Recording in Progress

This class is being recorded

Please turn off your video and/or video if you do not wish to be recorded

CMSC436: Programming Handheld Systems

Threads & Handlers

Today’s Topics

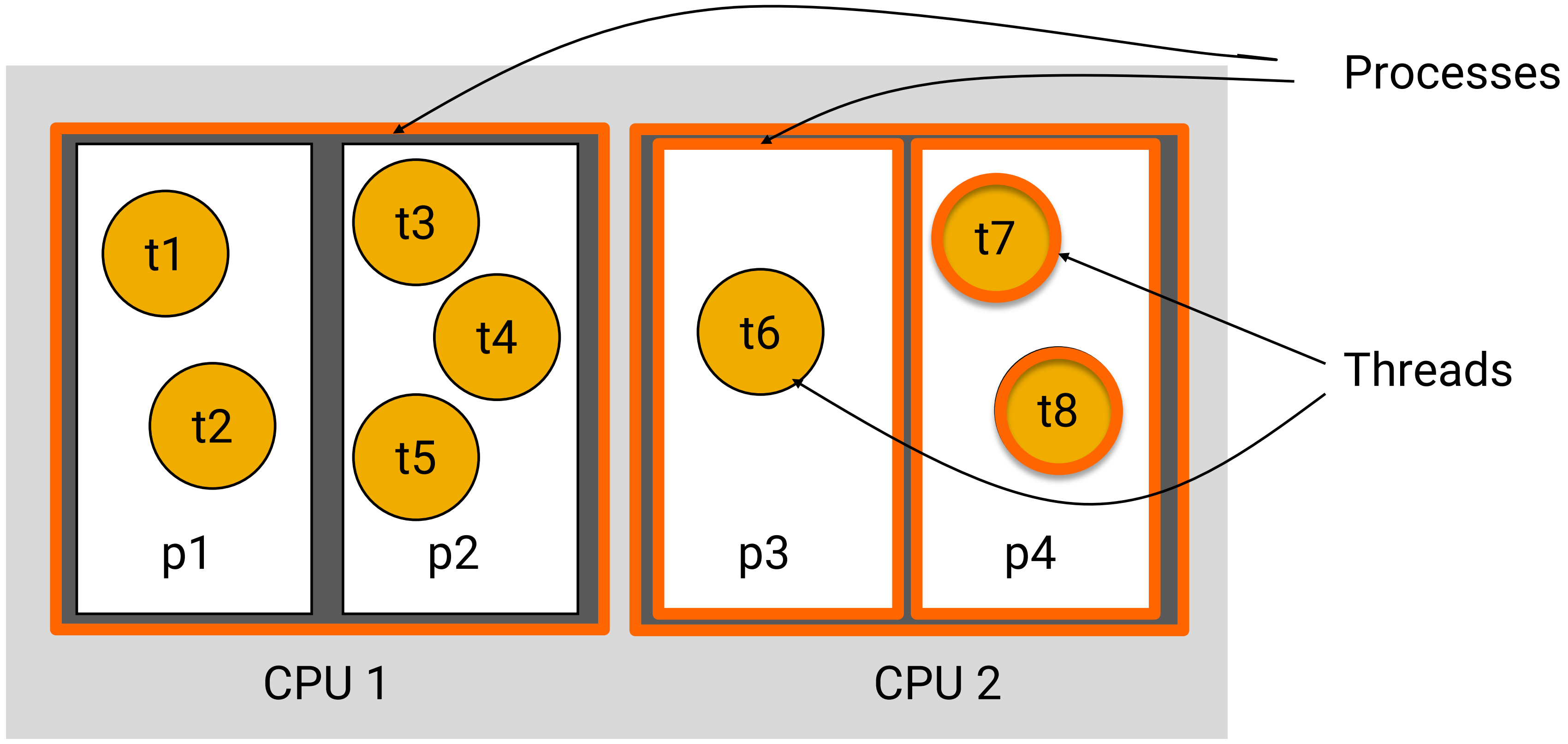
Threading overview

Android’s UI Thread

The Handler class

What is a Thread?

