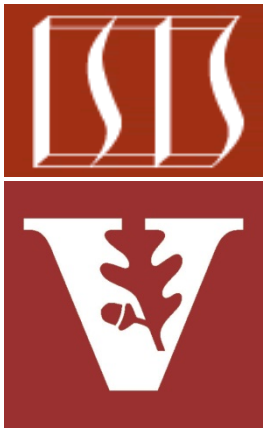
Android Concurrency: Posting & Processing Runnables with Android Handler

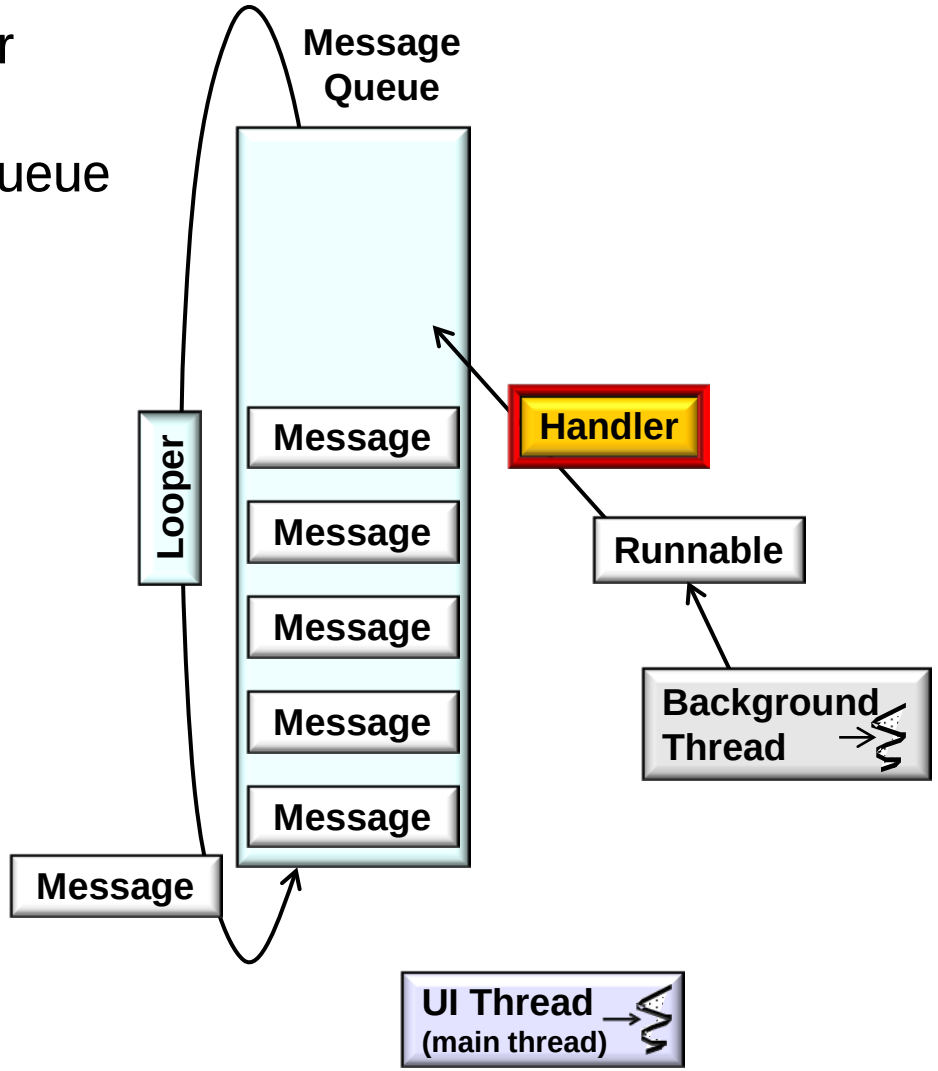


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

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

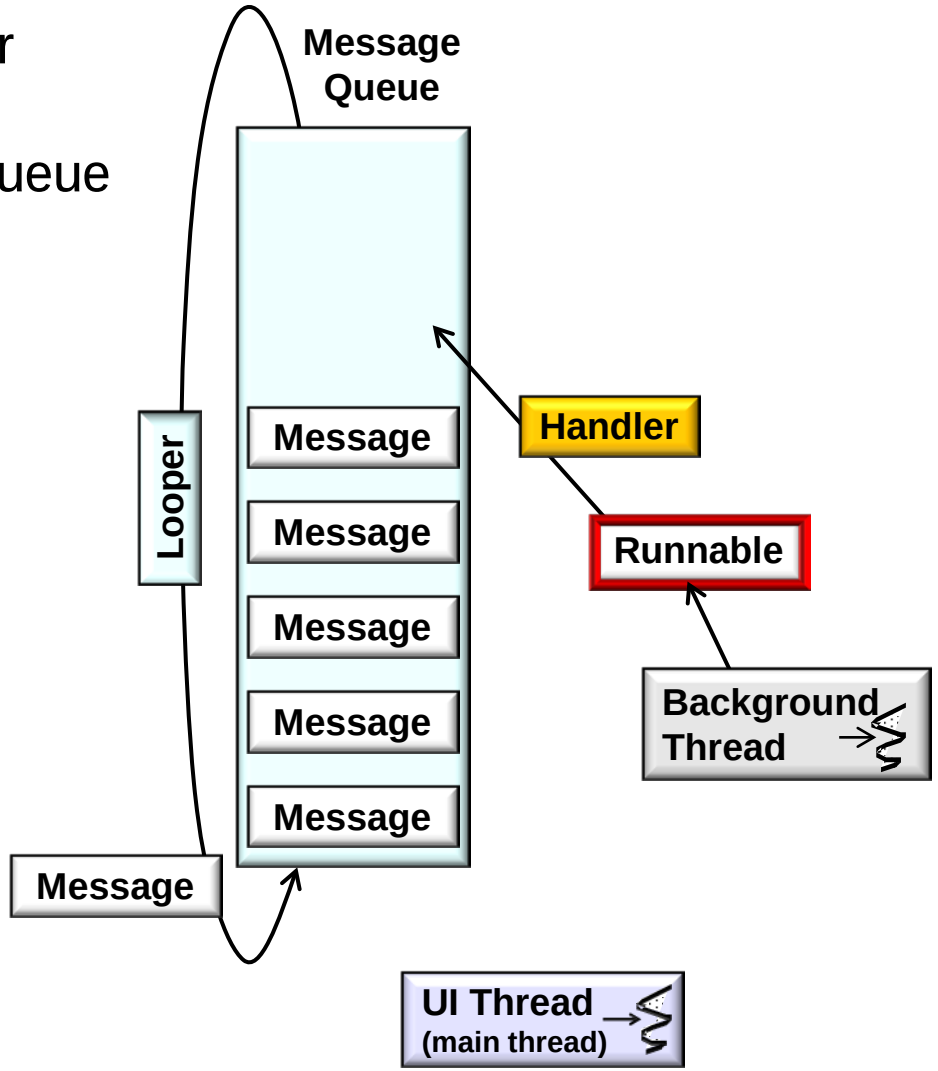
Learning Objectives in this Part of the Module

- Understand how an Android Handler enables the posting & processing of Runnable objects via the MessageQueue associated with a Thread's Looper



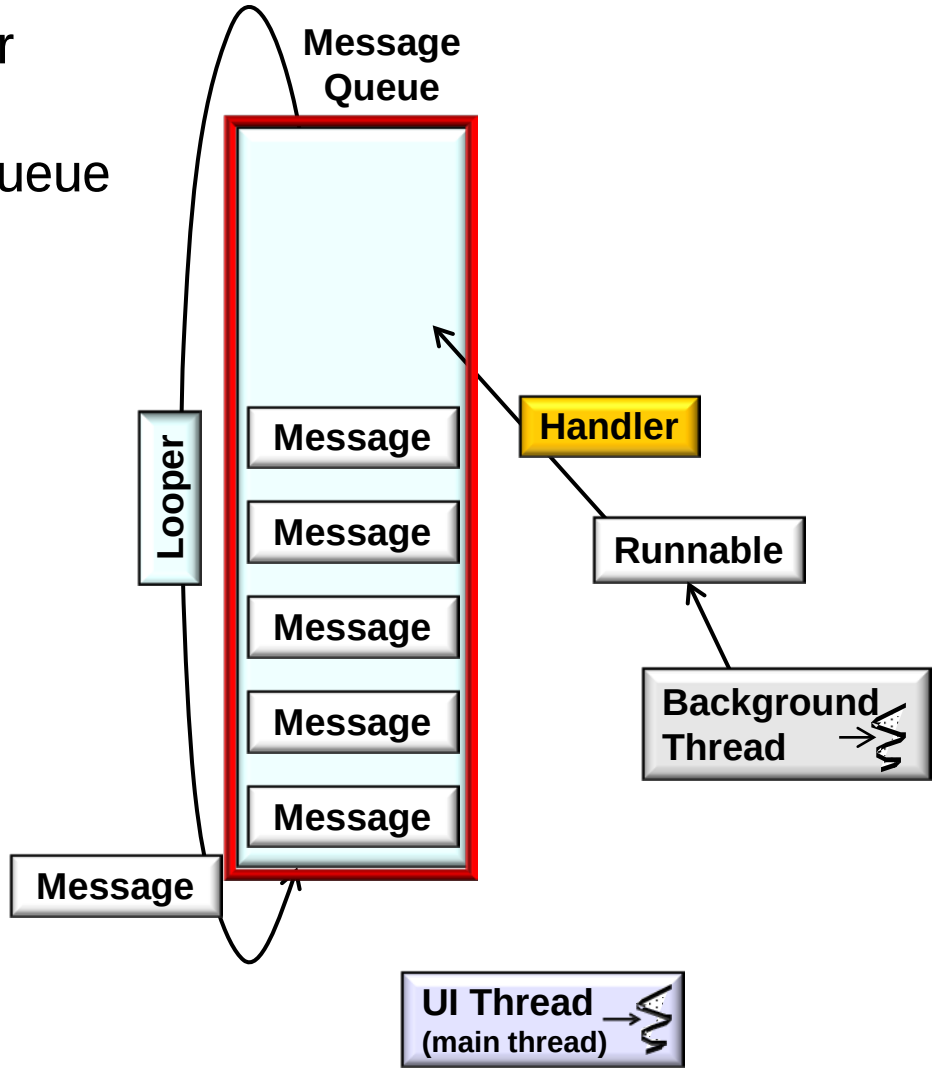
Learning Objectives in this Part of the Module

- Understand how an Android Handler enables the posting & processing of Runnable objects via the MessageQueue associated with a Thread's Looper



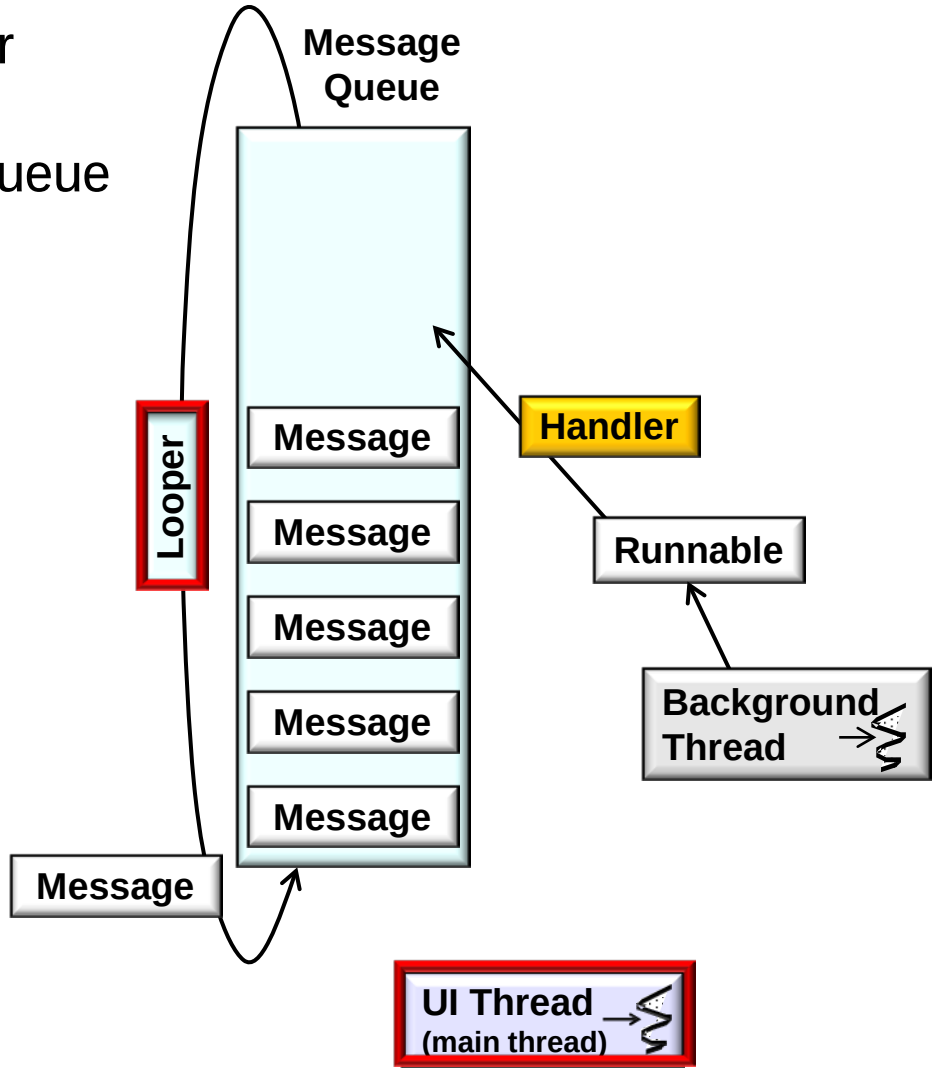
Learning Objectives in this Part of the Module

- Understand how an Android Handler enables the posting & processing of Runnable objects via the MessageQueue associated with a Thread's Looper



