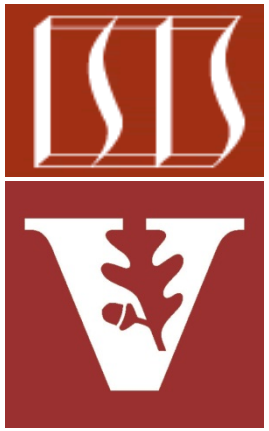
Android Concurrency: Overview of the Android Handler & HaMeR Framework

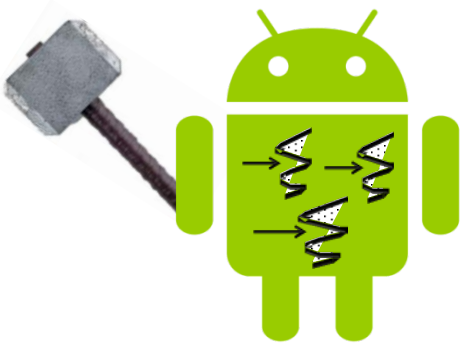


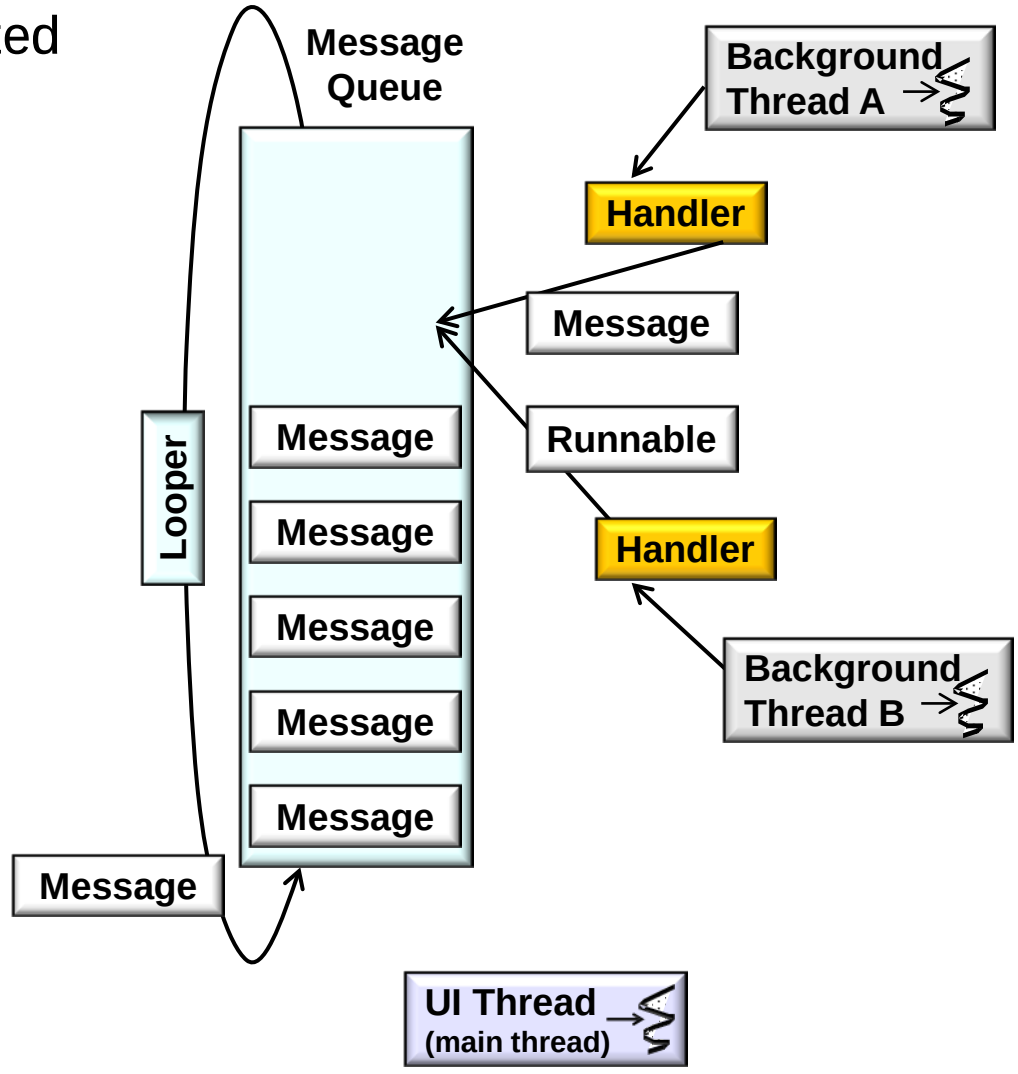
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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

Learning Objectives in this Part of the Module

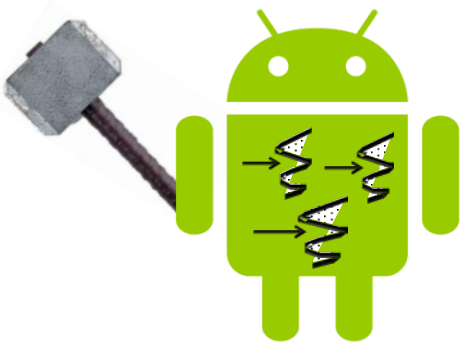
- Understand the concurrency idioms & programming mechanisms related to the Android HaMeR framework

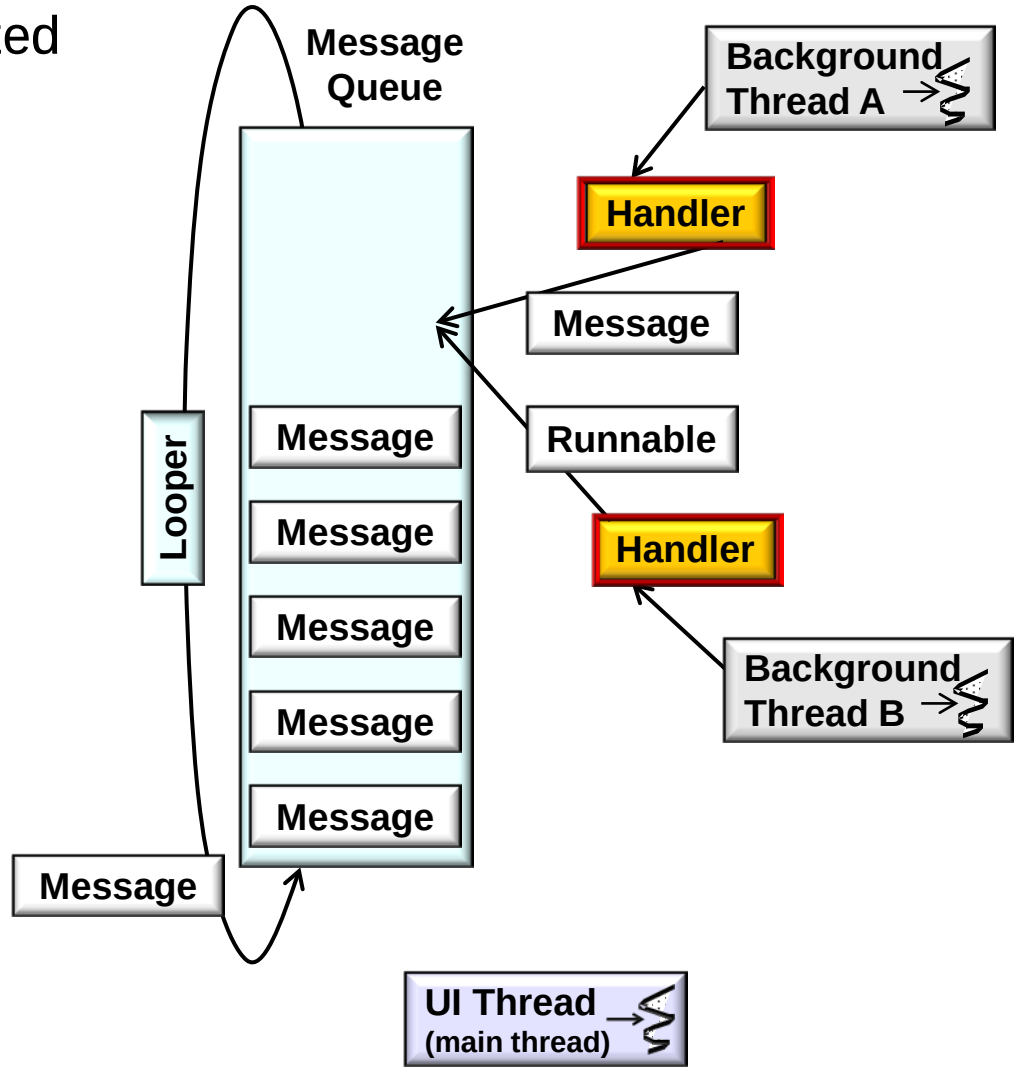




Learning Objectives in this Part of the Module

- Understand the concurrency idioms & programming mechanisms related to the Android HaMeR framework
  - Focusing largely on the Handler class



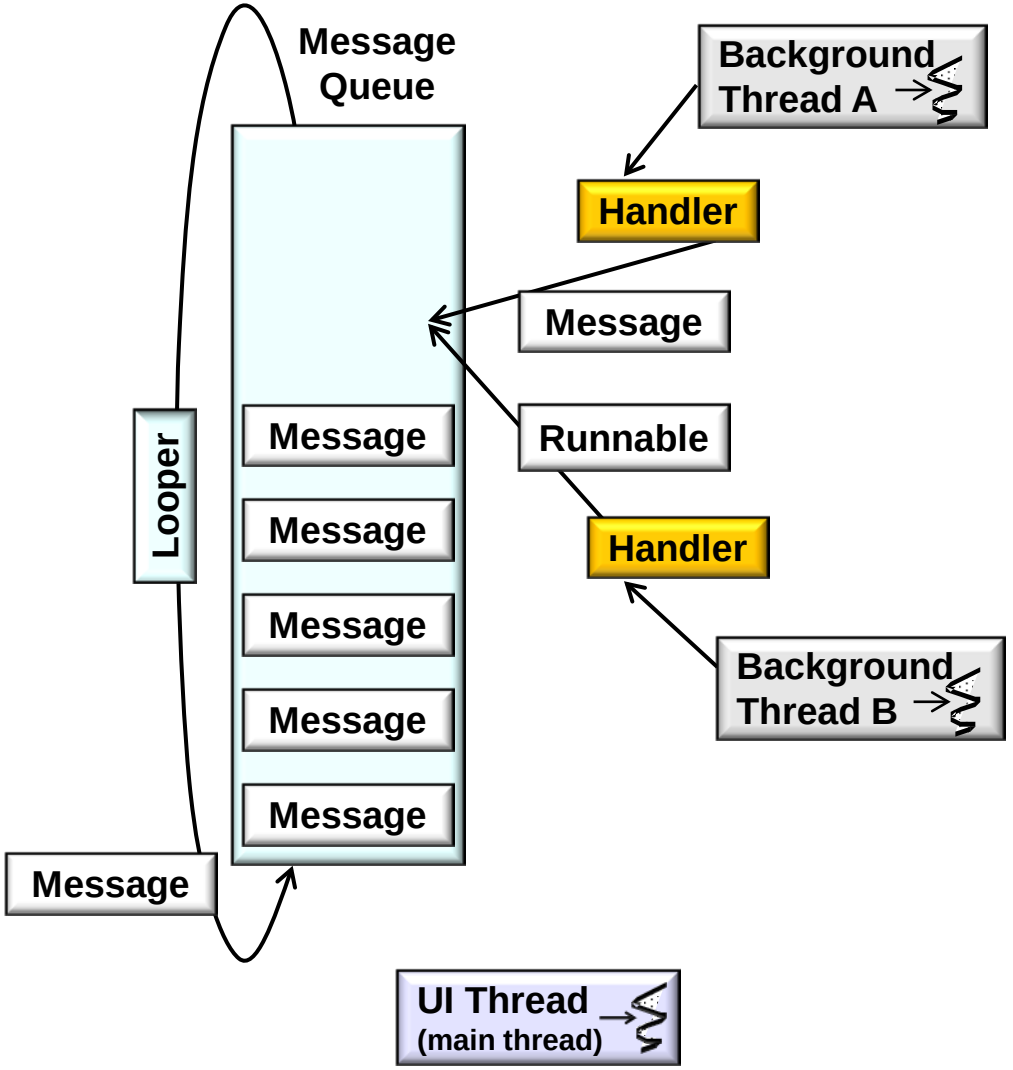


Other classes in the HaMeR framework will be covered in more detail later

Overview of the HaMeR Framework

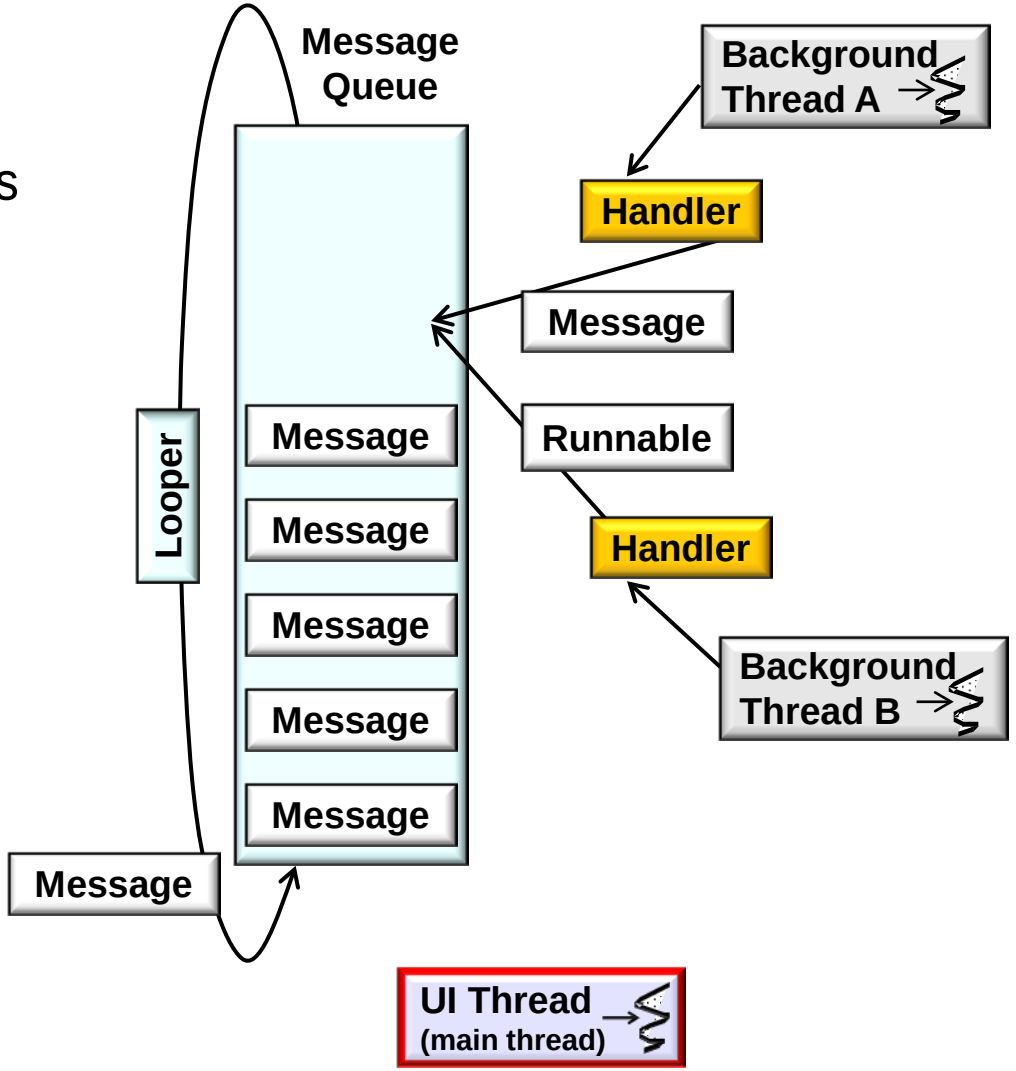
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons



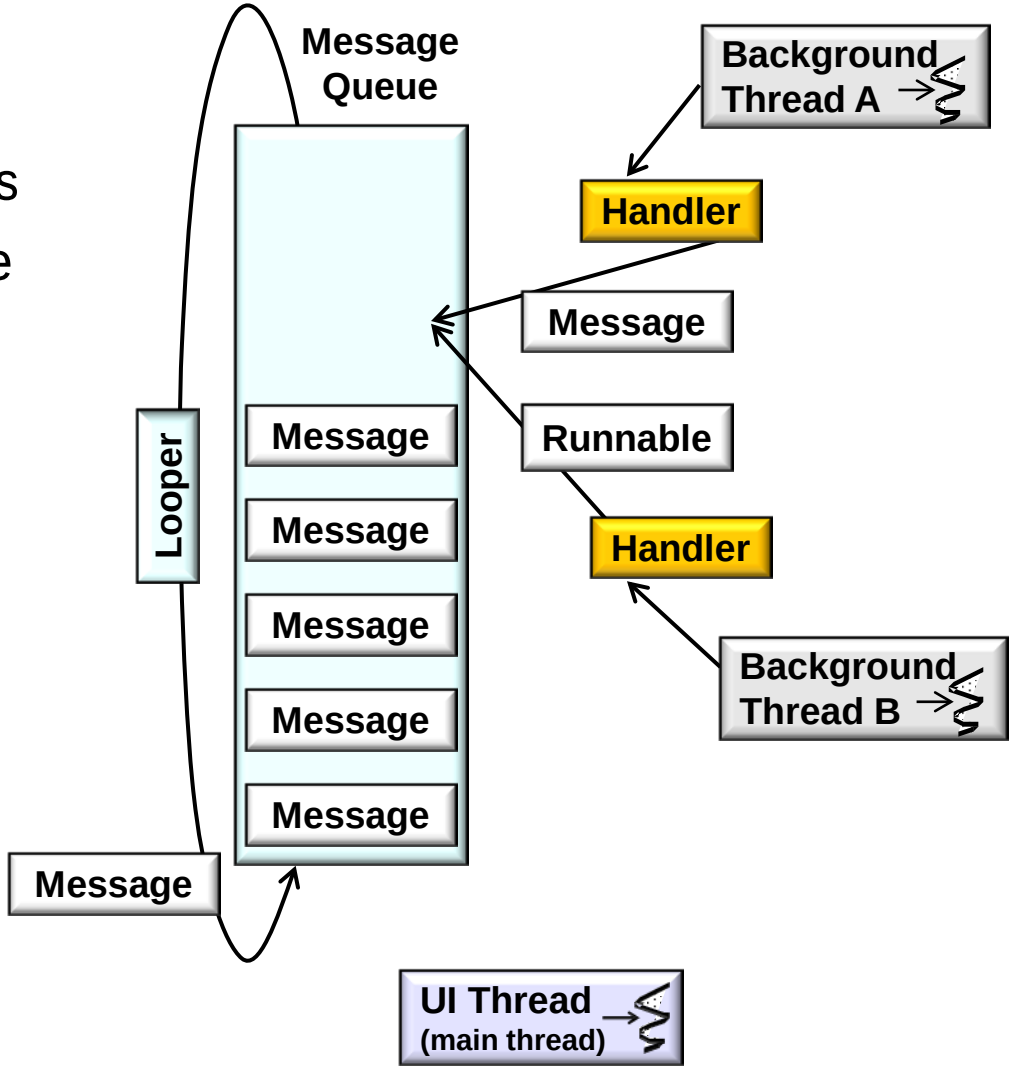
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
  - To initiate concurrent operations



