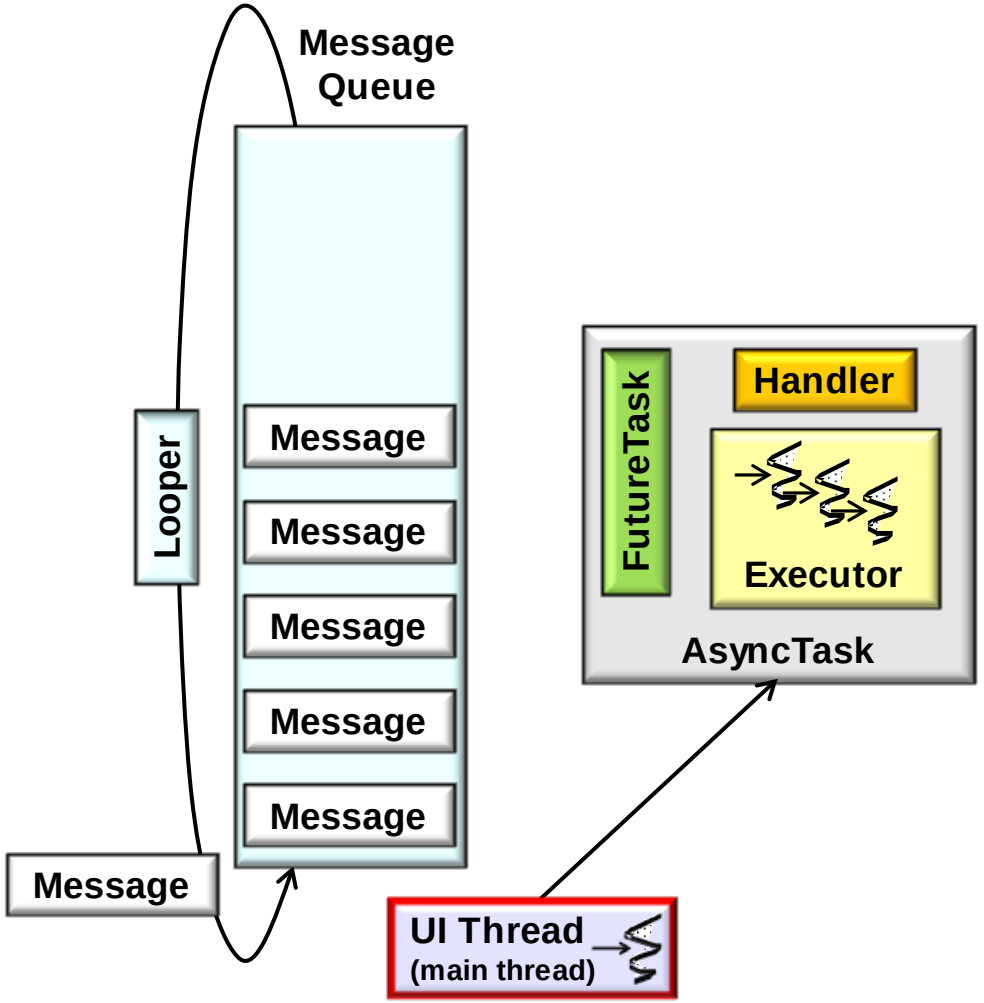




Overview of the AsyncTask Framework

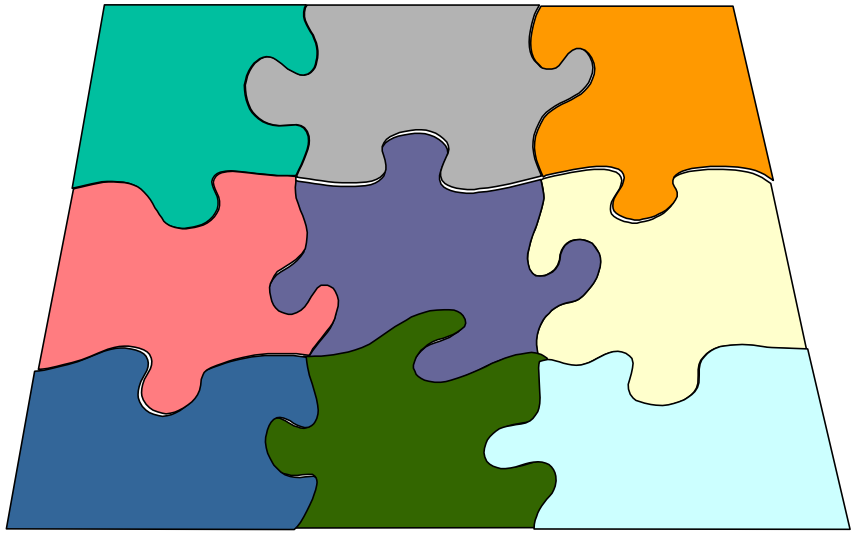
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected

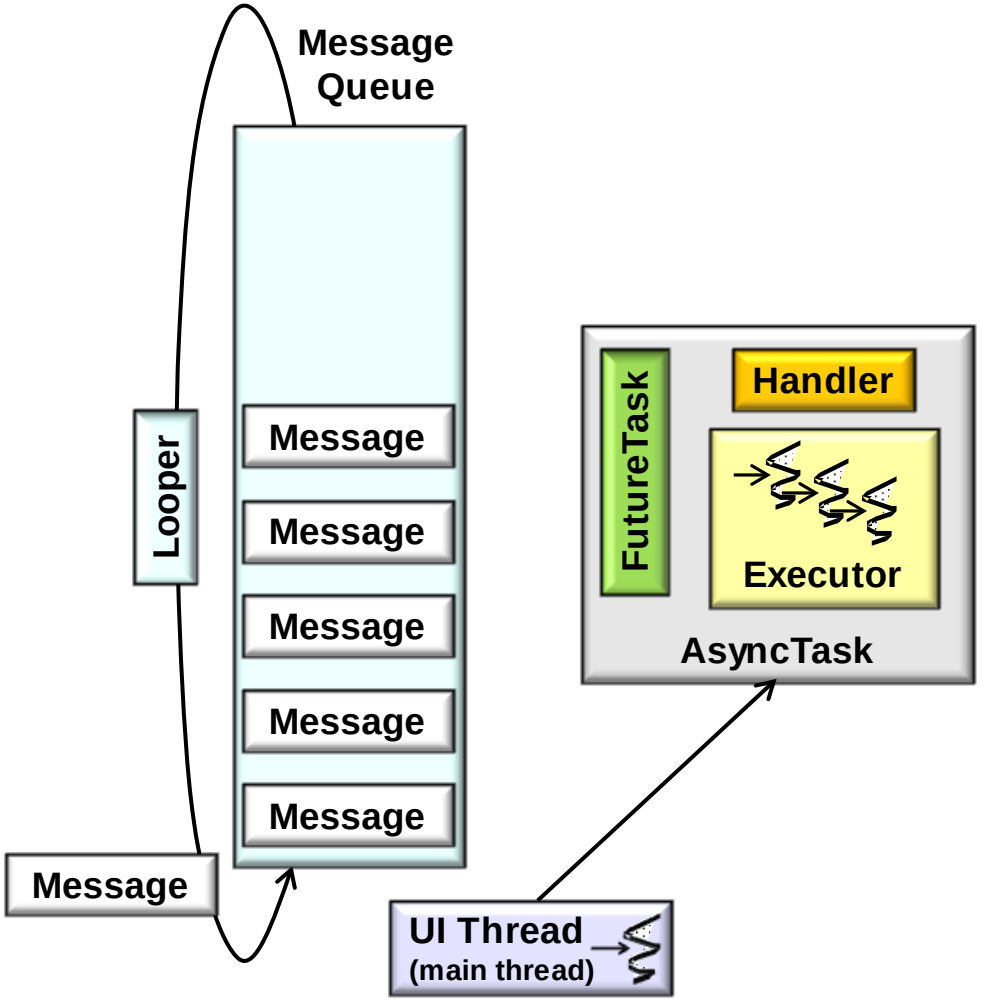




Overview of the AsyncTask Framework

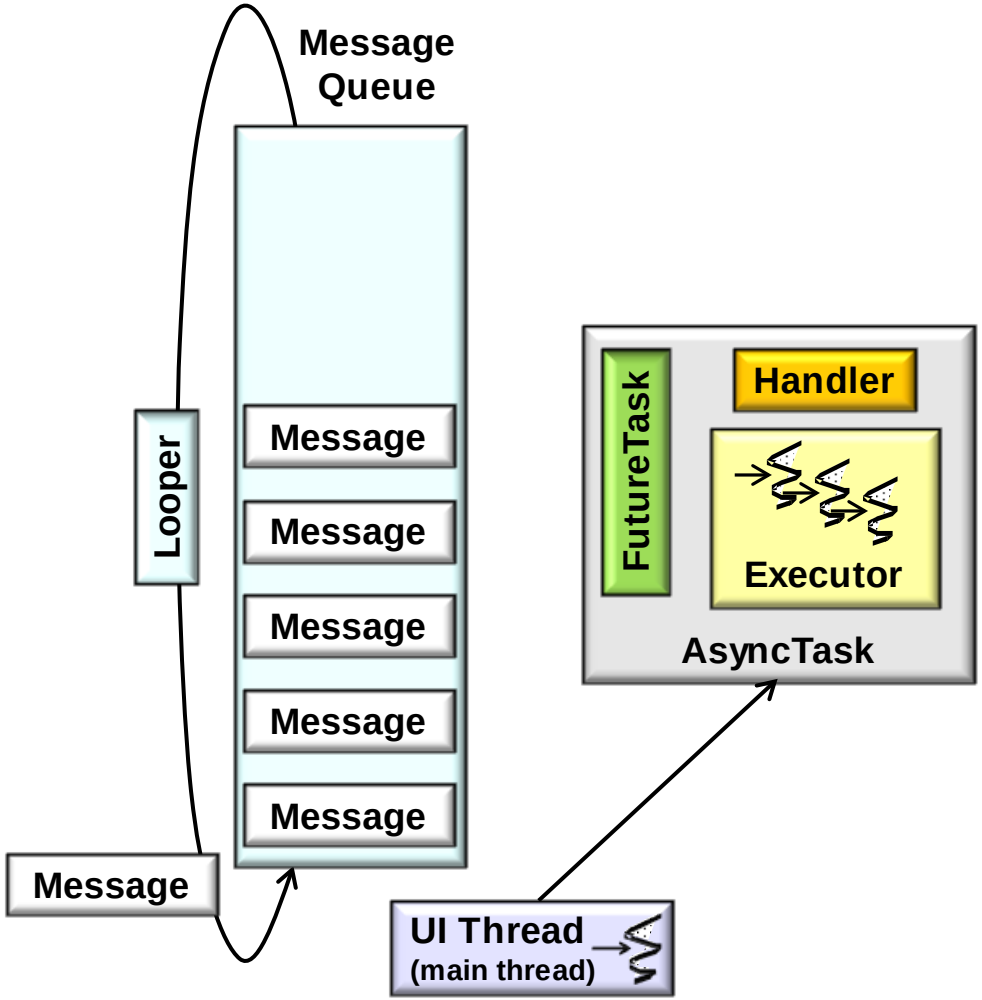
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - No direct manipulation of Threads, Handlers, Message, or Runnables





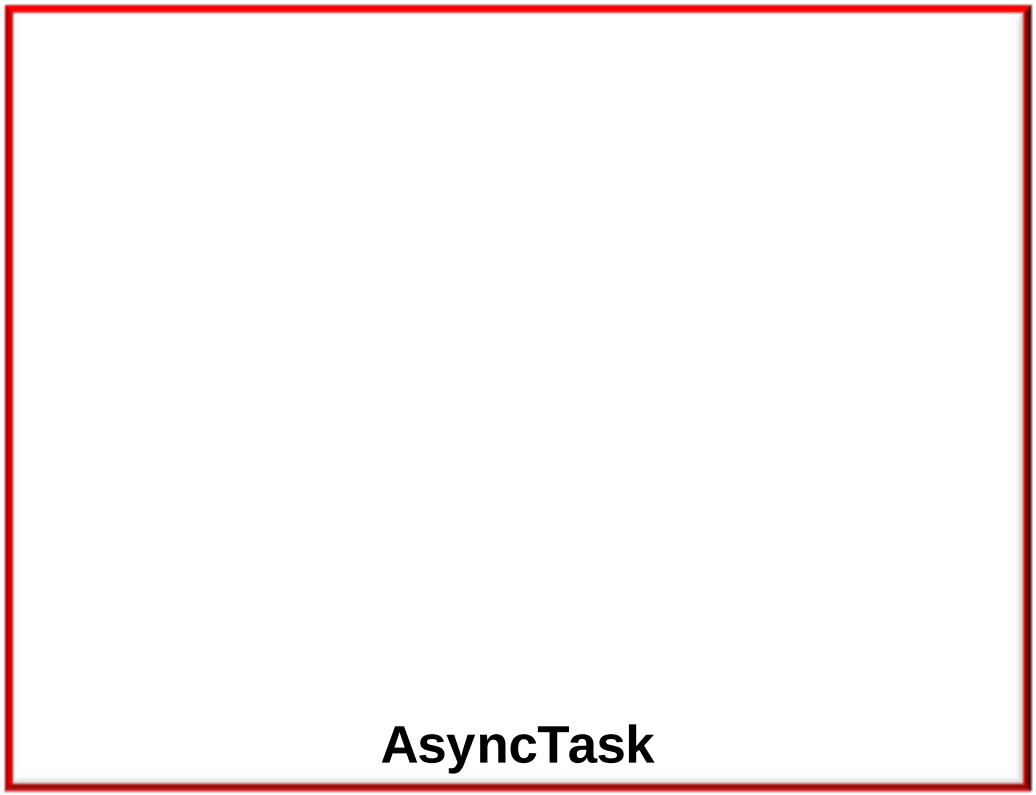
Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - No direct manipulation of Threads, Handlers, Message, or Runnables
  - Smaller “surface area”



Overview of the AsyncTask Framework

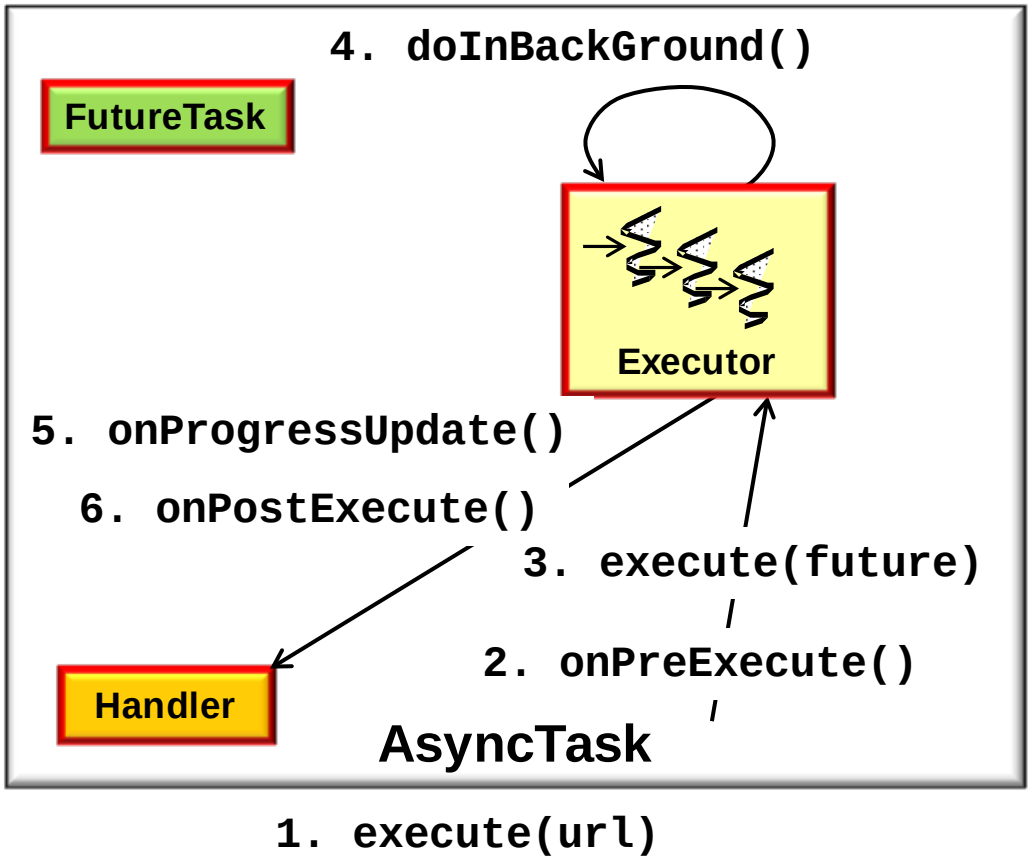
- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - No direct manipulation of Threads, Handlers, Message, or Runnables
  - Smaller “surface area”
  - Complex framework details are accessed via a facade




```
1. execute(url)
```