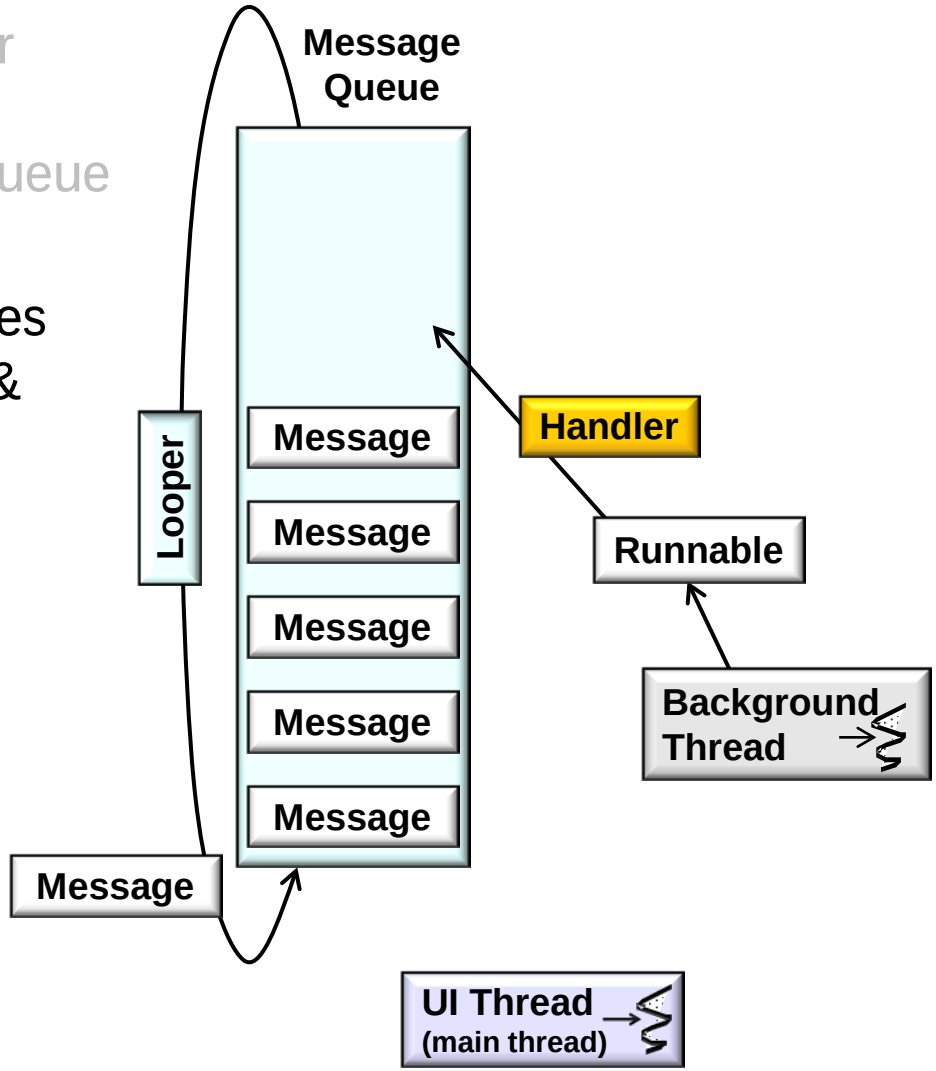
Learning Objectives in this Part of the Module

- Understand how an Android Handler enables the posting & processing of Runnable objects via the MessageQueue associated with a Thread's Looper



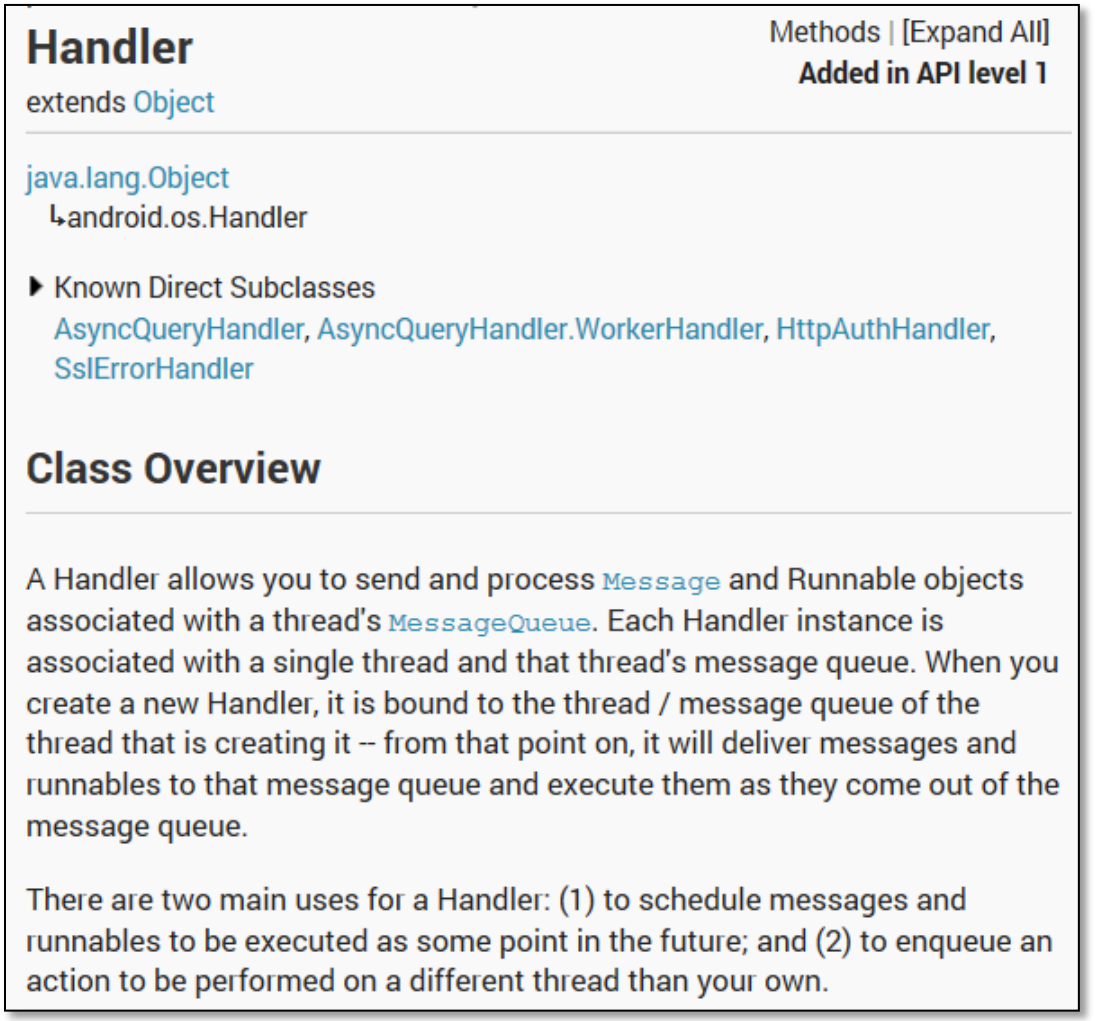
Learning Objectives in this Part of the Module

- Understand how an Android Handler enables the posting & processing of Runnable objects via the MessageQueue associated with a Thread's Looper
- Recognize how Handlers & Runnables are applied in Android applications & its HaMeR concurrency framework



Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler



Handler

Methods | [Expand All]
Added in API level 1

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.

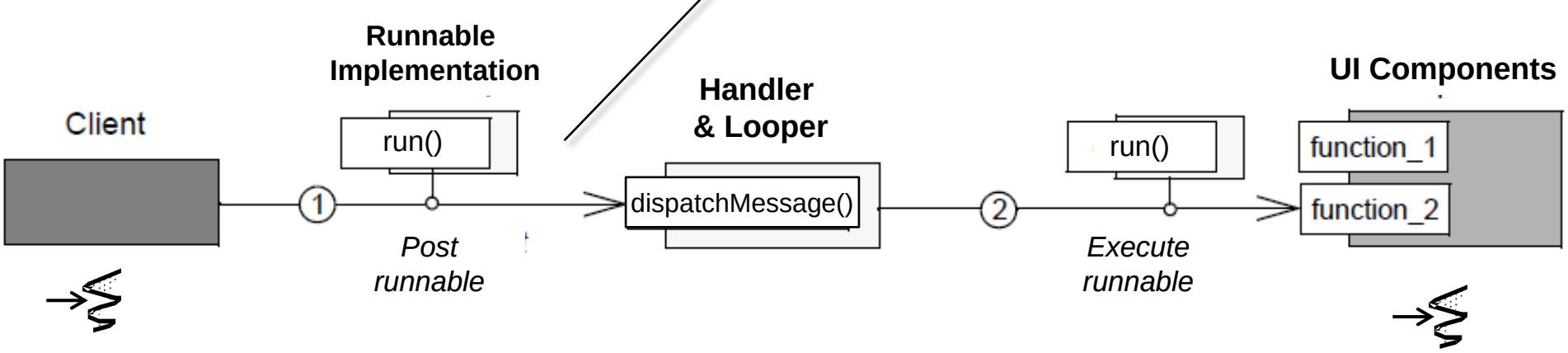
There are two main uses for a Handler: (1) to schedule messages and runnables to be executed as some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.

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

Handler Methods that Post Runnables

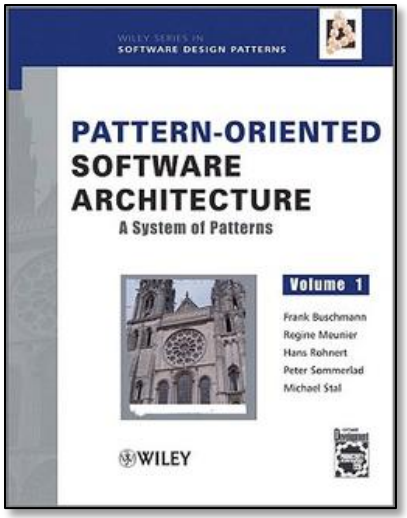
- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
  - Each Runnable is dequeued & its run() hook method is dispatched

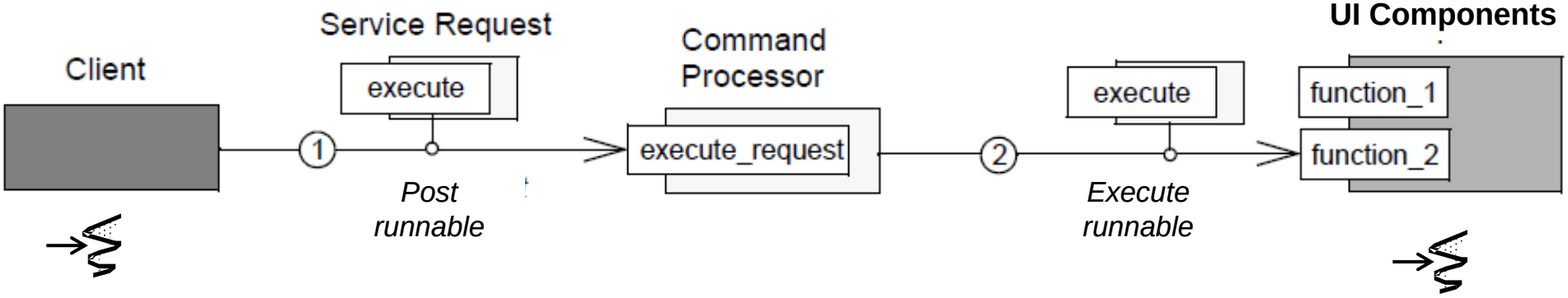
```
mHandler.post
  (new Runnable() {
     public void run()
       { ... }
  });
```



Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
  - Each Runnable is dequeued & its run() hook method is dispatched
  - Implements the *Command Processor* pattern





www.dre.vanderbilt.edu/~schmidt/PDF/CommandRevisited.pdf has more info

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()

`boolean post(Runnable r)`

- Add Runnable to rear of MessageQueue & run when MessageQueue is ready

`boolean postAtFrontOfQueue (Runnable r)`

- Add Runnable to front of MessageQueue & run when MessageQueue is ready

`boolean postDelayed(Runnable r, long delayMillis)`

- Add Runnable to MessageQueue & run after specified amount of time elapses

`boolean postAtTime(Runnable r, long uptimeMillis)`

- Add Runnable to MessageQueue & run at a specific time

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()
  - Process Runnable as soon as possible

`boolean post(Runnable r)`

- Add Runnable to rear of MessageQueue & run when MessageQueue is ready

`boolean postAtFrontOfQueue (Runnable r)`

- Add Runnable to front of MessageQueue & run when MessageQueue is ready

`boolean postDelayed(Runnable r, long delayMillis)`

- Add Runnable to MessageQueue & run after specified amount of time elapses

`boolean postAtTime(Runnable r, long uptimeMillis)`

- Add Runnable to MessageQueue & run at a specific time

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()
  - Process Runnable as soon as possible
  - Specify a delay using "relative time"

`boolean post(Runnable r)`

- Add Runnable to rear of MessageQueue & run when MessageQueue is ready

`boolean postAtFrontOfQueue (Runnable r)`

- Add Runnable to front of MessageQueue & run when MessageQueue is ready

`boolean postDelayed(Runnable r, long delayMillis)`

- Add Runnable to MessageQueue & run after specified amount of time elapses

`boolean postAtTime(Runnable r, long uptimeMillis)`

- Add Runnable to MessageQueue & run at a specific time

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()
  - Process Runnable as soon as possible
  - Specify a delay using "relative time"
  - Specific a delay using "absolute time"

`boolean post(Runnable r)`

- Add Runnable to rear of MessageQueue & run when MessageQueue is ready

`boolean postAtFrontOfQueue (Runnable r)`

- Add Runnable to front of MessageQueue & run when MessageQueue is ready

`boolean postDelayed(Runnable r, long delayMillis)`

- Add Runnable to MessageQueue & run after specified amount of time elapses

`boolean postAtTime(Runnable r, long uptimeMillis)`

- Add Runnable to MessageQueue & run at a specific time

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()
  - Process Runnable as soon as possible
  - Specify a delay using "relative time"
  - Specific a delay using "absolute time"

`boolean post(Runnable r)`

- Add Runnable to rear of MessageQueue & run when MessageQueue is ready

`boolean postAtFrontOfQueue (Runnable r)`

- Add Runnable to front of MessageQueue & run when MessageQueue is ready

`boolean postDelayed(Runnable r, long delayMillis)`

- Add Runnable to MessageQueue & run after specified amount of time elapses

`boolean postAtTime(Runnable r, long uptimeMillis)`

- Add Runnable to MessageQueue & run at a specific time

These methods support timing related behavior

Handler Methods that Post Runnables

- Handler defines methods for posting & removing Runnables from the MessageQueue associated with a Handler
- There are several variants of post()
- There are several variants of remove()