- Conceptual view
  - Parallel computation running in a process
- Implementation view
  - A program counter and a stack
  - Heap and static areas shared with other threads



Computing Device

Common Thread Model

Threads implement the Runnable interface

    public void run()

See:

https://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html

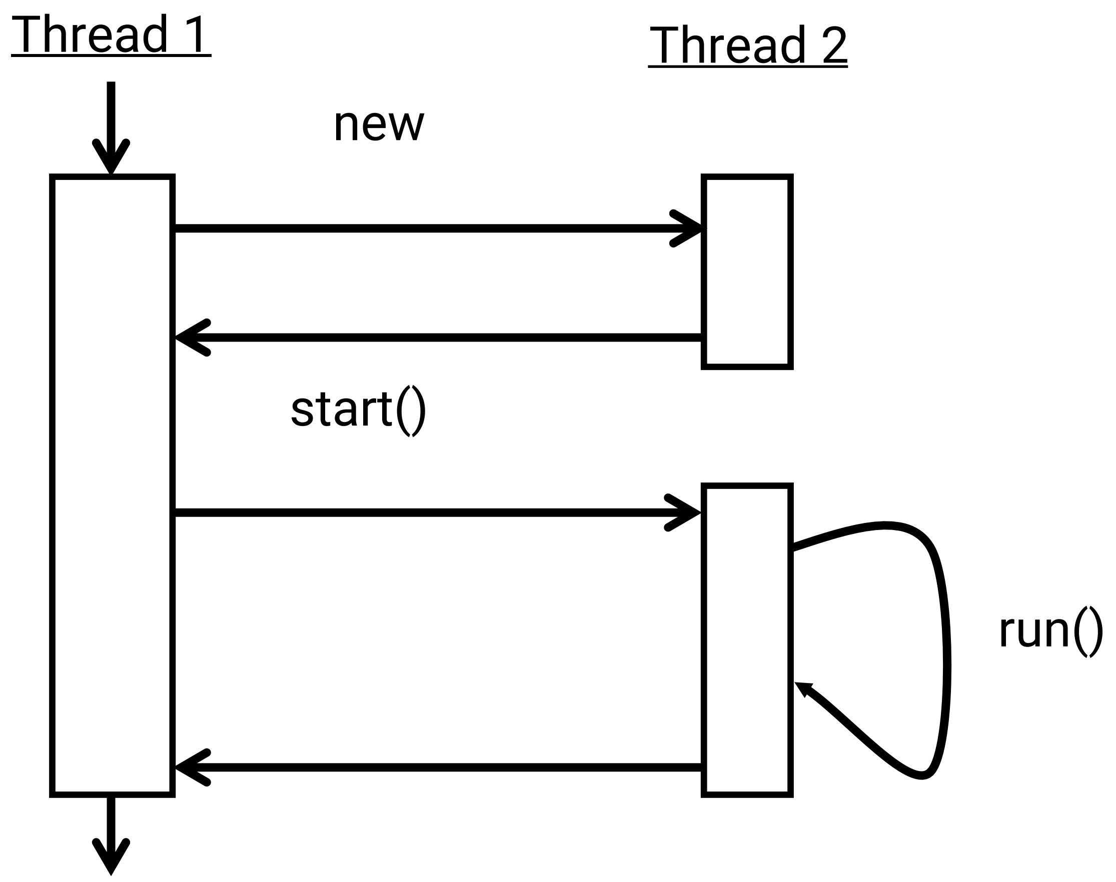
Some Commonly-used Thread Methods

- void start()
    - Starts the Thread
- void sleep(long time)
    - Sleeps for the given period

Basic Thread Use Case

- Instantiate a Thread object
- Invoke the Thread’s start() method
  - Thread’s run() method get called
  - Thread terminates when run() returns