Overview of the AsyncTask Framework

- Classes in HaMeR framework are loosely connected
- Classes in AsyncTask framework are strongly connected
  - No direct manipulation of Threads, Handlers, Message, or Runnables
  - Smaller "surface area"
  - Complex framework details are accessed via a facade

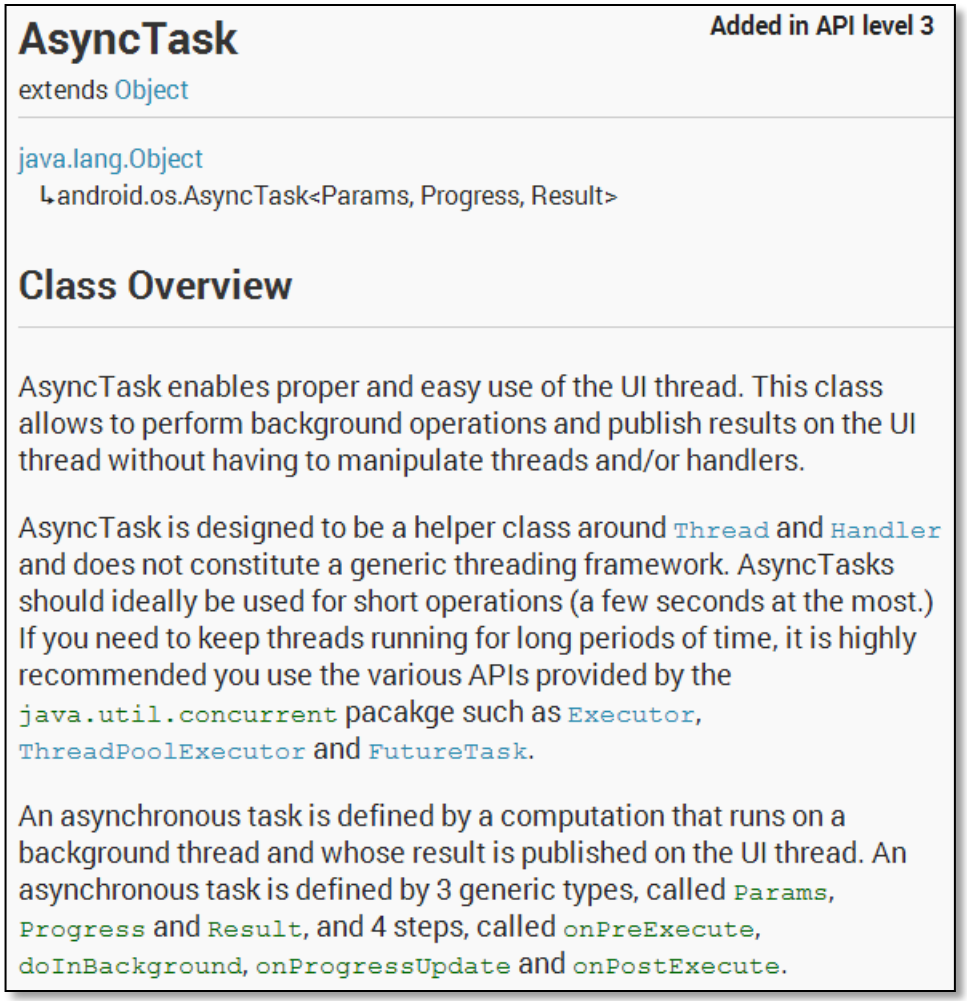


en.wikipedia.org/wiki/Facade_pattern has more on *Facade* pattern

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`.