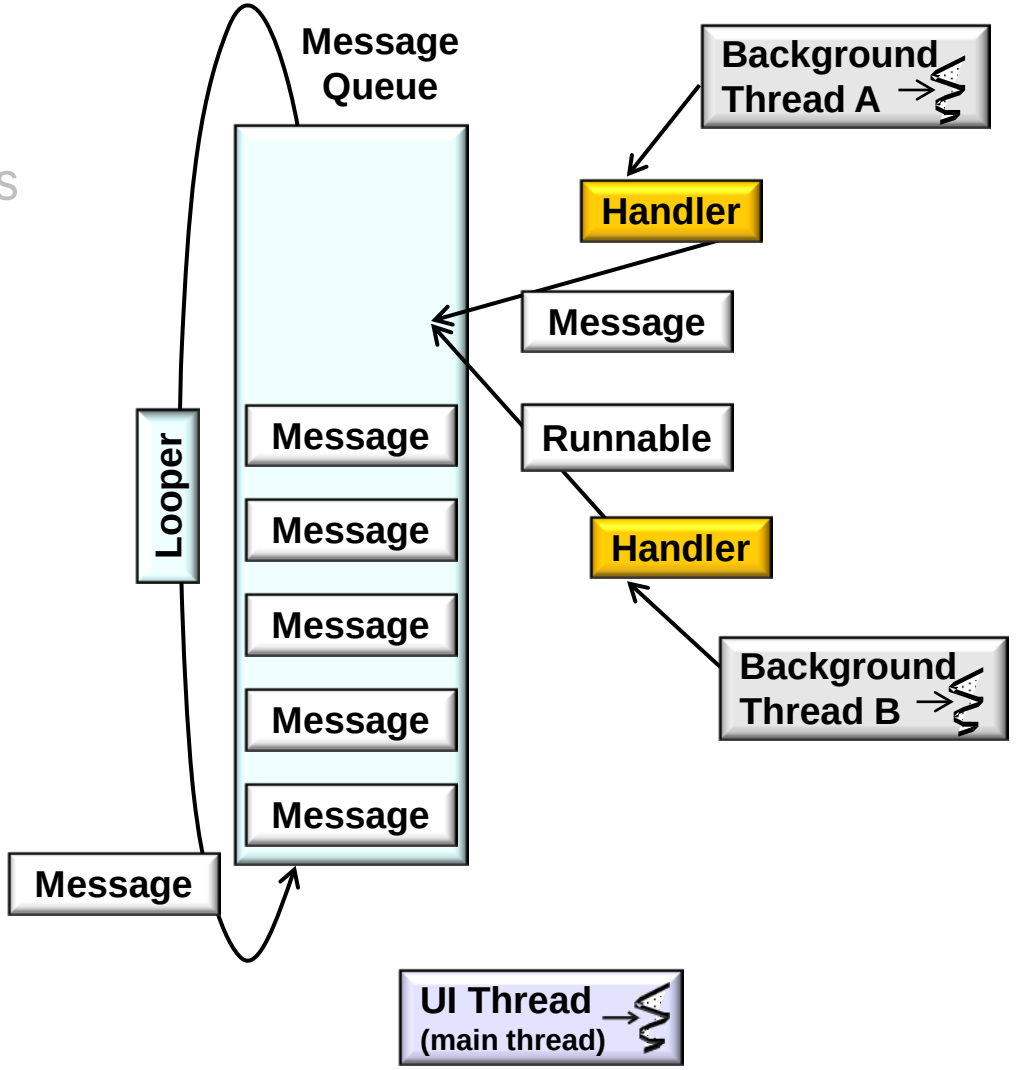
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
  - To initiate concurrent operations
    - e.g., UI thread can trigger the download of a large image in a background thread



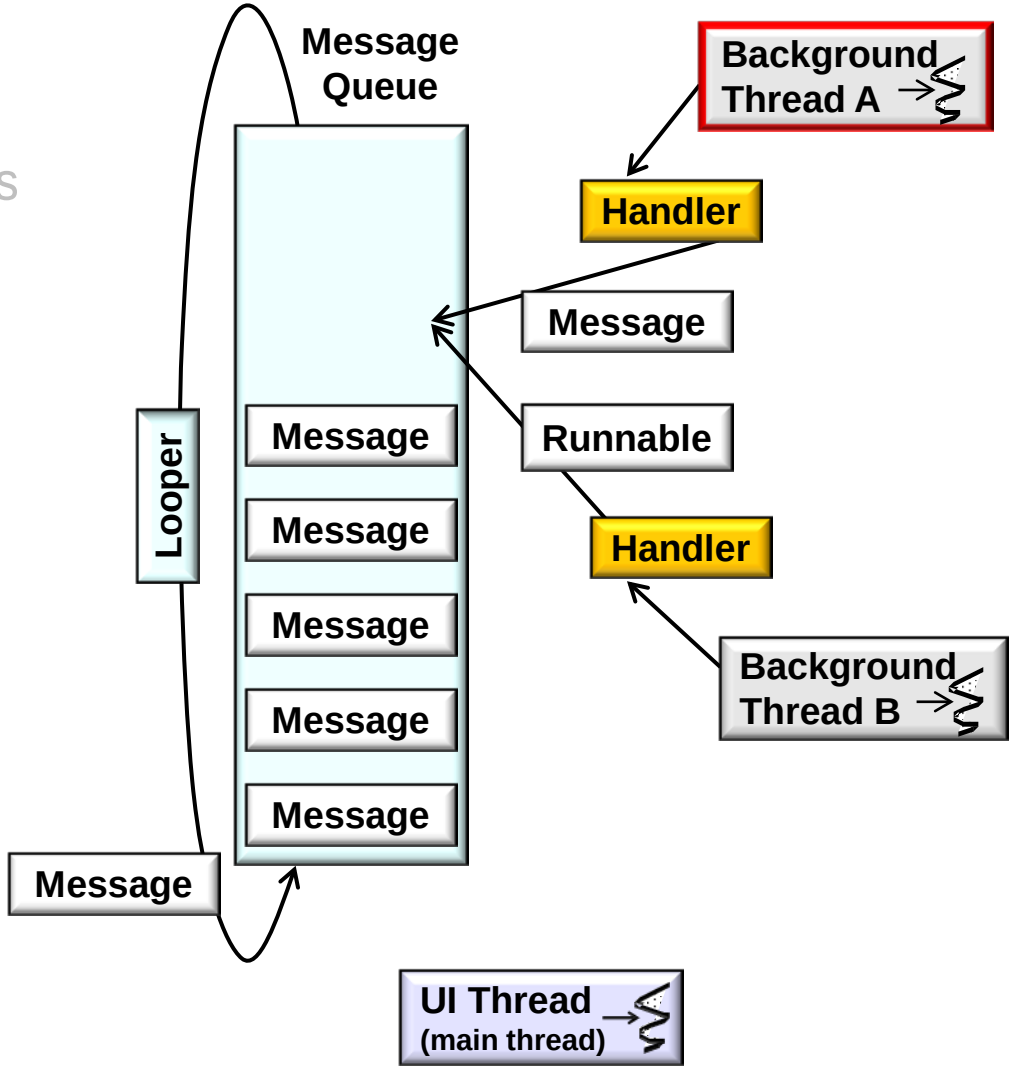
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
  - To initiate concurrent operations
  - To coordinate their behavior



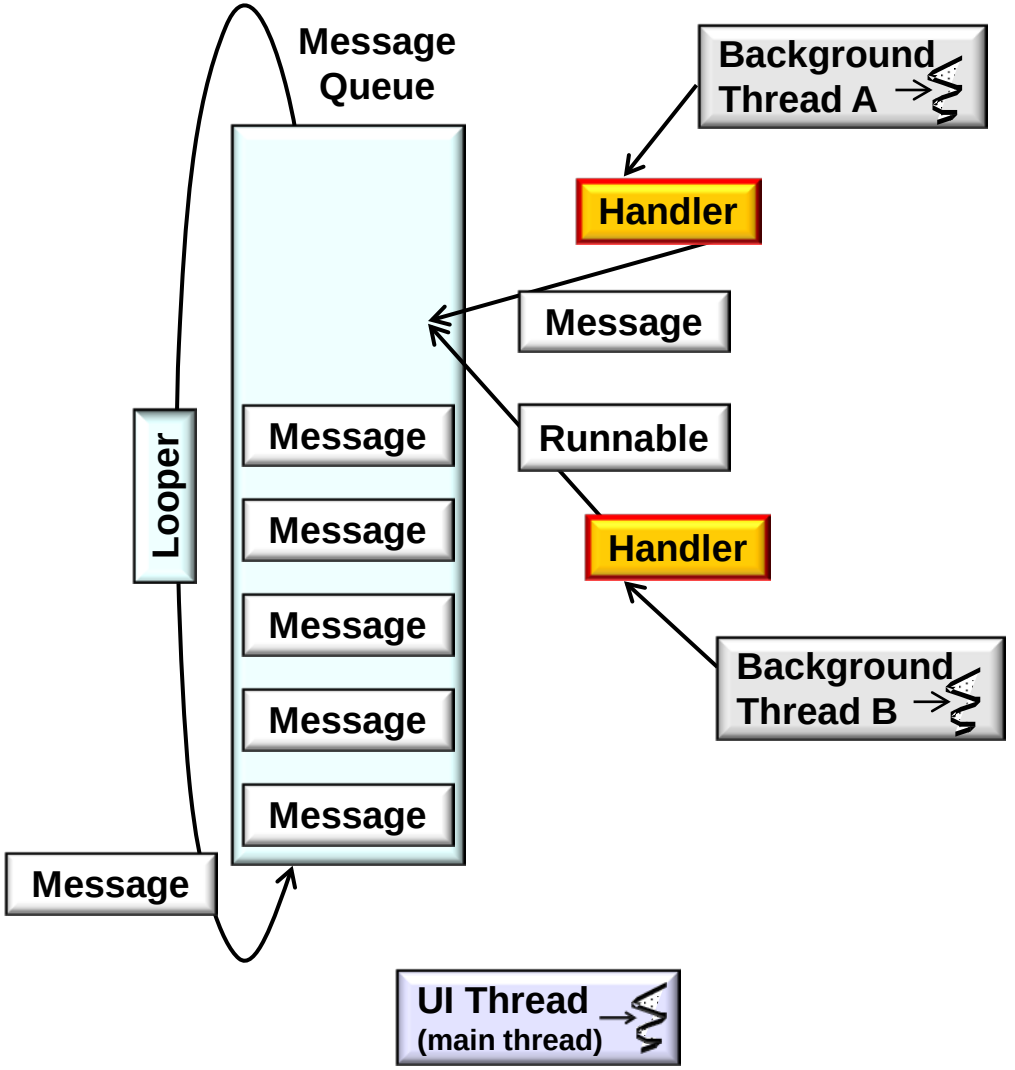
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
  - To initiate concurrent operations
  - To coordinate their behavior
    - e.g., background thread can inform UI thread when image download is complete



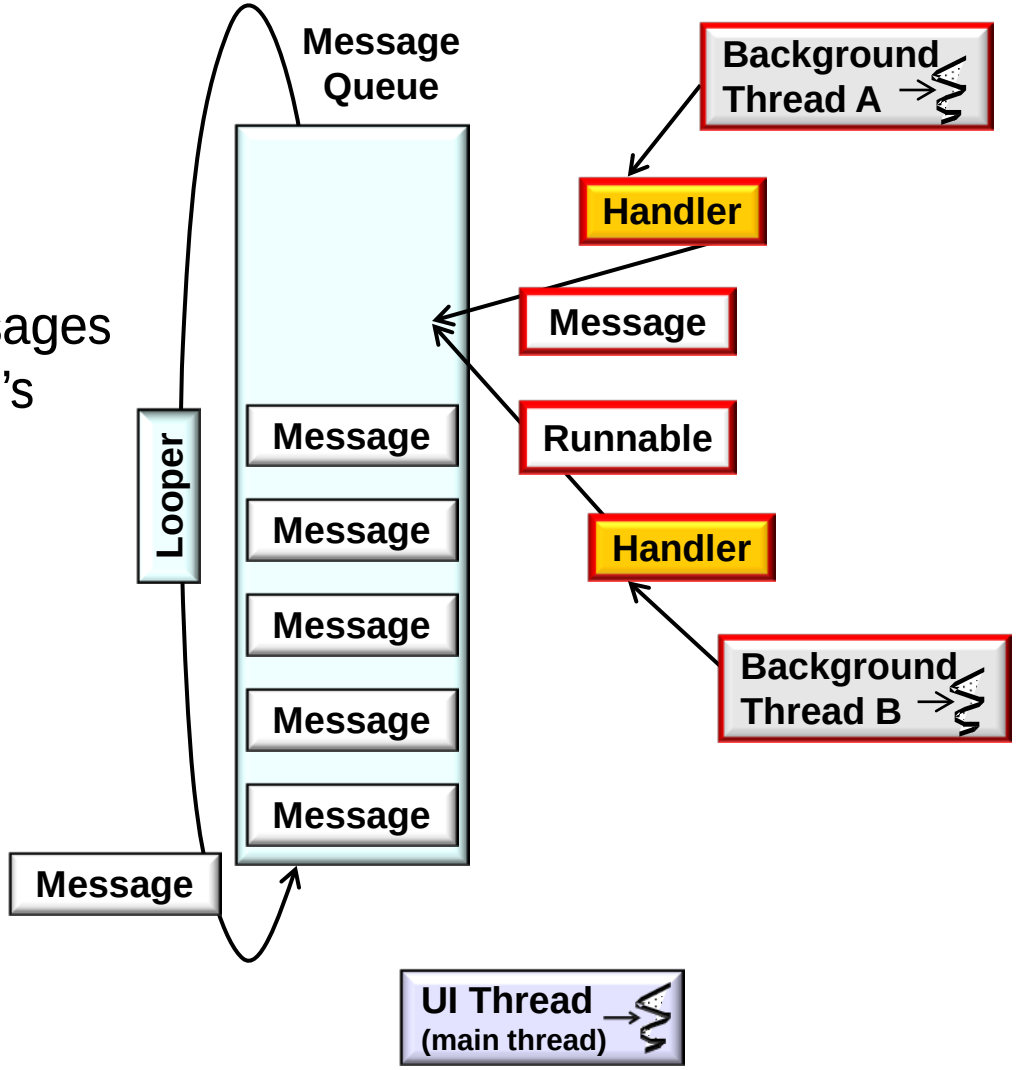
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
- The HaMeR framework supports these use cases



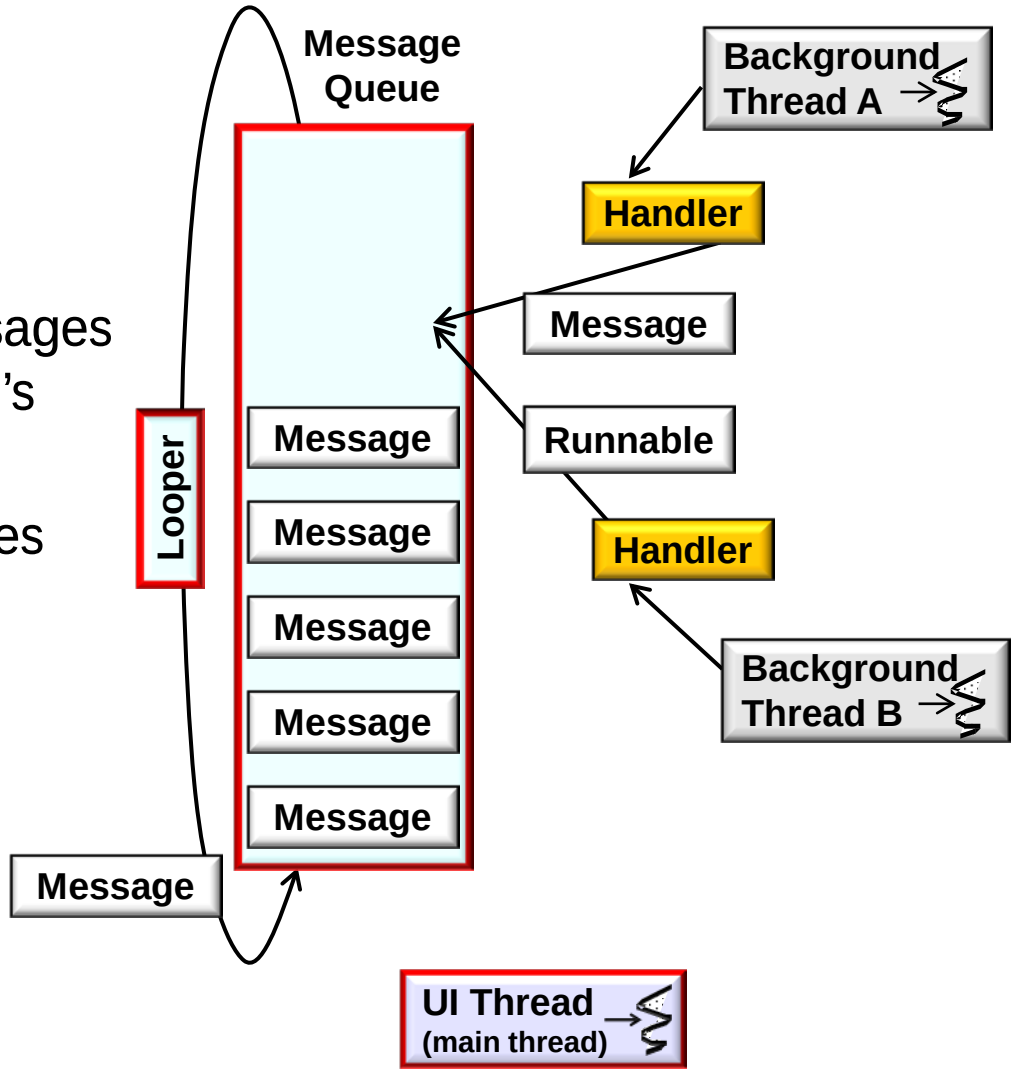
Overview of the HaMeR Framework

- UI & background threads need to interact for various reasons
- The HaMeR framework supports these use cases
  - Background Threads send Messages or post Runnables to UI Thread's MessageQueue via Handlers



Overview of the HaMeR Framework

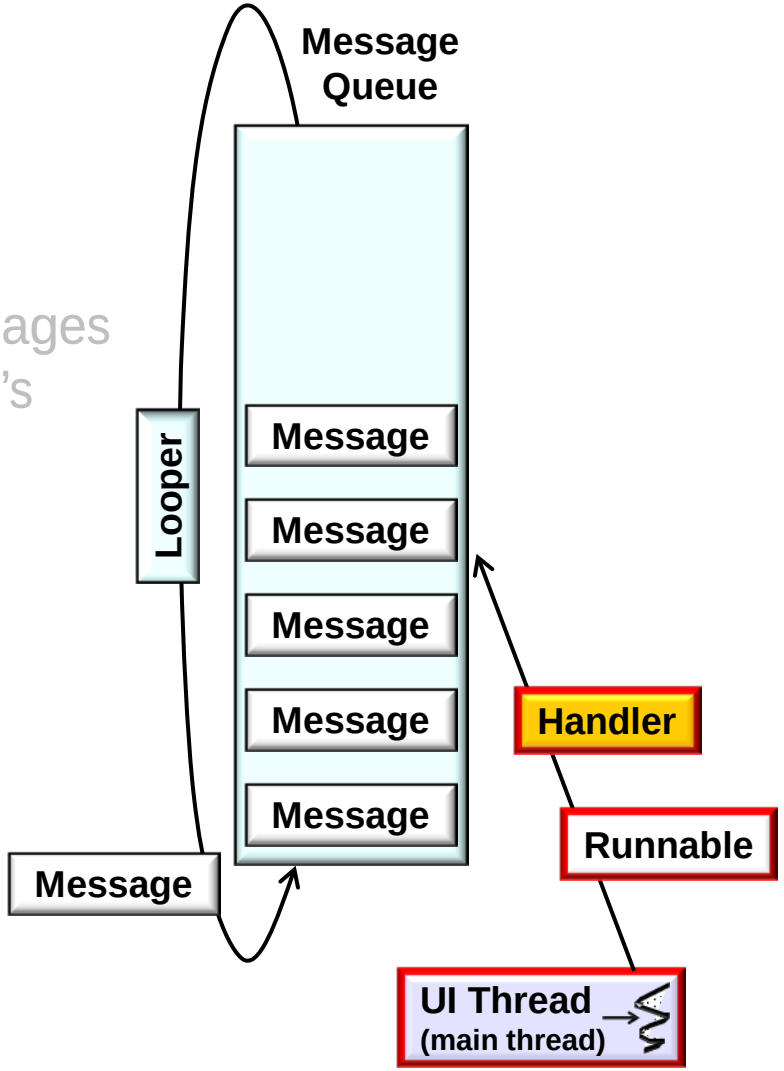
- UI & background threads need to interact for various reasons
- The HaMeR framework supports these use cases
  - Background Threads send Messages or post Runnables to UI Thread's MessageQueue via Handlers
    - UI Thread's Looper coordinates with Handler to process Messages/Runnables