`void removeCallbacks(Runnable r)`

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

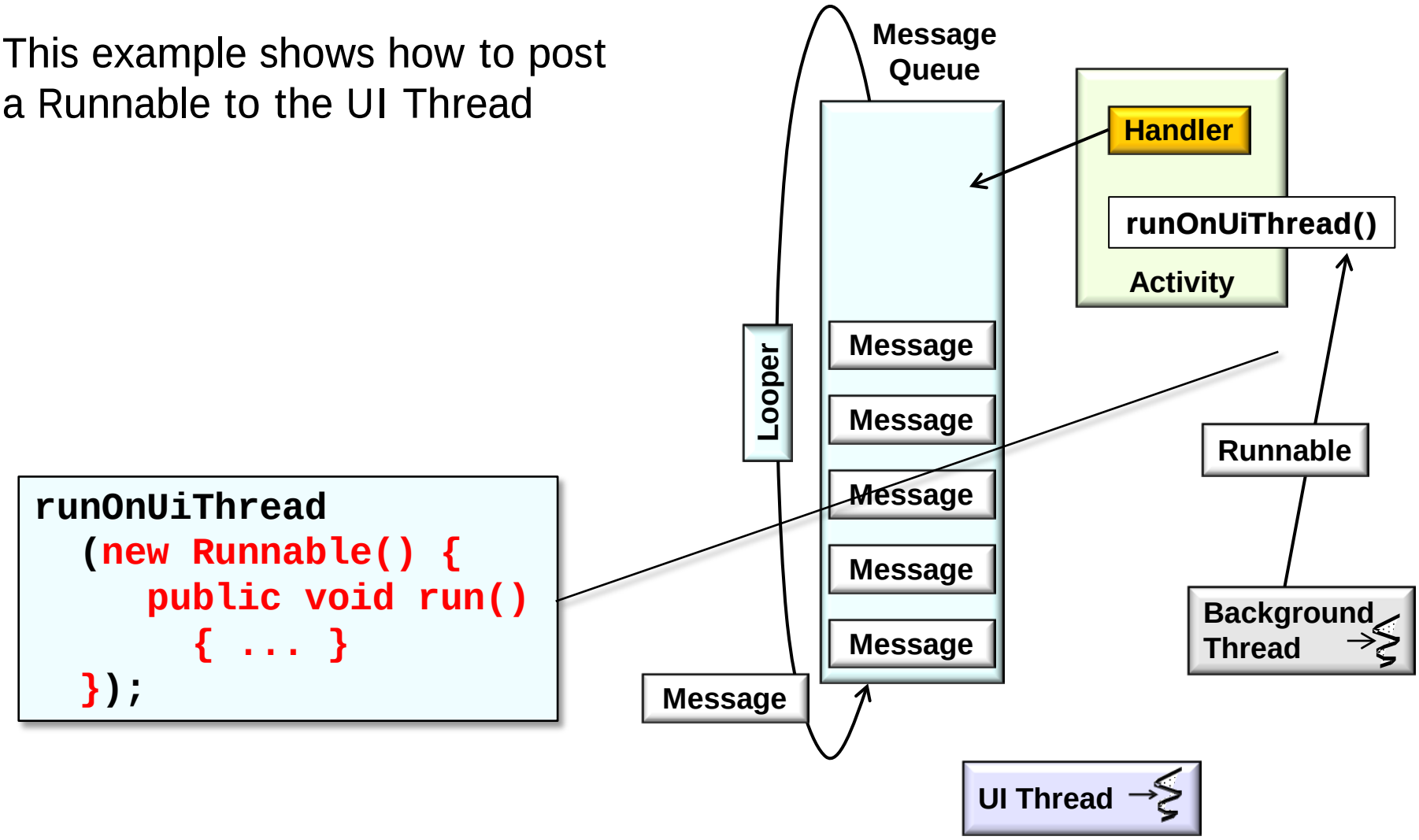
`void removeCallbacks(Runnable r, Object token)`

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

Posting Runnables to a Handler in the HaMeR Framework

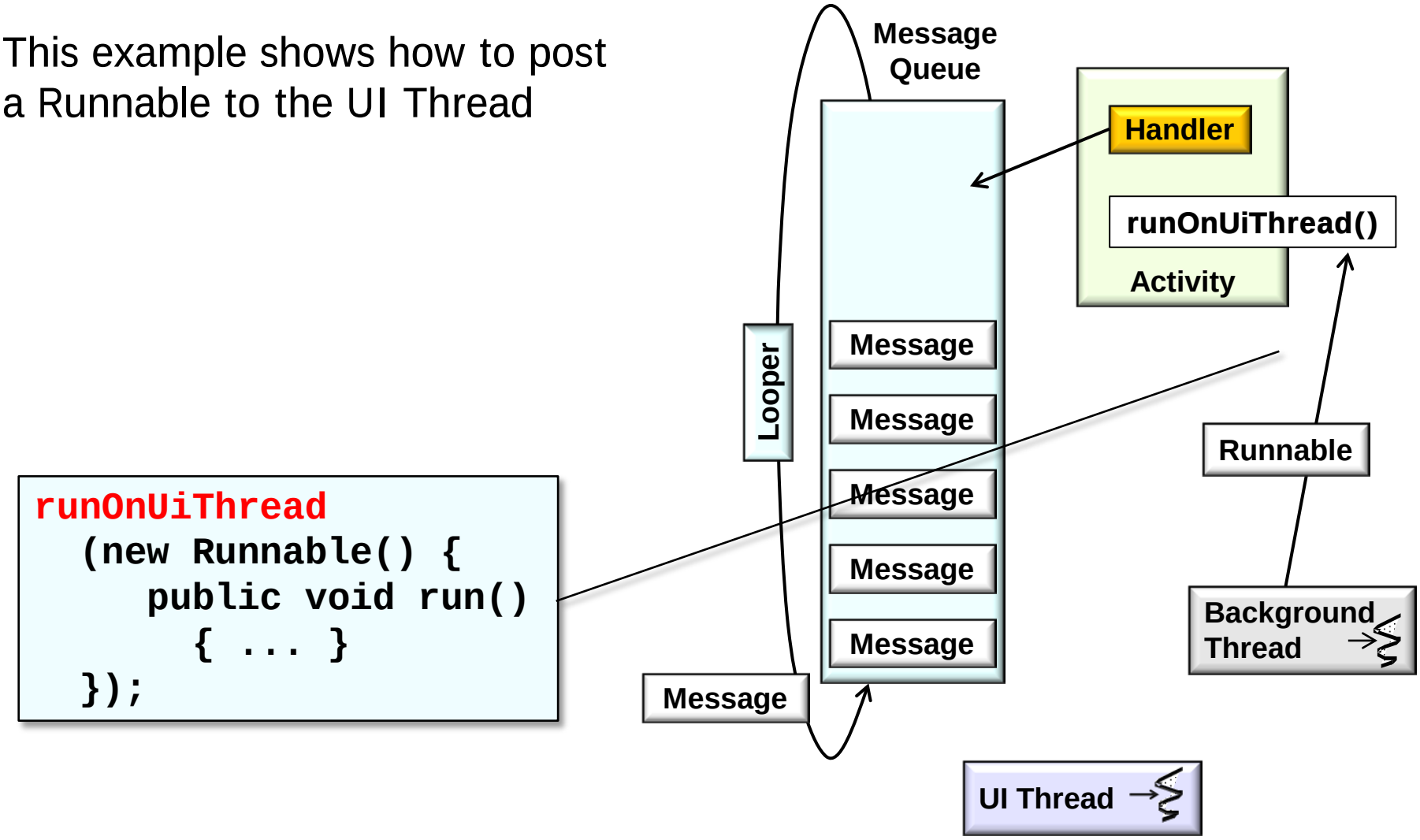
Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread



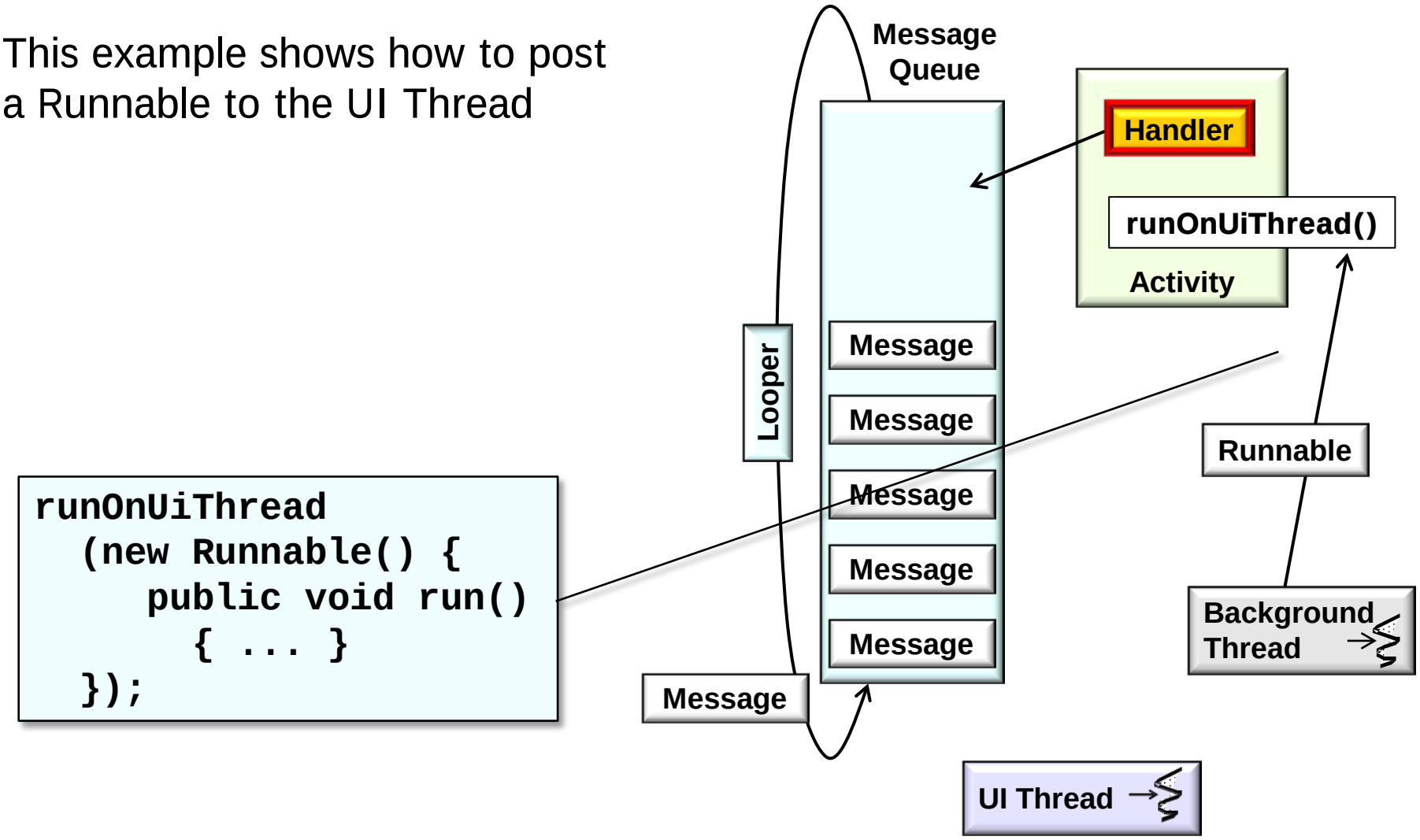
Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread



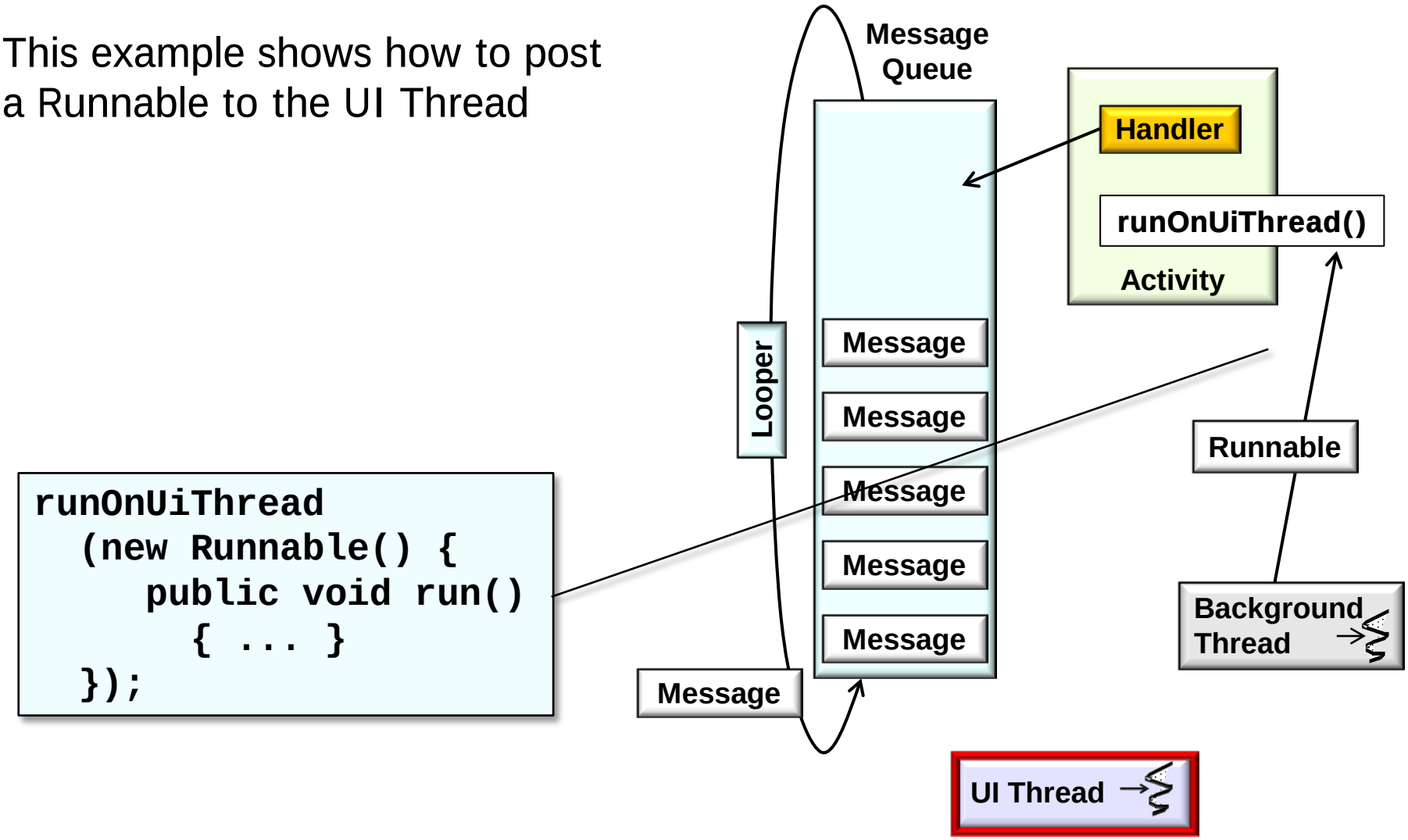
Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread



Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread



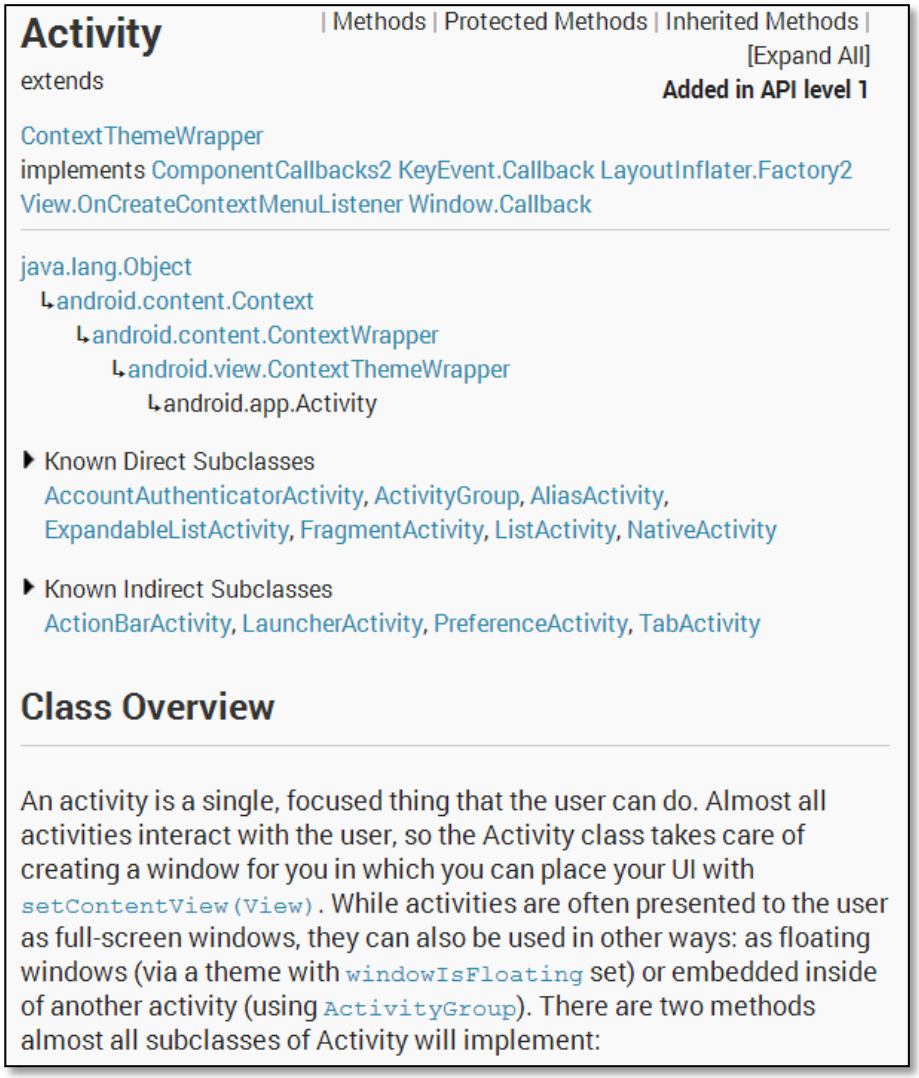
Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread

source.developer.android has all the Android source code

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do



Activity

| Methods | Protected Methods | Inherited Methods | [Expand All]

extends

Added in API level 1

ContextThemeWrapper

implements ComponentCallbacks2 KeyEvent.Callback LayoutInflater.Factory2 View.OnCreateContextMenuListener Window.Callback

java.lang.Object
↳android.content.Context
↳android.content.ContextWrapper
↳android.view.ContextThemeWrapper
↳android.app.Activity