developer.android.com/reference/android/os/AsyncTask.html has more info

Categories of Methods in the AsyncTask Class

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

`static void execute(Runnable runnable)`

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

`final 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 applications

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

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

```
final 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 applications

`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
  - Protected methods
    - Invoked by framework at different points of time & in different 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
  - Public methods
  - Protected methods
    - Invoked by framework at different points of time & in different 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()

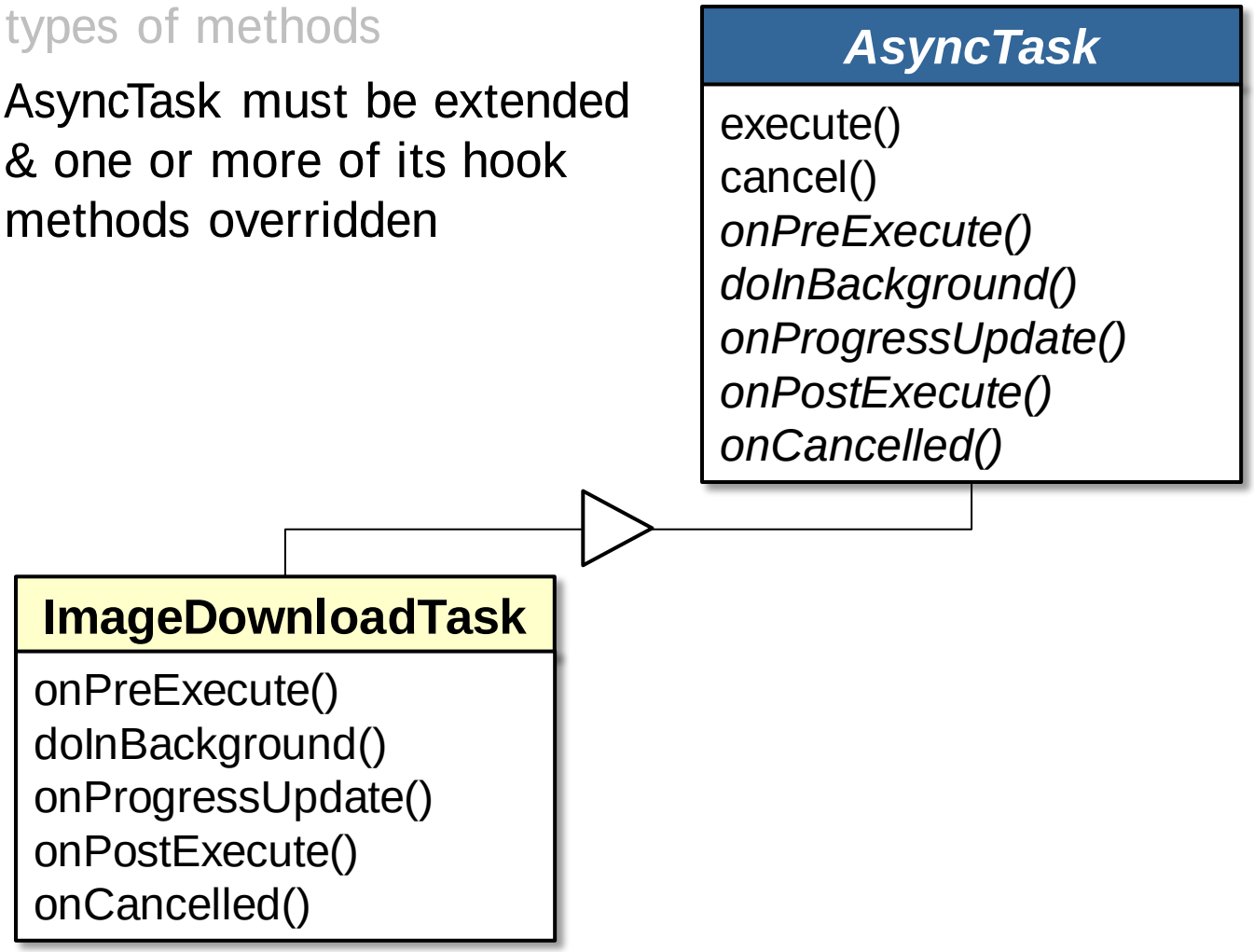
`void onProgressUpdate(Progress... values)`
- Runs on UI thread after publishProgress() called

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

...

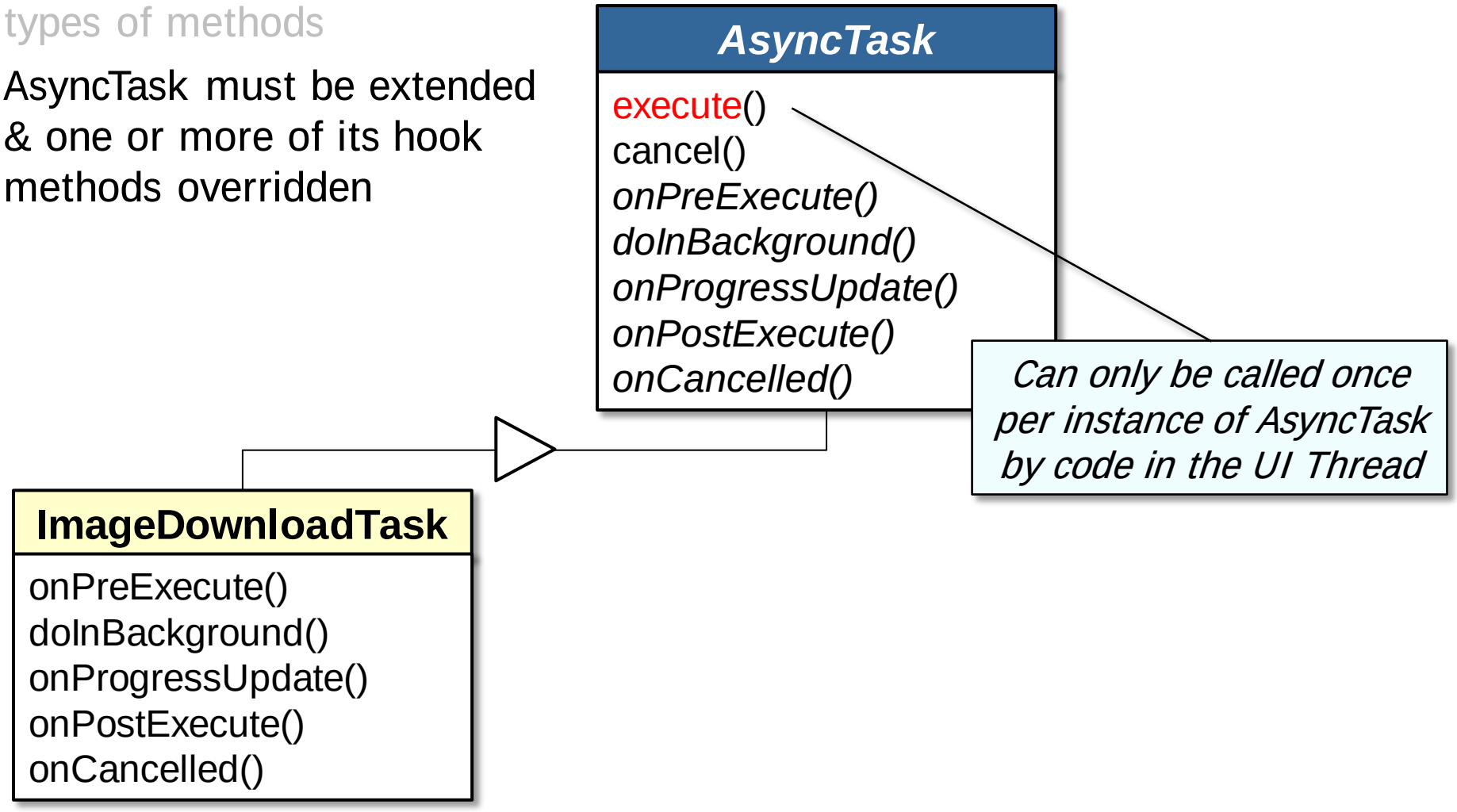
Categories of Methods in the AsyncTask Class

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



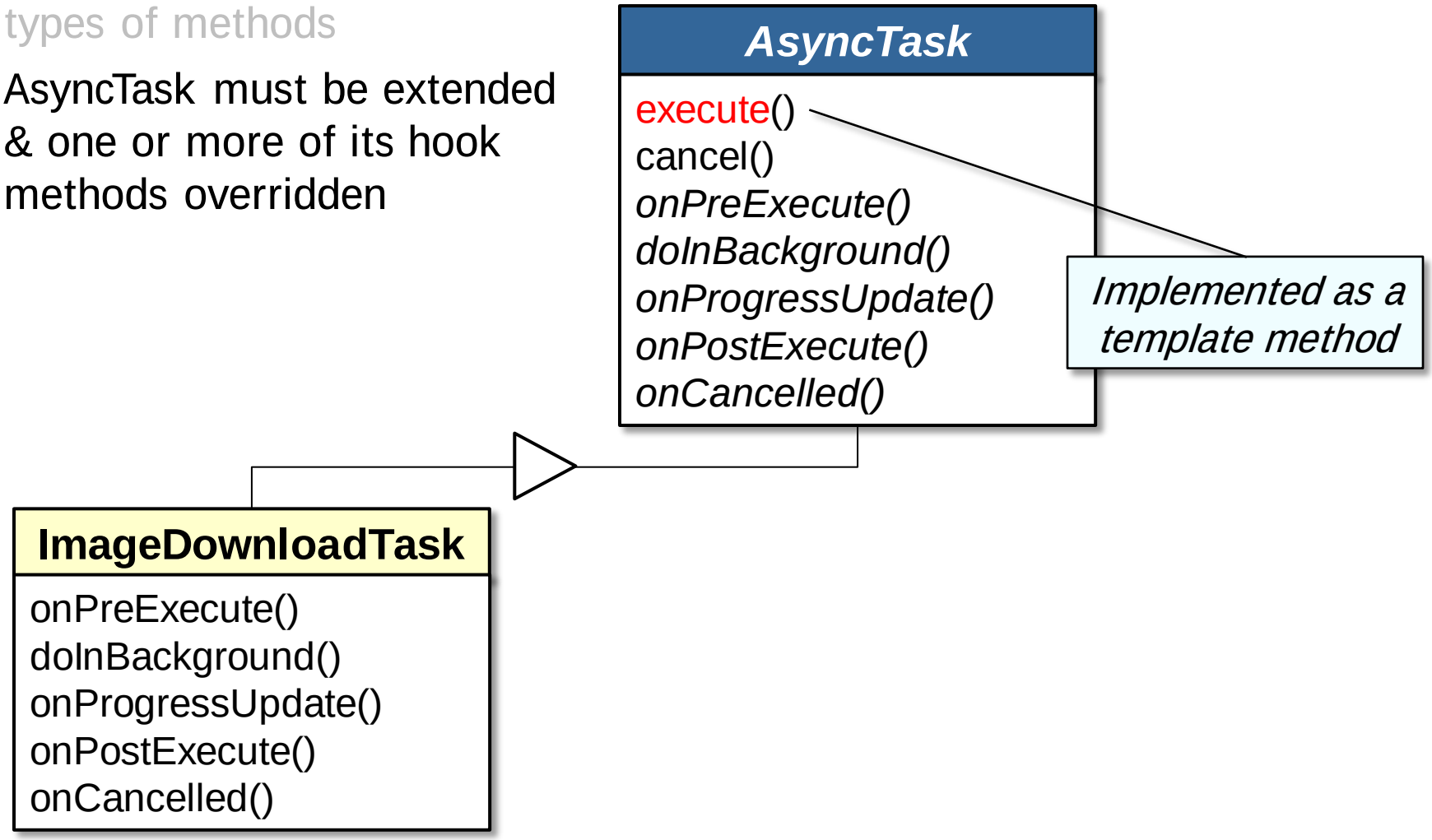
Categories of Methods in the AsyncTask Class

- 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

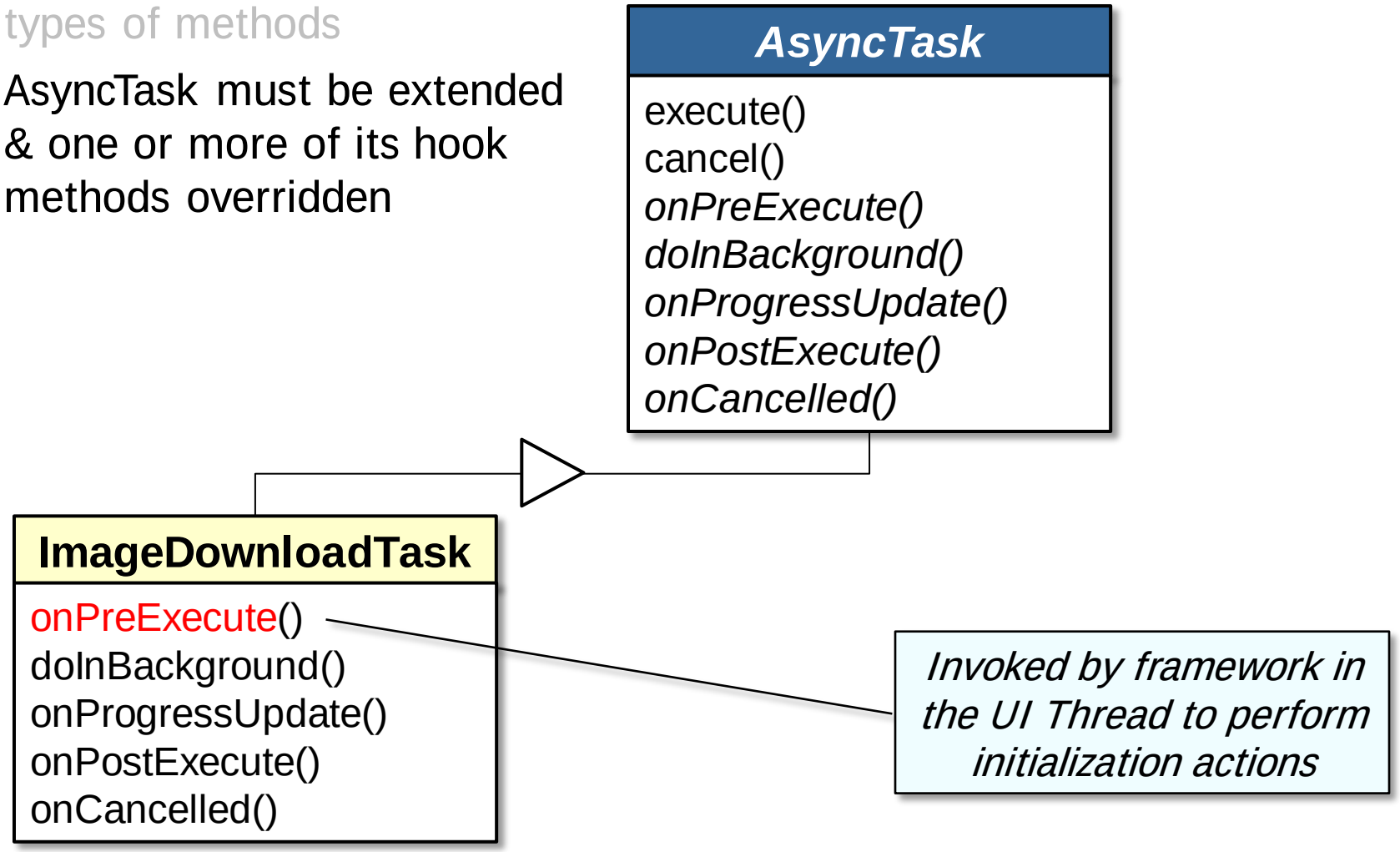
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



en.wikipedia.org/wiki/Template_method_pattern has more info

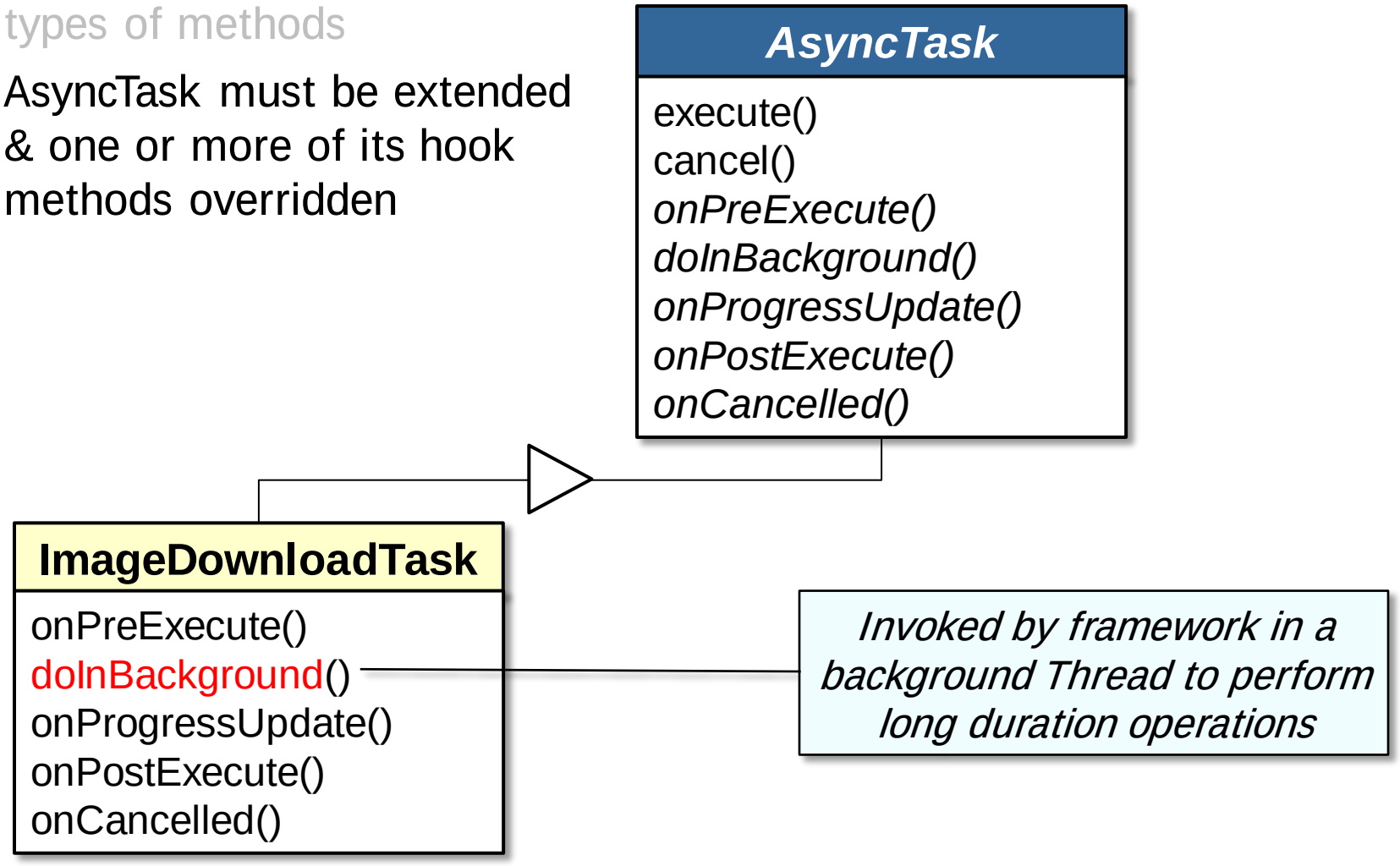
Categories of Methods in the AsyncTask Class

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