Applications using HaMeR framework may not need to use Java synchronization mechanisms directly

Overview of the HaMeR Framework

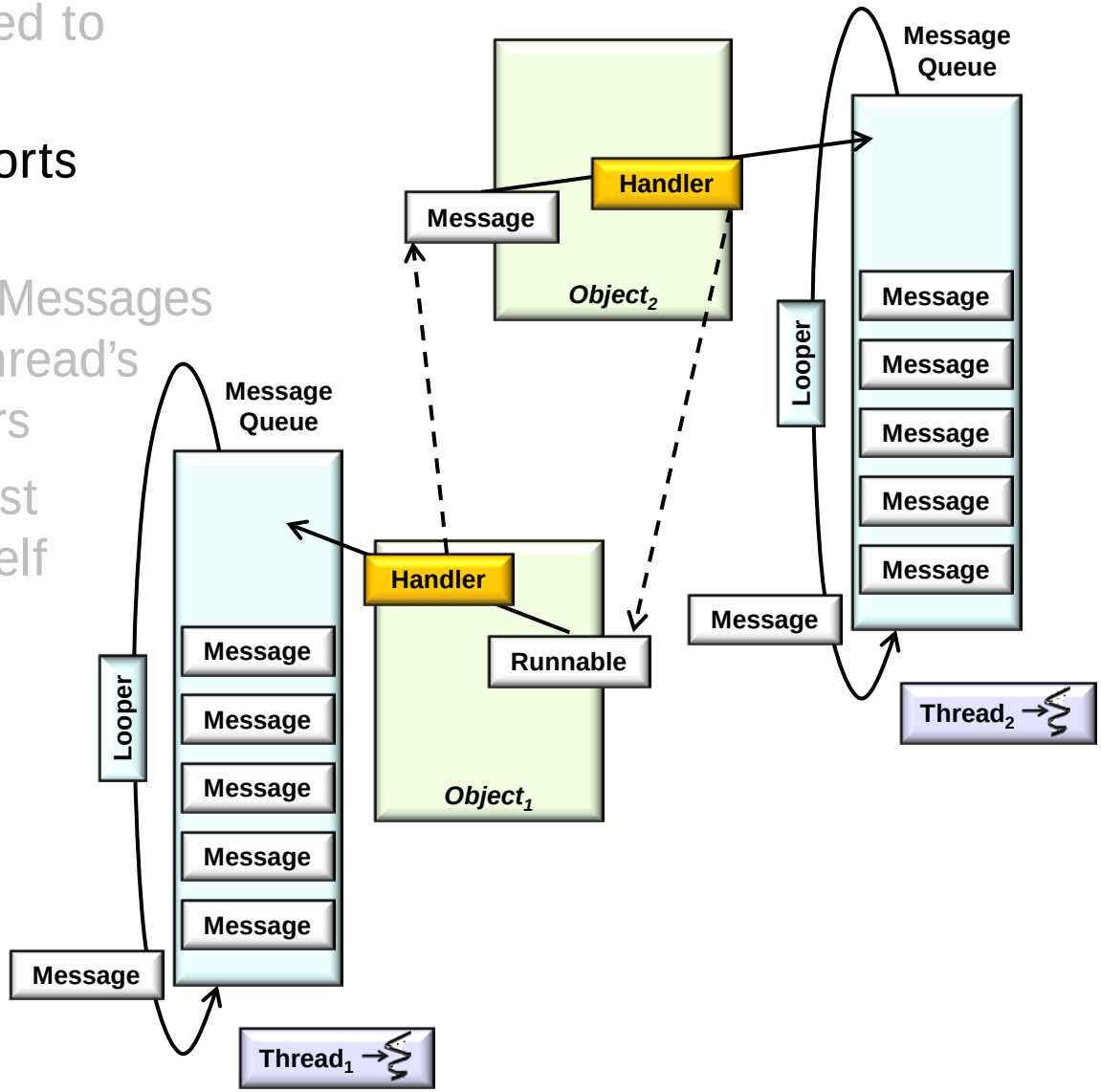
- UI & background threads need to interact for various reasons
- The HaMeR framework supports these use cases
  - Background Threads send Messages or post Runnables to UI Thread's MessageQueue via Handlers
  - A Thread can also send/post Messages/Runnables to itself



Used for deferred processing of Runnables/Messages

Overview of the HaMeR Framework

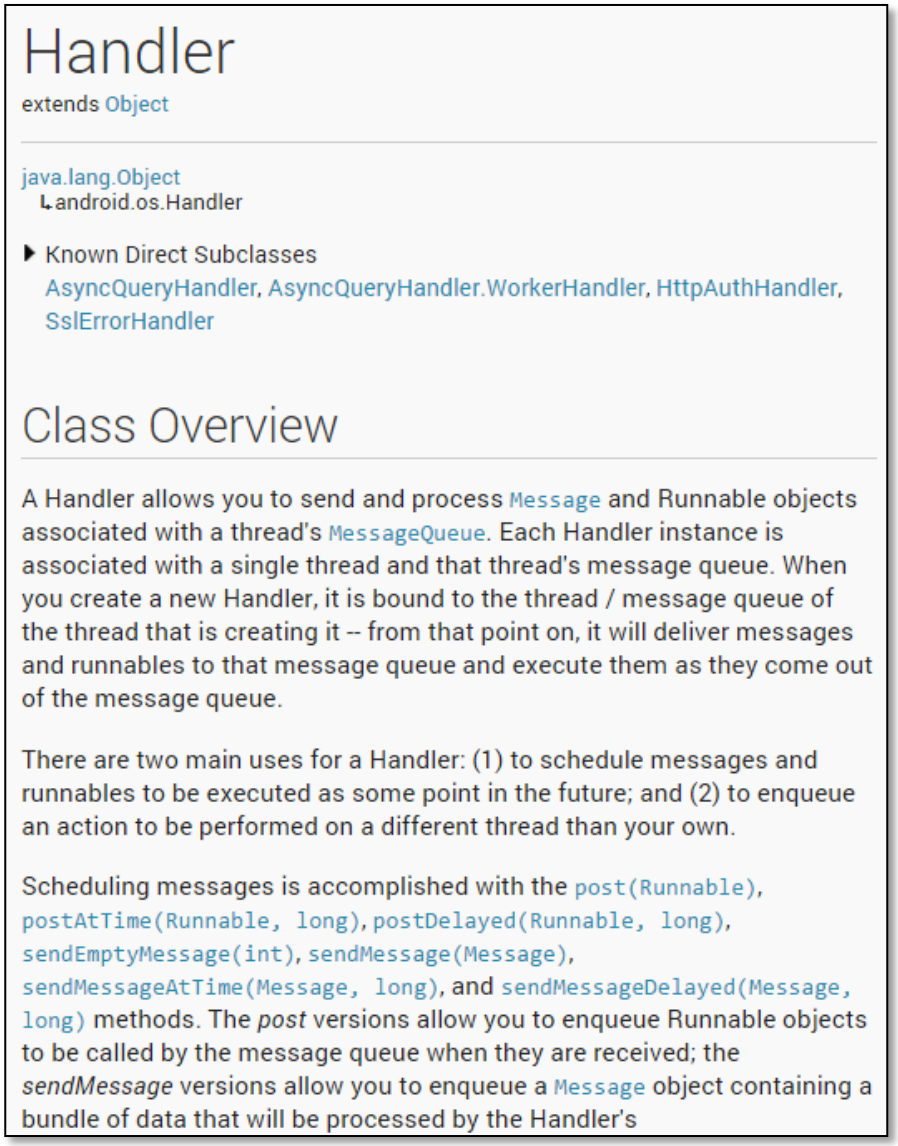
- UI & background threads need to interact for various reasons
- The HaMeR framework supports these use cases
  - Background Threads send Messages or post Runnables to UI Thread's MessageQueue via Handlers
  - A Thread can also send/post Messages/Runnables to itself
  - Background Threads can also interact via Handlers



Overview of the Handler Class

Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads



Handler

extends Object

java.lang.Object
↳ android.os.Handler

▸ Known Direct Subclasses
AsyncQueryHandler, AsyncQueryHandler.WorkerHandler, HttpAuthHandler, SslErrorHandler

Class Overview

A Handler allows you to send and process `Message` and Runnable objects associated with a thread's `MessageQueue`. Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler, it is bound to the thread / message queue of the thread that is creating it -- from that point on, it will deliver messages and runnables to that message queue and execute them as they come out of the message queue.

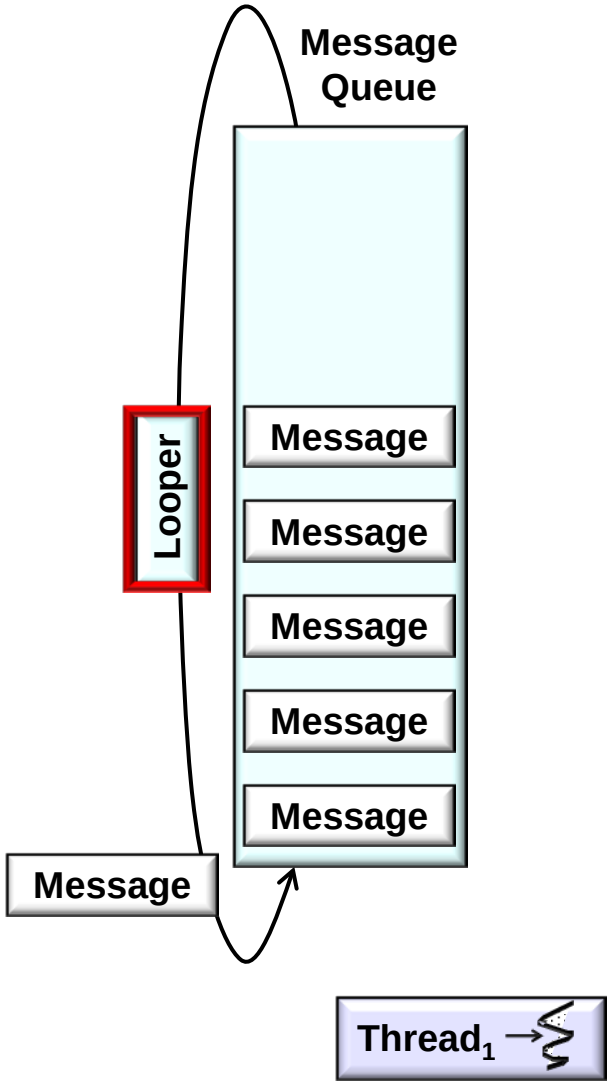
There are two main uses for a Handler: (1) to schedule messages and runnables to be executed as some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.

Scheduling messages is accomplished with the `post(Runnable)`, `postAtTime(Runnable, long)`, `postDelayed(Runnable, long)`, `sendEmptyMessage(int)`, `sendMessage(Message)`, `sendMessageAtTime(Message, long)`, and `sendMessageDelayed(Message, long)` methods. The *post* versions allow you to enqueue Runnable objects to be called by the message queue when they are received; the *sendMessage* versions allow you to enqueue a `Message` object containing a bundle of data that will be processed by the Handler's

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

Overview of the Handler Class

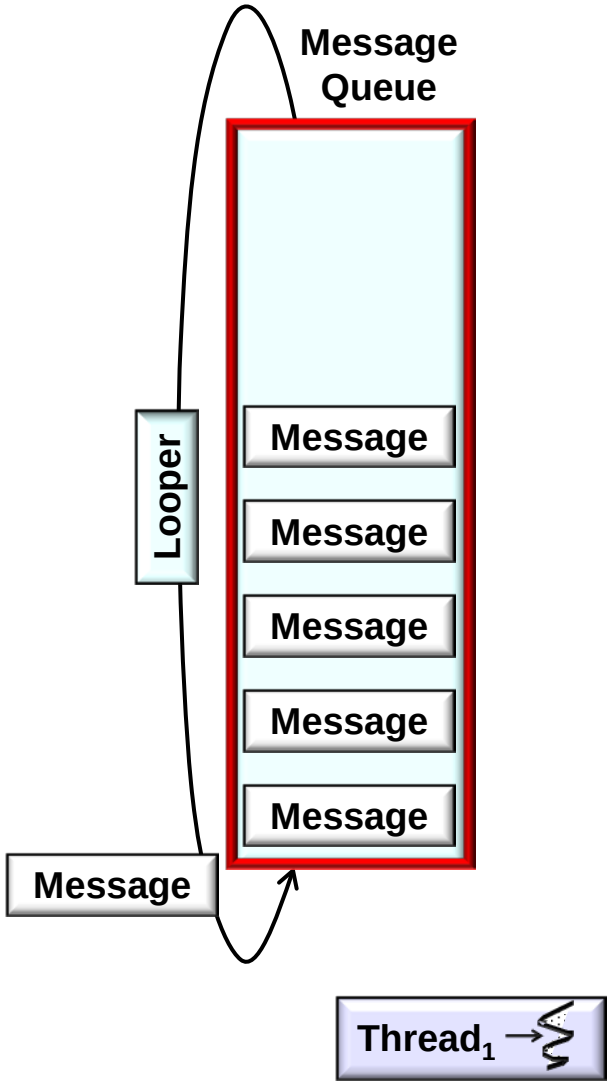
- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue



See upcoming part on "The Android Looper"

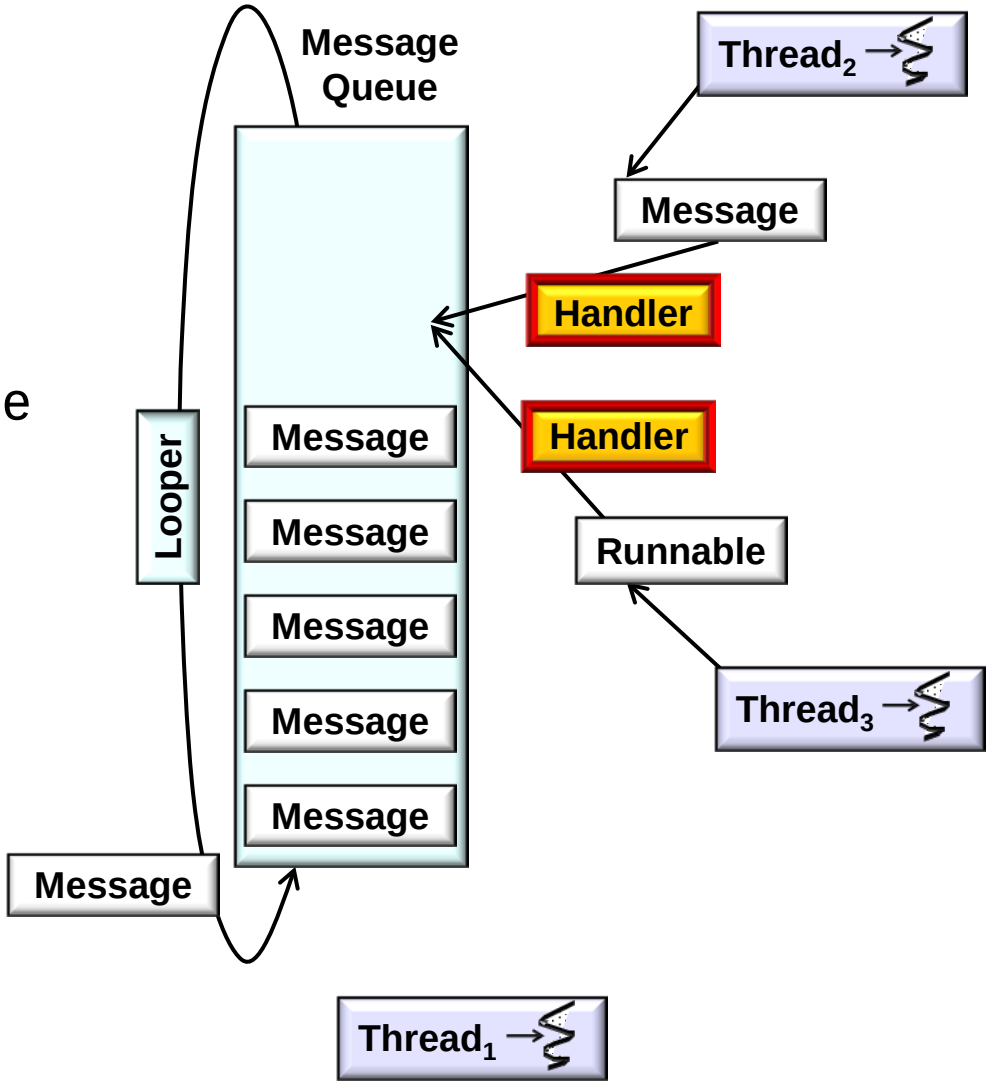
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue



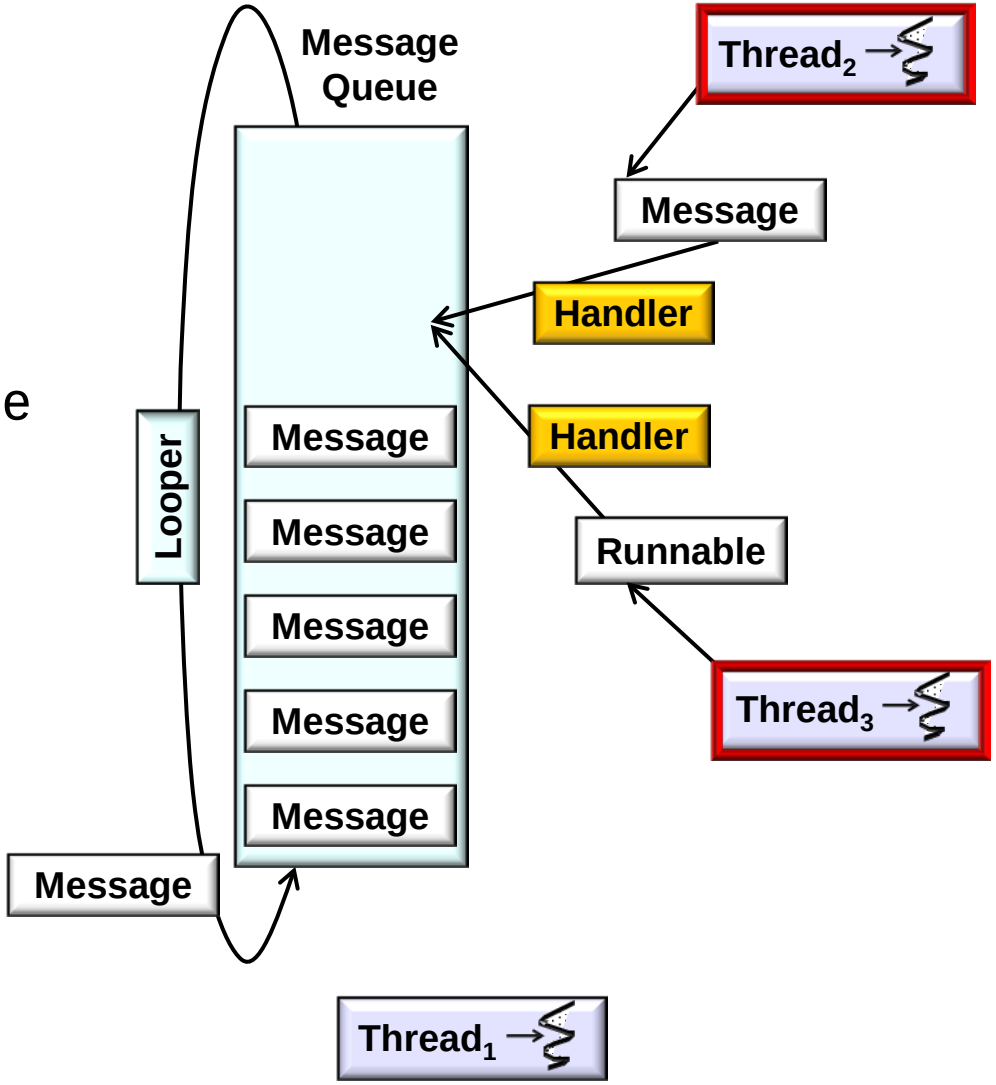
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
    - Used to process Messages & Runnables placed on the queue by one or more Threads