Basic Thread Use Case



ThreadingNoThreading

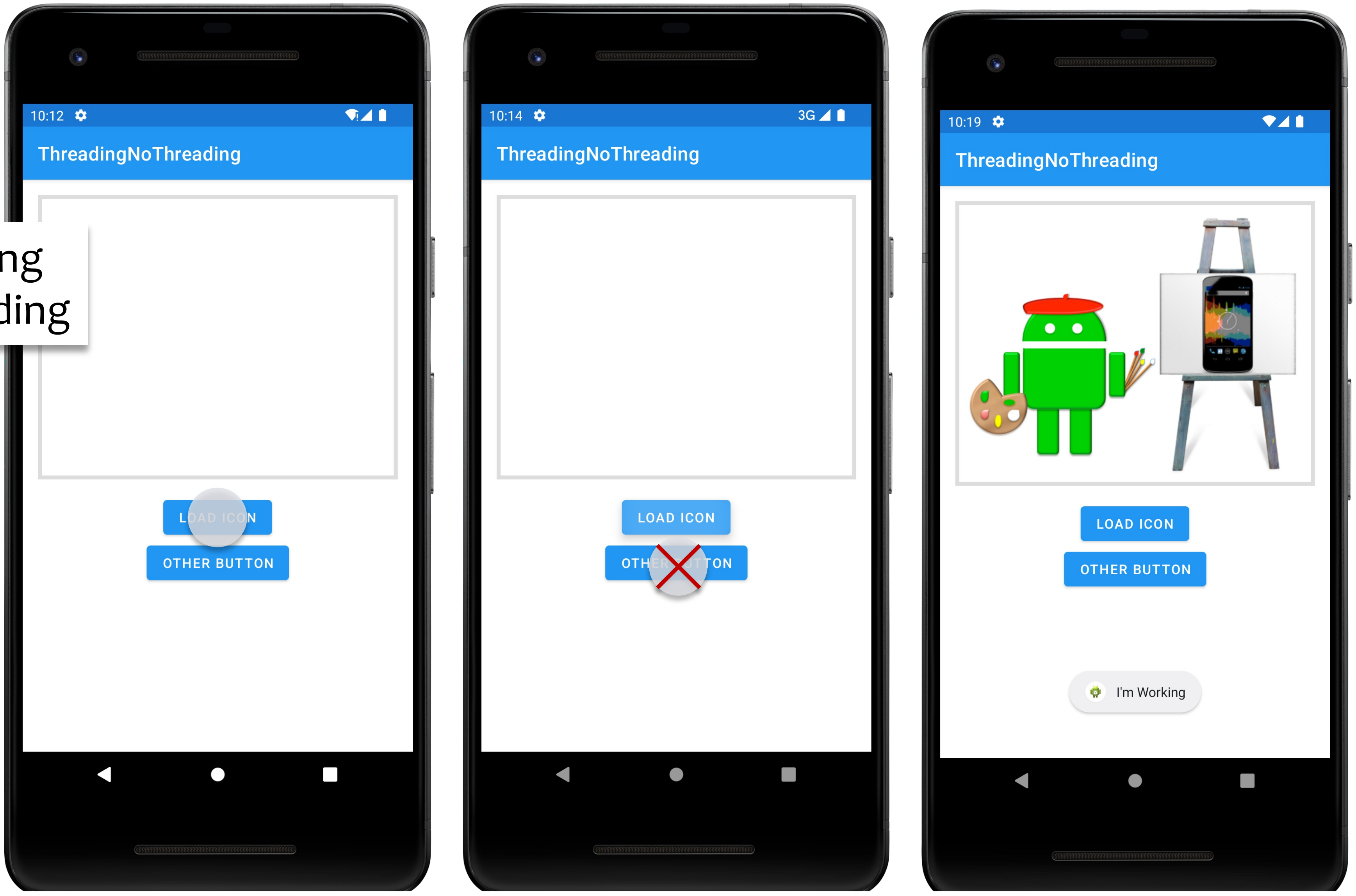
Application displays two buttons

LoadIcon: Load and show bitmap from a resource file & display

Other Button: Display a Toast message

Problem: The Other Button doesn’t respond right after LoadIcon button is pressed

Threading NoThreading