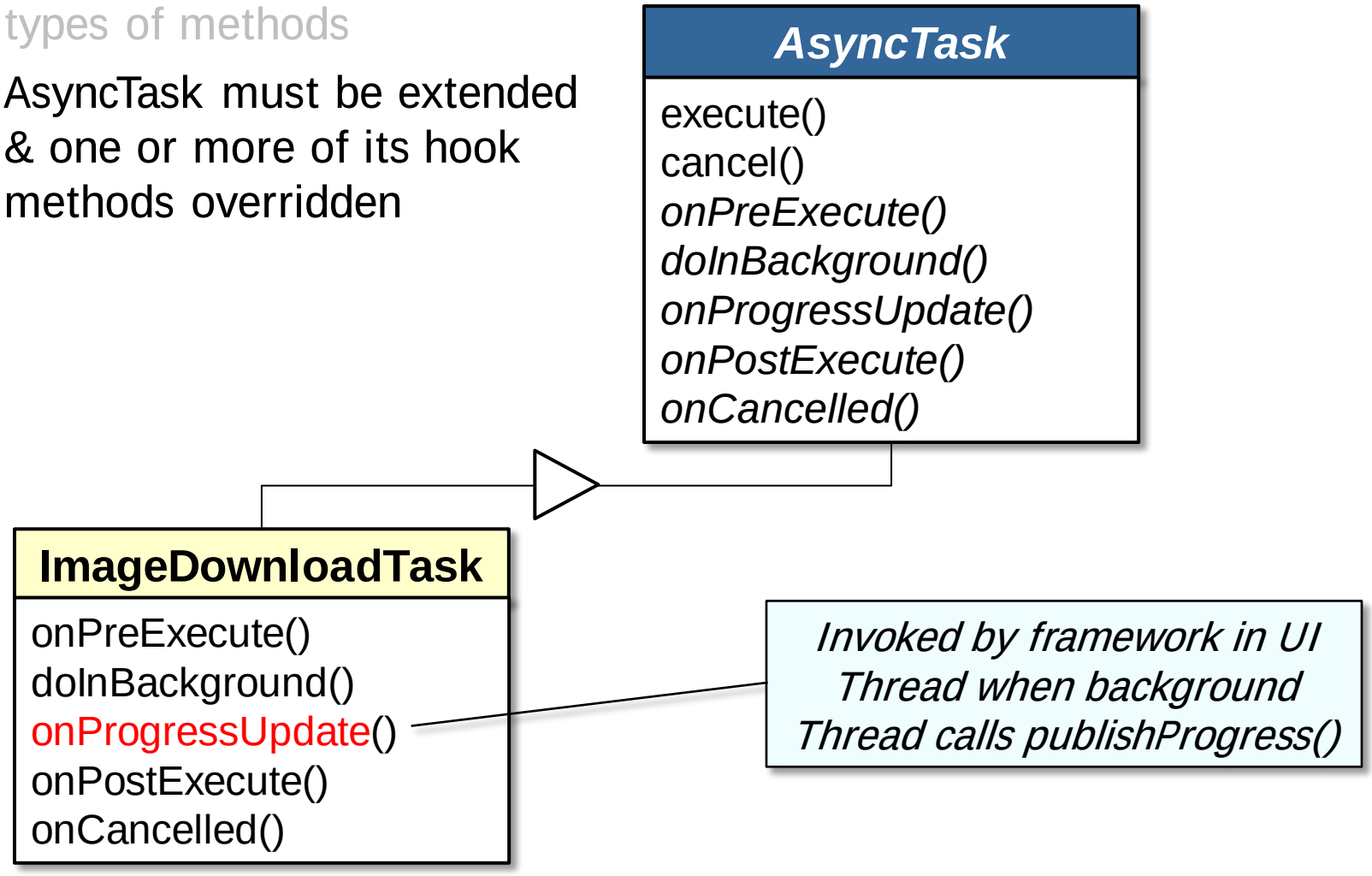
Categories of Methods in the AsyncTask Class

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



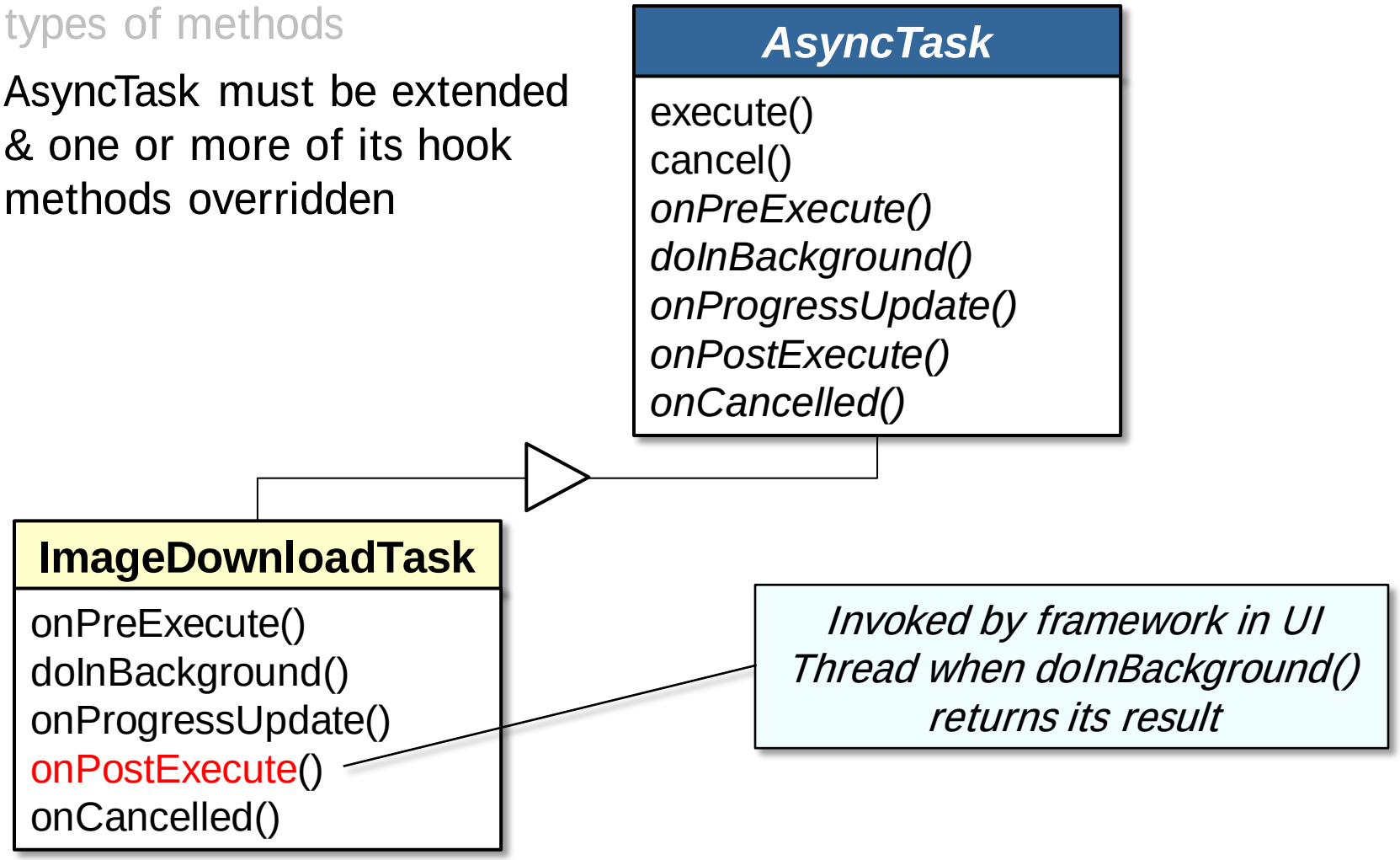
Categories of Methods in the AsyncTask Class

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



Categories of Methods in the AsyncTask Class

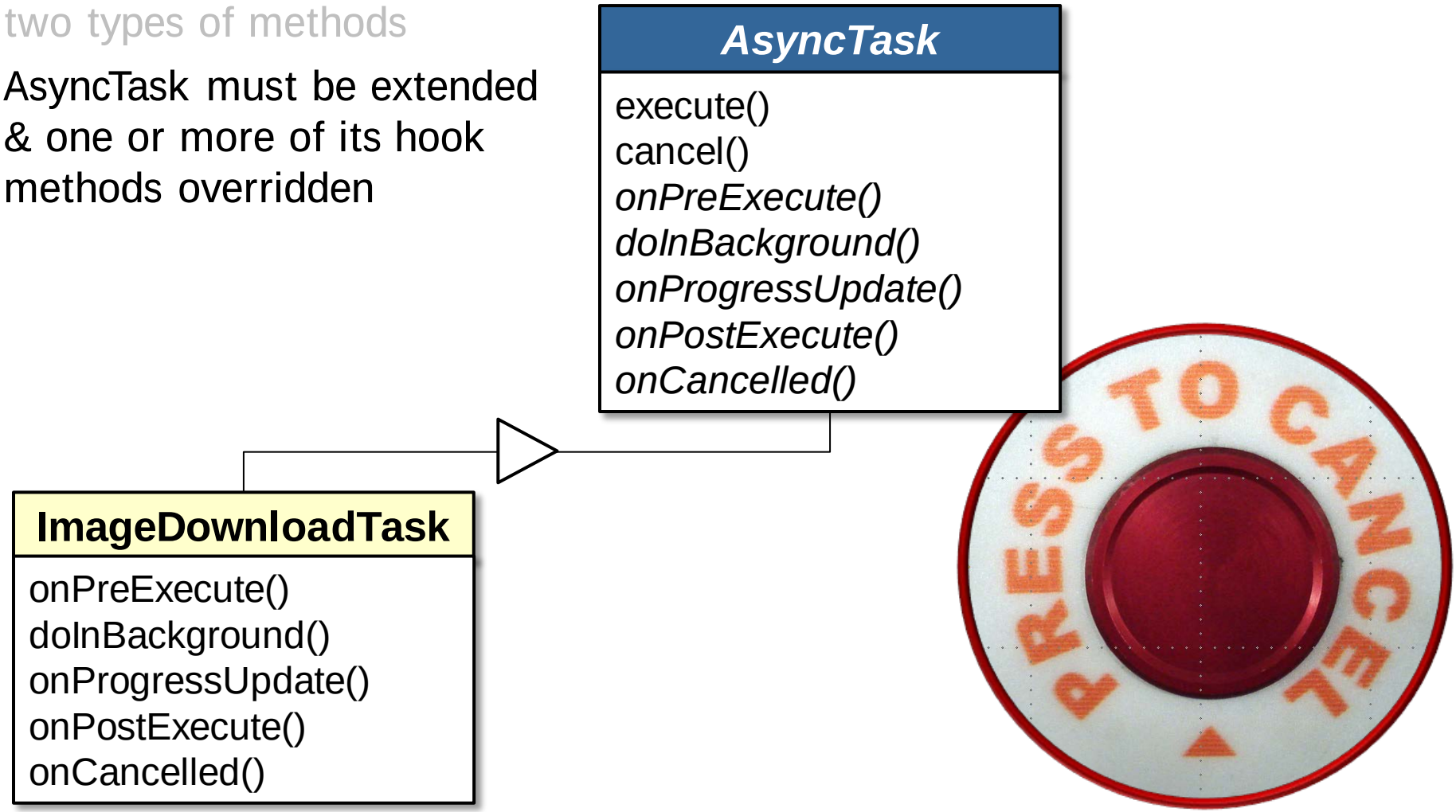
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Categories of Methods in AsyncTask (Part 2)

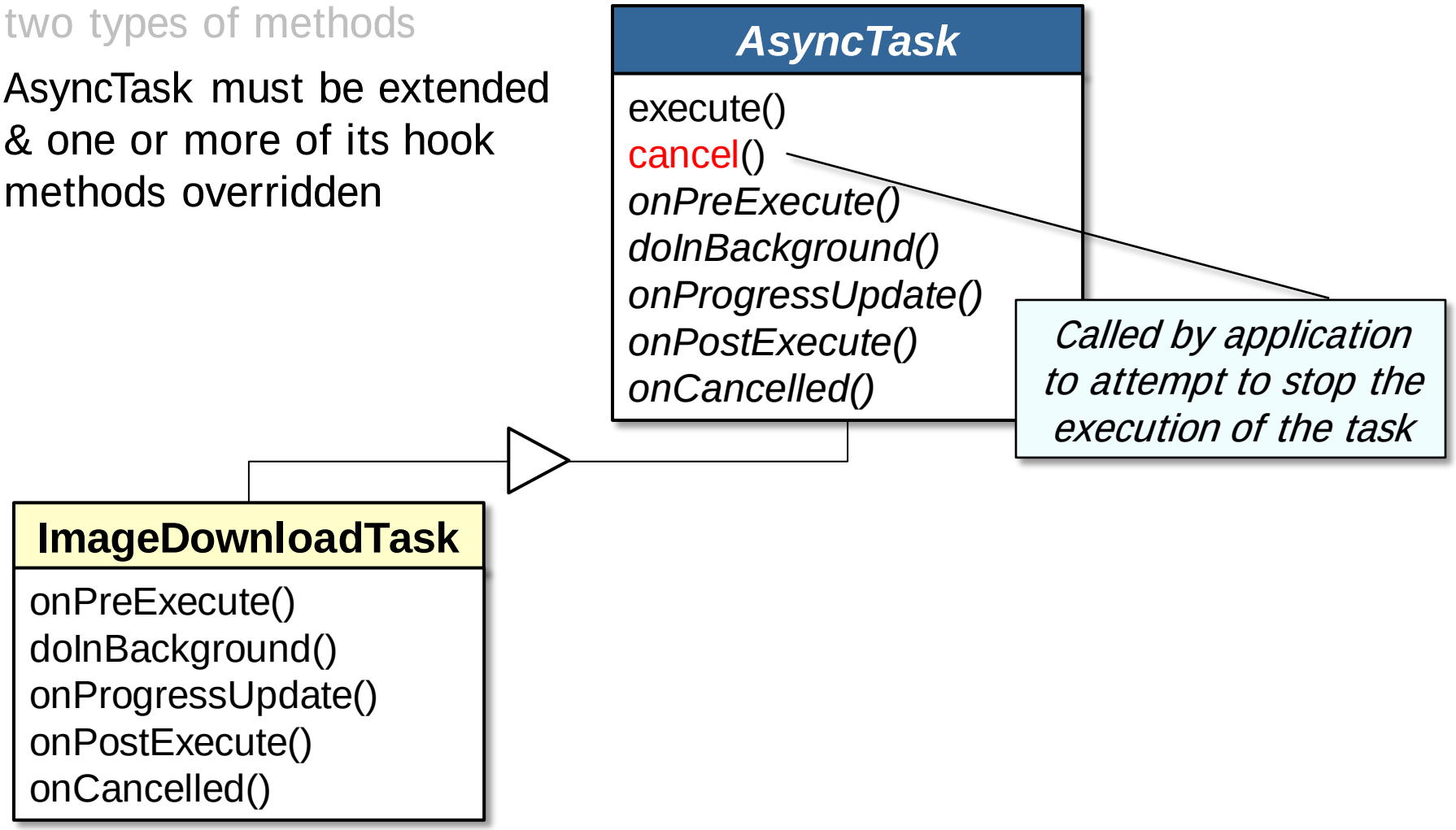
Categories of Methods in the AsyncTask Class

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



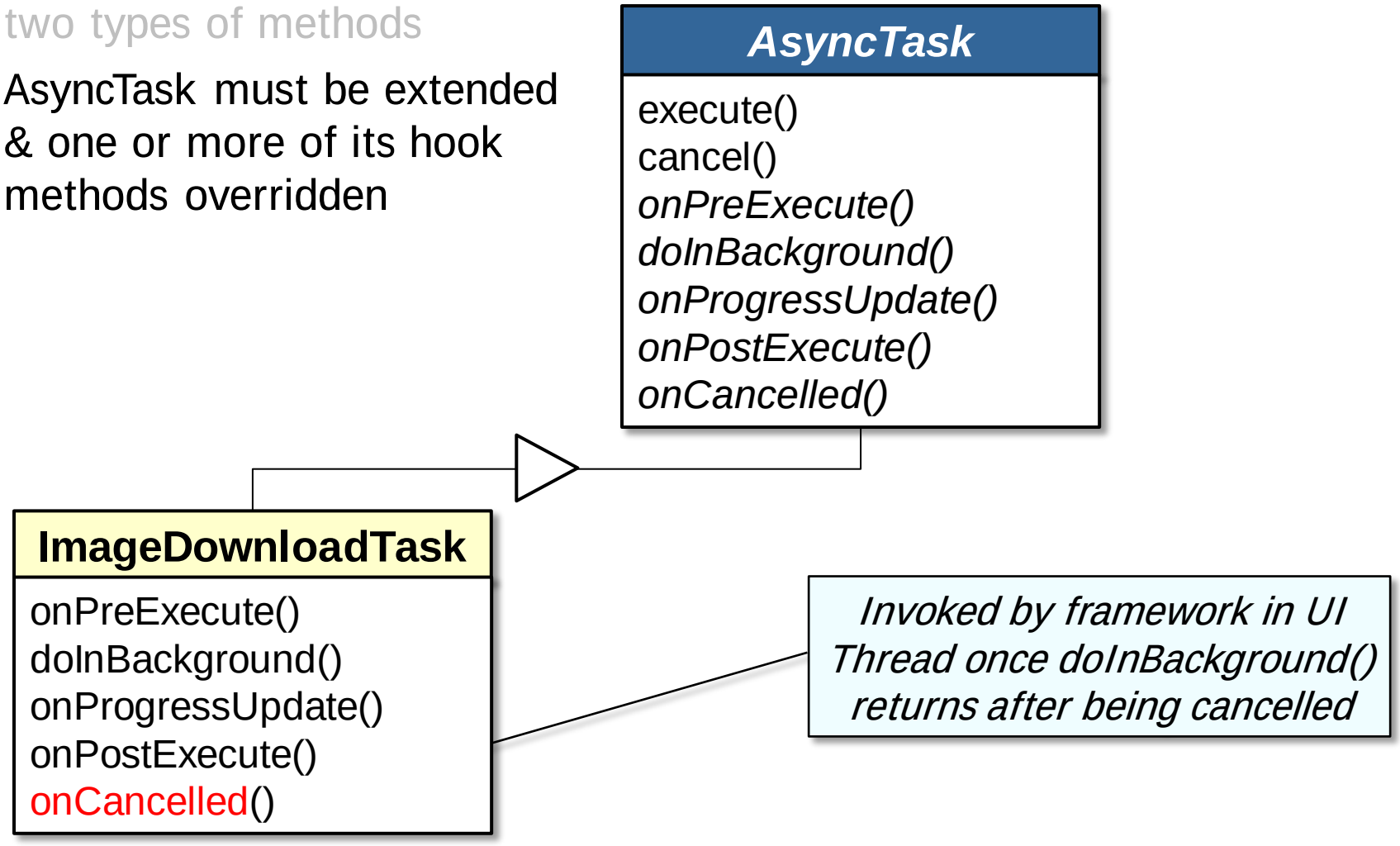
Categories of Methods in the AsyncTask Class

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



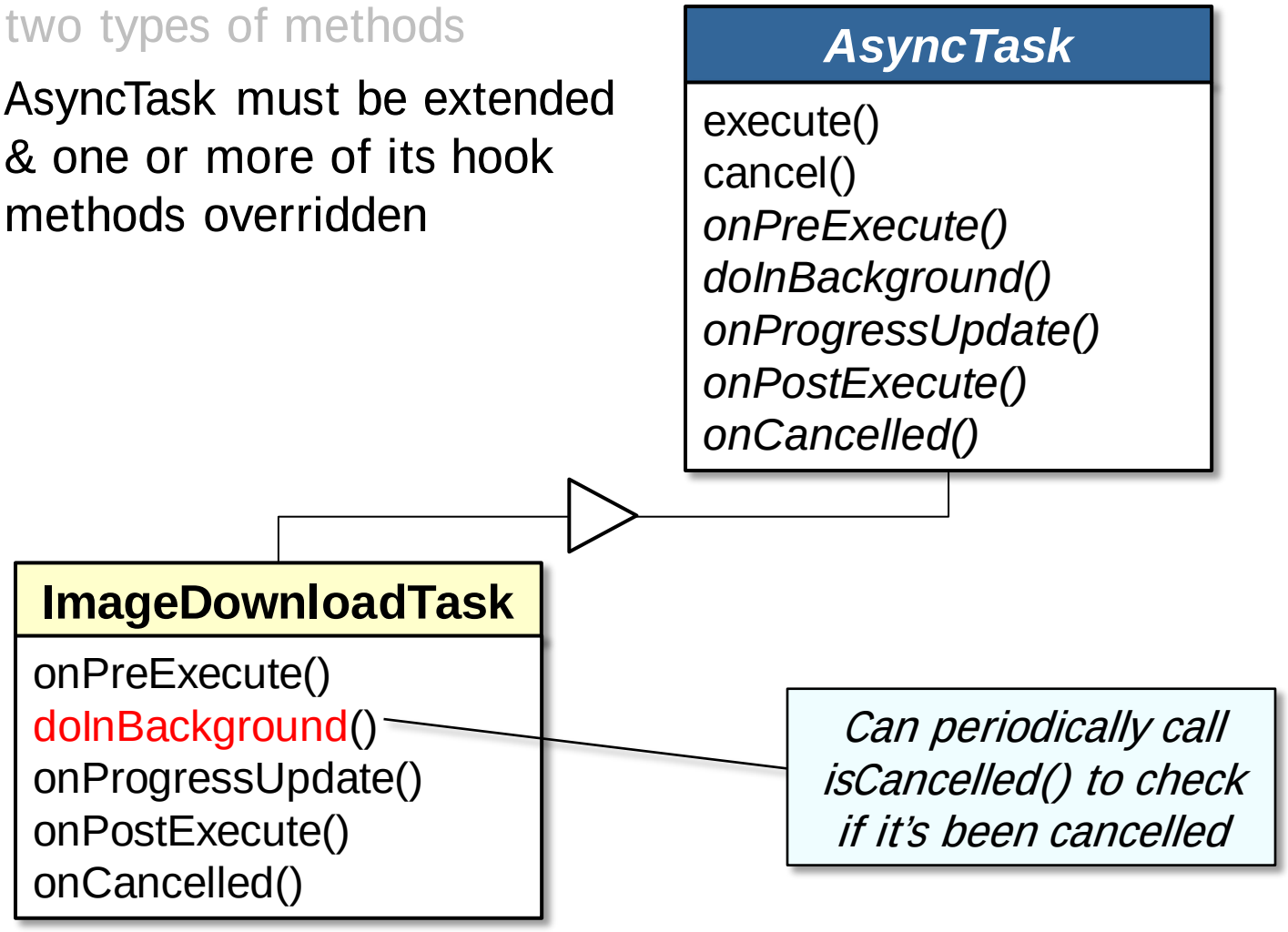
Categories of Methods in the AsyncTask Class

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



Categories of Methods in the AsyncTask Class

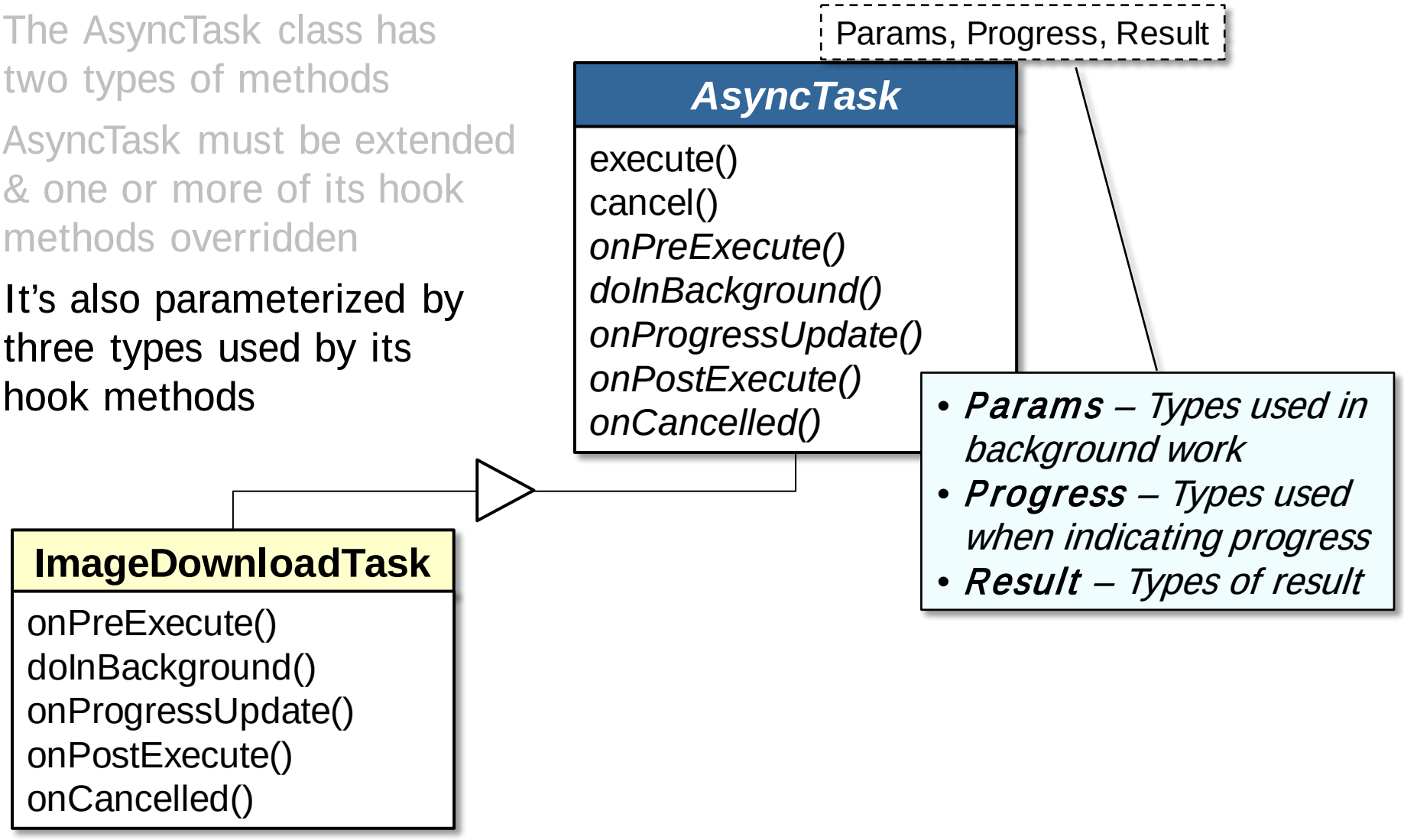
- The AsyncTask class has two types of methods
- AsyncTask must be extended & one or more of its hook methods overridden



Earlier part on "Overview of Java Threads (Part 2)" shows how to stop Threads

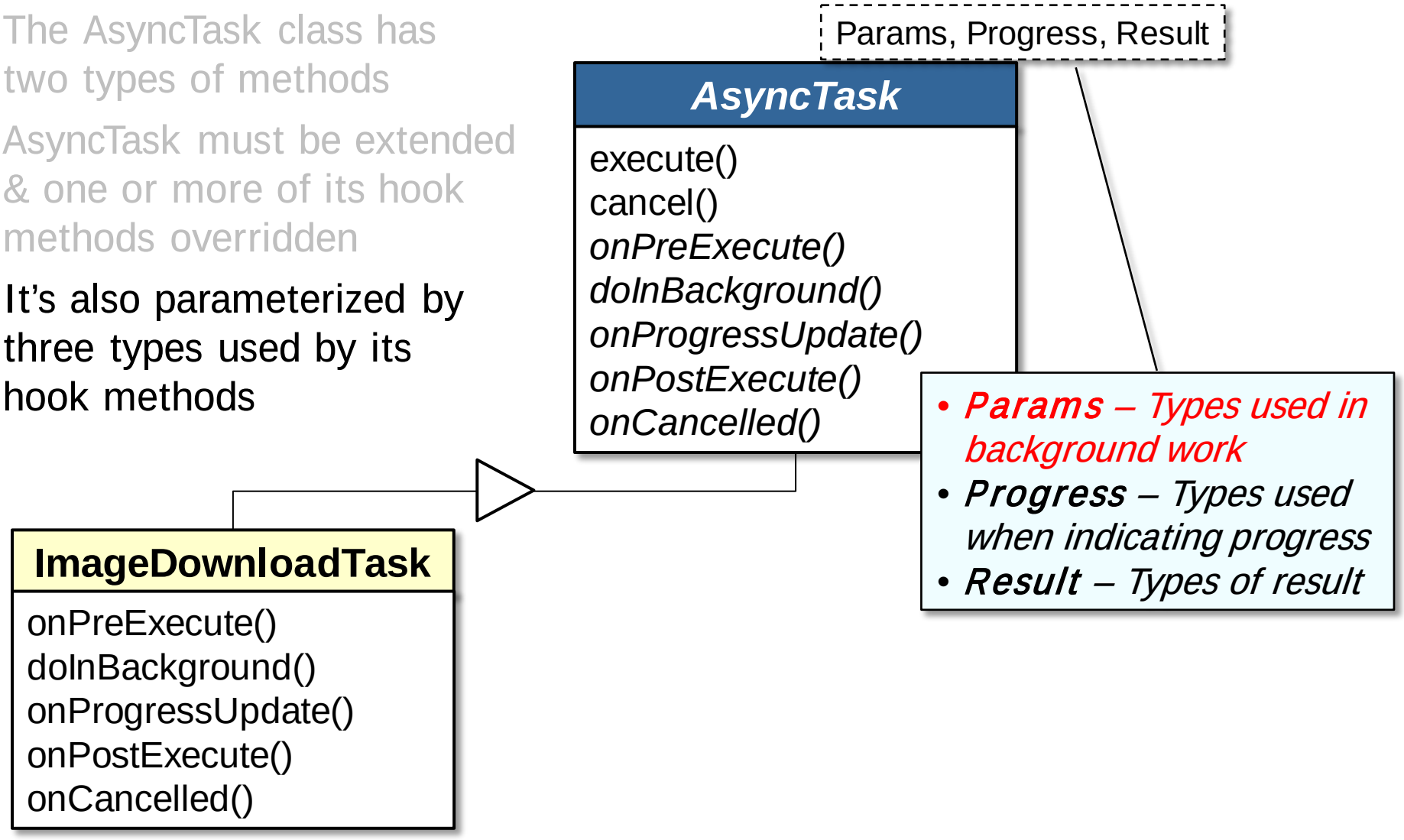
Categories of Methods in the AsyncTask Class

- 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



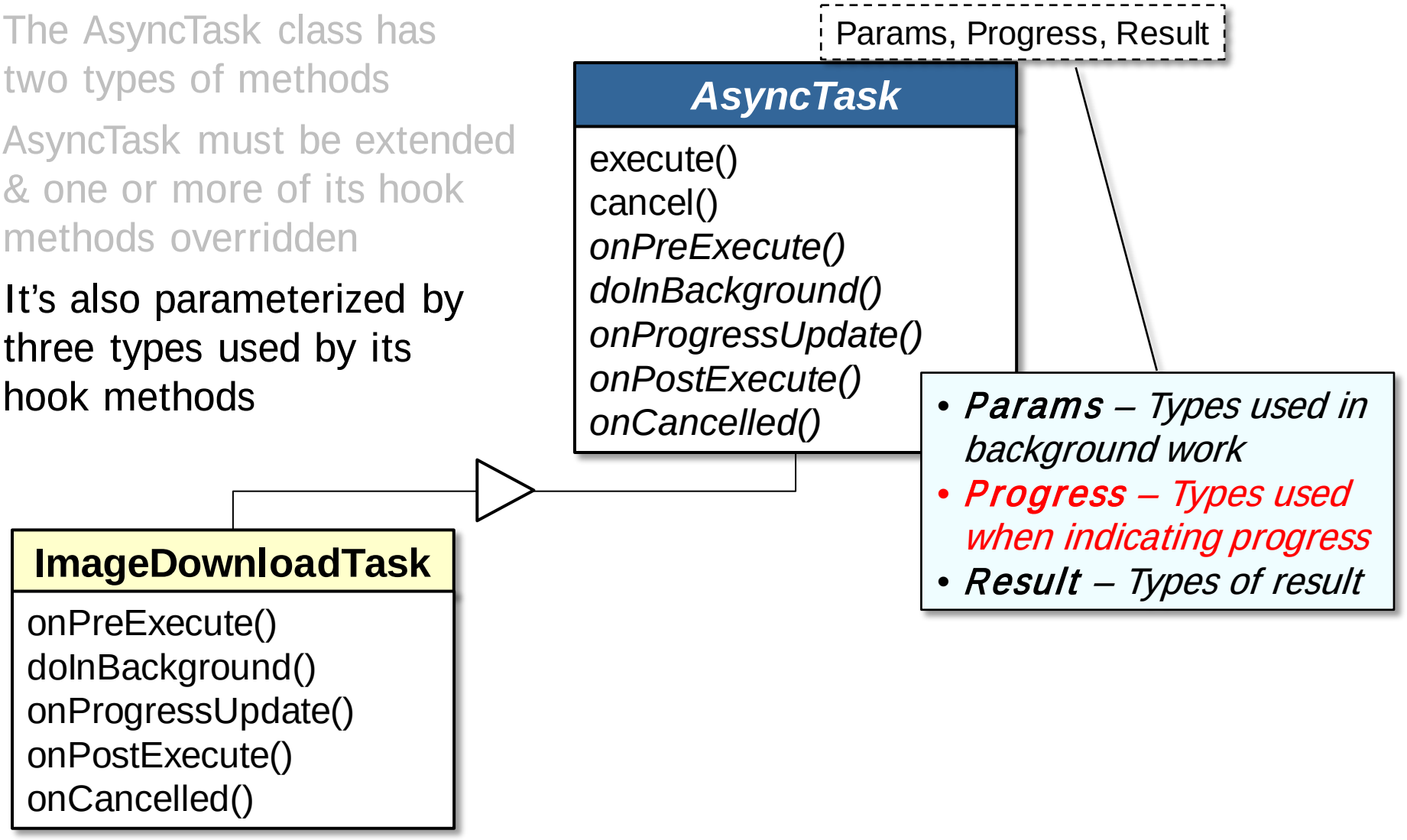
Categories of Methods in the AsyncTask Class

- 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



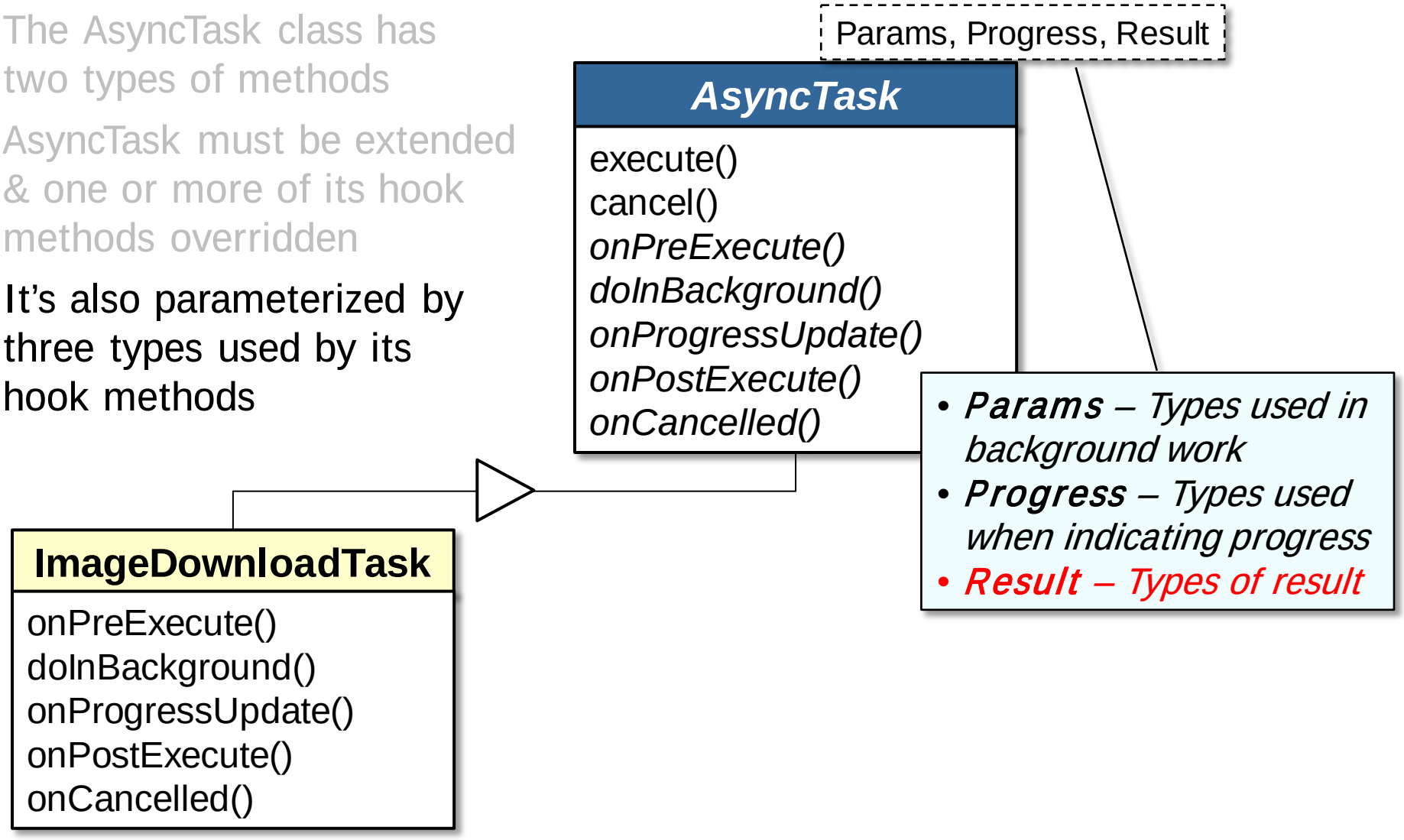
Categories of Methods in the AsyncTask Class

- 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
- Applications can customize AsyncTask to meet their concurrency needs