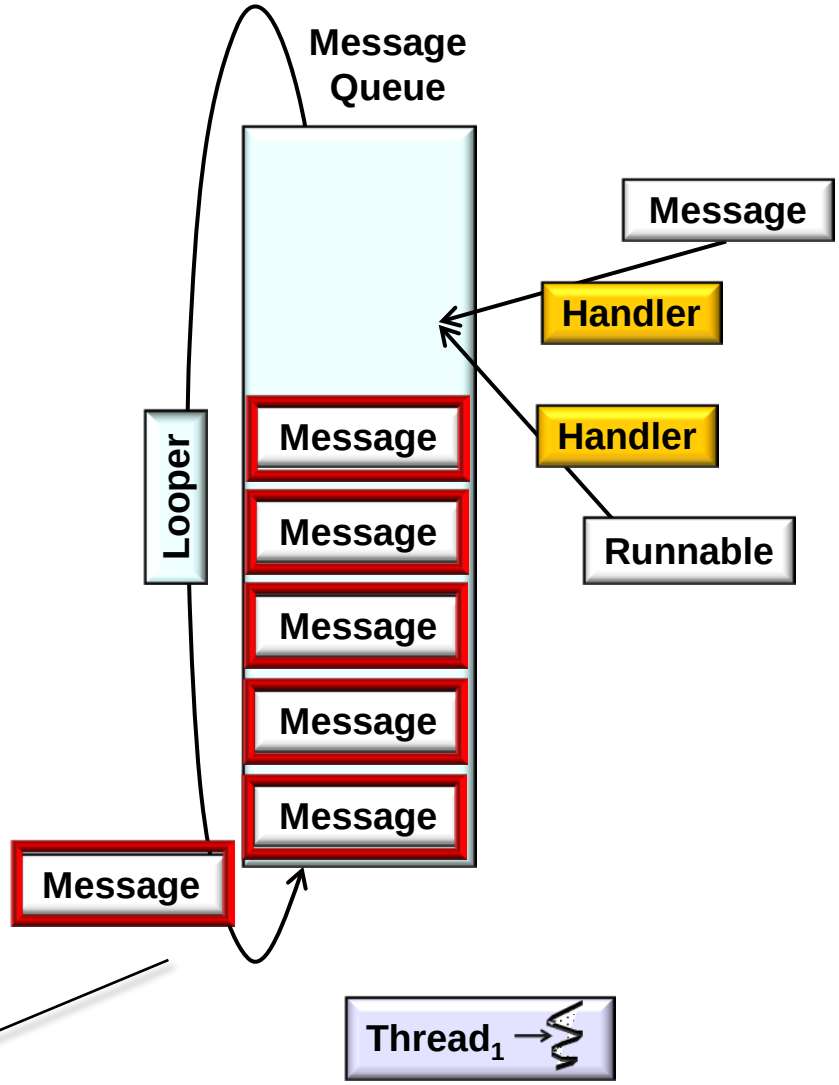
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
    - Used to process Messages & Runnables placed on the queue by one or more Threads



Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
  - MessageQueue management is done by an instance of Handler (in conjunction with a Looper)

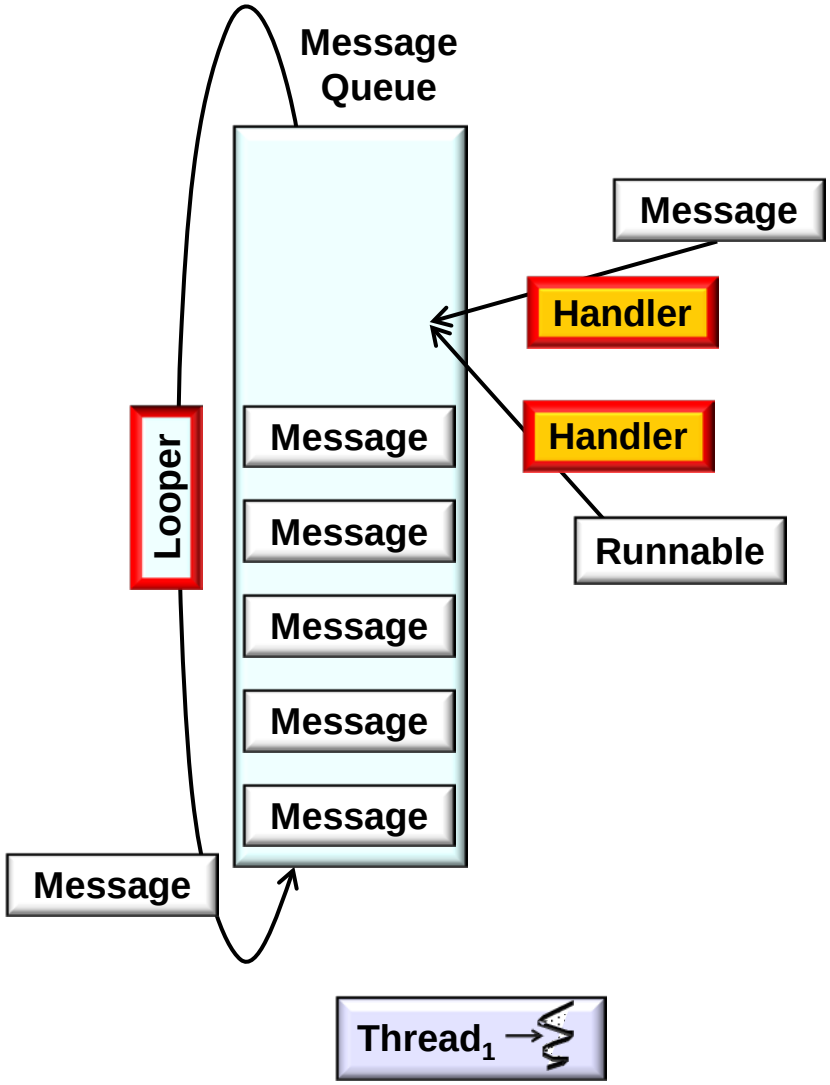


A Handler adds & removes Messages in a MessageQueue & dispatches Messages to their intended targets

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

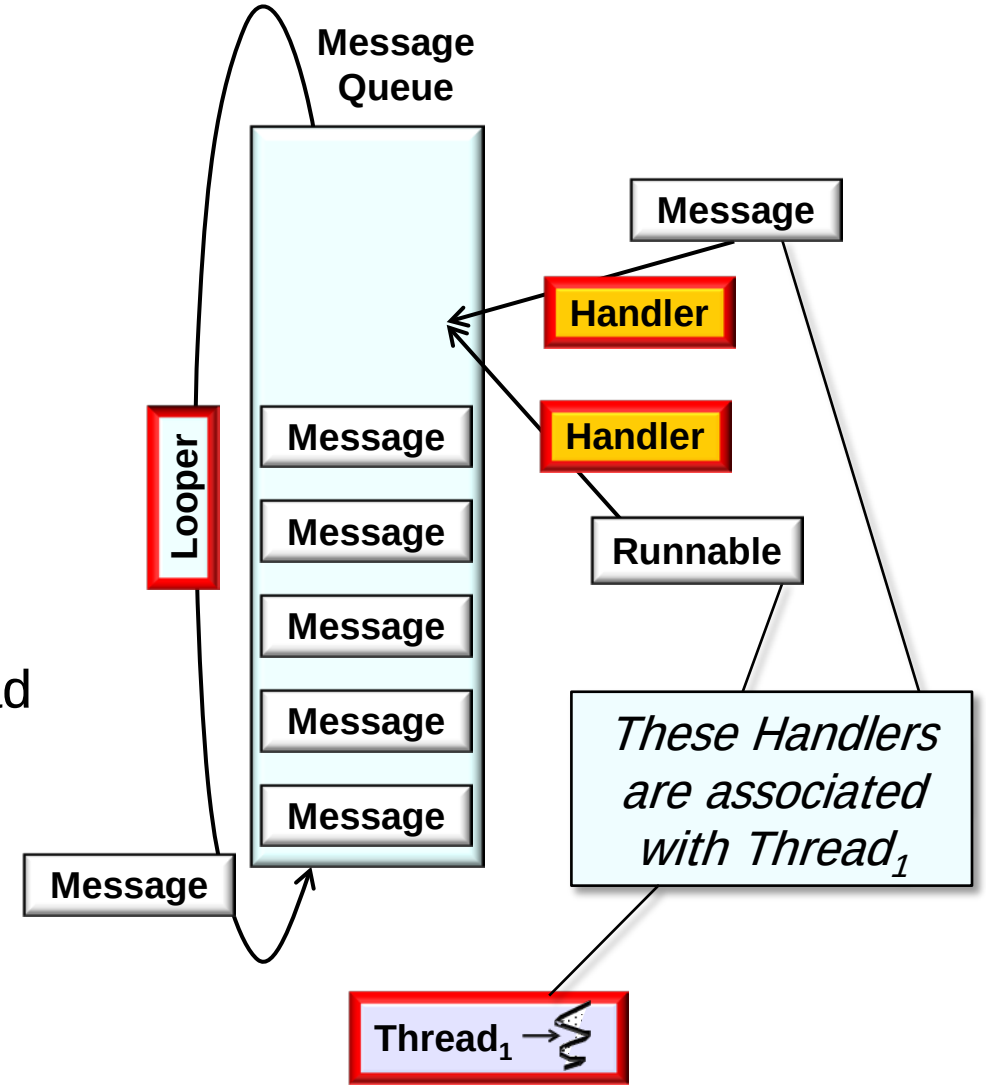
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
  - MessageQueue management is done by an instance of Handler (in conjunction with a Looper)
    - A Handler is associated with a particular Looper



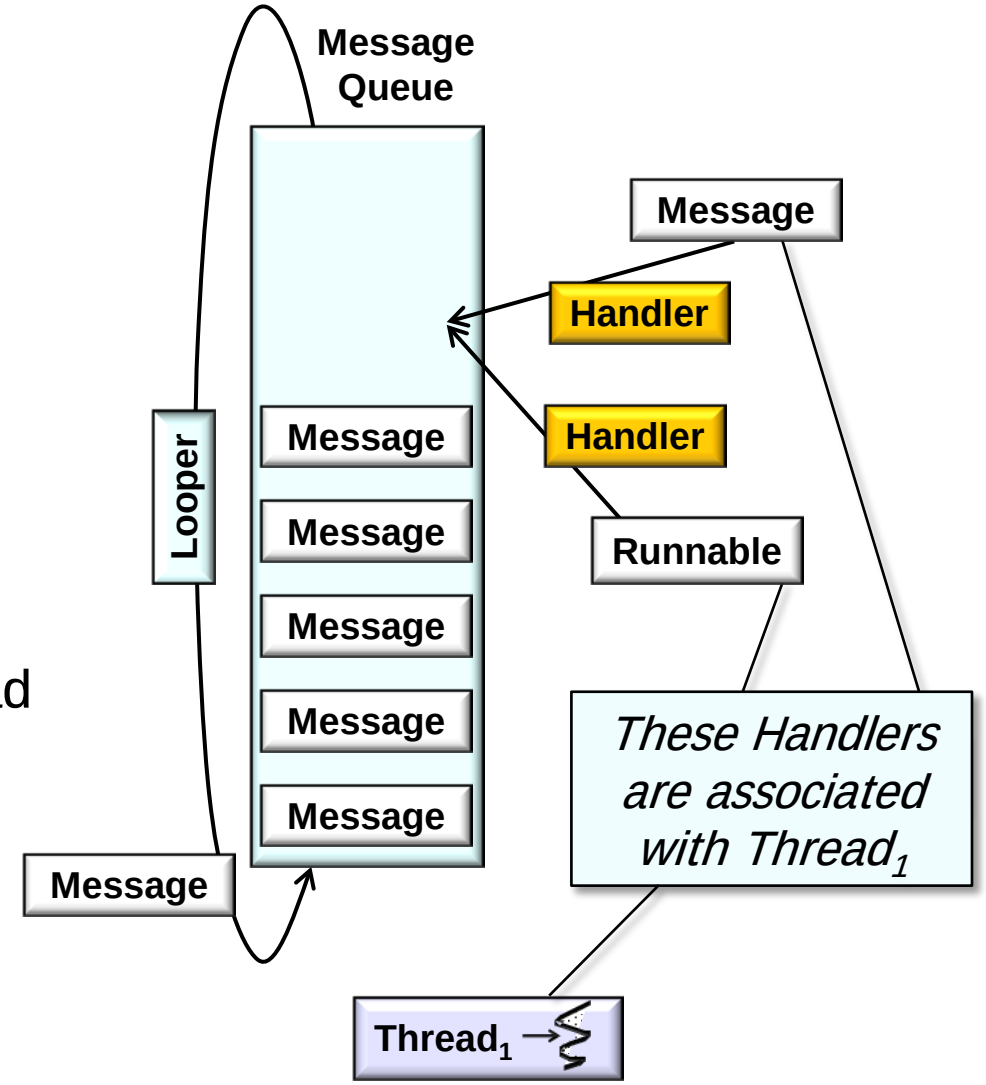
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
  - MessageQueue management is done by an instance of Handler (in conjunction with a Looper)
    - A Handler is associated with a particular Looper
      - Defaults to Looper in Thread where Handler was created



Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
  - A Looper has a MessageQueue
  - MessageQueue management is done by an instance of Handler (in conjunction with a Looper)
    - A Handler is associated with a particular Looper
      - Defaults to Looper in Thread where Handler was created



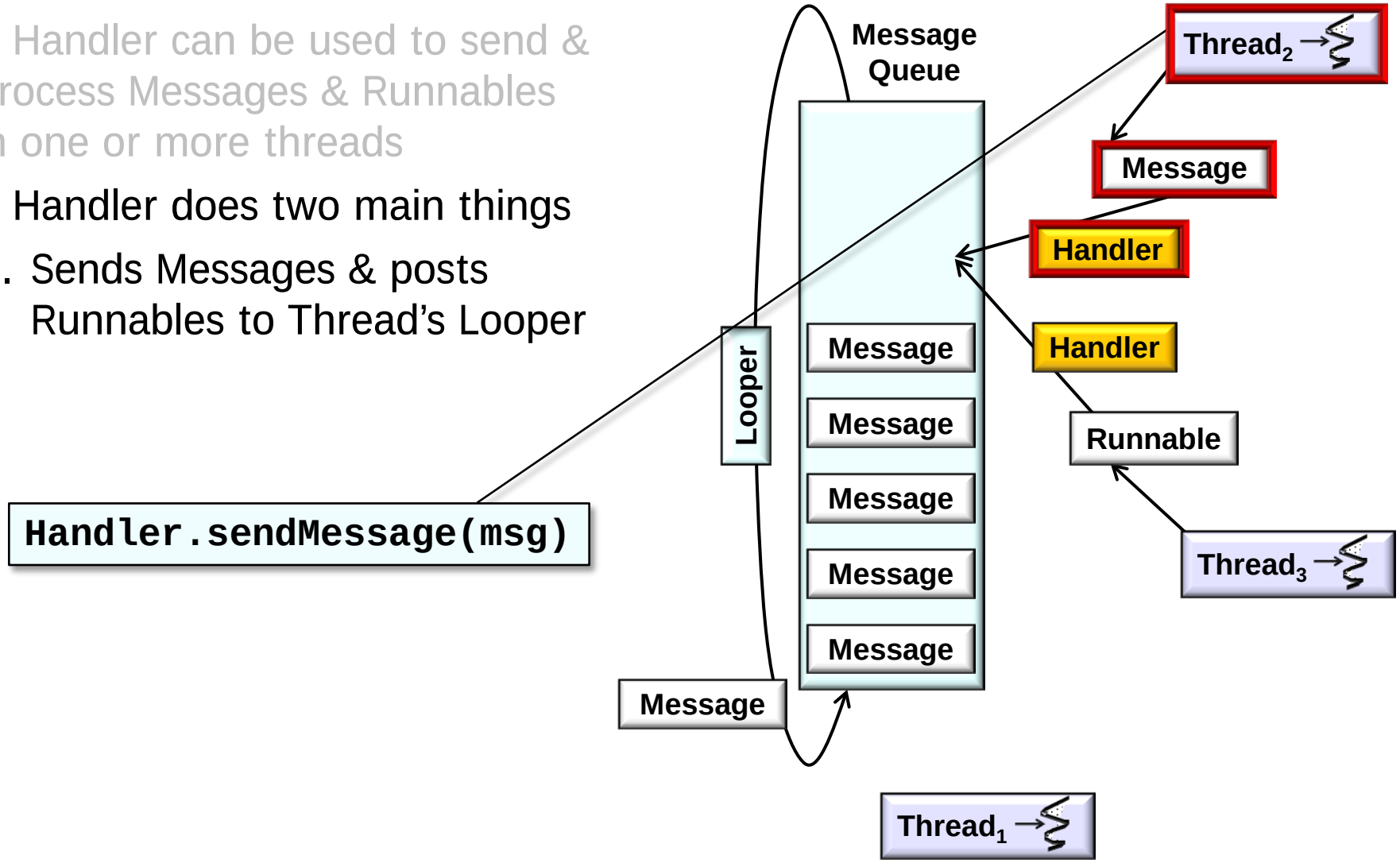
A Handler can be associated with a different Looper & a different Thread by passing the Looper as a parameter to its constructor

Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things

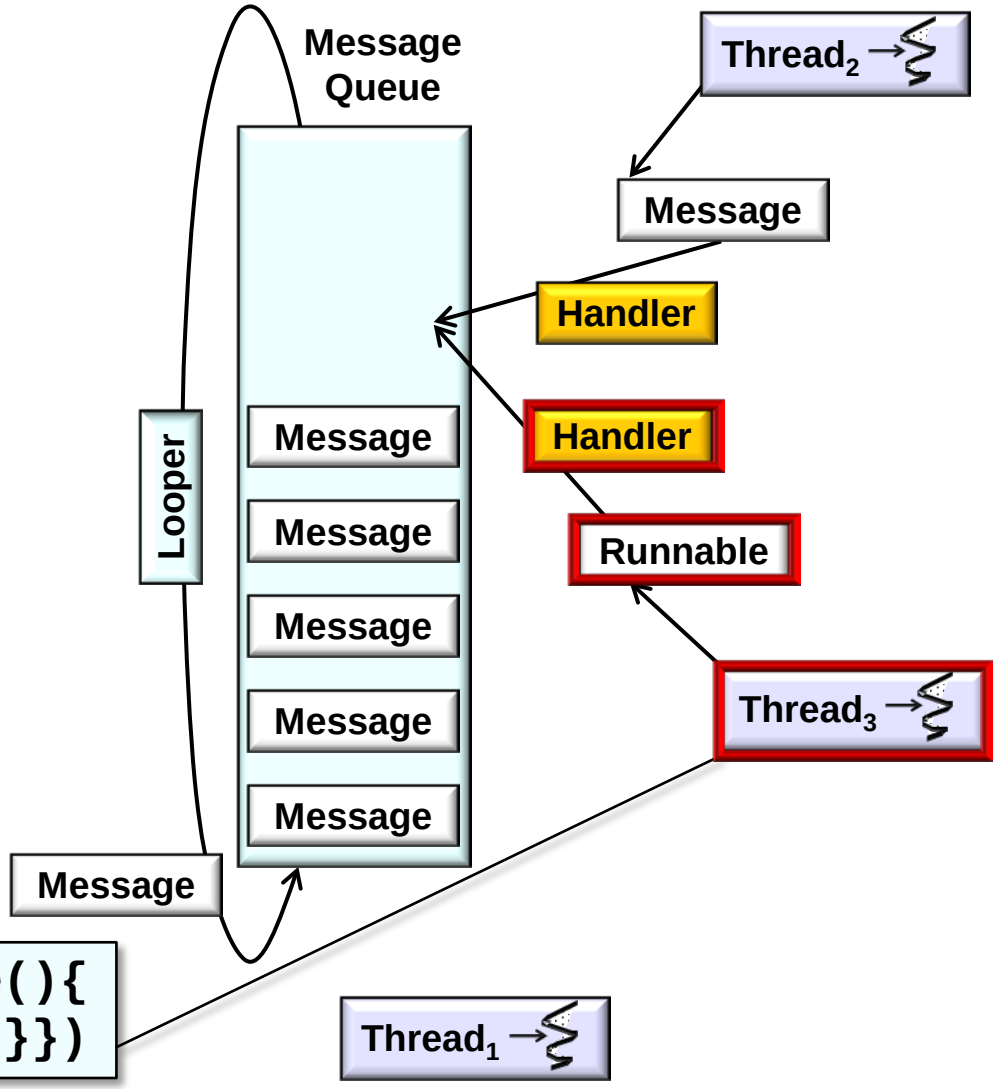
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper