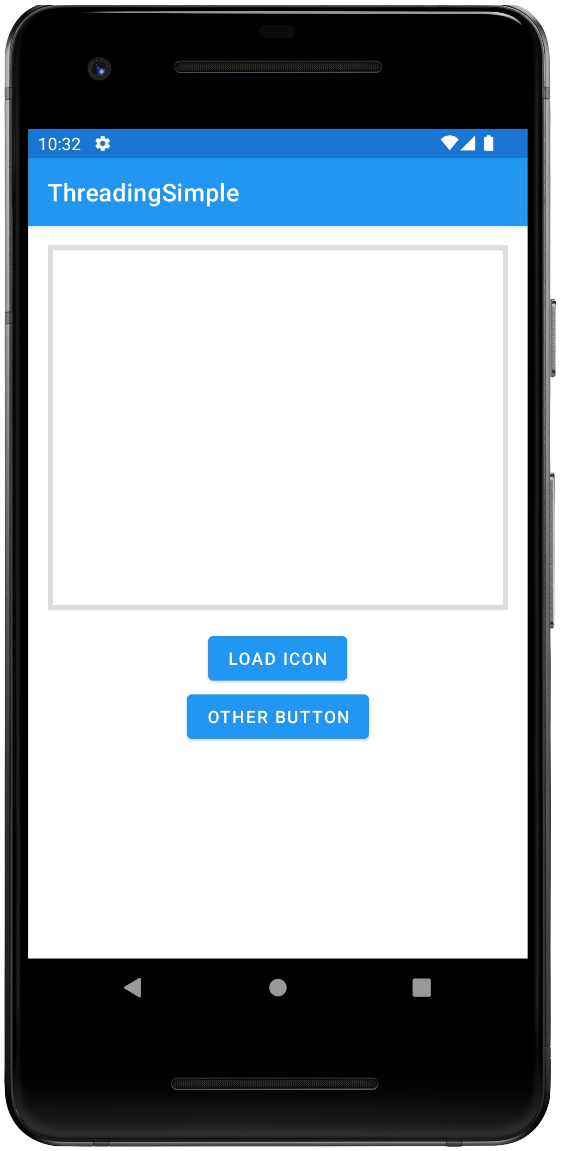


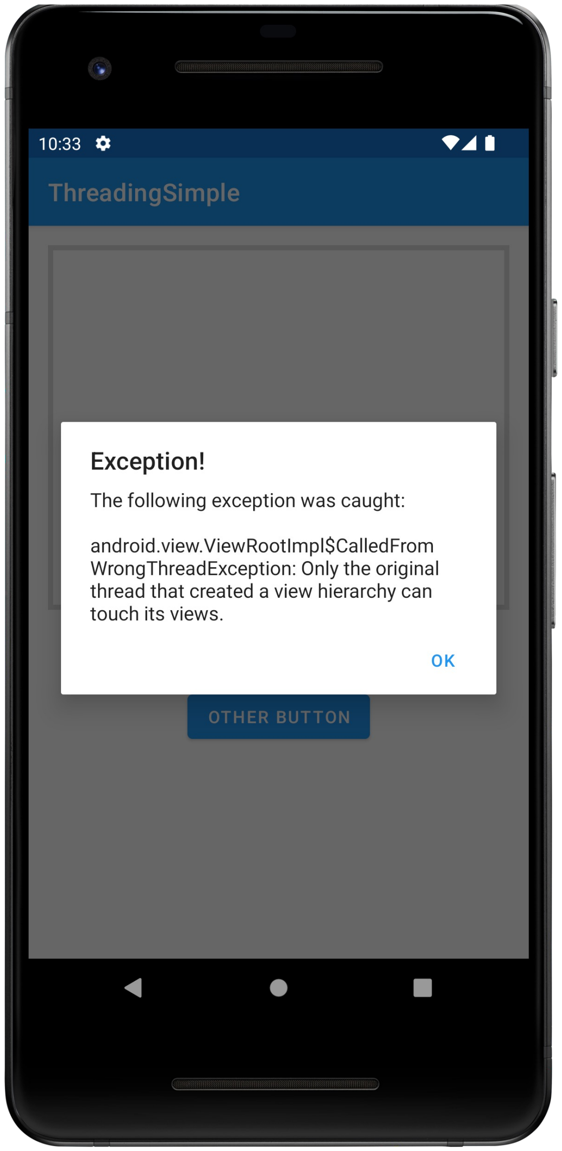
ThreadingSimple

Seemingly obvious, but incorrect, solution:

Button listener spawns a separate Thread to load bitmap & display it

Threading Simple





Exception!

The following exception was caught:

android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.