```
class DownloadTask extends
      AsyncTask<URL, Integer, Long> {
  protected Long doInBackground
                        (URL... 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
- Applications can customize AsyncTask to meet their concurrency needs

```
class DownloadTask extends
      AsyncTask<URL, Integer, Long> {
  protected Long doInBackground
                          (URL... 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
- Applications can customize AsyncTask to meet their concurrency needs

```
class DownloadTask extends
      AsyncTask<URL, Integer, Long> {
  protected Long doInBackground
                          (URL... 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
- Applications can customize AsyncTask to meet their concurrency needs

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

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

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

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

developer.android.com/reference/android/os/AsyncTask.html shows example

Using AsyncTask in Android

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls

com.android.phone

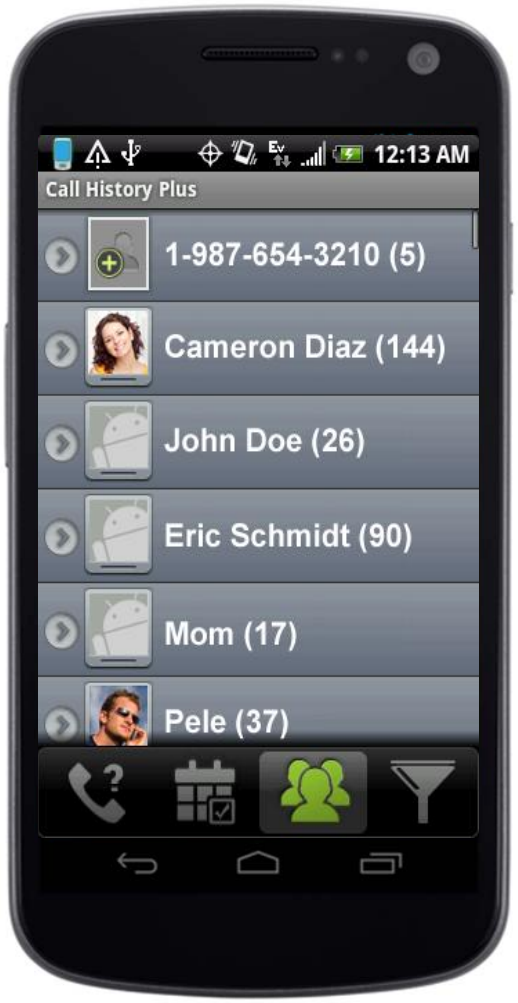
Class CallLogAsync

```
java.lang.Object
  └ com.android.phone.CallLogAsync
```

```
public class CallLogAsync
extends java.lang.Object
```

Class to access the call logs database asynchronously since database ops can take a long time depending on t
load. It uses AsyncTask which has its own thread pool.

```
 Typical usage:
 ==============

  // From an activity...
  String mLastNumber = "";

  CallLogAsync log = new CallLogAsync();

  CallLogAsync.AddCallArgs addCallArgs = new CallLogAsync.AddCallArgs(
      this, ci, number, presentation, type, timestamp, duration);

  log.addCall(addCallArgs);

  CallLogAsync.GetLastOutgoingCallArgs lastCallArgs = new CallLogAsync.GetLastOutgoing
      this, new CallLogAsync.OnLastOutgoingCallComplete() {
                public void lastOutgoingCall(String number) { mLastNumber = number; }
            });
  log.getLastOutgoingCall(lastCallArgs);
```



packages/apps/Phone/src/com/android/phone/CallLogAsync.java has source

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls

```
public class CallLogAsync {
  ...
  public AsyncTask addCall
              (AddCallArgs args) {
    ...
  }

  public AsyncTask
    getLastOutgoingCall
      (GetLastOutgoingCallArgs args)
  {
    ...
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
  - Database operations need to run in a background Thread since they can take a long time to run

```
public class CallLogAsync {
  ...
  public AsyncTask addCall
              (AddCallArgs args) {
    ...
  }

  public AsyncTask
    getLastOutgoingCall
      (GetLastOutgoingCallArgs args)
  {
    ...
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync log =
      new CallLogAsync();

    CallLogAsync.AddCallArgs
      addCallArgs = new
        CallLogAsync.AddCallArgs
          (this, ci, number,
           presentation, type,
           timestamp, duration);

    log.addCall(addCallArgs);
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
  - Contains meta-data about a call

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync log =
      new CallLogAsync();

    CallLogAsync.AddCallArgs
      addCallArgs = new
        CallLogAsync.AddCallArgs
          (this, ci, number,
           presentation, type,
           timestamp, duration);

    log.addCall(addCallArgs);
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
  - Contains meta-data about a call

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync log =
      new CallLogAsync();

    CallLogAsync.AddCallArgs
      addCallArgs = new
        CallLogAsync.AddCallArgs
          (this, ci, number,
           presentation, type,
           timestamp, duration);

    log.addCall(addCallArgs);
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
  - Contains meta-data about a call

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync log =
      new CallLogAsync();

    CallLogAsync.AddCallArgs
      addCallArgs = new
        CallLogAsync.AddCallArgs
          (this, ci, number,
           presentation, type,
           timestamp, duration);

    log.addCall(addCallArgs);
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
  - Contains meta-data about a call

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync log =
      new CallLogAsync();