Overview of the Handler Class

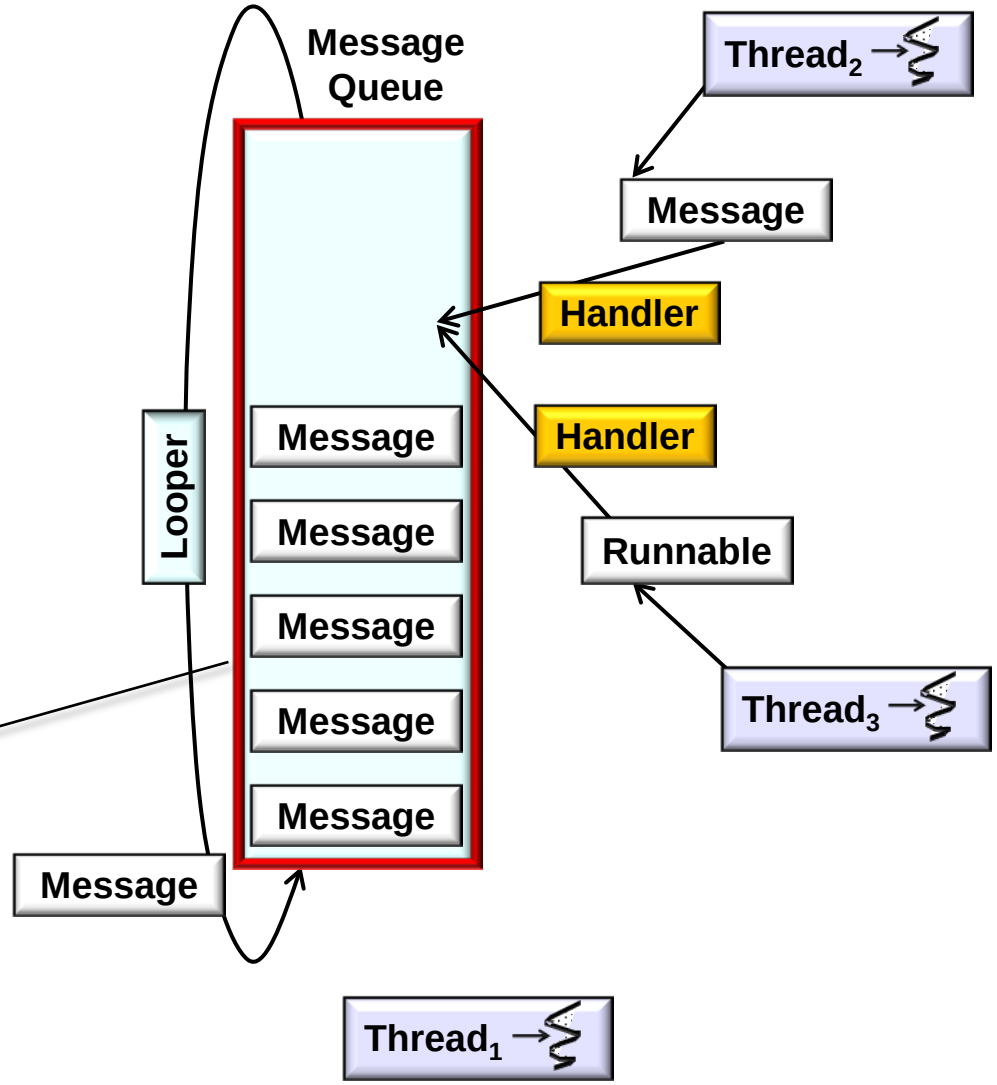
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper




```
Handler.post(new Runnable(){
  public void run(){ ... }})
```

Overview of the Handler Class

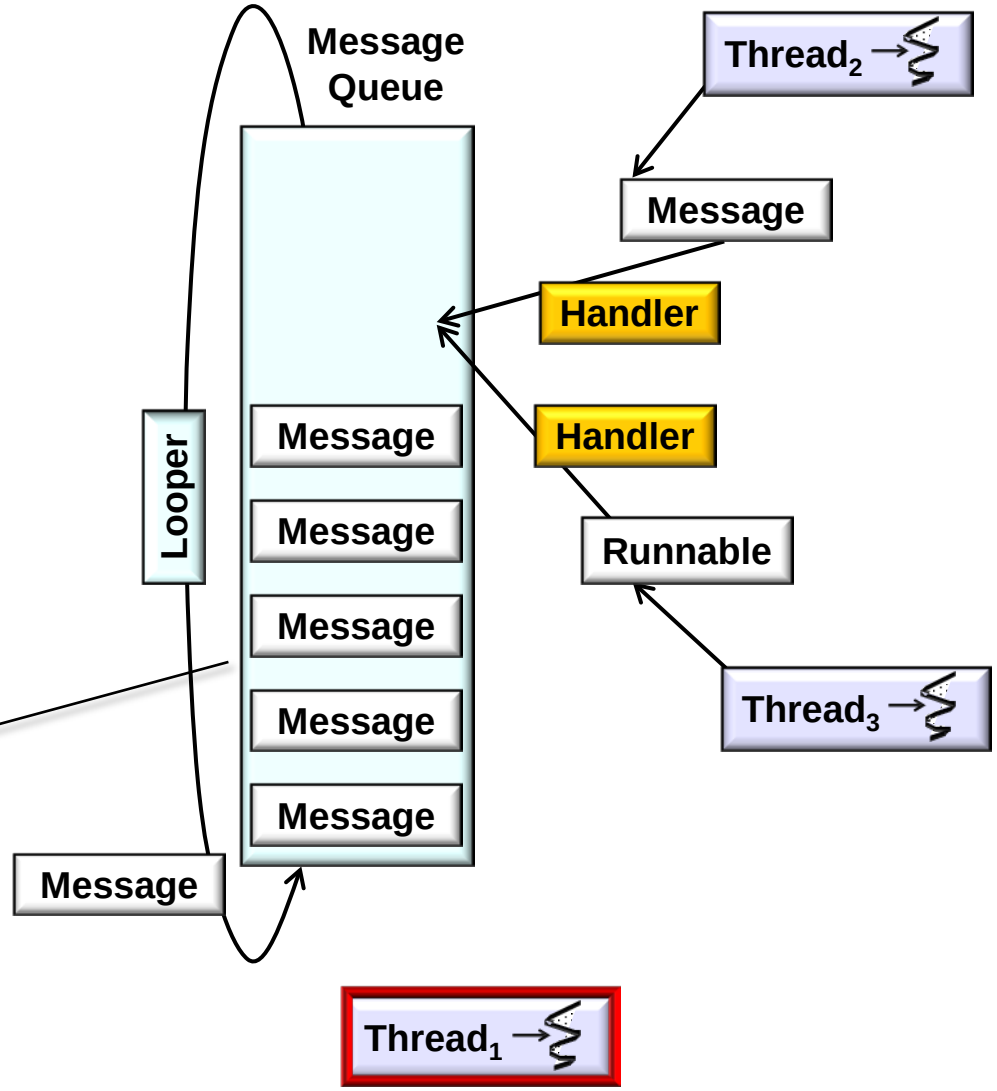
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
     - The Looper's MessageQueue enqueues & schedules them for future execution




```
queue.enqueueMessage
  (msg, uptimeMillis)
```

Overview of the Handler Class

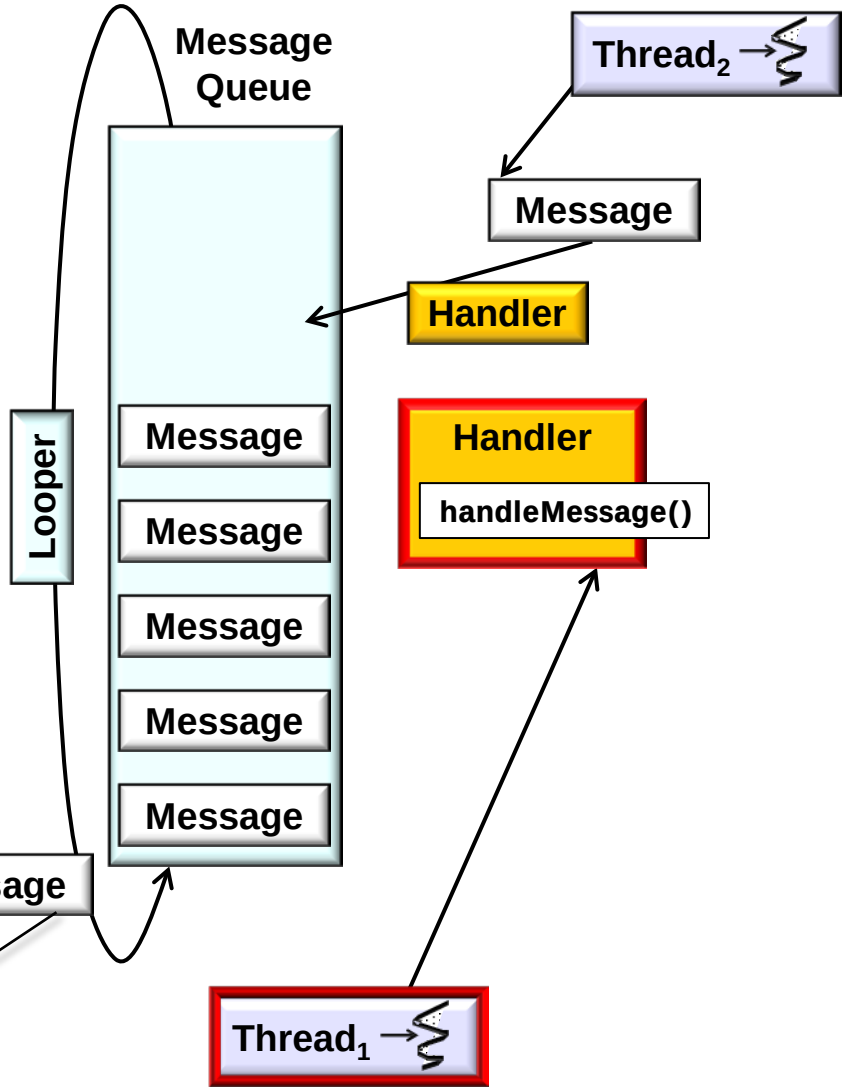
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
     - The Looper's MessageQueue enqueues & schedules them for future execution




```
queue.enqueueMessage
  (msg, uptimeMillis)
```

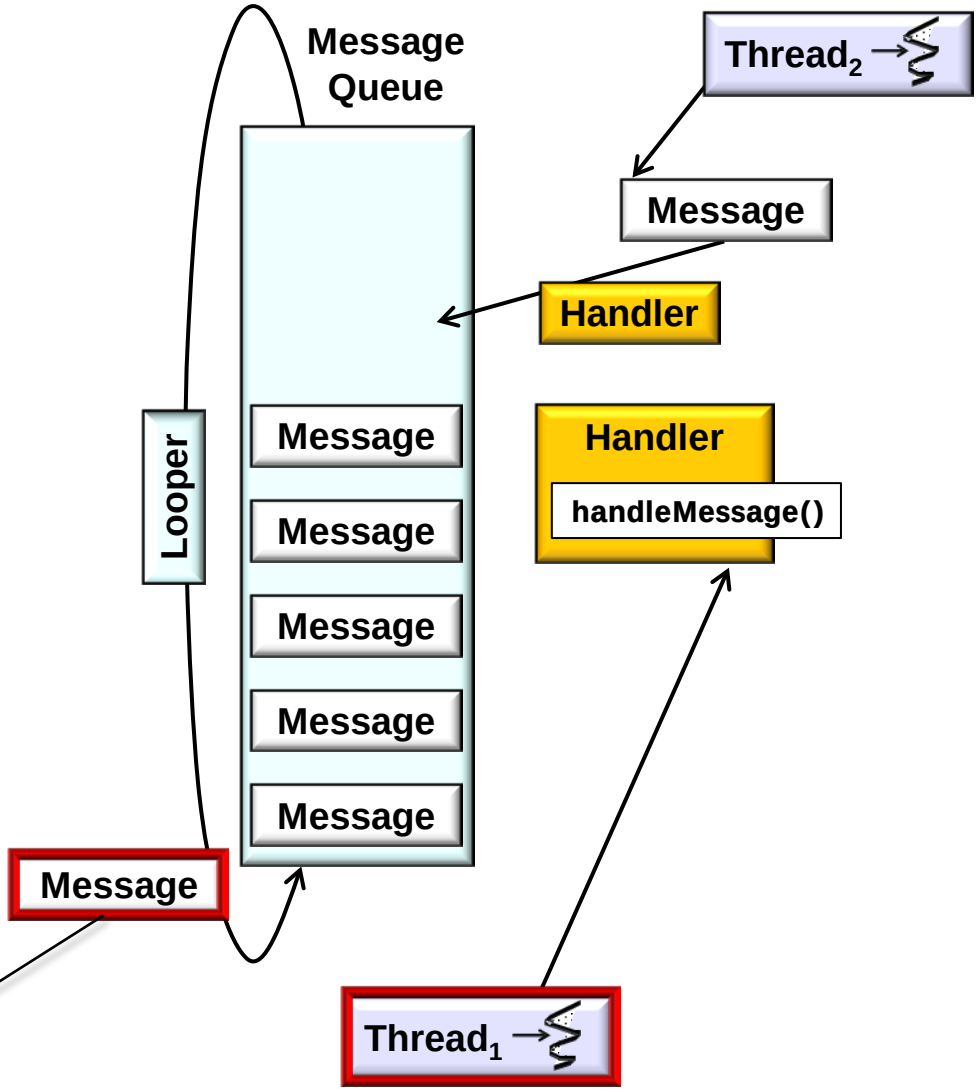
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
  2. Collaborates with Looper to serialize the processing of Messages in a Thread



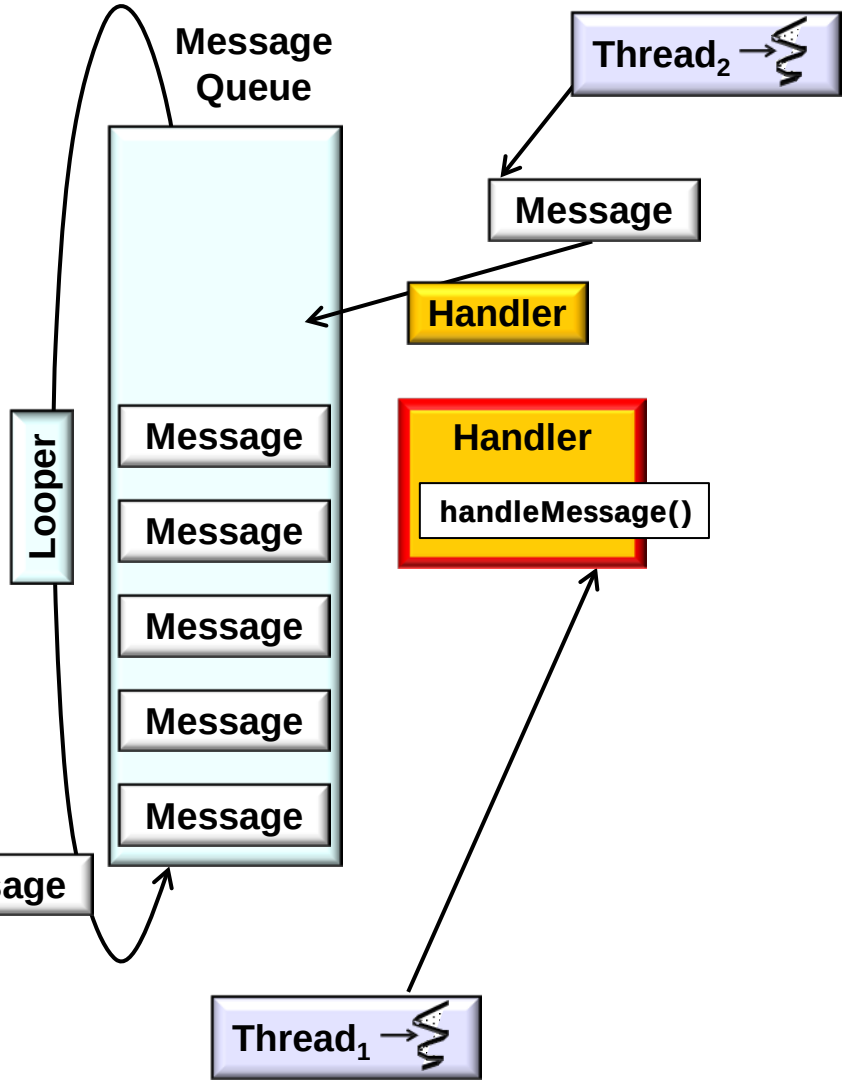
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
  2. Collaborates with Looper to serialize the processing of Messages in a Thread
     - Each Message (& Runnable) tracks its associated Handler



Overview of the Handler Class

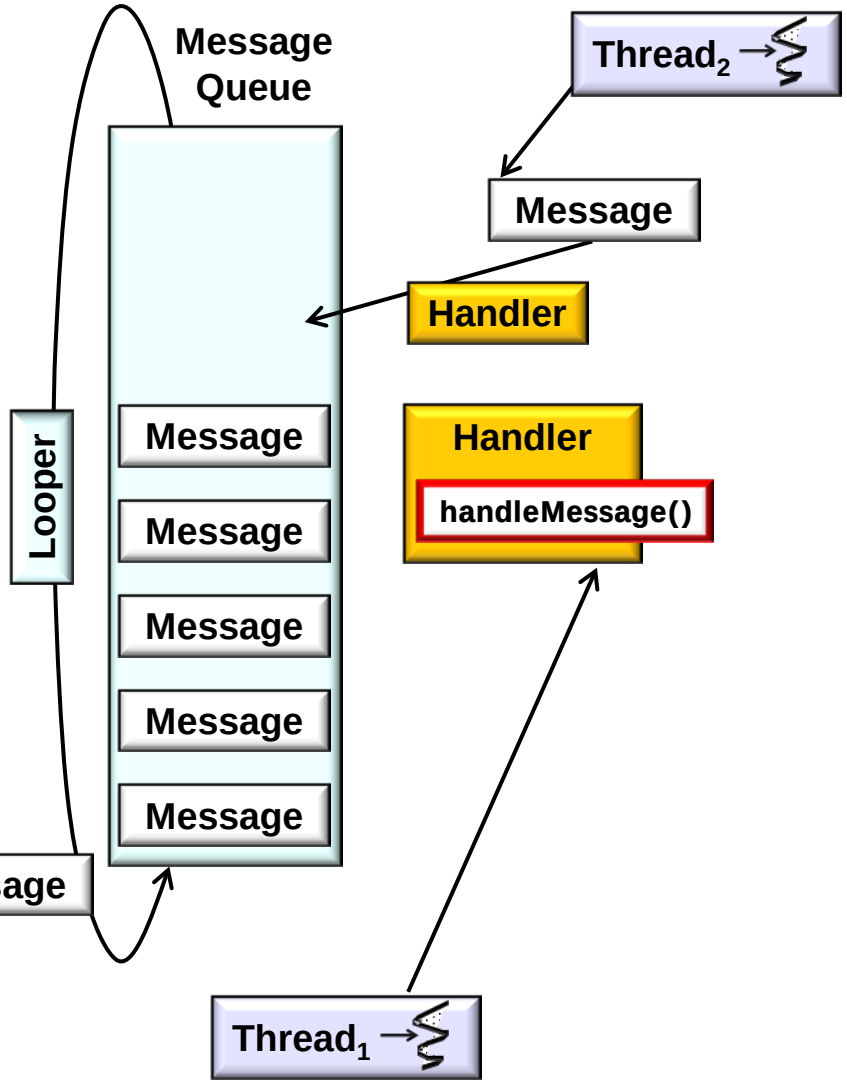
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
  2. Collaborates with Looper to serialize the processing of Messages in a Thread
     - Each Message (& Runnable) tracks its associated Handler




```
msg.target.
dispatchMessage(msg)
```