▸ Known Direct Subclasses
AccountAuthenticatorActivity, ActivityGroup, AliasActivity, ExpandableListActivity, FragmentActivity, ListActivity, NativeActivity

▸ Known Indirect Subclasses
ActionBarActivity, LauncherActivity, PreferenceActivity, TabActivity

Class Overview

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with `setContentView(View)`. While activities are often presented to the user as full-screen windows, they can also be used in other ways: as floating windows (via a theme with `windowIsFloating` set) or embedded inside of another activity (using `ActivityGroup`). There are two methods almost all subclasses of Activity will implement:

developer.android.com/reference/android/app/Activity.html has more info

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods

```
public class Activity ... {
  ...
  public onConfigurationChanged
       (Configuration newConfig)
  public onAttachedFragment
            (Fragment fragment)
  public onBackPressed()
  public onTouchEvent
            (MotionEvent event)
  protected void onCreate
     (Bundle savedInstanceState)
  protected void onStart()
  protected void onRestart()
  protected void onResume()
  protected void onPause()
  protected void onStop()
  protected void onDestroy()
  ...
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods

```
public class Activity ... {
  ...
  public onConfigurationChanged
       (Configuration newConfig)
  public onAttachedFragment
            (Fragment fragment)
  public onBackPressed()
  public onTouchEvent
            (MotionEvent event)
  protected void onCreate
     (Bundle savedInstanceState)
  protected void onStart()
  protected void onRestart()
  protected void onResume()
  protected void onPause()
  protected void onStop()
  protected void onDestroy()
  ...
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods

```
public class Activity ... {
  ...
  public onConfigurationChanged
       (Configuration newConfig)
  public onAttachedFragment
            (Fragment fragment)
  public onBackPressed()
  public onTouchEvent
            (MotionEvent event)
  protected void onCreate
     (Bundle savedInstanceState)
  protected void onStart()
  protected void onRestart()
  protected void onResume()
  protected void onPause()
  protected void onStop()
  protected void onDestroy()
  ...
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods

```
public class Activity ... {
  ...
  public onConfigurationChanged
       (Configuration newConfig)
  public onAttachedFragment
            (Fragment fragment)
  public onBackPressed()
  public onTouchEvent
            (MotionEvent event)
  protected void onCreate
     (Bundle savedInstanceState)
  protected void onStart()
  protected void onRestart()
  protected void onResume()
  protected void onPause()
  protected void onStop()
  protected void onDestroy()
  ...
```

frameworks/base/core/java/android/app/Activity.java has the source code

Posting to a Handler in the HaMeR Framework

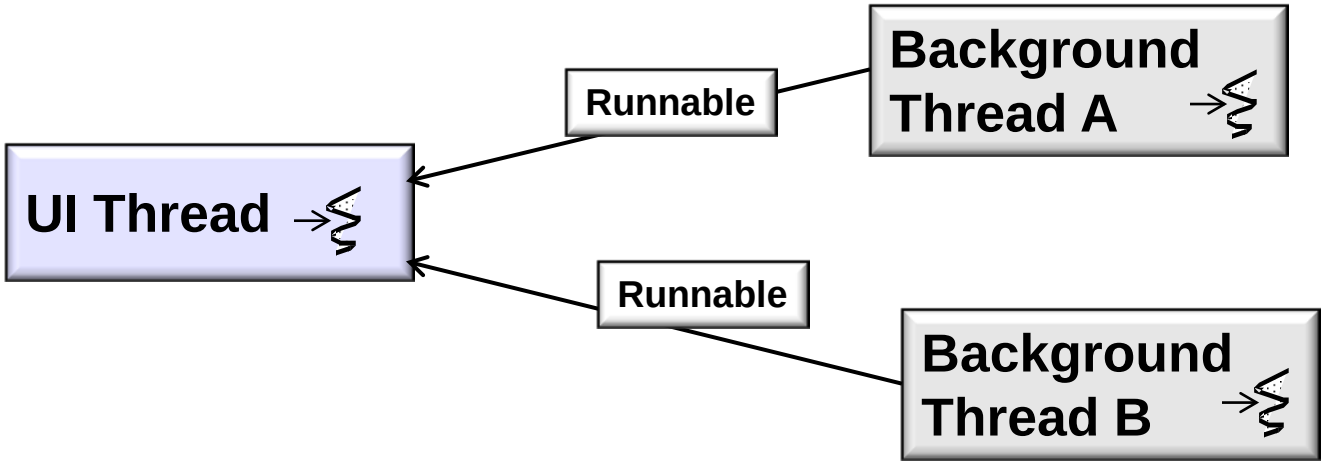
- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread

```
public class Activity ... {
   ...
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread

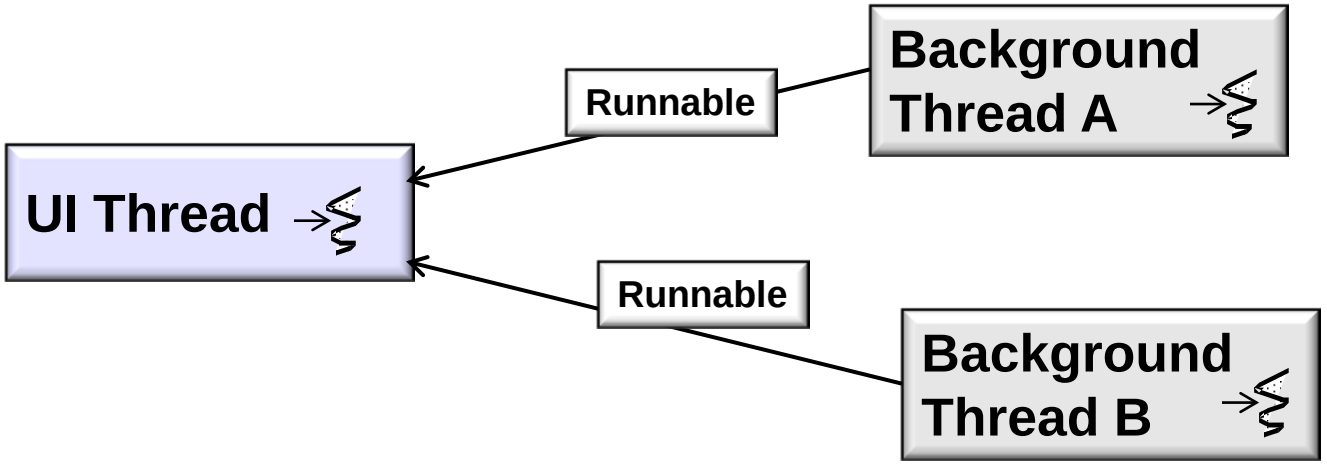
```
public class Activity ... {
   ...
```



Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread

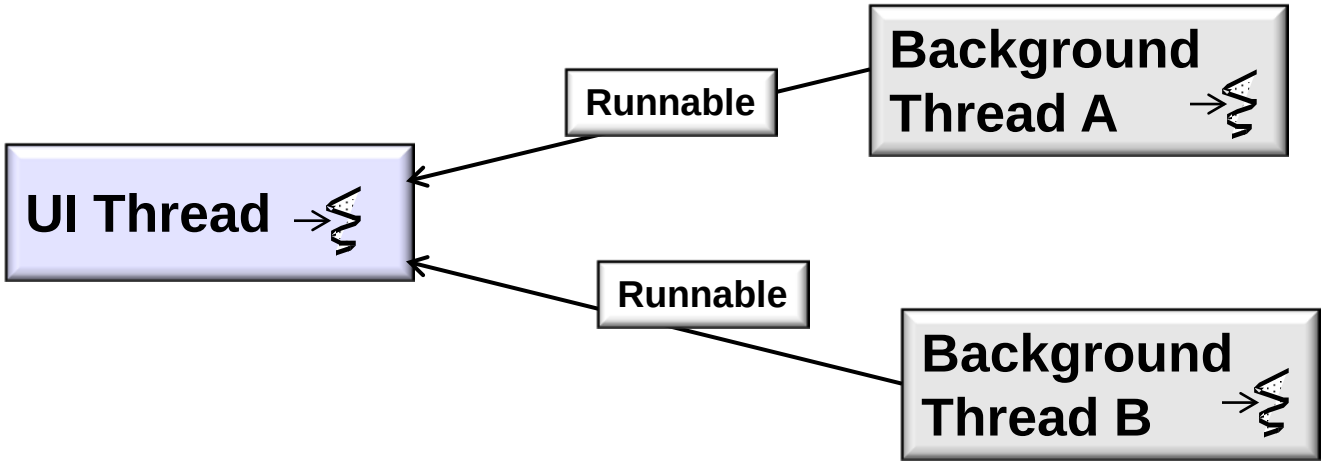
```
public class Activity ... {
  ...

  public final void runOnUiThread
    (Runnable action) {
    ...
```



Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread

```
public class Activity ... {
  ...

  public final void runOnUiThread
    (Runnable action) {
    ...
```



Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread

```
public class Activity ... {
  ...

public final void runOnUiThread
  (Runnable action) {
  ...
```

```
void print(final String output) {
  ....runOnUiThread (new Runnable() { public void run() {
        mTextView.append(output);
  }});
  ...
}
```

See earlier part on “Java Synchronization & Scheduling Example”

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread
  - Activity contains a Handler associated with the UI Thread's Looper

```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    ...
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread
  - Activity contains a Handler associated with the UI Thread's Looper

```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    if (Thread.currentThread()
        != mUiThread) {
            mHandler.post(action);
        } else {
            action.run();
        }
    }
  }
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread
  - Activity contains a Handler associated with the UI Thread's Looper

```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    if (Thread.currentThread()
        != mUiThread) {
          mHandler.post(action);
        } else {
          action.run();
        }
    }
  }
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread
  - Activity contains a Handler associated with the UI Thread's Looper