    CallLogAsync.AddCallArgs
      addCallArgs = new
        CallLogAsync.AddCallArgs
          (this, ci, number,
           presentation, type,
           timestamp, duration);

    log.addCall(addCallArgs);
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info

```
public class CallLogAsync {
  ...
  public AsyncTask addCall
              (AddCallArgs args) {
    assertUiThread();
    return new AddCallTask().
                  execute(args);
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
public class CallLogAsync {
  ...
  public AsyncTask addCall
              (AddCallArgs args) {
    assertUiThread();
    return new AddCallTask().
                  execute(args);
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected Uri[] doInBackground
        (AddCallArgs... callList) {
    ...
    Uri[] result = new Uri[count];
    ...
      result[i]=
        Calls.addCall
          (c.ci, c.context,
           c.number, c.presentation,
           c.callType, c.timestamp,
           c.durationInSec);
    ...
    return result;
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected Uri[] doInBackground
        (AddCallArgs... callList) {
    ...
    Uri[] result = new Uri[count];
    ...
      result[i]=
        Calls.addCall
          (c.ci, c.context,
           c.number, c.presentation,
           c.callType, c.timestamp,
           c.durationInSec);
    ...
    return result;
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected Uri[] doInBackground
        (AddCallArgs... callList) {
    ...
    Uri[] result = new Uri[count];
    ...
      result[i]=
        Calls.addCall
          (c.ci, c.context,
           c.number, c.presentation,
           c.callType, c.timestamp,
           c.durationInSec);
    ...
    return result;
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
public class CallLog {
  ...
  public static class Calls ... {
    public static Uri addCall(...) {
      final ContentResolver resolver =
        context.getContentResolver();
      ...
      Uri result = resolver.insert
        (CONTENT_URI, values);
      ...
      return result;
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
public class CallLog {
  ...
  public static class Calls ... {
    public static Uri addCall(...) {
      final ContentResolver resolver =
        context.getContentResolver();
      ...
      Uri result = resolver.insert
        (CONTENT_URI, values);
      ...
      return result;
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread

```
public class CallLog {
  ...
  public static class Calls ... {
    public static Uri addCall(...) {
      final ContentResolver resolver =
        context.getContentResolver();
      ...
      Uri result = resolver.insert
        (CONTENT_URI, values);
      ...
      return result;
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread
  - Inserts call meta-data into the CallLog's Content Provider & SQLite database

```
public class CallLog {
  ...
  public static class Calls ... {
    public static Uri addCall(...) {
      final ContentResolver resolver =
        context.getContentResolver();
      ...
      Uri result = resolver.insert
        (CONTENT_URI, values);
      ...
      return result;
```

May block!

frameworks/base/core/java/android/provider/CallLog.java has source

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread
  - Inserts call meta-data into the CallLog's Content Provider & SQLite database

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected Uri[] doInBackground
        (AddCallArgs... callList) {
    ...
    Uri[] result = new Uri[count];
    ...
      result[i]=
        Calls.addCall
          (c.ci, c.context,
           c.number, c.presentation,
           c.callType, c.timestamp,
           c.durationInSec);
    ...
    return result;
  }
```

packages/apps/Phone/src/com/android/phone/CallLogAsync.java has source

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread
  - Inserts call meta-data into the CallLog's Content Provider & SQLite database
  - Perform a sanity check to make sure the call was written to the database

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected void onPostExecute
                        (Uri[] result) {
    for (Uri uri : result) {
      if (uri == null) {
        Log.e(TAG, ...);
      }
    }
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread
  - Inserts call meta-data into the CallLog's Content Provider & SQLite database
  - Perform a sanity check to make sure the call was written to the database

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected void onPostExecute
                        (Uri[] result) {
    for (Uri uri : result) {
      if (uri == null) {
        Log.e(TAG, ...);
      }
    }
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
  - Runs in background Thread to avoid blocking UI Thread
  - Inserts call meta-data into the CallLog's Content Provider & SQLite database
  - Perform a sanity check to make sure the call was written to the database

```
class AddCallTask extends AsyncTask
      <AddCallArgs, Void, Uri[]> {
  ...
  protected void onPostExecute
                        (Uri[] result) {
    for (Uri uri : result) {
      if (uri == null) {
        Log.e(TAG, ...);
      }
    }
  }
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call

```
public class CallLogAsync {
  ...
  public AsyncTask
    getLastOutgoingCall
      (GetLastOutgoingCallArgs args)
  {
     assertUiThread();
     return new
       GetLastOutgoingCallTask
         (args.callback).
                    execute(args);
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call

```
public class CallLogAsync {
  ...
  public AsyncTask
    getLastOutgoingCall
      (GetLastOutgoingCallArgs args)
  {
     assertUiThread();
     return new
       GetLastOutgoingCallTask
         (args.callback).
                   execute(args);
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
  - doInBackground() again can initiate blocking calls to the SQLite database since it runs in a background Thread

```
class GetLastOutgoingCallTask
      extends AsyncTask
        <GetLastOutgoingCallArgs,
         Void, String> {
  ...
  protected String doInBackground
    (GetLastOutgoingCallArgs...
     list) {
    ...
    String number;
    ...
    number =
      Calls.getLastOutgoingCall
        (args.context);
    ...
    return number;
    ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
  - doInBackground() again can initiate blocking calls to the SQLite database since it runs in a background Thread

```
class GetLastOutgoingCallTask
      extends AsyncTask
        <GetLastOutgoingCallArgs,
         Void, String> {
  ...
  protected String doInBackground
    (GetLastOutgoingCallArgs...
     list) {
    ...
    String number;
    ...
    number =
      Calls.getLastOutgoingCall
        (args.context);
    ...
    return number;
    ...
```

May block!

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
  - doInBackground() again can initiate blocking calls to the SQLite database since it runs in a background Thread
  - onPostExecute() invokes a hook method to pass the last outgoing call number back to registered callback object

```
class GetLastOutgoingCallTask
      extends AsyncTask
        <GetLastOutgoingCallArgs,
         Void, String> {
  ...
  protected void onPostExecute
                    (String number) {
    ...
    mCallback.lastOutgoingCall
      (number);
  }
  ...
```

Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
  - doInBackground() again can initiate blocking calls to the SQLite database since it runs in a background Thread
  - onPostExecute() invokes a hook method to pass the last outgoing call number back to registered callback object

```
class GetLastOutgoingCallTask
      extends AsyncTask
        <GetLastOutgoingCallArgs,
         Void, String> {
  ...
  protected void onPostExecute
                      (String number) {
    ...
    mCallback.lastOutgoingCall
      (number);
  }
  ...
```

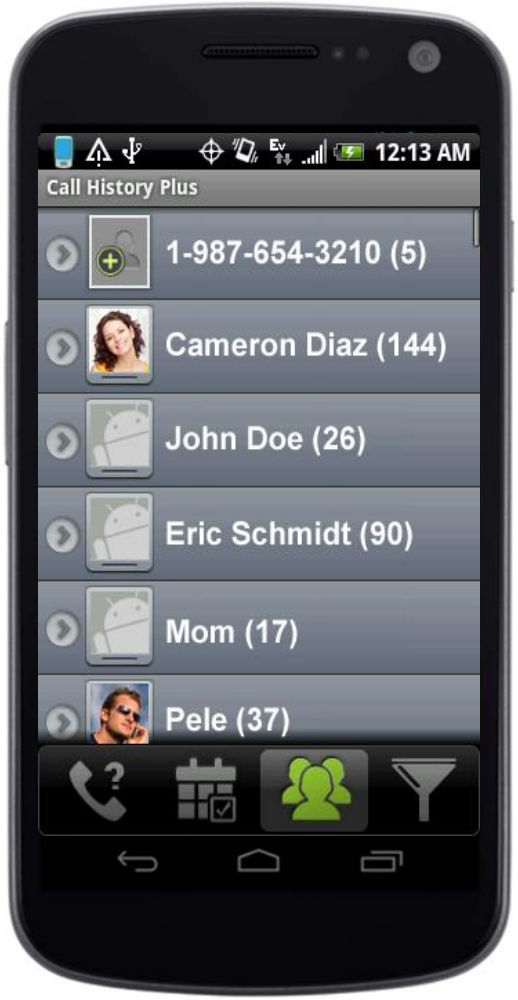
Using AsyncTask in Android

- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
  - doInBackground() again can initiate blocking calls to the SQLite database since it runs in a background Thread
  - onPostExecute() invokes a hook method to pass the last outgoing call number back to registered callback object