Overview of the Handler Class

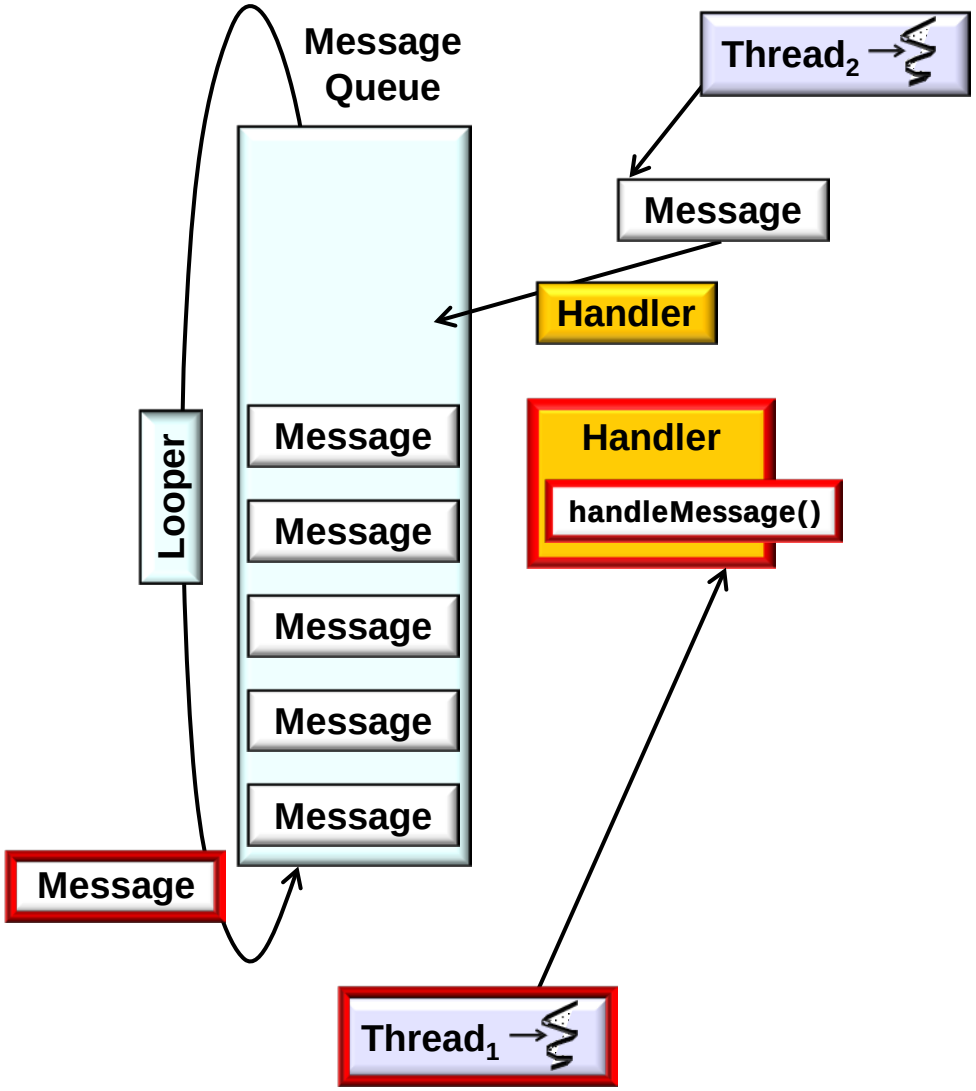
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
  2. Collaborates with Looper to serialize the processing of Messages in a Thread
     - Each Message (& Runnable) tracks its associated Handler




```
msg.target.
dispatchMessage(msg)
```

Overview of the Handler Class

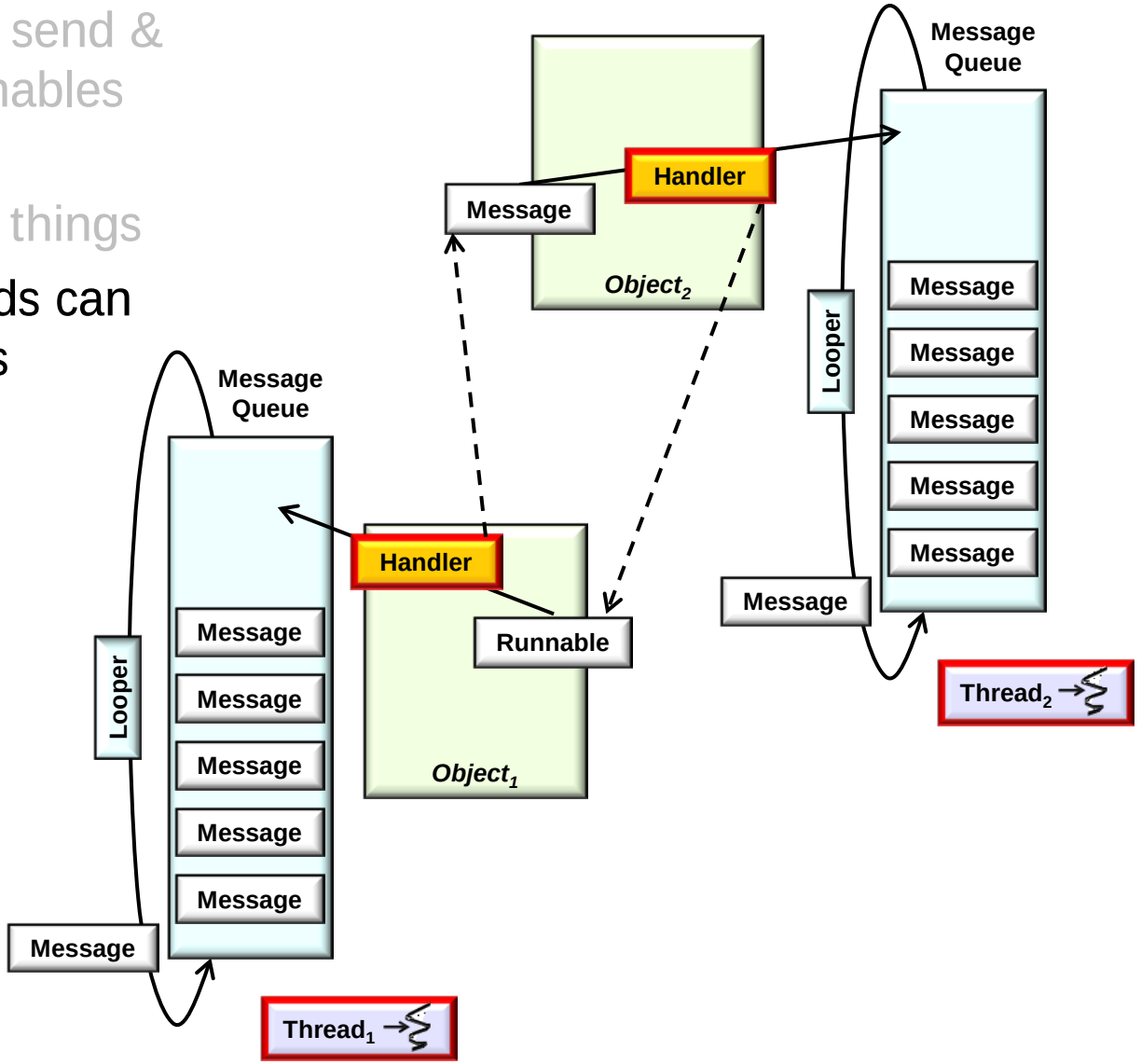
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
  1. Sends Messages & posts Runnables to Thread's Looper
  2. Collaborates with Looper to serialize the processing of Messages in a Thread
     - Each Message (& Runnable) tracks its associated Handler
     - Can simplify concurrency control if design rules & idioms are followed



See developer.android.com/training/multiple-threads/communicate-ui.html

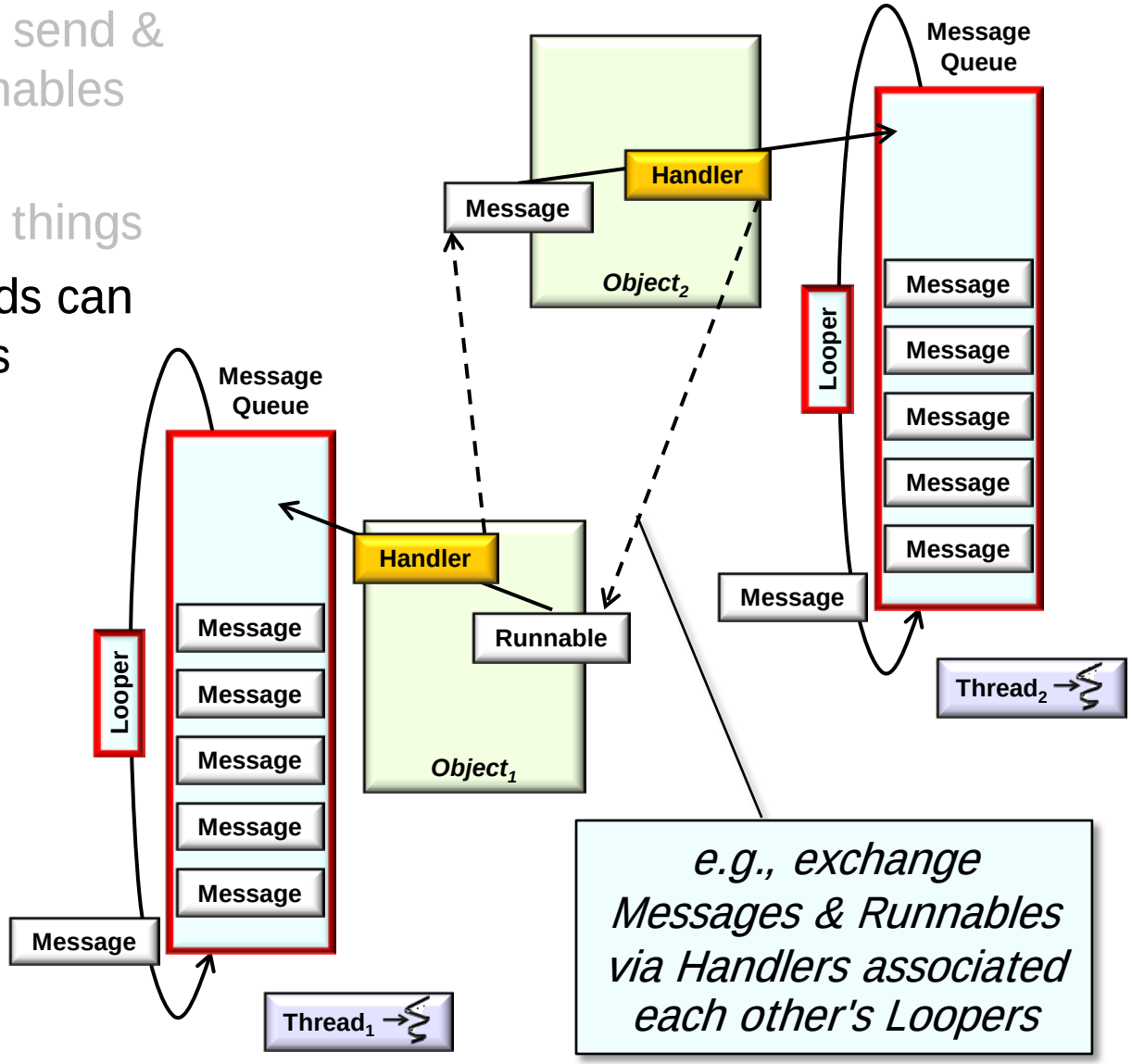
Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
- Objects in different threads can interact via their Handlers



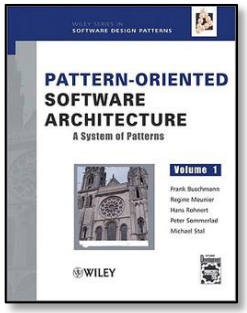
Overview of the Handler Class

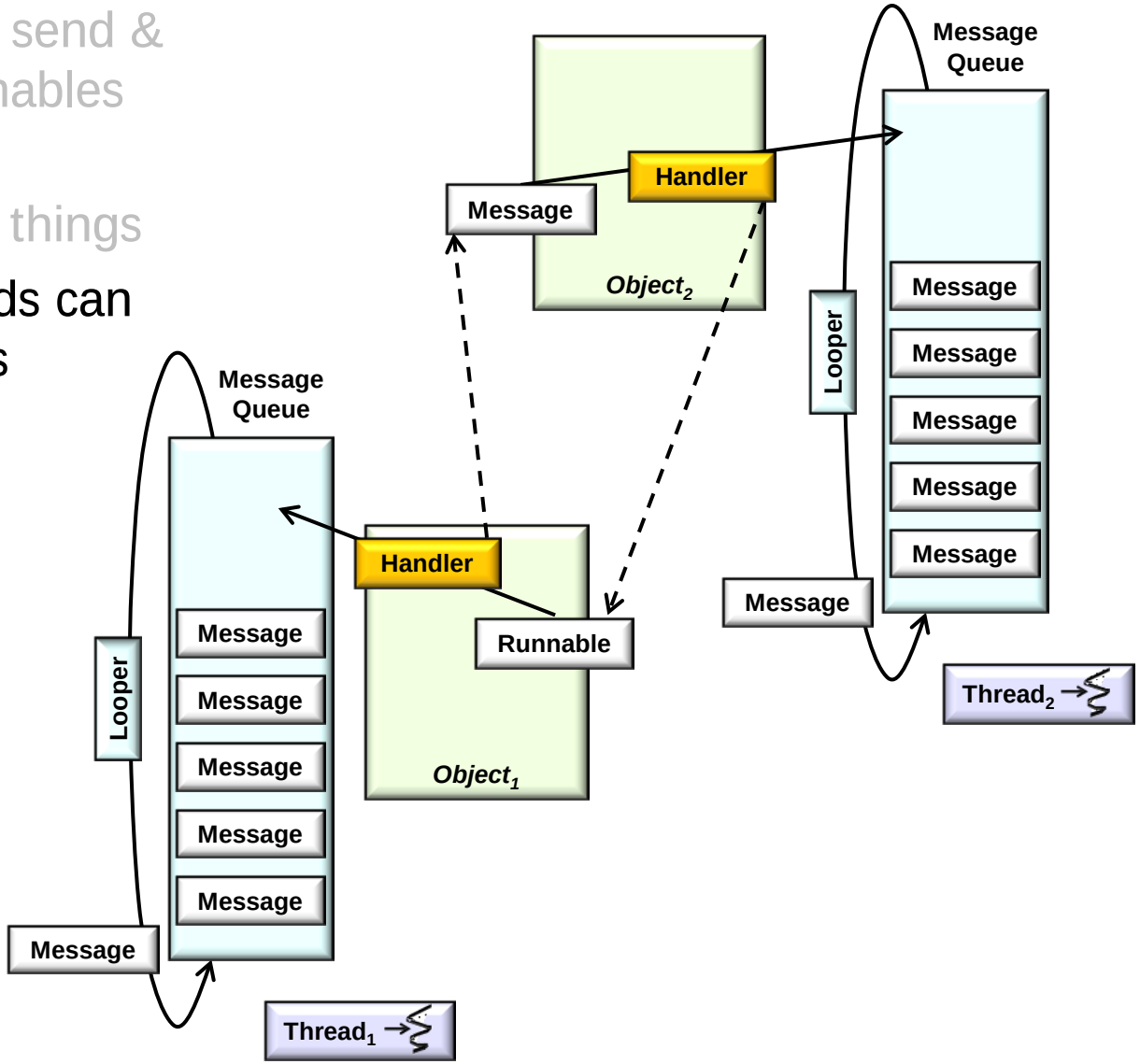
- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
- Objects in different threads can interact via their Handlers



Overview of the Handler Class

- A Handler can be used to send & process Messages & Runnables in one or more threads
- A Handler does two main things
- Objects in different threads can interact via their Handlers
  - Android's concurrency frameworks implement key patterns





See upcoming discussions on "The *Command Processor* Pattern" & "The *Active Object* Pattern"

Categories of Methods in the Handler Class

Categories of Methods in the Handler Class

- Handler has many methods

```
public class Handler {
  public Handler()
  public Handler(Callback callback)
  public Handler(Looper looper)
  public void handleMessage
                    (Message msg)
  public void dispatchMessage
                    (Message msg)
  public Message obtainMessage()
  public boolean post(Runnable r)
  public boolean postDelayed
      (Runnable r, long delayMillis)
  public boolean sendMessage
                    (Message msg)
  public boolean sendMessageDelayed
    (Message msg, long delayMillis)
  ...
```

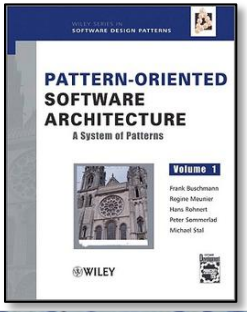
Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories

```
public class Handler {
   public Handler()
   public Handler(Callback callback)
   public Handler(Looper looper)
   public void handleMessage
```

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories



See upcoming discussions on "The *Command Processor* Pattern" & "The *Active Object* Pattern"

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables

`boolean post(Runnable r)`

- Add Runnable to end of MessageQueue & run when it's ready

`boolean post(Runnable r, long delayMillis)`

- Add Runnable to end of MessageQueue, to be run after specified amount of time elapses

`boolean postAtFrontOfQueue (Runnable r)`

- Posts a Runnable to the front of MessageQueue & run when it's ready

`void removeCallbacks(Runnable r)`

- Remove any pending posts of Runnable r that are in the MessageQueue

…

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
    - Insert/delete a Runnable into/from MessageQueue associated with the Handler

`boolean post(Runnable r)`

- Add Runnable to end of MessageQueue & run when it's ready

`boolean post(Runnable r, long delayMillis)`

- Add Runnable to end of MessageQueue, to be run after specified amount of time elapses

`boolean postAtFrontOfQueue (Runnable r)`

- Posts a Runnable to the front of MessageQueue & run when it's ready

`void removeCallbacks(Runnable r)`

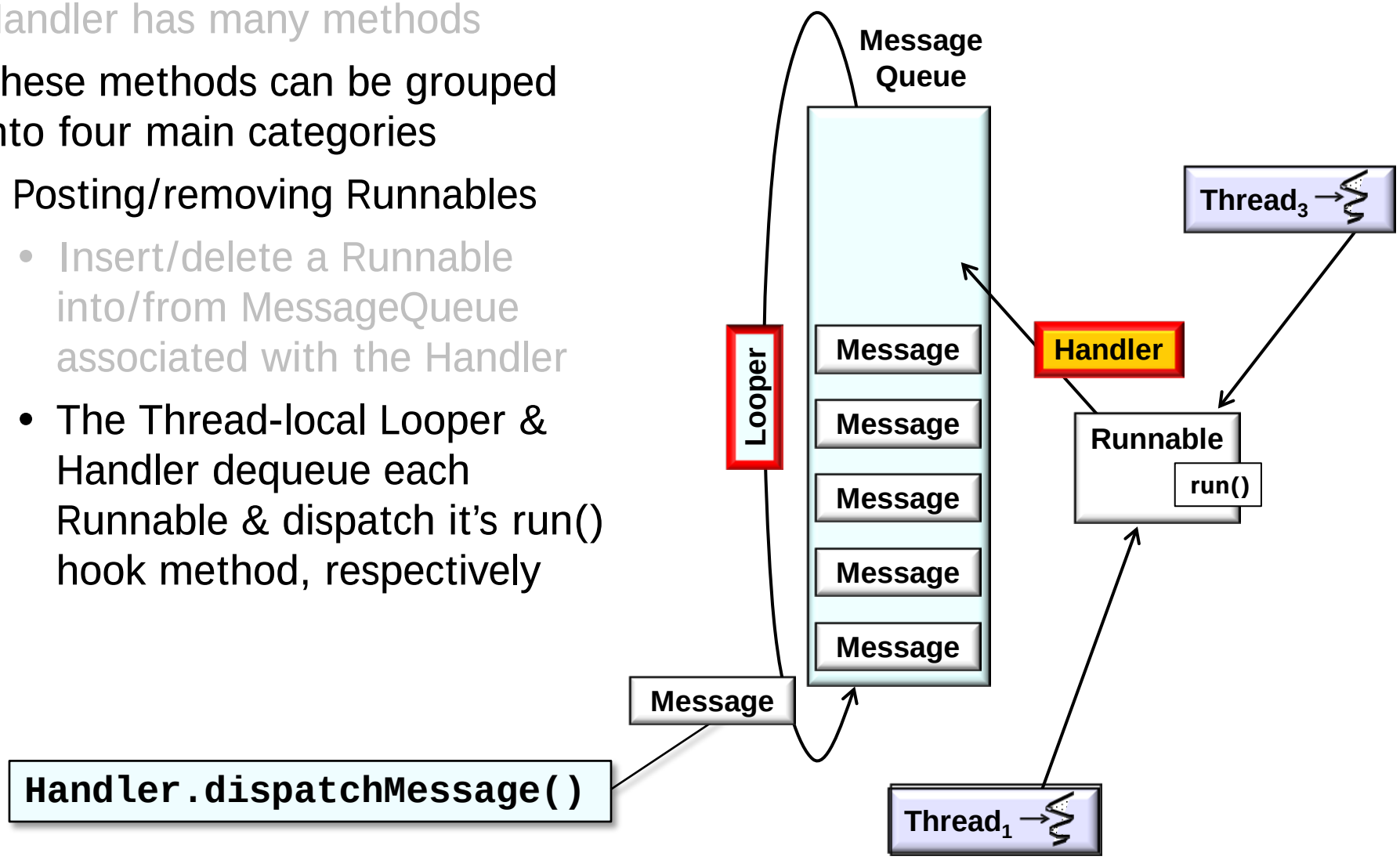
- Remove any pending posts of Runnable r that are in the MessageQueue

...