```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    if (Thread.currentThread()
        != mUiThread) {
            mHandler.post(action);
        } else {
            action.run();
        }
    }
  }
```

Posting to a Handler in the HaMeR Framework

- This example shows how to post a Runnable to the UI Thread
- An Android Activity provides a single, focused thing a user can do
  - It contains dozens of methods
  - Its methods run in the UI Thread
  - runOnUiThread() executes a specified action on the UI Thread
  - Activity contains a Handler associated with the UI Thread's Looper

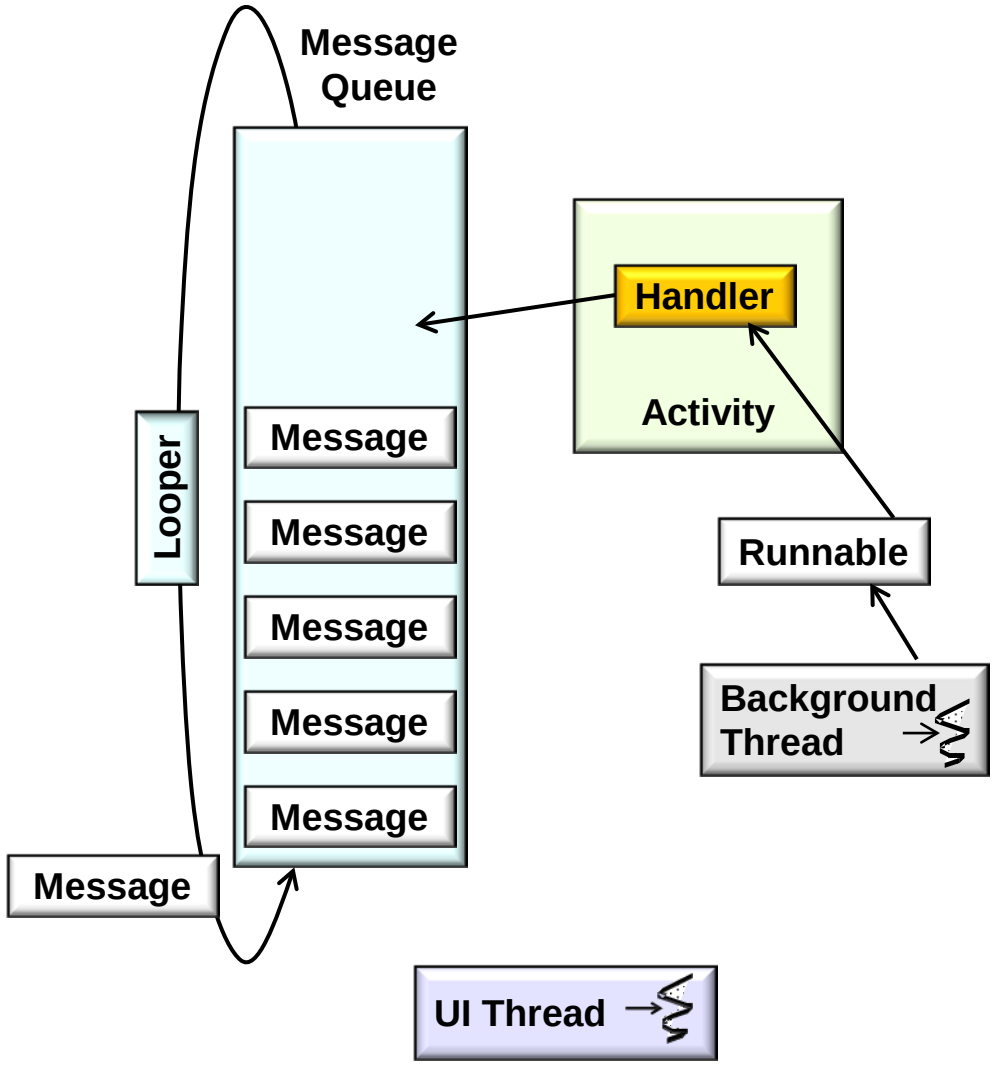
```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    if (Thread.currentThread()
        != mUiThread) {
          mHandler.post(action);
        } else {
          action.run();
        }
    }
  }
```

This use of post() is not limited to Activity – it's a common Android idiom

The Handler's post() Method Implementation

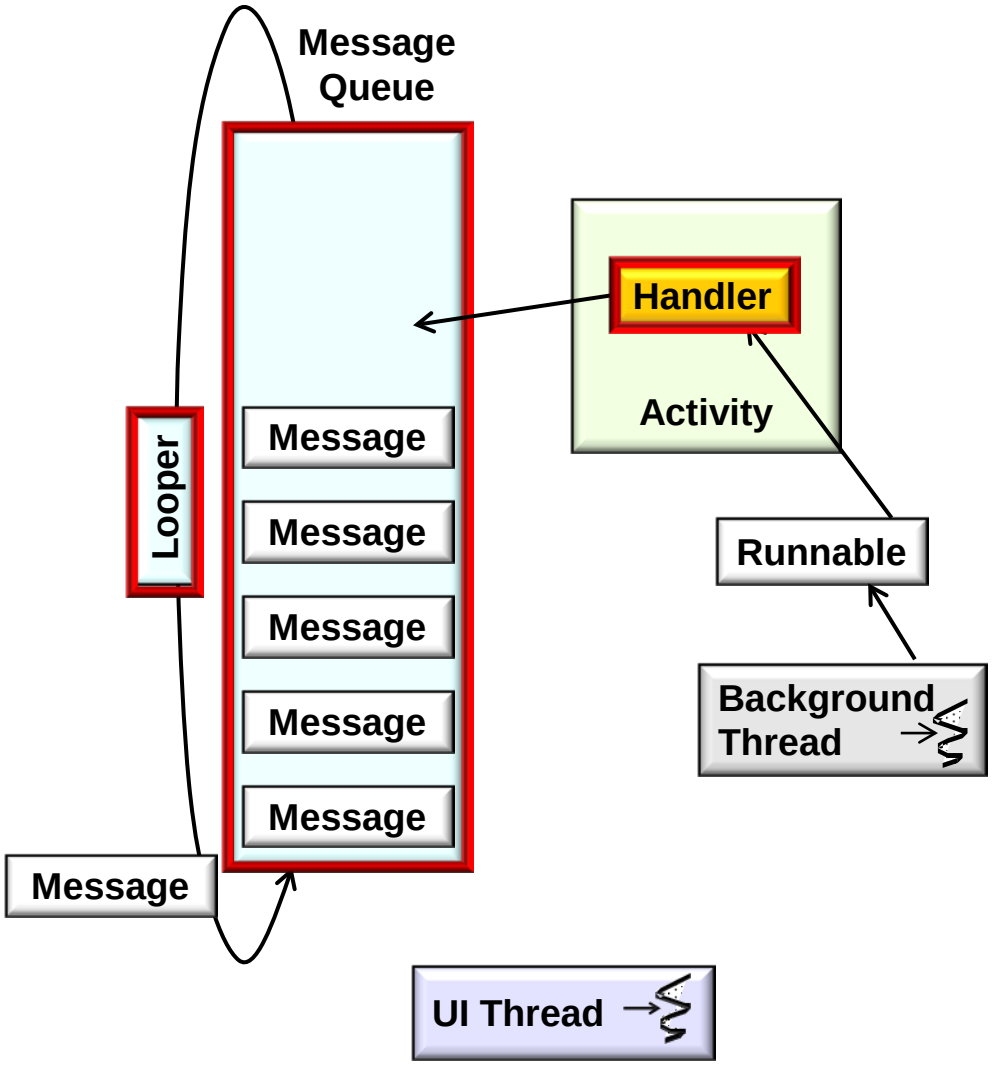
The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation



The Handler's post() Method Implementation

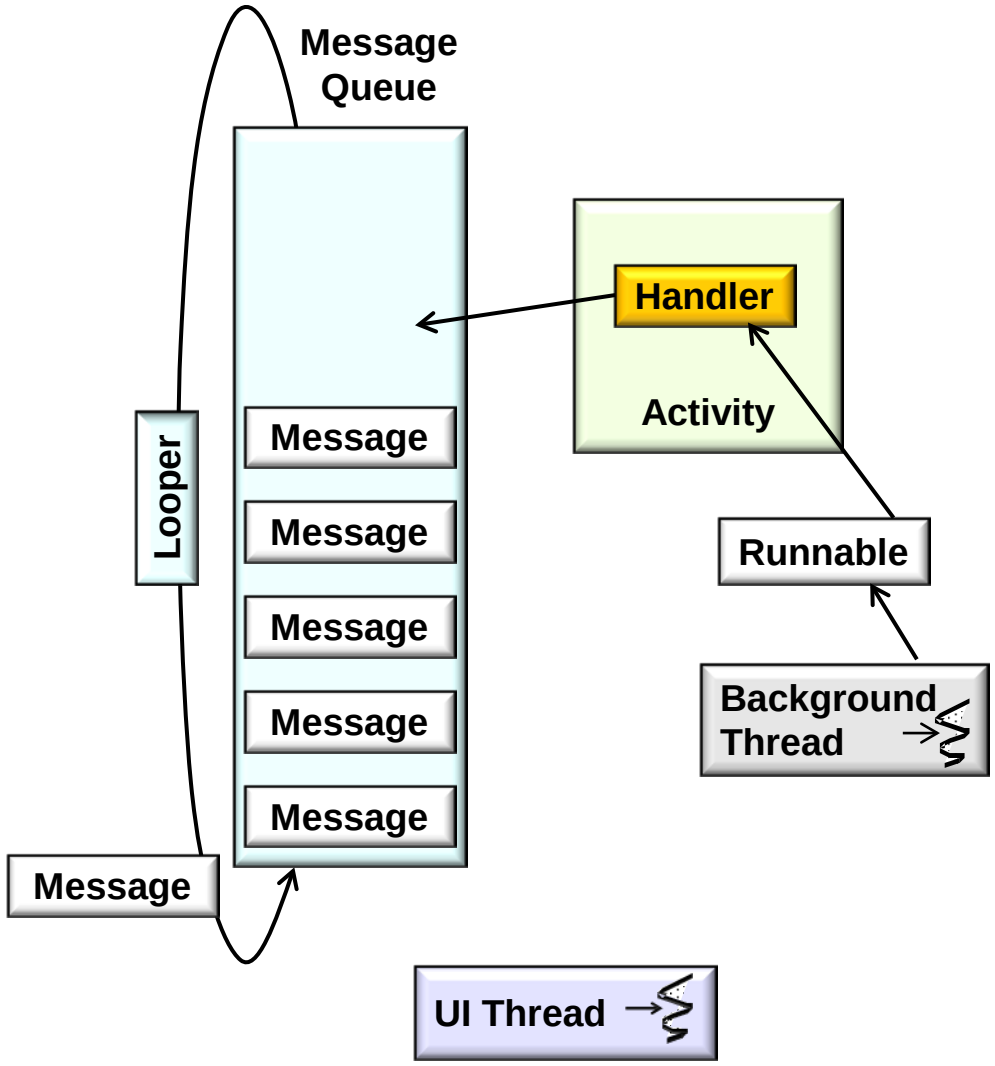
- This example shows the Handler's post() method implementation
  - It also shows how classes in the Android HaMeR concurrency framework collaborate



See earlier part in "Overview of Android Concurrency Frameworks & Idioms"

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
  - It also shows how classes in the Android HaMeR concurrency framework collaborate



You don't need to understand all these steps to use the HaMeR framework

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler


```
public class Activity ... {
  ...
  final Handler mHandler =
    new Handler();

  public final void runOnUiThread
    (Runnable action) {
    if (Thread.currentThread()
        != mUiThread) {
            mHandler.post(action);
        } else {
            action.run();
        }
    }
}
```

This code runs in the calling Thread, i.e., in a background Thread

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }
```

frameworks/base/core/java/android/os/Handler.java has the source code

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }

  private final Message
    getPostMessage(Runnable r) {
      Message m =
        Message.obtain();
      m.callback = r;
      return m;
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }

  private final Message
    getPostMessage(Runnable r) {
      Message m =
        Message.obtain();
      m.callback = r;
      return m;
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }

  private final Message
    getPostMessage(Runnable r) {
      Message m =
        Message.obtain();
      m.callback = r;
      return m;
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
                  post(Runnable r) {
    return sendMessageDelayed
      (getPostMessage(r), 0);
  }
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
    sendMessageAtTime
      (Message msg,
       long uptimeMillis) {
    ...
    MessageQueue queue = mQueue;
    ...
    msg.target = this;
    queue.enqueueMessage
      (msg, uptimeMillis);
    ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
    sendMessageAtTime
      (Message msg,
       long uptimeMillis) {
    ...
    MessageQueue queue = mQueue;
    ...
    msg.target = this;
    queue.enqueueMessage
      (msg, uptimeMillis);
    ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
    sendMessageAtTime
      (Message msg,
       long uptimeMillis) {
    ...
    MessageQueue queue = mQueue;
    ...
    msg.target = this;
    queue.enqueueMessage
      (msg, uptimeMillis);
    ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public final boolean
    sendMessageAtTime
      (Message msg,
       long uptimeMillis) {
    ...
    MessageQueue queue = mQueue;
    ...
    msg.target = this;
    queue.enqueueMessage
      (msg, uptimeMillis);
    ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Looper {
  ...
  final MessageQueue mQueue;

  public static void loop() {
    Looper me = myLooper();
    ...
    MessageQueue queue =
      me.mQueue;

    for (;;) {
      Message msg =
        queue.next();
      ...
      msg.target.
        dispatchMessage(msg);
      ...
```

This code runs in the User Interface (UI) Thread

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Looper {
  ...
  final MessageQueue mQueue;

  public static void loop() {
    Looper me = myLooper();
    ...
    MessageQueue queue =
      me.mQueue;

    for (;;) {
      Message msg =
        queue.next();
      ...
      msg.target.
        dispatchMessage(msg);
      ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Looper {
  ...
  final MessageQueue mQueue;

  public static void loop() {
    Looper me = myLooper();
    ...
    MessageQueue queue =
      me.mQueue;

    for (;;) {
      Message msg =
        queue.next();
      ...
      msg.target.
        dispatchMessage(msg);
      ...
```

See previous part on "Android Looper"

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Looper {
  ...
  final MessageQueue mQueue;

  public static void loop() {
    Looper me = myLooper();
    ...
    MessageQueue queue =
      me.mQueue;

    for (;;) {
      Message msg =
        queue.next();
      ...
      msg.target.
        dispatchMessage(msg);
      ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
  ...
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                    handleCallback
                     (Message message)
{
    message.callback.run();
}
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                  (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                  handleCallback
                   (Message message)
{
    message.callback.run();
}
```

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                    handleCallback
                     (Message message)
{
  message.callback.run();
}
```

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

See earlier part on "Java Synchronization & Scheduling Example"

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                    handleCallback
                     (Message message)
{
  message.callback.run();
}
```

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

See earlier part on "Java Synchronization & Scheduling Example"

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods

UI Thread

Note the inversion of control in this code

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Looper {
  ...
  final MessageQueue mQueue;

  public static void loop() {
    Looper me = myLooper();
    ...
    MessageQueue queue =
      me.mQueue;

    for (;;) {
      Message msg =
        queue.next();
      ...
      msg.target.
        dispatchMessage(msg);
      ...
```

Note the inversion of control in this code

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods


```
public class Handler {
  ...
  public void dispatchMessage
                  (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                  handleCallback
                   (Message message)
{
    message.callback.run();
}
```

Note the inversion of control in this code

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
  - Handler's post() methods share code with the sendMessage() methods

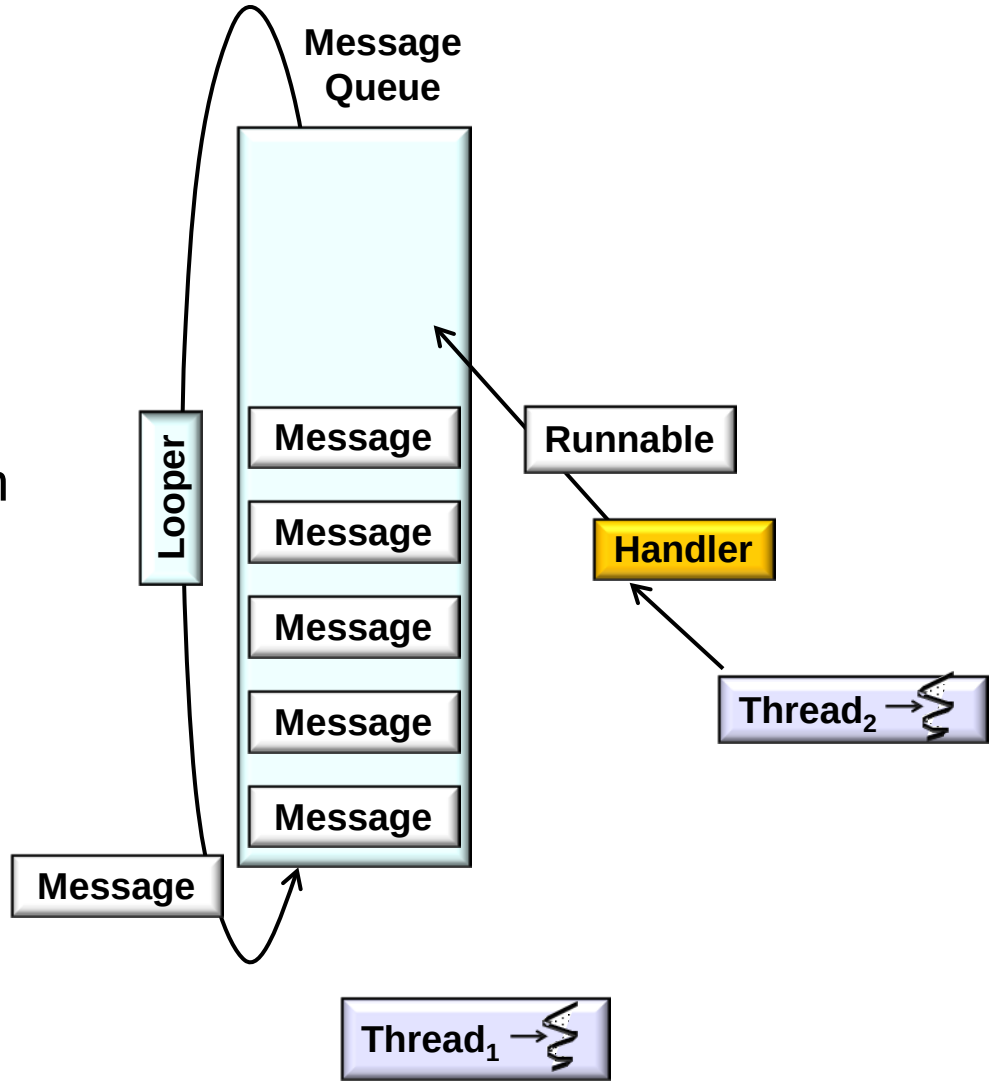

```
public class Handler {
  ...
  public void dispatchMessage
                    (Message msg) {
    if (msg.callback != null)
      handleCallback(msg);
    ...

  private final void
                    handleCallback
                     (Message message)
{
  message.callback.run();
}
```