```
class CallLogActivity
      extends Activity {
  ...
  void updateCallLog(String number) {
    ...
    CallLogAsync.
          GetLastOutgoingCallArgs
      lastCallArgs = new
        CallLogAsync.
          GetLastOutgoingCallArgs
            (this,
             new CallLogAsync.
      OnLastOutgoingCallComplete()
      { public void
        lastOutgoingCall
          (String number) { ... );
      }
  ...
```

Using AsyncTask in Android

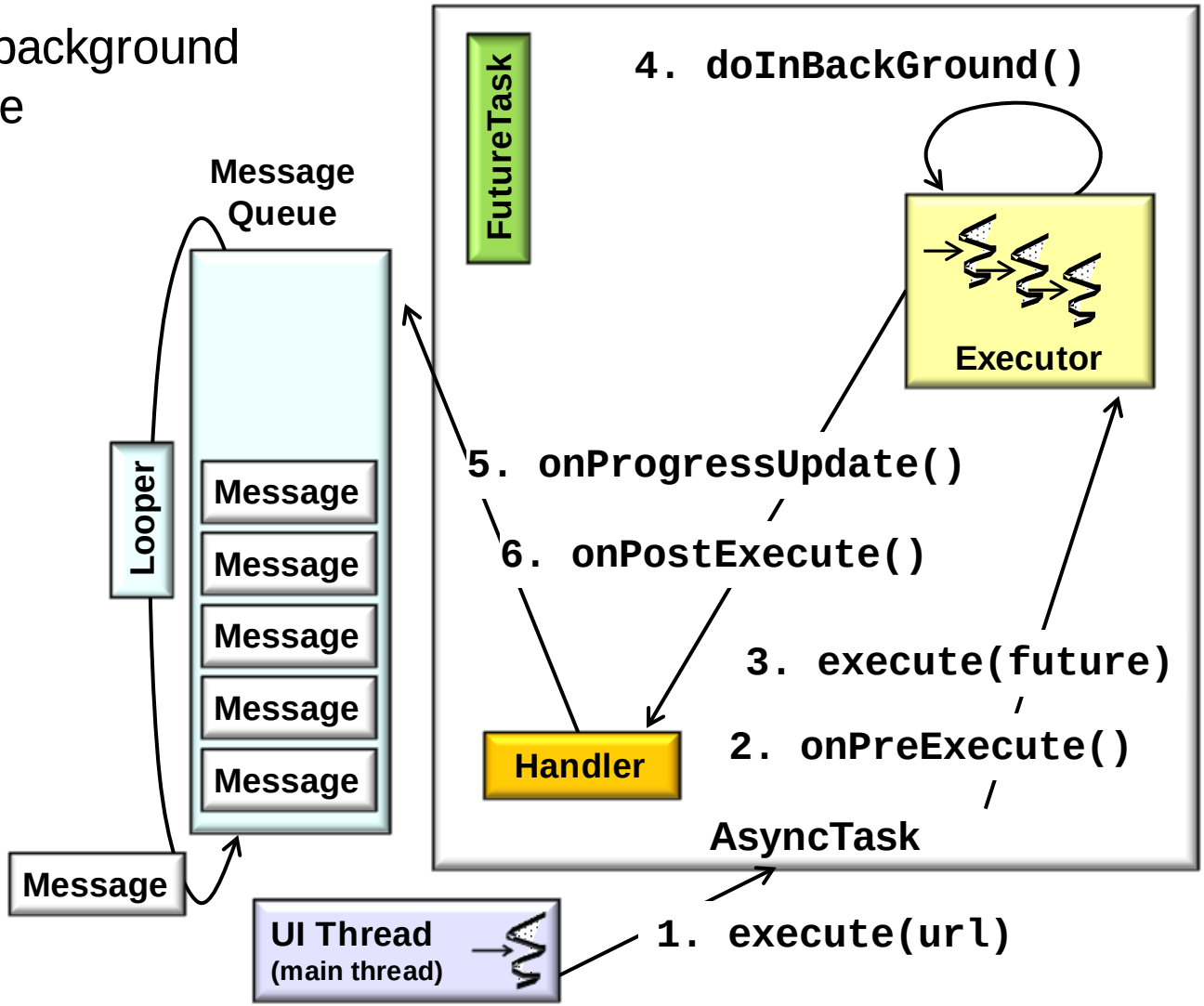
- Android's Phone application uses AsyncTask to log calls
- An Activity creates a new CallLogAsync object
- CallLogAsync's addCall() uses AsyncTask to store call log info
- CallLogAsync also uses AsyncTask to get the last outgoing call
- The AsyncTask makes it easy to implement the Android Phone application's CallLogAsync methods concurrently



Summary

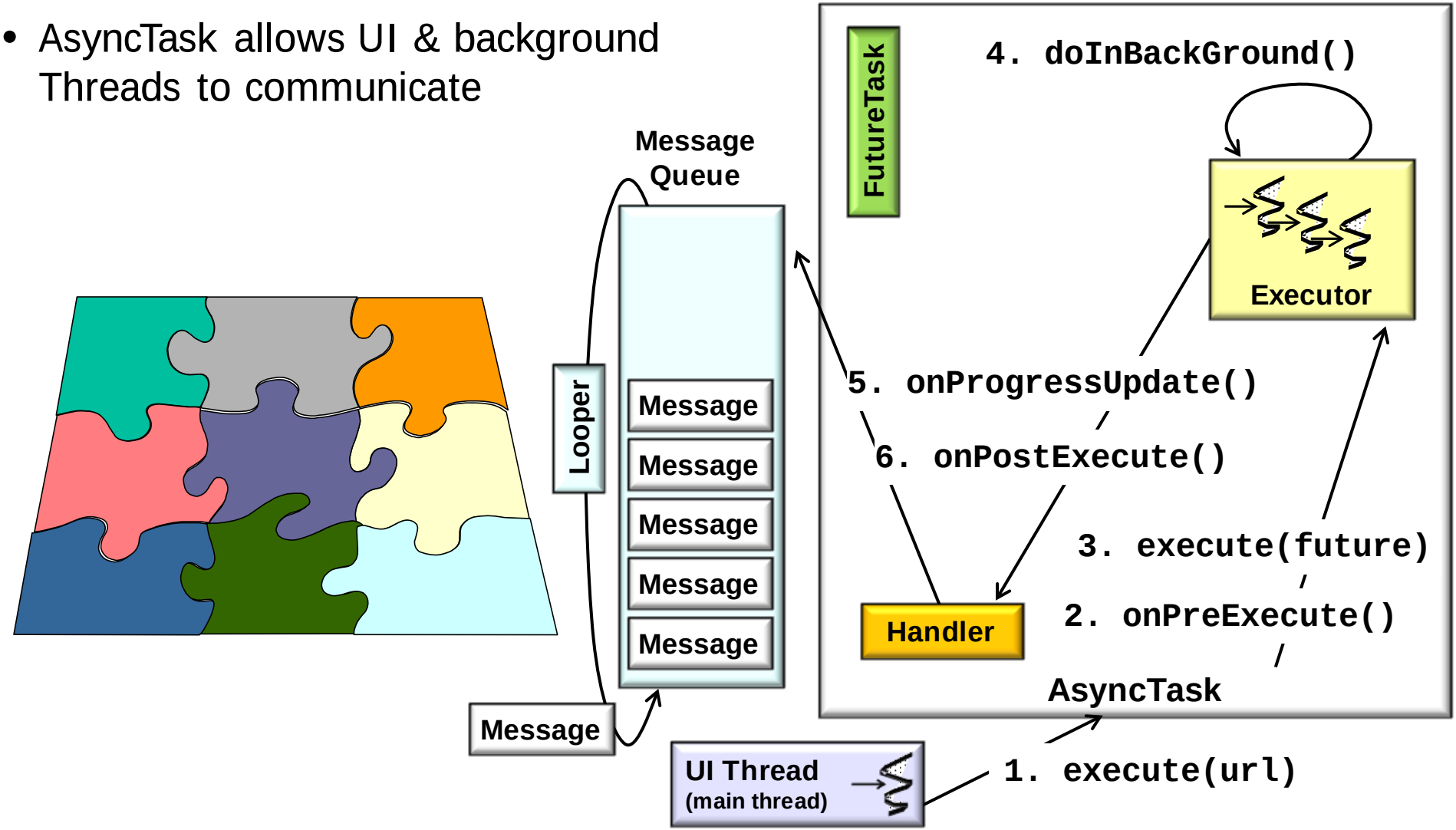
Summary

- AsyncTask allows UI & background Threads to communicate



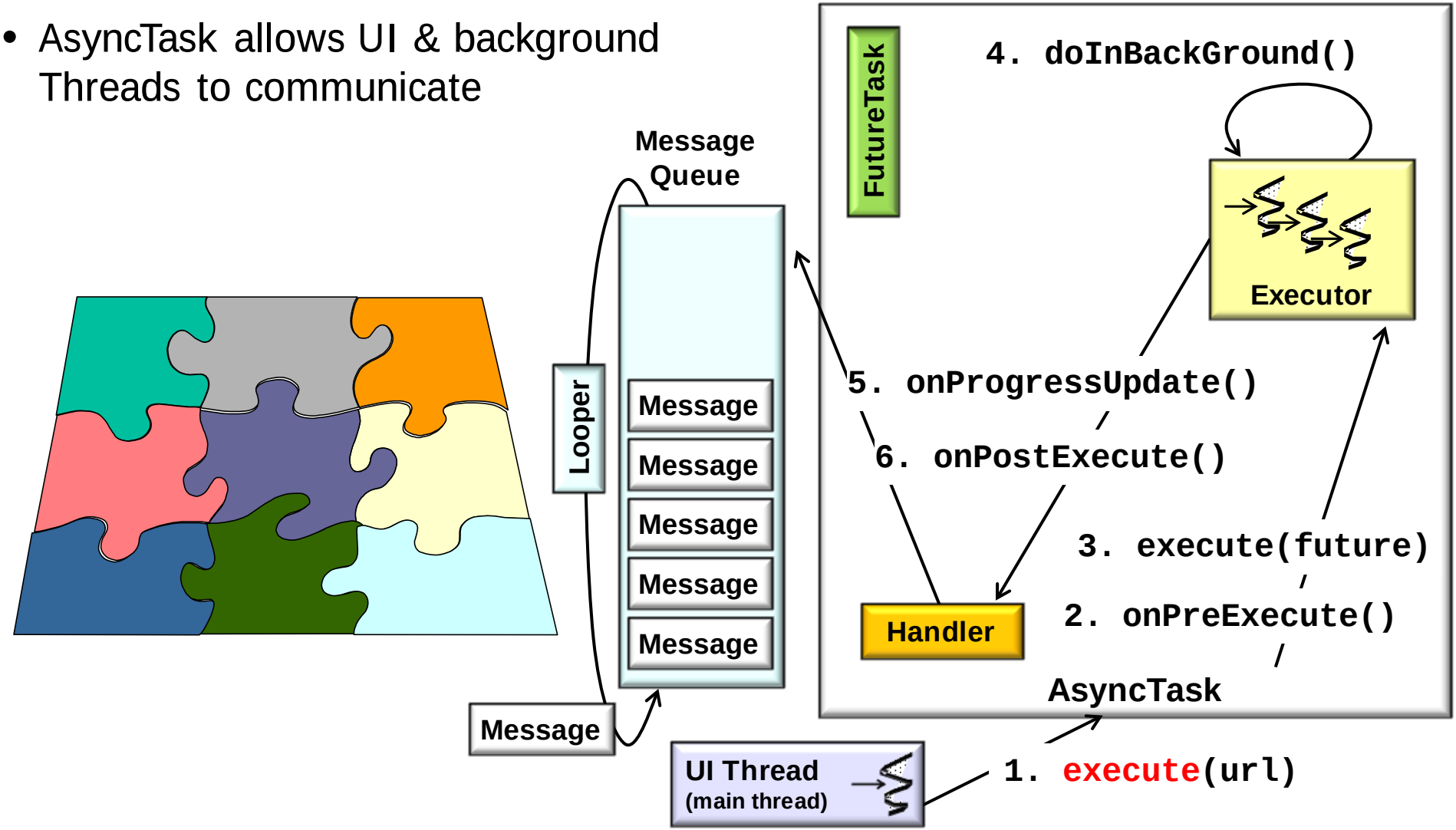
Summary

- AsyncTask allows UI & background Threads to communicate



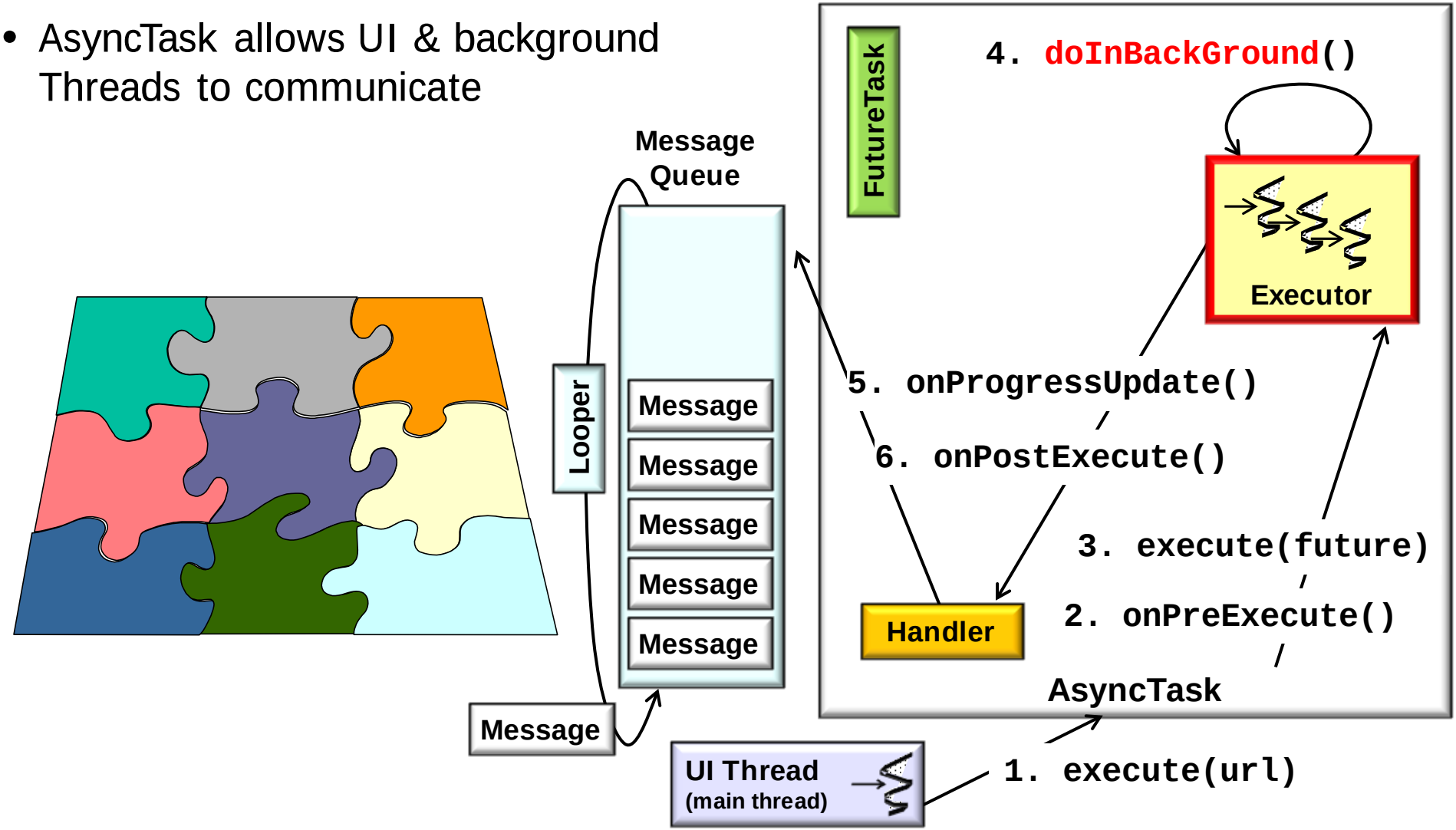
Summary

- AsyncTask allows UI & background Threads to communicate



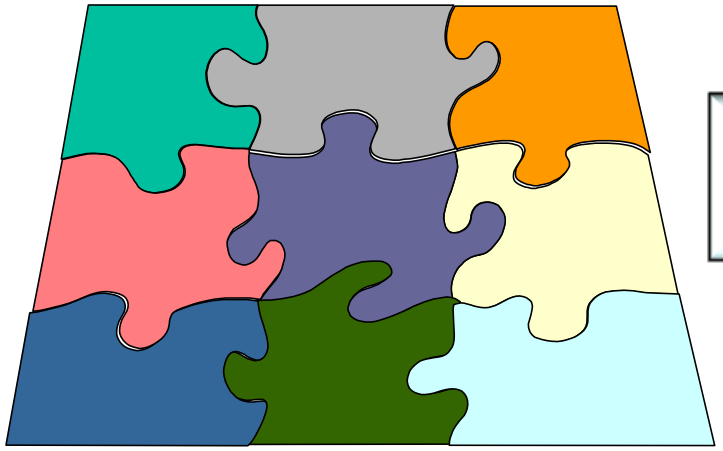
Summary