Supports timed Runnables

Categories of Methods in the Handler Class

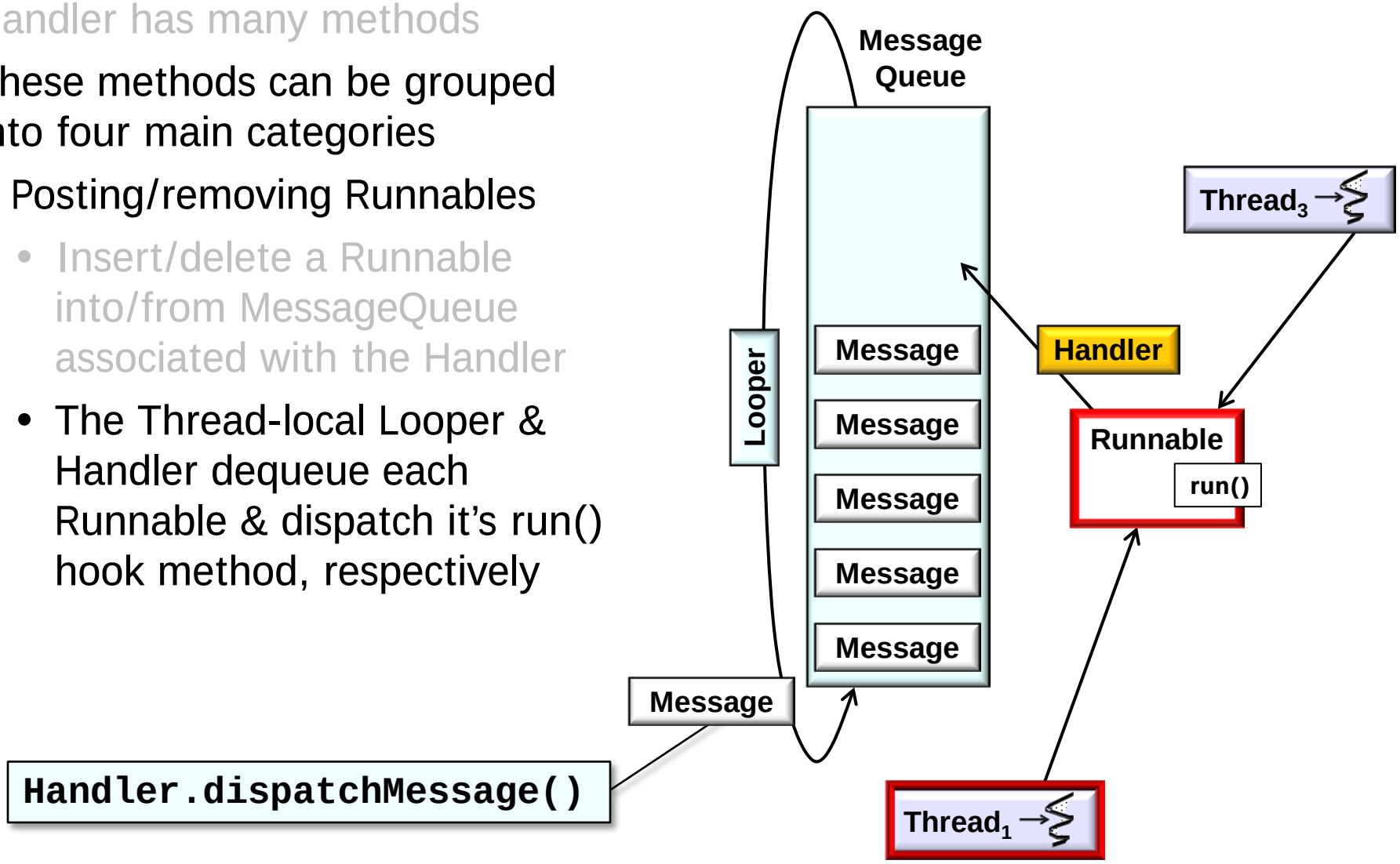
- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
    - Insert/delete a Runnable into/from MessageQueue associated with the Handler
    - The Thread-local Looper & Handler dequeue each Runnable & dispatch it's run() hook method, respectively



See upcoming discussions on the "*Command Processor* pattern"

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
    - Insert/delete a Runnable into/from MessageQueue associated with the Handler
    - The Thread-local Looper & Handler dequeue each Runnable & dispatch it's run() hook method, respectively



Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages

`boolean sendMessage(Message msg)`

- Puts msg at end of MessageQueue immediately

`boolean sendMessageDelayed (Message msg)`

- Put msg into the MessageQueue after all pending Messages before (current time + delayMillis)

`boolean sendMessageAtFrontOfQueue (Message msg)`

- Put msg at the front of the MessageQueue

`void removeMessages(int what)`

- Remove any pending posts of Messages with code ‘what’ that are in the MessageQueue

…

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
    - Insert/delete a Message into/from MessageQueue associated with the Handler

`boolean sendMessage(Message msg)`

- Puts msg at end of MessageQueue immediately

`boolean sendMessageDelayed (Message msg)`

- Put msg into the MessageQueue after all pending Messages before (current time + delayMillis)

`boolean sendMessageAtFrontOfQueue (Message msg)`

- Put msg at the front of the MessageQueue

`void removeMessages(int what)`

- Remove any pending posts of Messages with code ‘what’ that are in the MessageQueue

...

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
    - Insert/delete a Message into/from MessageQueue associated with the Handler

`boolean sendMessage(Message msg)`

- Puts msg at end of MessageQueue immediately

`boolean sendMessageDelayed (Message msg)`

- Put msg into the MessageQueue after all pending Messages before (current time + delayMillis)

`boolean sendMessageAtFrontOfQueue (Message msg)`

- Put msg at the front of the MessageQueue

`void removeMessages(int what)`

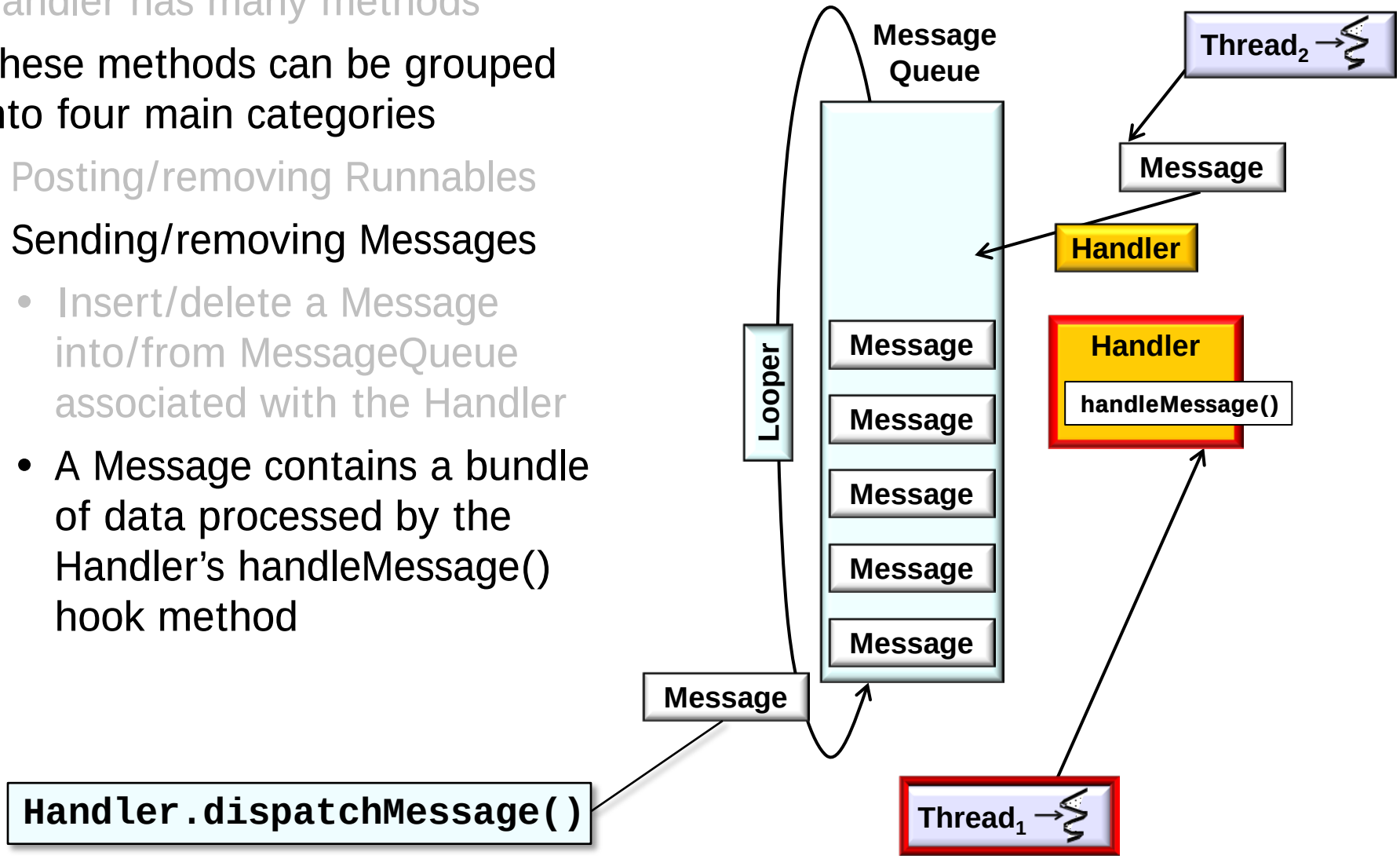
- Remove any pending posts of Messages with code ‘what’ that are in the MessageQueue

...

Supports timed Messages

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
    - Insert/delete a Message into/from MessageQueue associated with the Handler
    - A Message contains a bundle of data processed by the Handler's handleMessage() hook method



See upcoming discussions on the "*Active Object* pattern"

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
  - Obtaining Messages

`Message obtainMessage()`

- Returns a new Message from the global message pool

`Message obtainMessage (int what)`

- Same as obtainMessage(), except that it also sets the what member of the returned Message

`Message obtainMessage (int what, int arg1, int arg2, Object obj)`

- Same as obtainMessage(), except that it also sets the what, obj, arg1, & arg2 values on the returned Message

...

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
  - Obtaining Messages
    - Factories that create Messages passed via sendMessage()

`Message obtainMessage()`

- Returns a new Message from the global message pool

`Message obtainMessage (int what)`

- Same as obtainMessage(), except that it also sets the what member of the returned Message

`Message obtainMessage (int what, int arg1, int arg2, Object obj)`

- Same as obtainMessage(), except that it also sets the what, obj, arg1, & arg2 values on the returned Message

...

See en.wikipedia.org/wiki/Creational_pattern

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
  - Obtaining Messages
  - Dispatching Messages/Runnables

`void handleMessage(Message msg)`

- Subclasses can implement this method to receive messages

`void dispatchMessage(Message msg)`

- Invoke the appropriate callback (e.g., run() or handleMessage()) based on the type of the Message

Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
  - Obtaining Messages
  - Dispatching Messages/Runnables
    - dispatchMessage() is used for both Messages & Runnables

`void handleMessage(Message msg)`

- Subclasses can implement this method to receive messages

`void dispatchMessage(Message msg)`

- Invoke the appropriate callback (e.g., run() or handleMessage()) based on the type of the Message

```
if (msg.callback != null)
   handleCallback(msg);
 else {
   if (mCallback != null) {
     if (mCallback.
         handleMessage(msg))
       return;
   }
   handleMessage(msg);
 }
```

See discussions in upcoming parts on HaMeR framework internals

Categories of Methods in the Handler Class

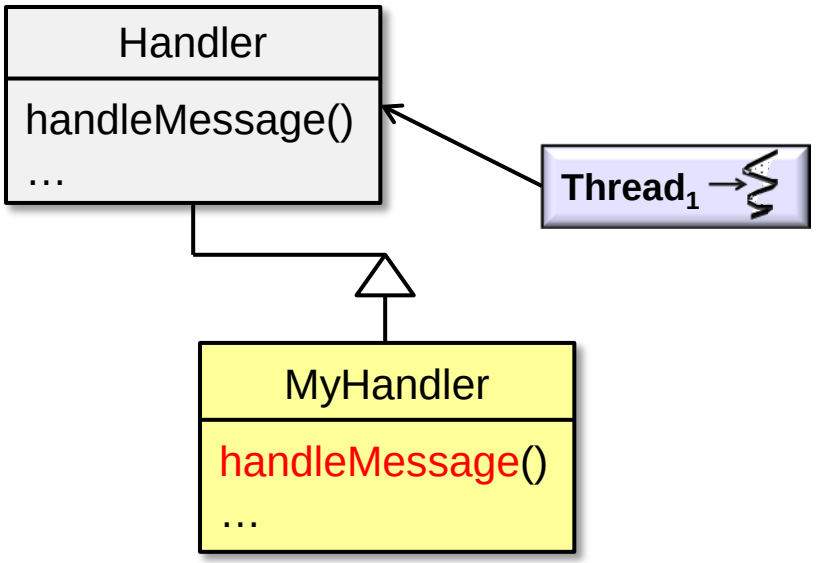
- Handler has many methods
- These methods can be grouped into four main categories
  - Posting/removing Runnables
  - Sending/removing Messages
  - Obtaining Messages
  - Dispatching Messages/Runnables
    - dispatchMessage() is used for both Messages & Runnables
    - Messages & Runnables are processed in context of Thread associated with Handler

`void handleMessage(Message msg)`

- Subclasses can implement this method to receive messages

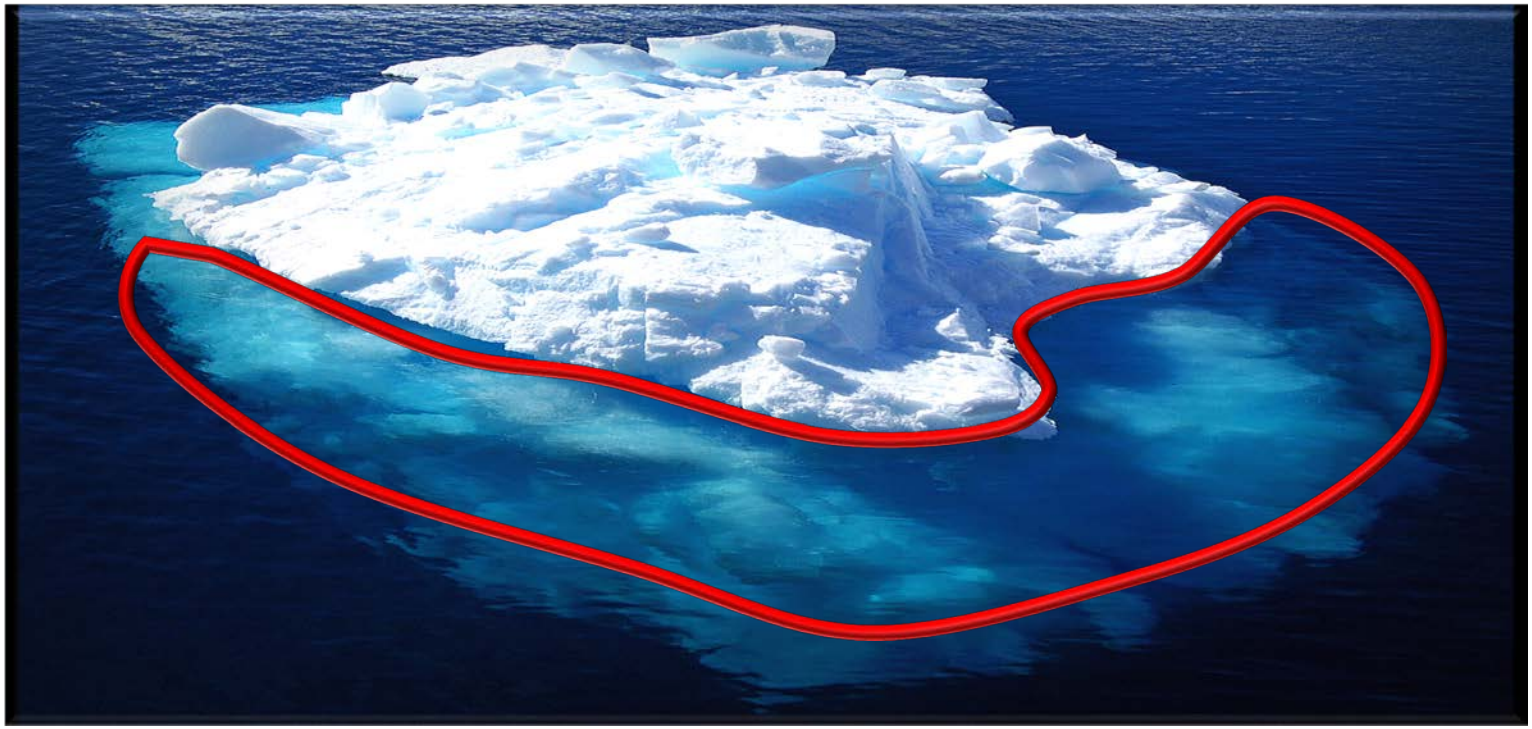
`void dispatchMessage(Message msg)`

- Invoke the appropriate callback (e.g., run() or handleMessage()) based on the type of the Message



Categories of Methods in the Handler Class

- Handler has many methods
- These methods can be grouped into four main categories