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

Note the inversion of control in this code

The Handler's post() Method Implementation

- This example shows the Handler's post() method implementation
- We show what happens after Activity.runOnUiThread() posts a Runnable on its Handler
- We'll explore the entire code path used to post, schedule, & dispatch Runnables later in this module

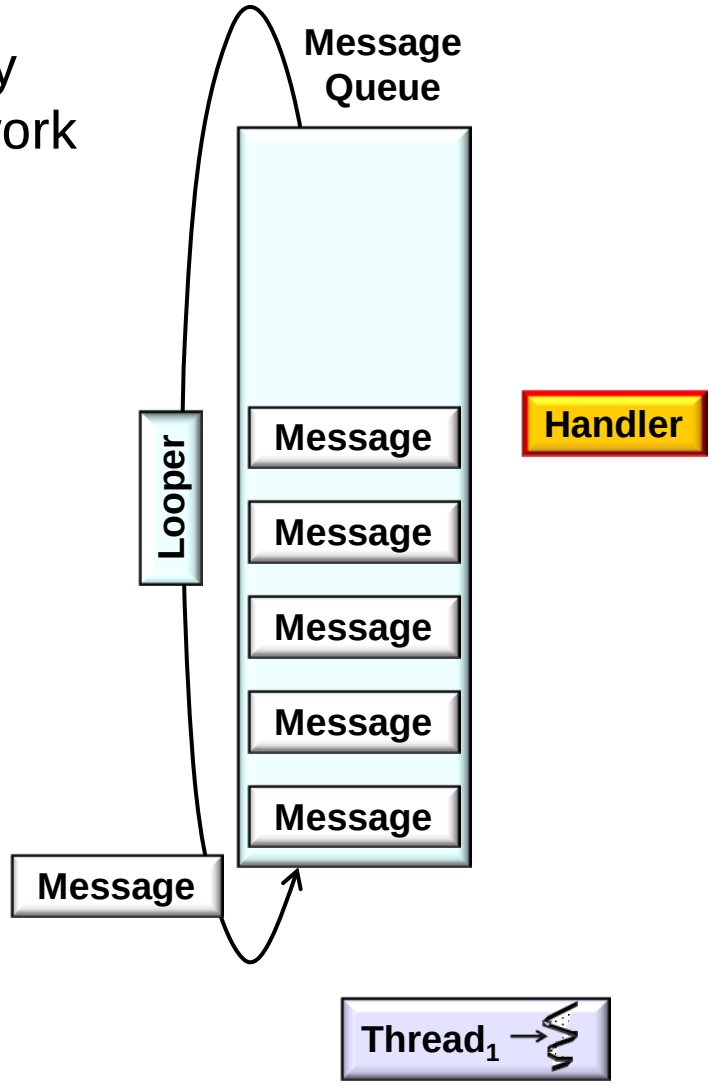


See upcoming parts on "the *Command Processor* pattern"

Summary

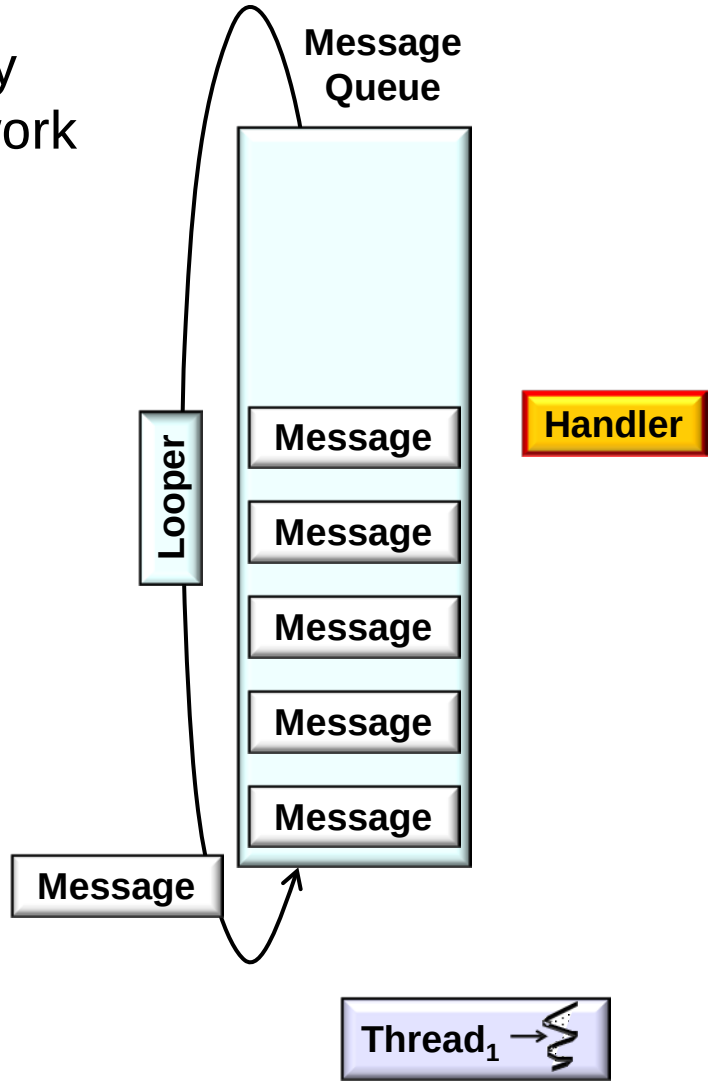
Summary

- Handler's post() methods form key portion of Android HaMeR framework



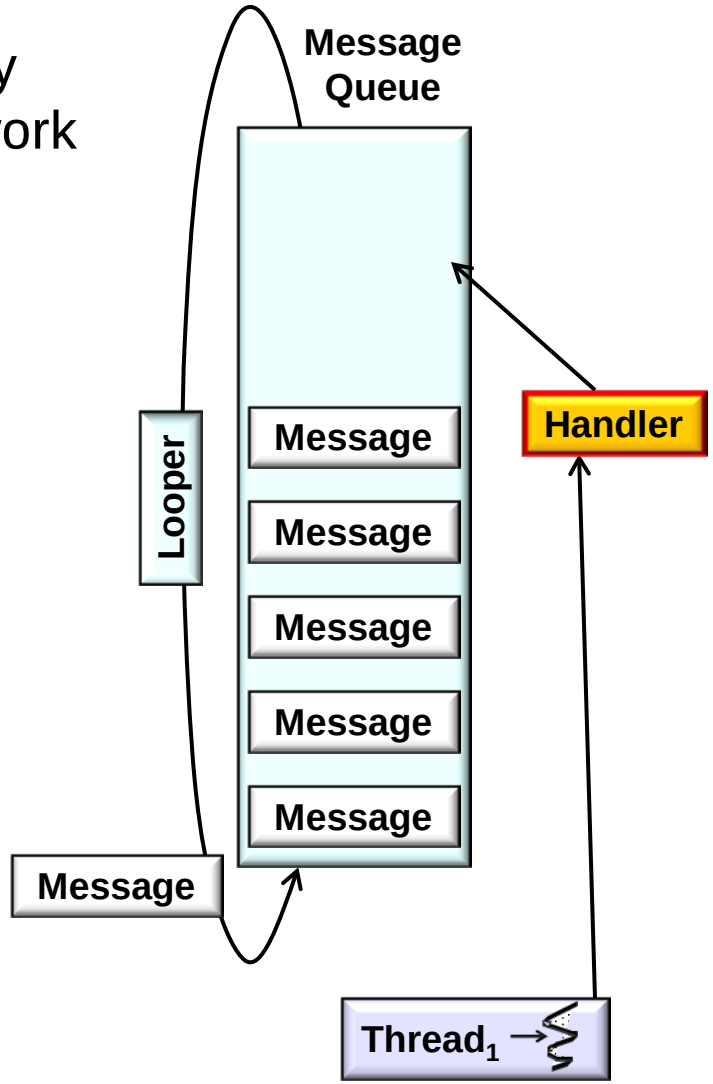
Summary

- Handler's post() methods form key portion of Android HaMeR framework
  - They can enqueue & later process Runnables posted from



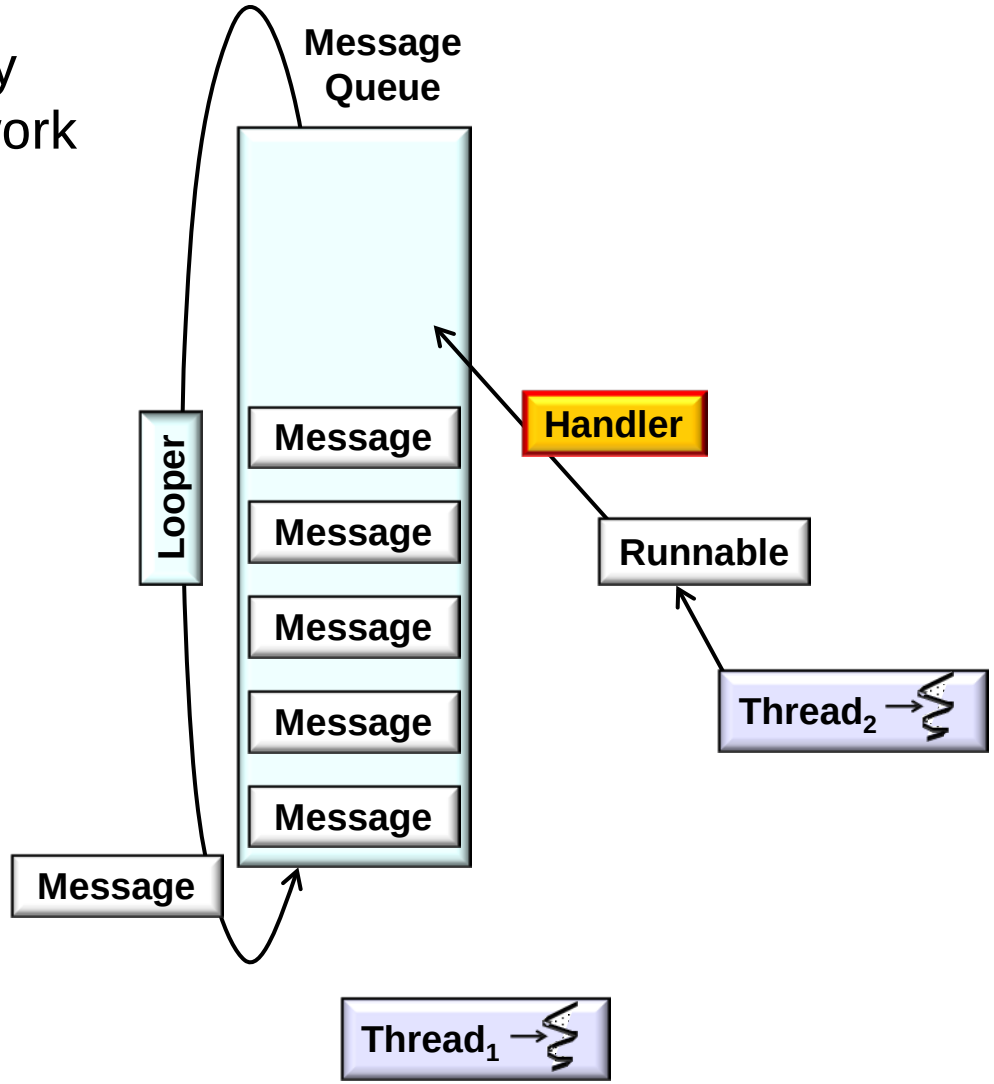
Summary

- Handler's post() methods form key portion of Android HaMeR framework
  - They can enqueue & later process Runnables posted from
    - within a single Thread to itself or



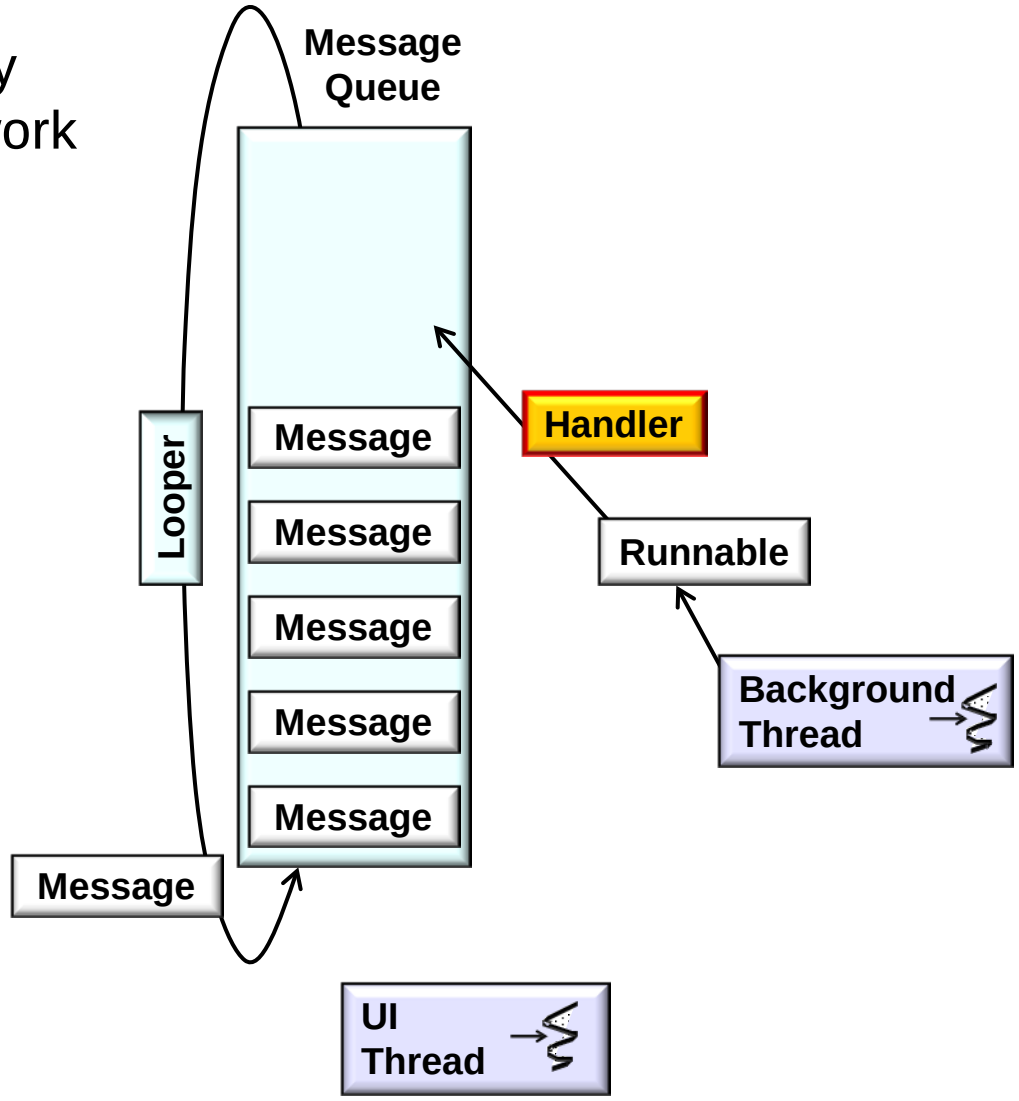
Summary

- Handler's post() methods form key portion of Android HaMeR framework
  - They can enqueue & later process Runnables posted from
    - within a single Thread to itself or
    - one Thread to another