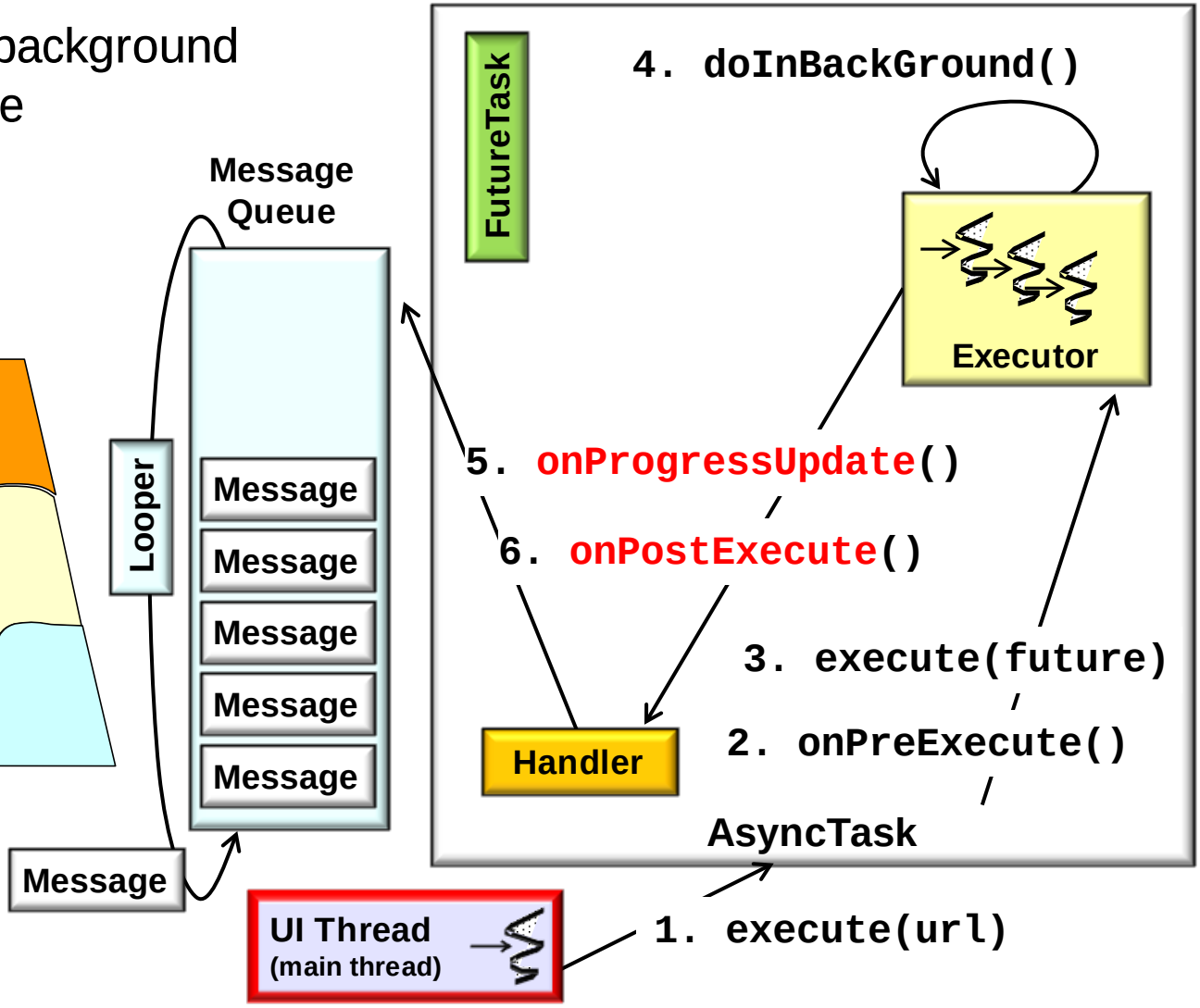
- AsyncTask allows UI & background Threads to communicate



Summary

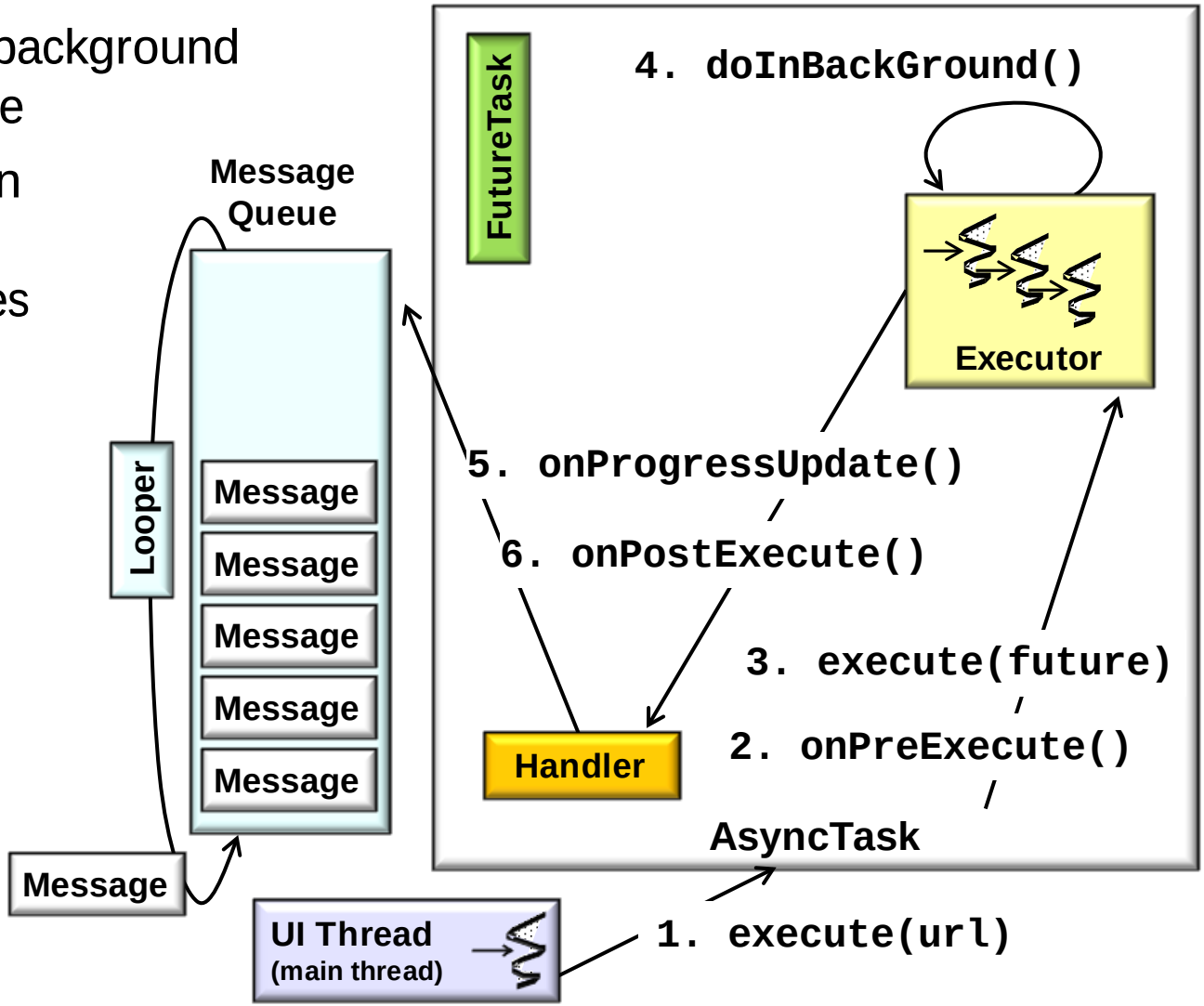
- AsyncTask allows UI & background Threads to communicate





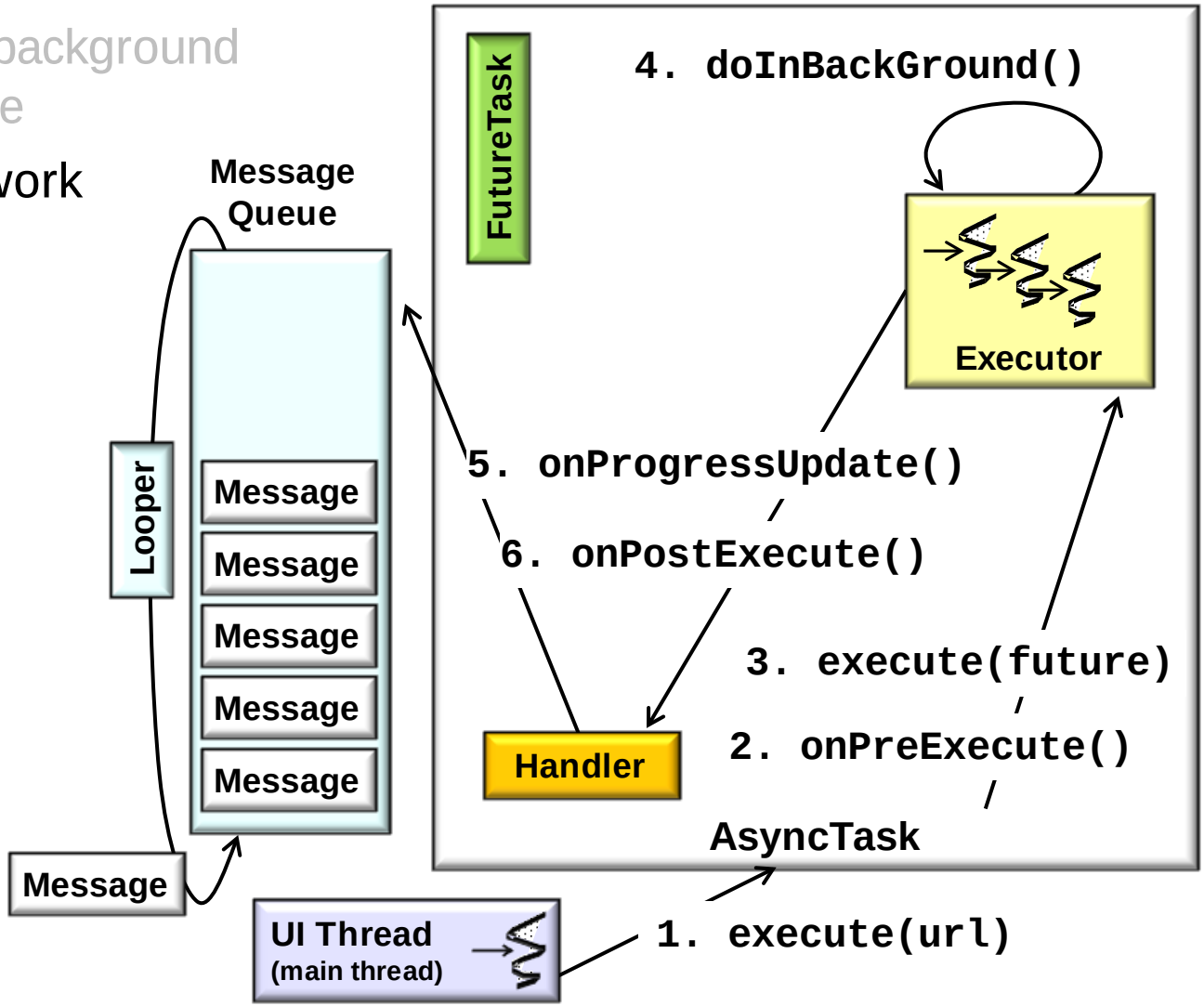
Summary

- AsyncTask allows UI & background Threads to communicate
  - No direct manipulation of Threads, Handlers, Message, or Runnables



Summary

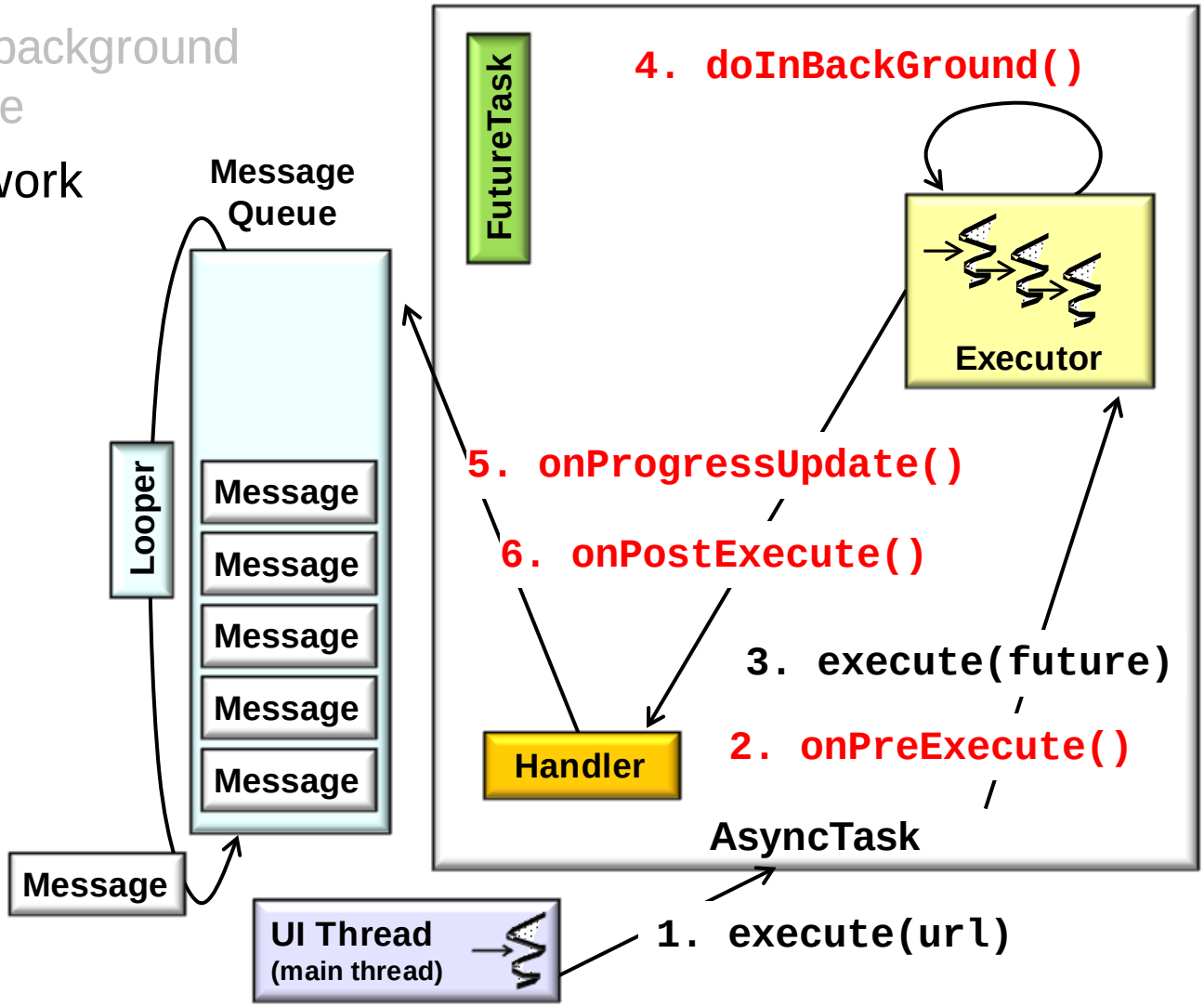
- AsyncTask allows UI & background Threads to communicate
- It embodies key framework characteristics



See earlier part on “Overview of Patterns & Frameworks”

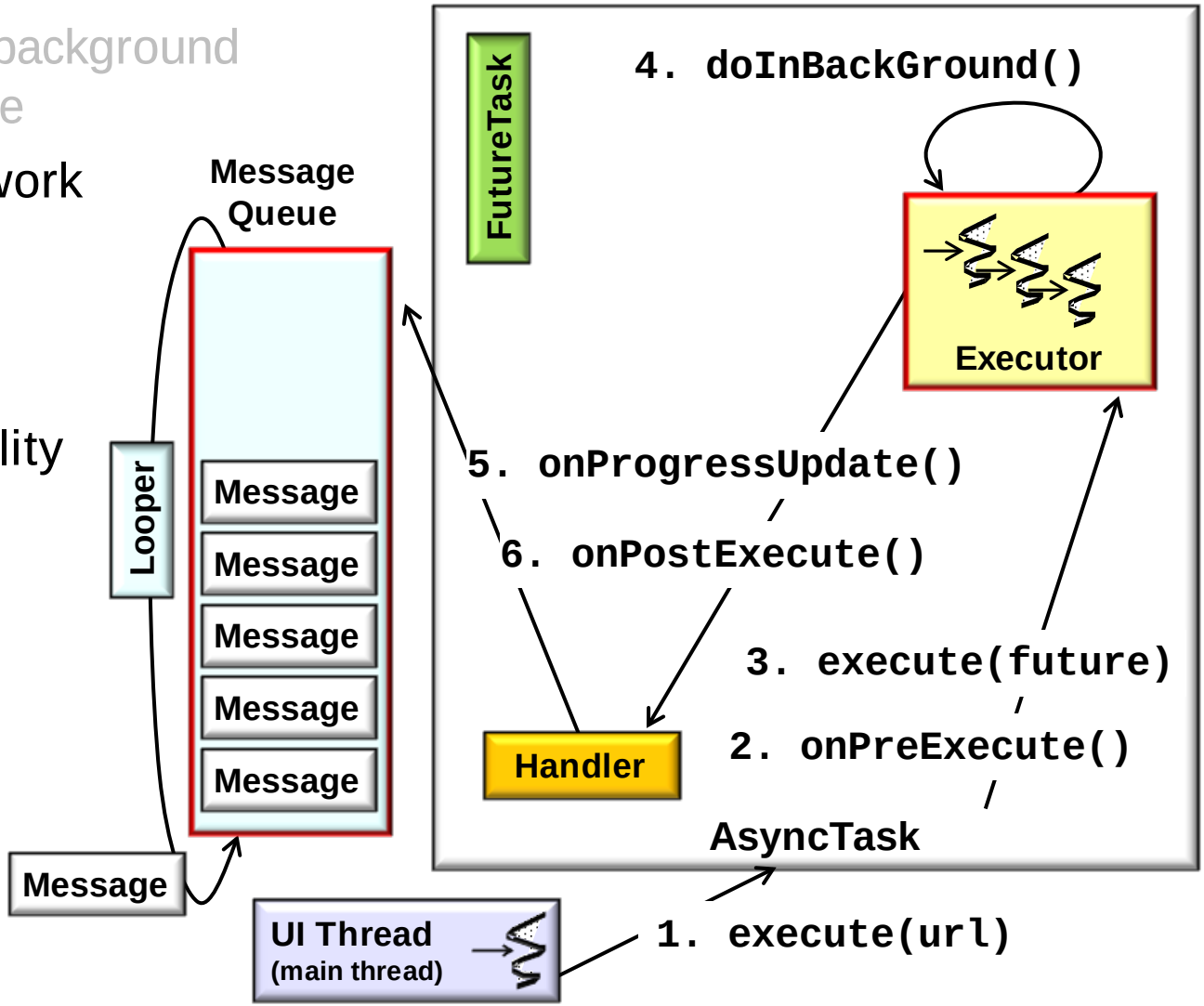
Summary

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



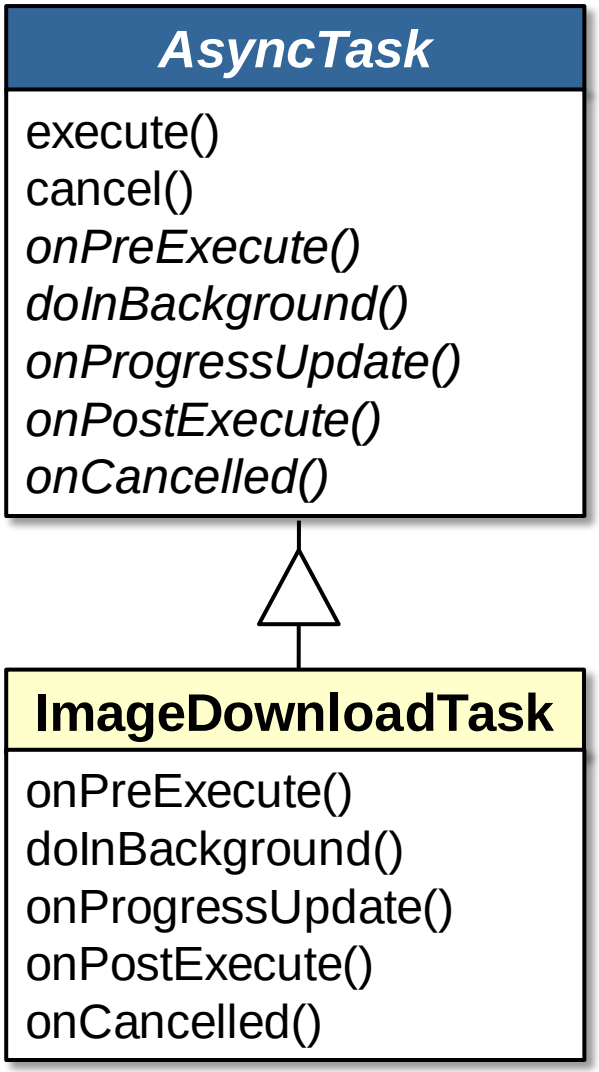
Summary

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



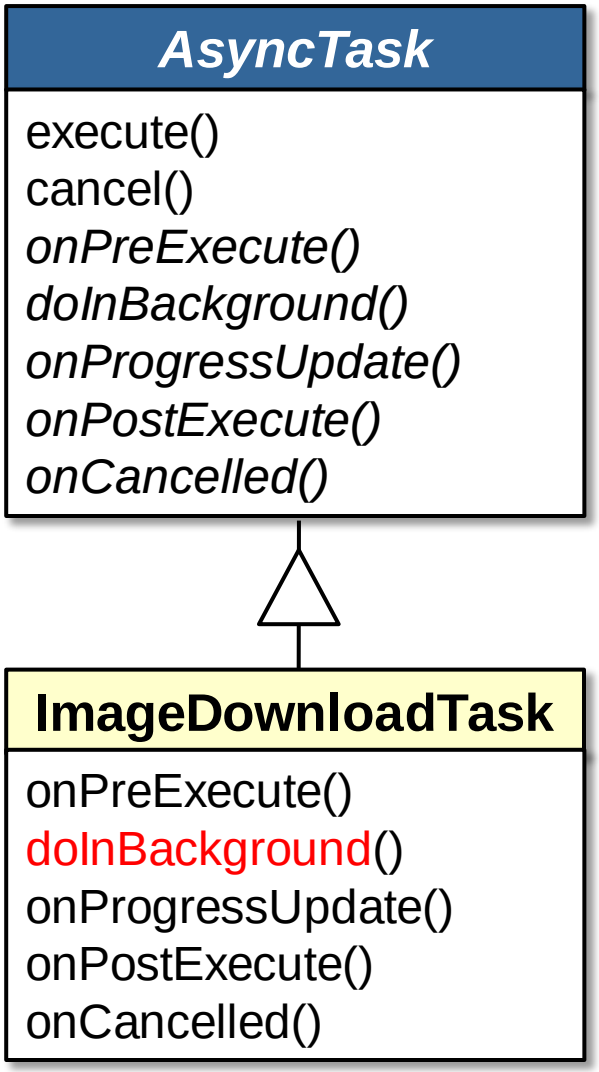
Summary

- 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



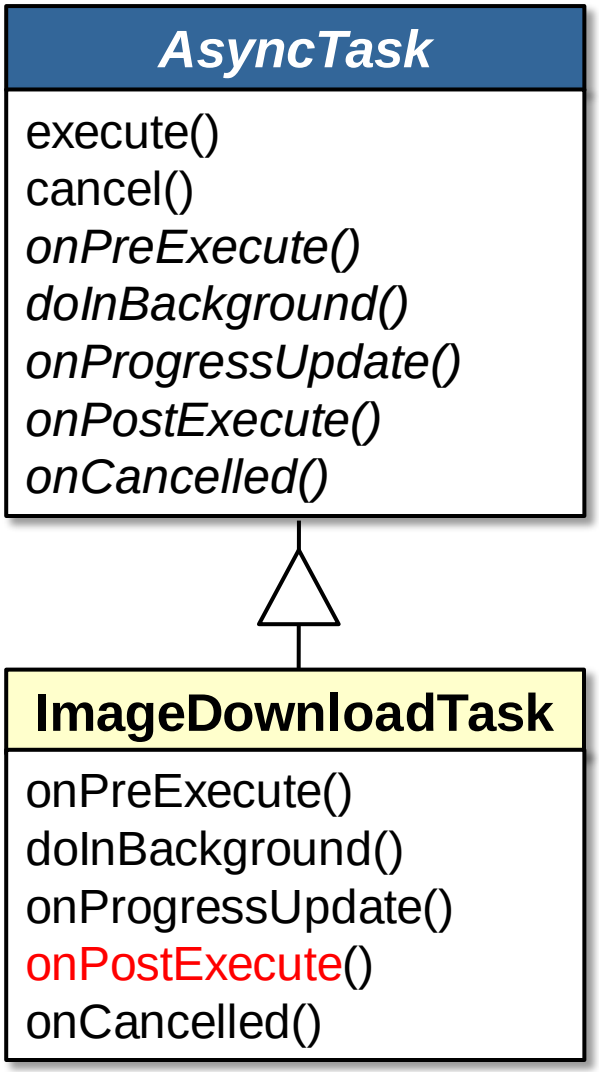
Summary

- 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



Summary

- 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



Summary

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

Summary

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