OK

The UI Thread

Applications have a main thread (the UI thread)

Application components in the same process use the same UI thread

User interaction, system callbacks, and lifecycle methods handled on the UI thread

In addition, UI toolkit is not thread-safe

Implications

- Blocking the UI thread hurts application responsiveness
  - Long-running ops should run in background threads
- Don’t access the UI toolkit from a non-UI thread

Improved Solution

Do work on a background thread, but update the UI on the UI Thread

Android provides several methods that are guaranteed to run in the UI Thread, e.g.,

- open fun View.post (action: Runnable!): Boolean
- fun Activity.runOnUiThread(action: Runnable!): Unit

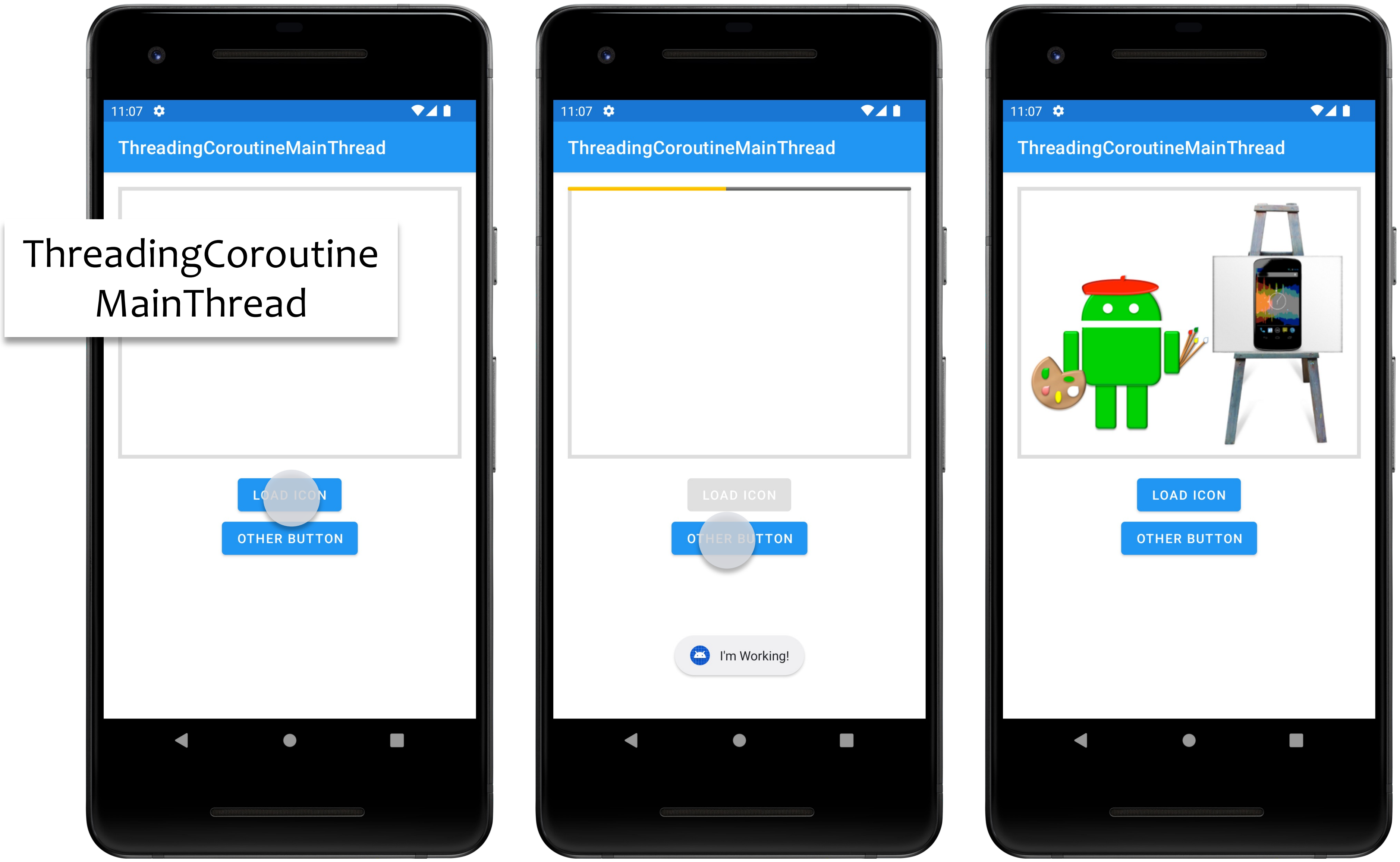
Can also use other approaches to ensure updates happen on UI thread