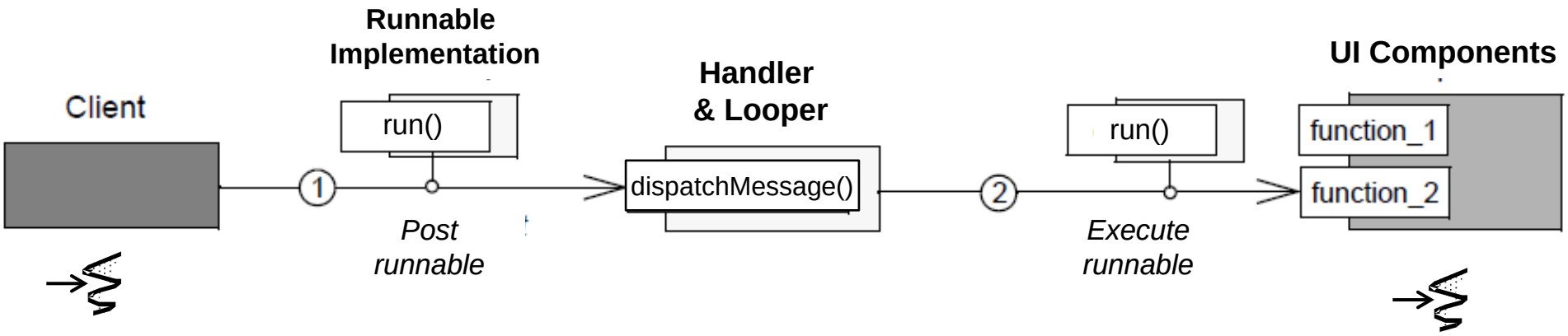
Summary

- Handler's post() methods form key portion of Android HaMeR framework
  - They can enqueue & later process Runnables posted from
    - within a single Thread to itself or
    - one Thread to another
  - They are often used to send commands from background Threads to the UI Thread



Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*

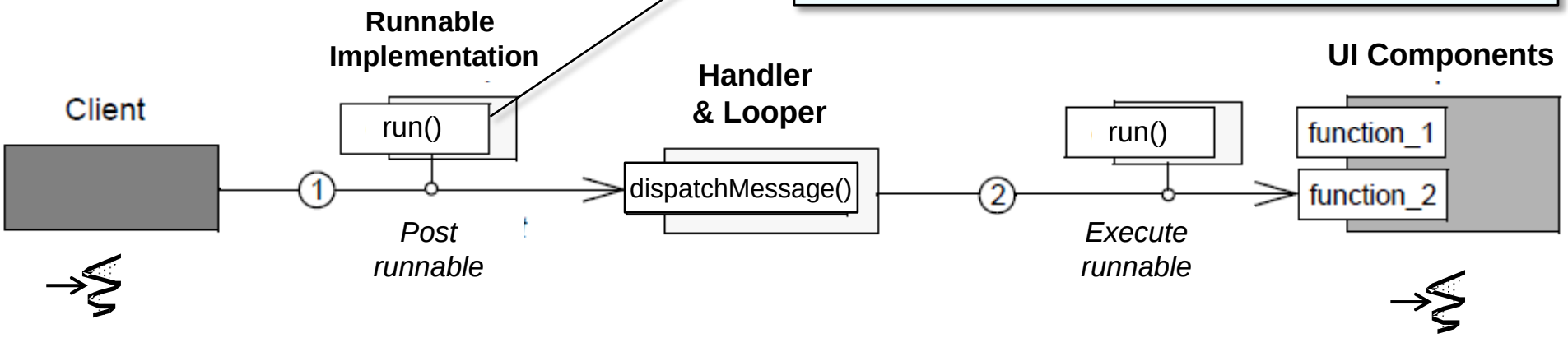


See upcoming parts on "the *Command Processor* pattern"

Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
  - Leverages Java's local class capabilities

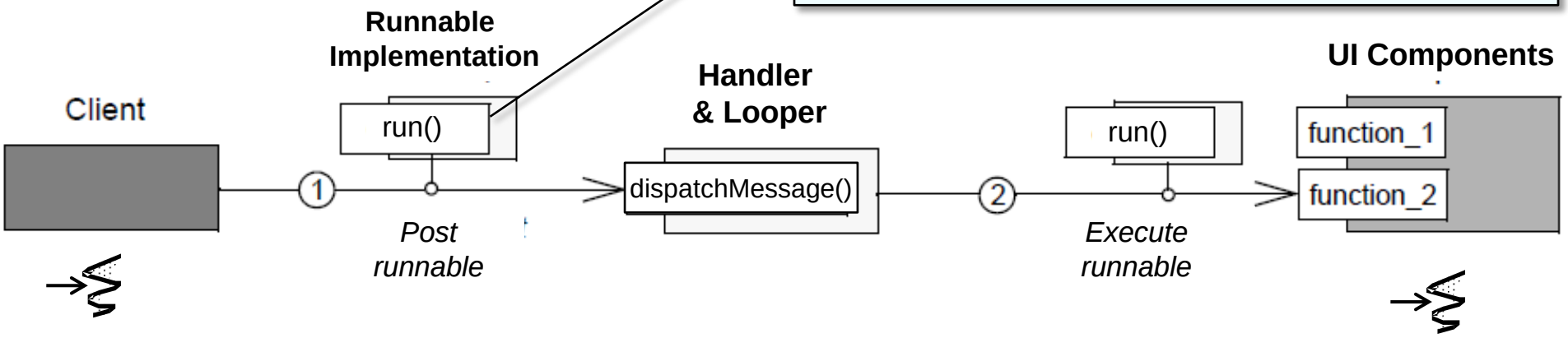
```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
      }}); ...
```



Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
  - Leverages Java's local class capabilities

```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
       }}); ...
```

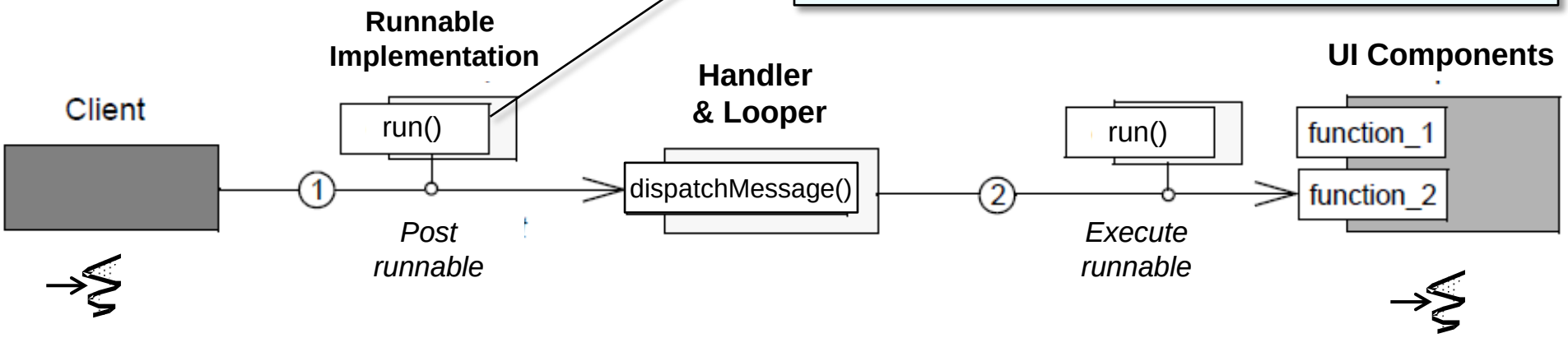


See earlier part on "Java Synchronization & Scheduling Example"

Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
  - Leverages Java's local class capabilities
  - Provides a variant of Closures

```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
       }}); ...
```

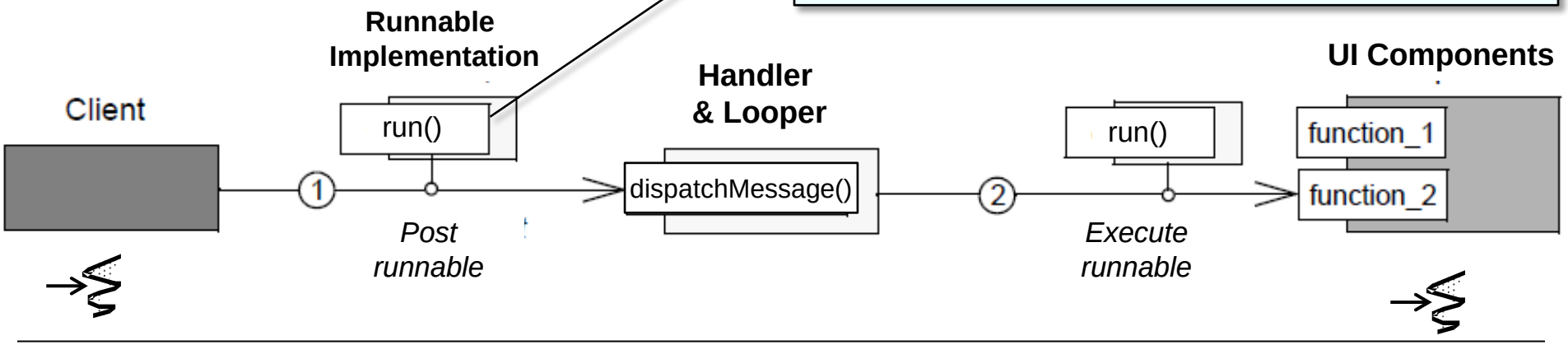


en.wikipedia.org/wiki/Closure_(computer_programming)

Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
  - Leverages Java's local class capabilities
  - Provides a variant of Closures

```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
       }}); ...
```

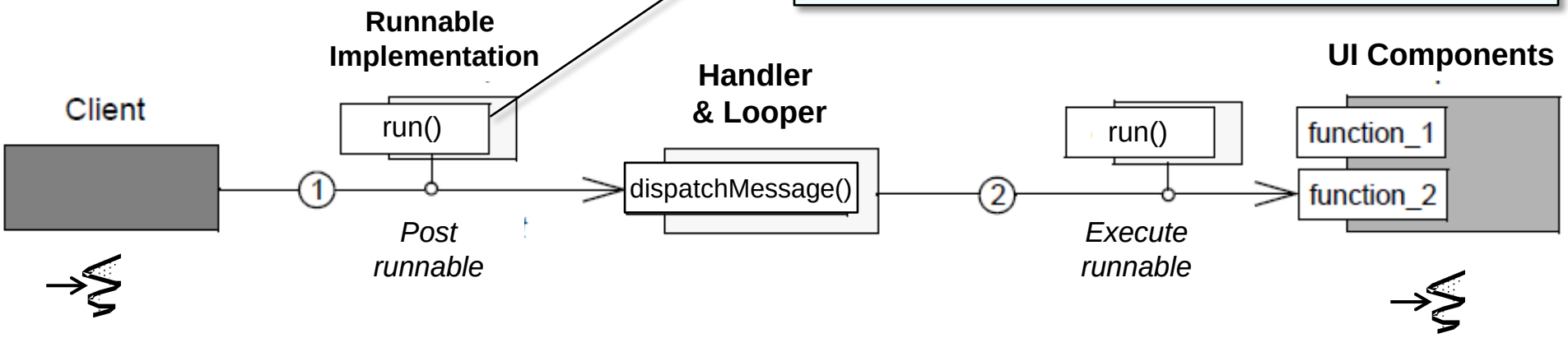


en.wikipedia.org/wiki/Closure_(computer_programming)#Local_classes_.28Java.29

Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at point where post() is called

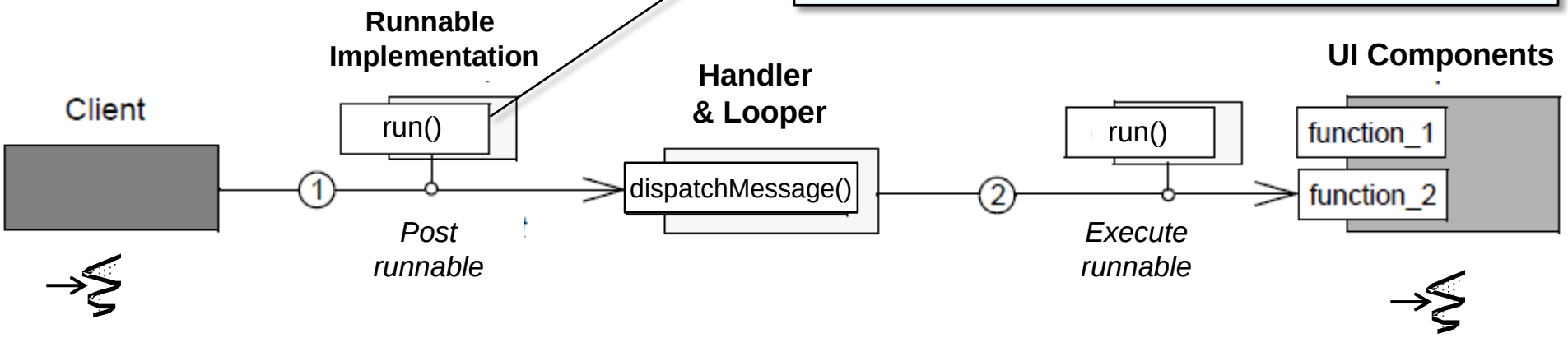
```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
      }}); ...
```



Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at point where post() is called
  - Doesn't require writing any receiver logic to handle post()