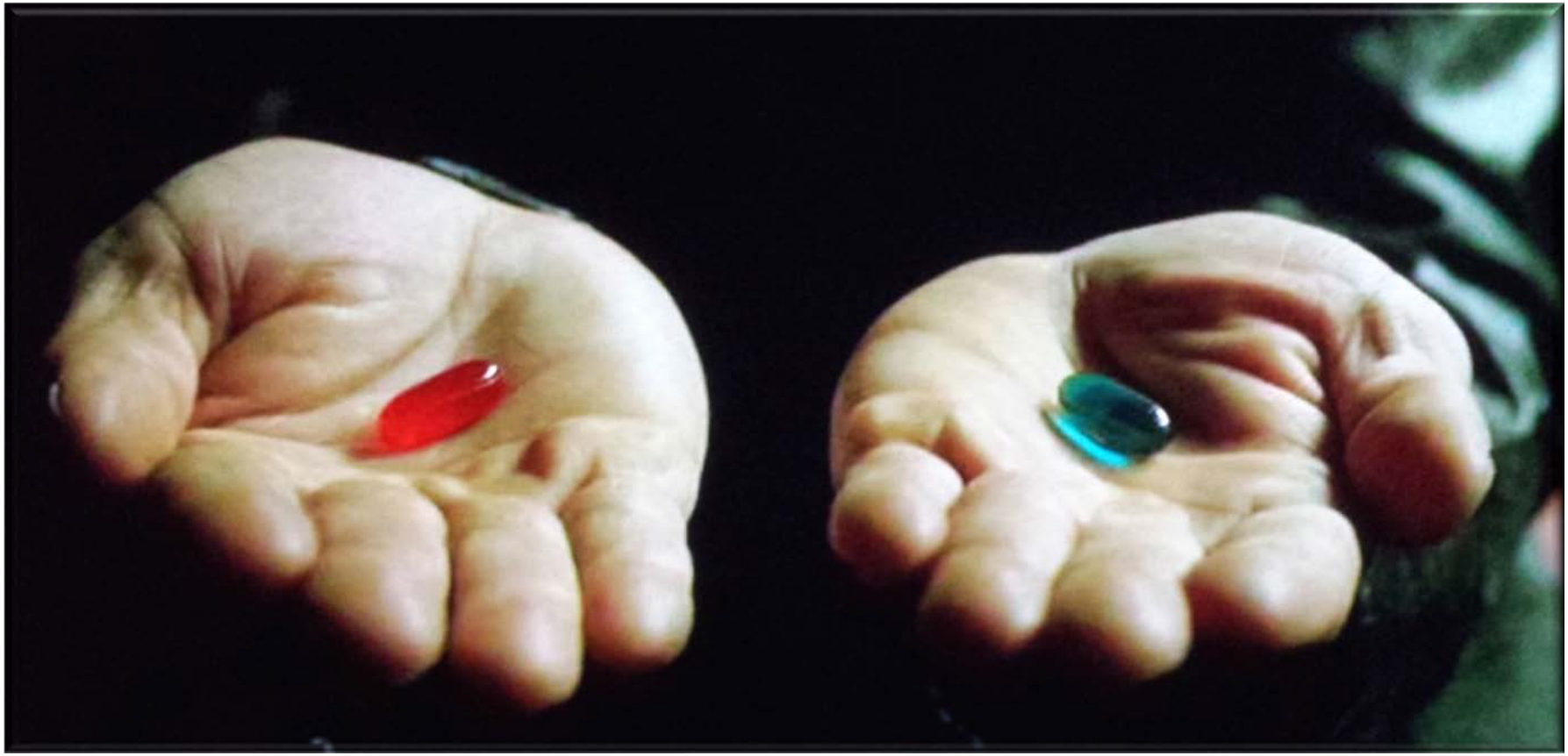


These Handler methods are covered in more detail later

Handler Usage Considerations

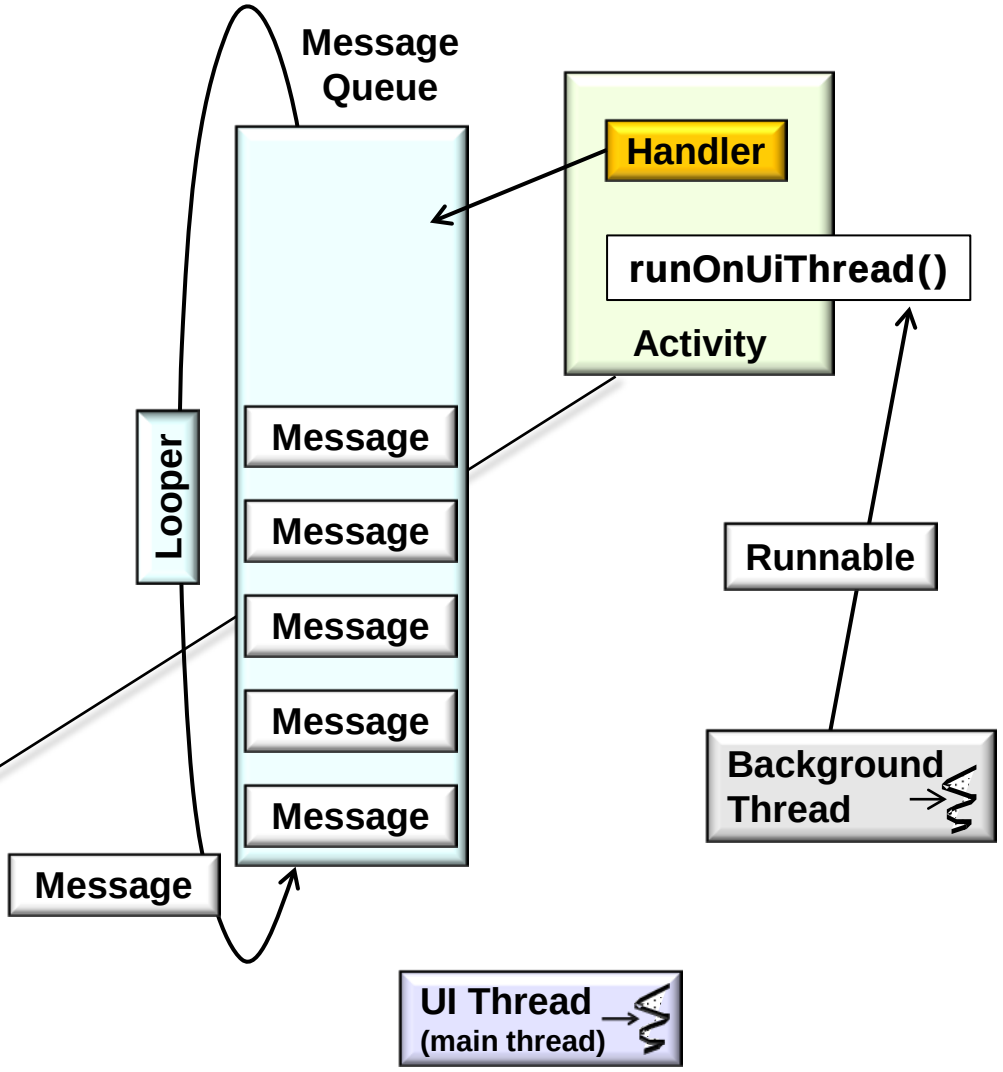
Handler Usage Considerations

- Selecting between posting Runnables vs sending Messages



Handler Usage Considerations

- Selecting between posting Runnables vs sending Messages
  - post() methods are used when *senders* know what operations to perform

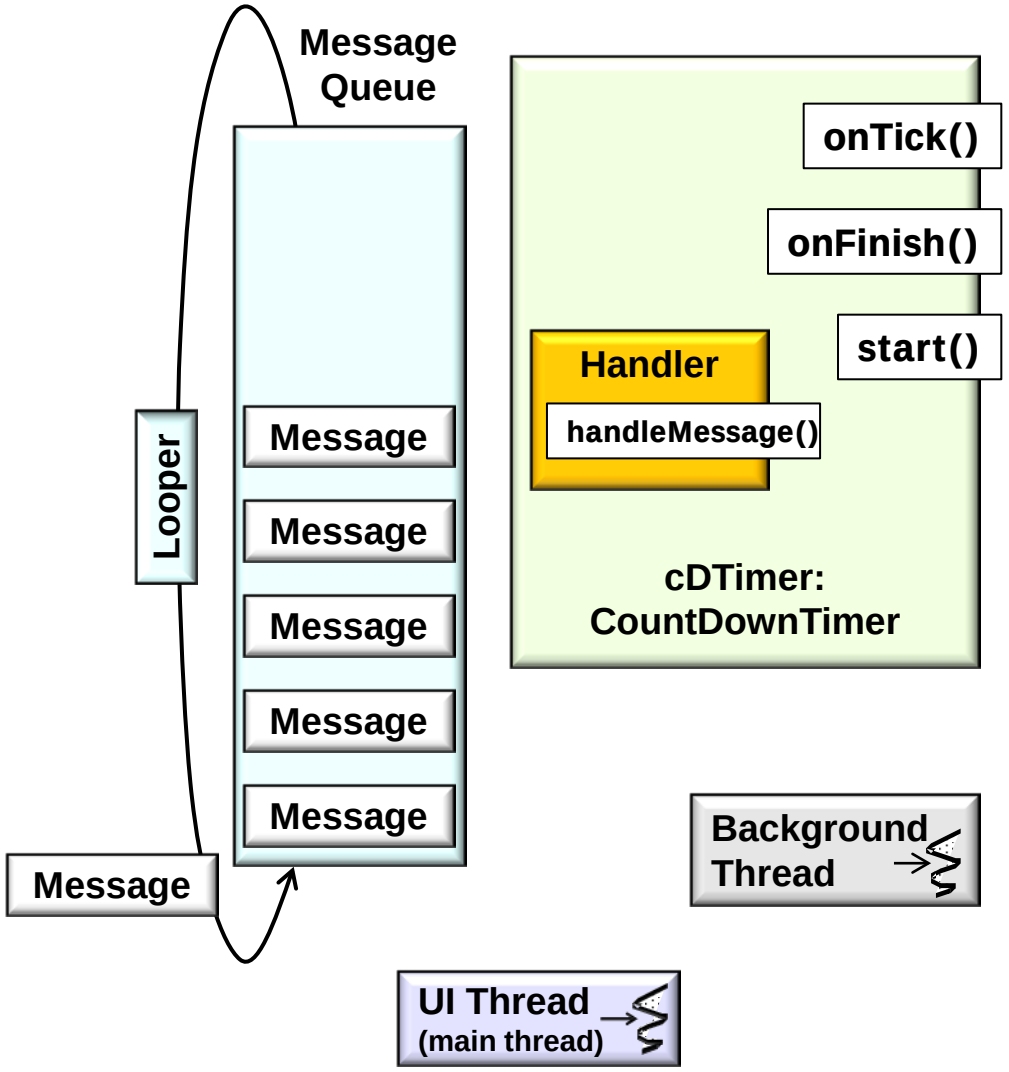



```
runOnUiThread
  (new Runnable() {
     public void run()
       { ... }
  });
```

See upcoming parts on “Posting & Processing Runnables with the Android Handler & HaMeR Framework”

Handler Usage Considerations

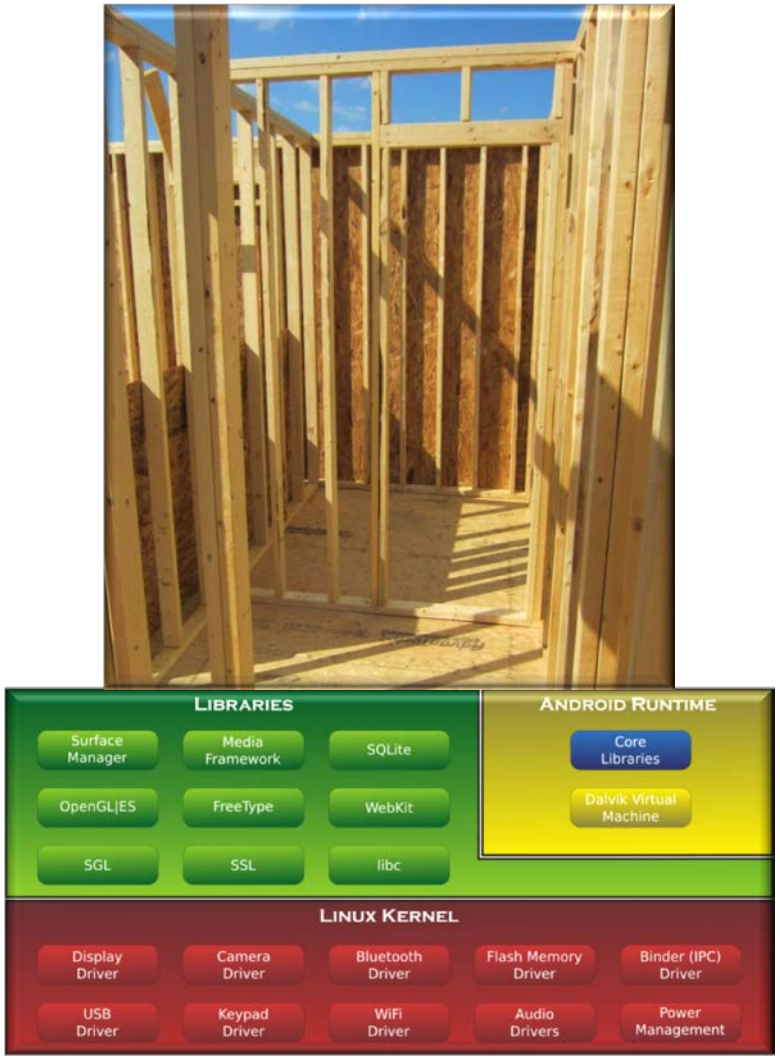
- Selecting between posting Runnables vs sending Messages
  - post() methods are used when *senders* know what operations to perform
  - sendMessage() methods are used when *receivers* know what operations to perform



See upcoming parts on “Sending & Handling Messages with the Android Handler & HaMeR Framework”

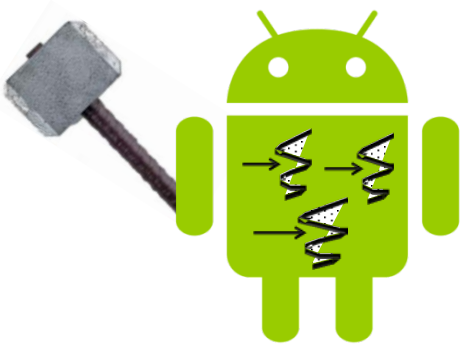
Handler Usage Considerations

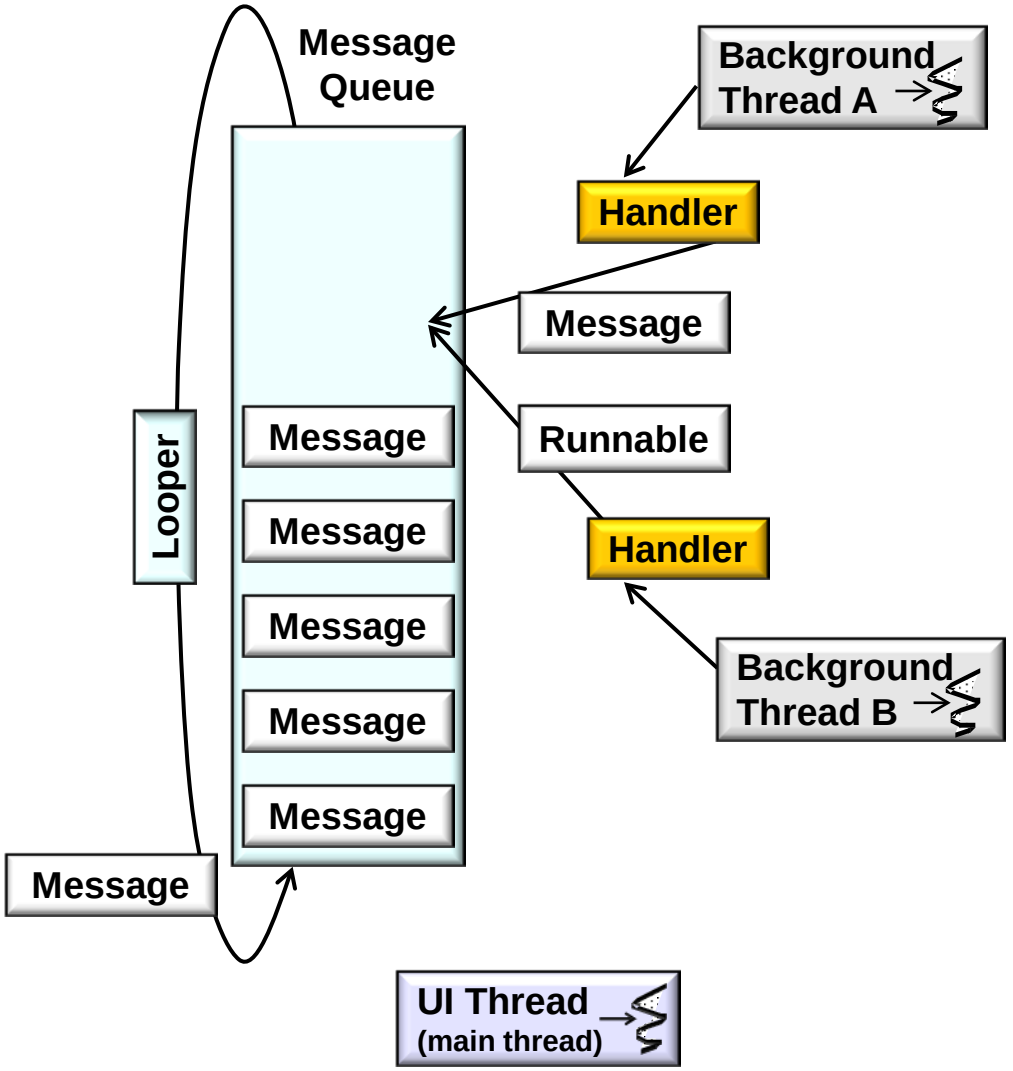
- Selecting between posting Runnables vs sending Messages
- Handlers used in many Android applications & frameworks



Handler Usage Considerations

- Selecting between posting Runnables vs sending Messages
- Handlers used in many Android applications & frameworks, e.g.
  - HaMeR framework

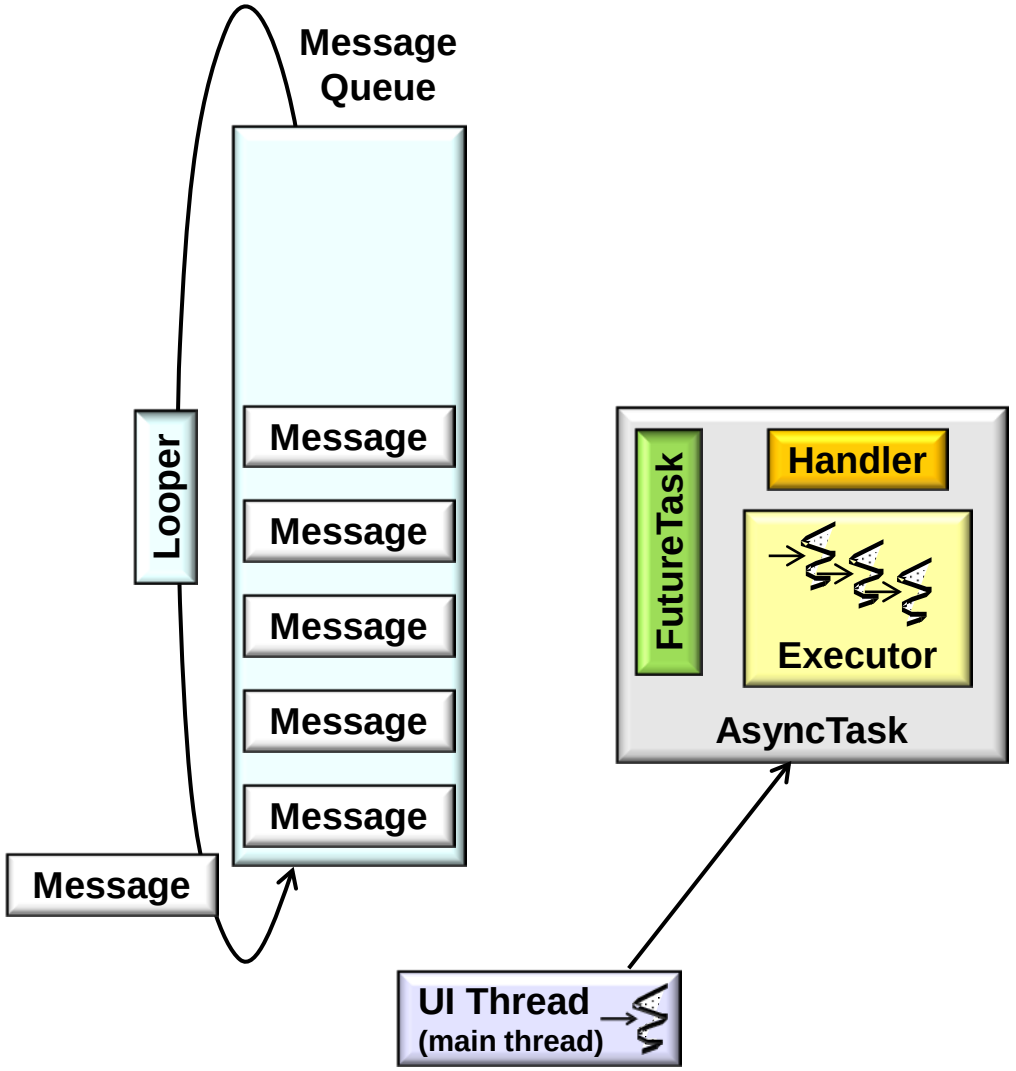




Next two parts focus on the HaMeR framework's use of Handlers et al

Handler Usage Considerations

- Selecting between posting Runnables vs sending Messages
- Handlers used in many Android applications & frameworks, e.g.
  - HaMeR framework
  - AsyncTask framework



See upcoming part on "the AsyncTask Framework"

Handler Usage Considerations

- Selecting between posting Runnables vs sending Messages
- Handlers used in many Android applications & frameworks
- Other resources explain more about how to use Handlers in Android's concurrency frameworks, services, & applications

See www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html