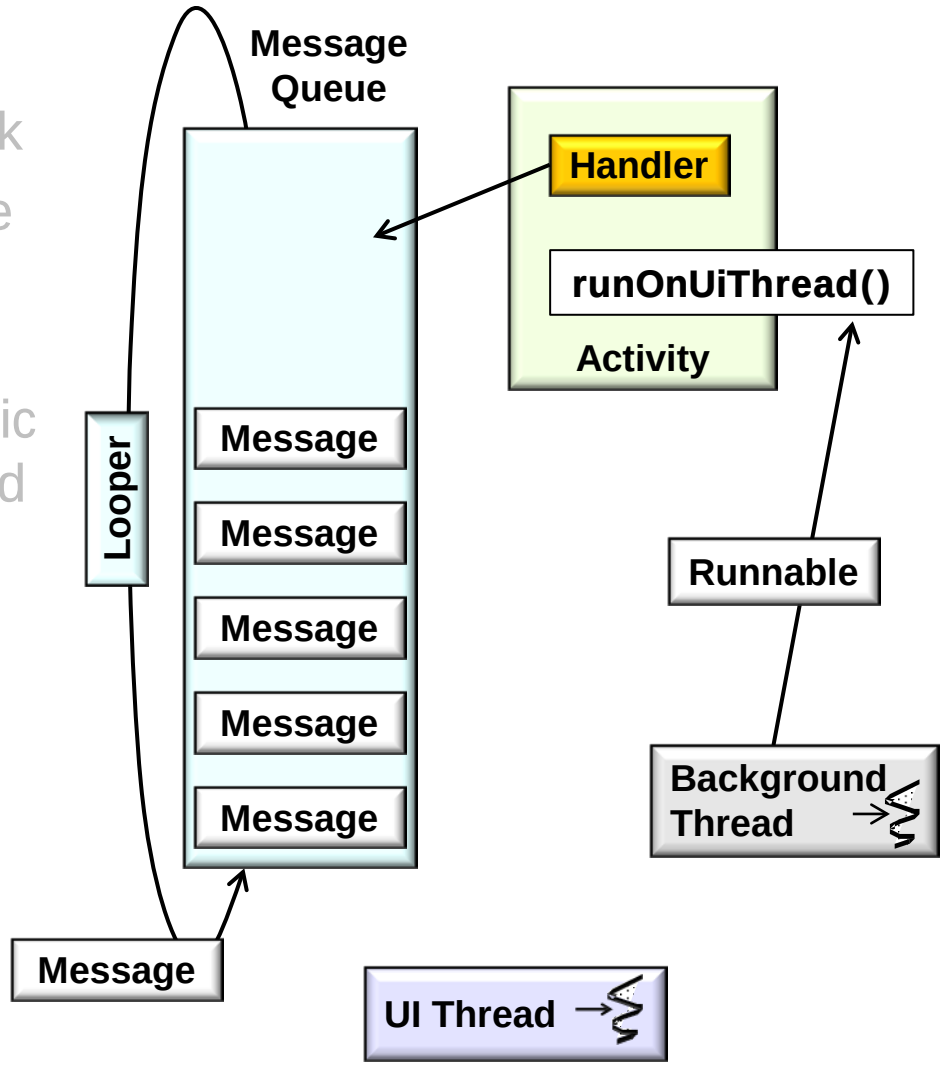
```
class AndroidPlatformStrategy
  ...
  TextView mTextView;

  public void print
       (final String output) {
    ....runOnUiThread
      (new Runnable() {
        public void run() {
          mTextView.
             append(output);
       }}); ...
```



See next part on "Sending & Handling Messages with Android Handler"

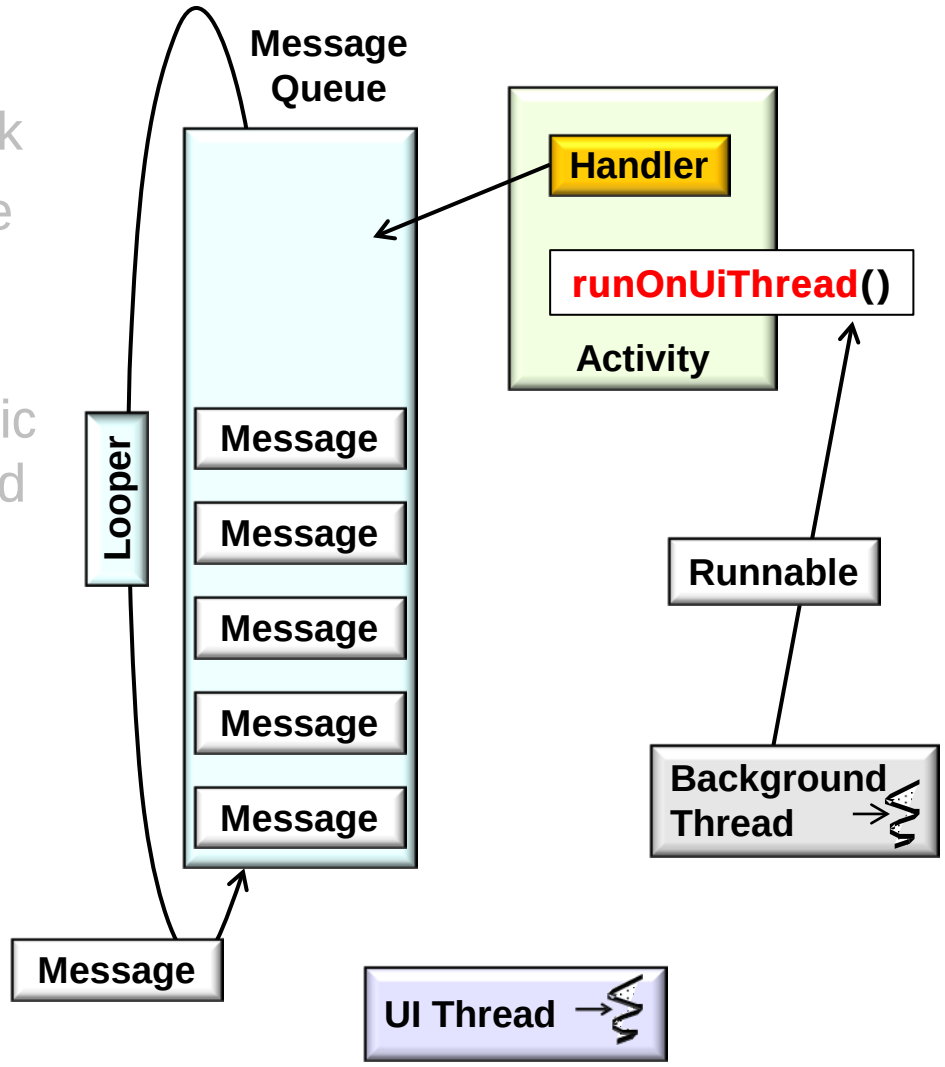
Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler



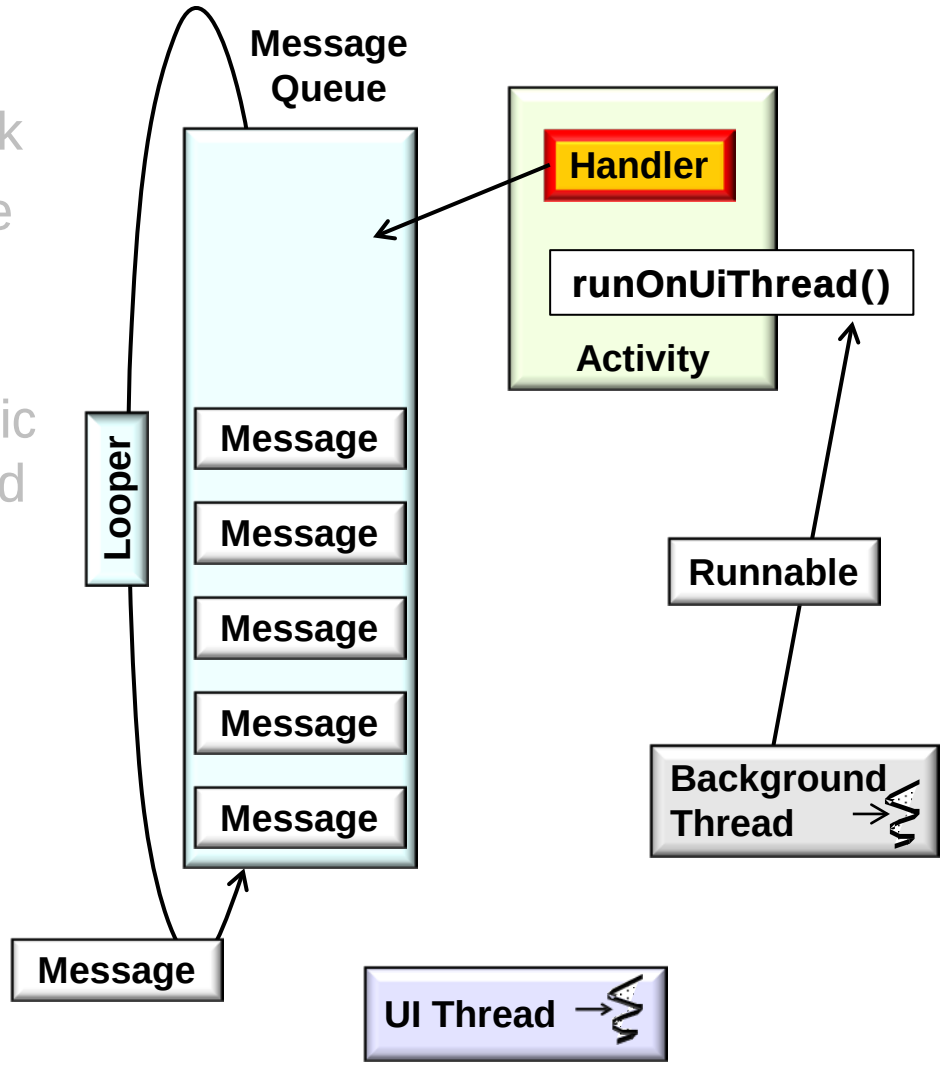
Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler



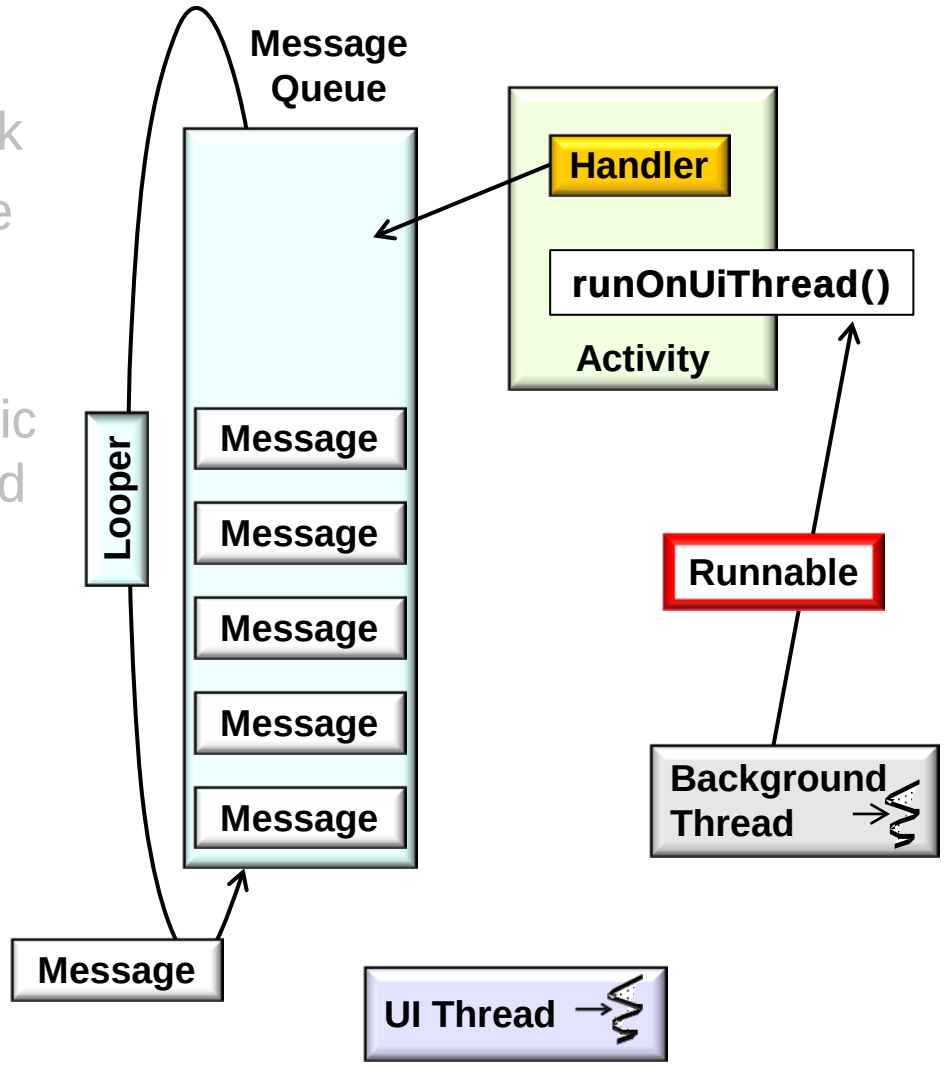
Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler



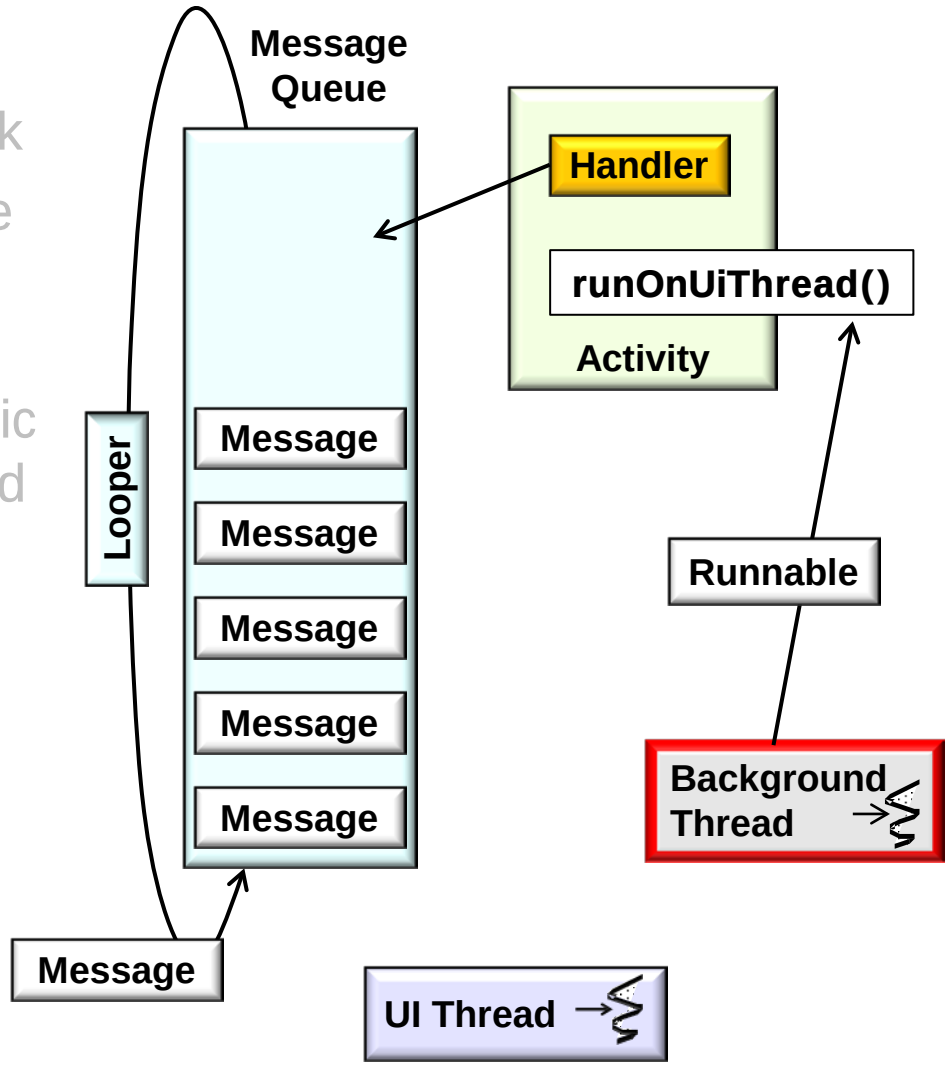
Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler



Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler



Summary

- Handler's post() methods form key portion of Android HaMeR framework
- Handler's post() methods collaborate with MessageQueue & Looper to implement *Command Processor*
- Commands centralize processing logic at the point where the post() method is called
- Android's Activity class was used as an example to showcase command processing features of Handler