Kotlin Coroutines

A concurrent, suspendable computation

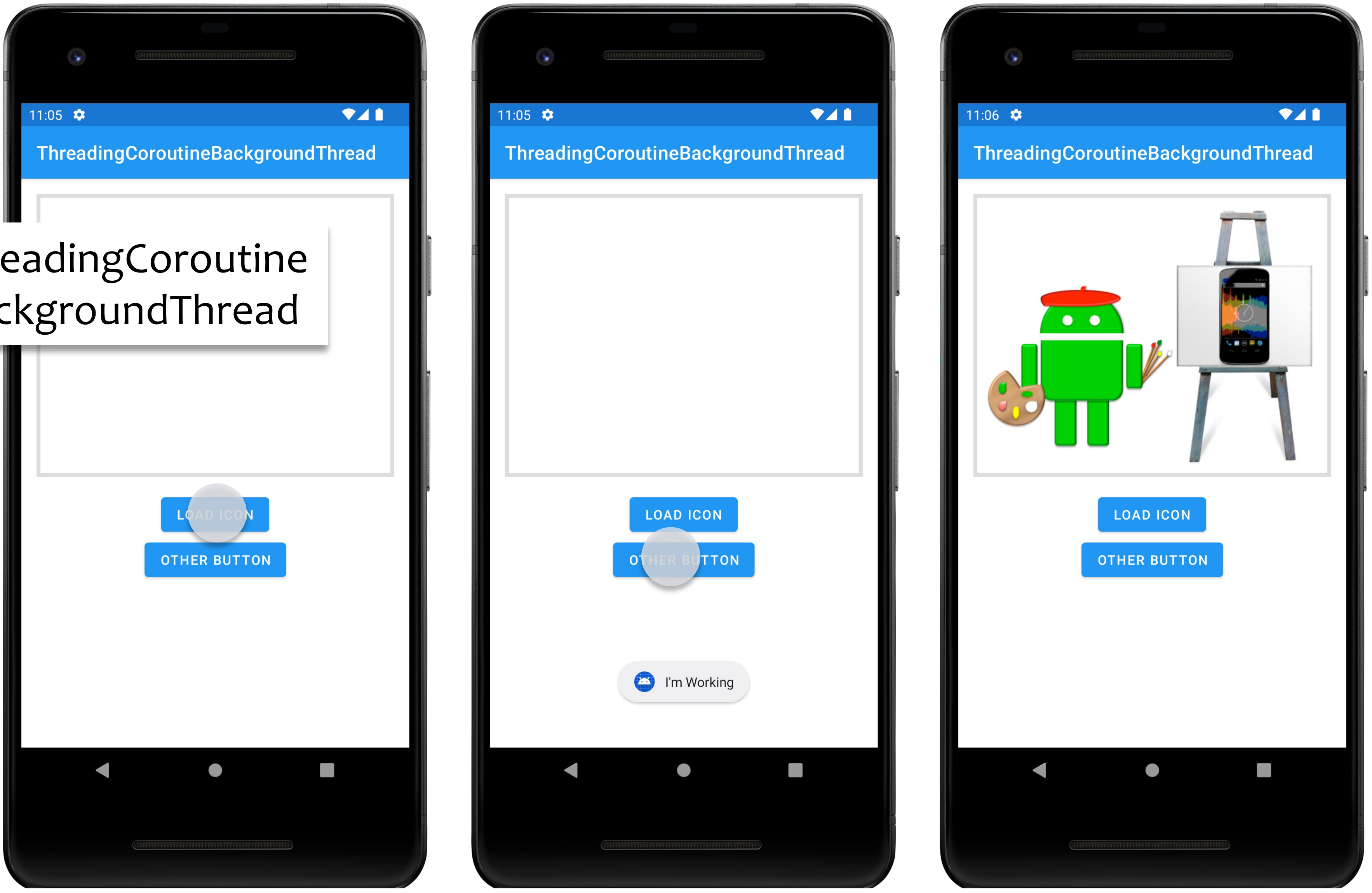
Can be thought of as a light-weight thread, but is not bound to a specific OS thread

See: https://developer.android.com/kotlin/coroutines/

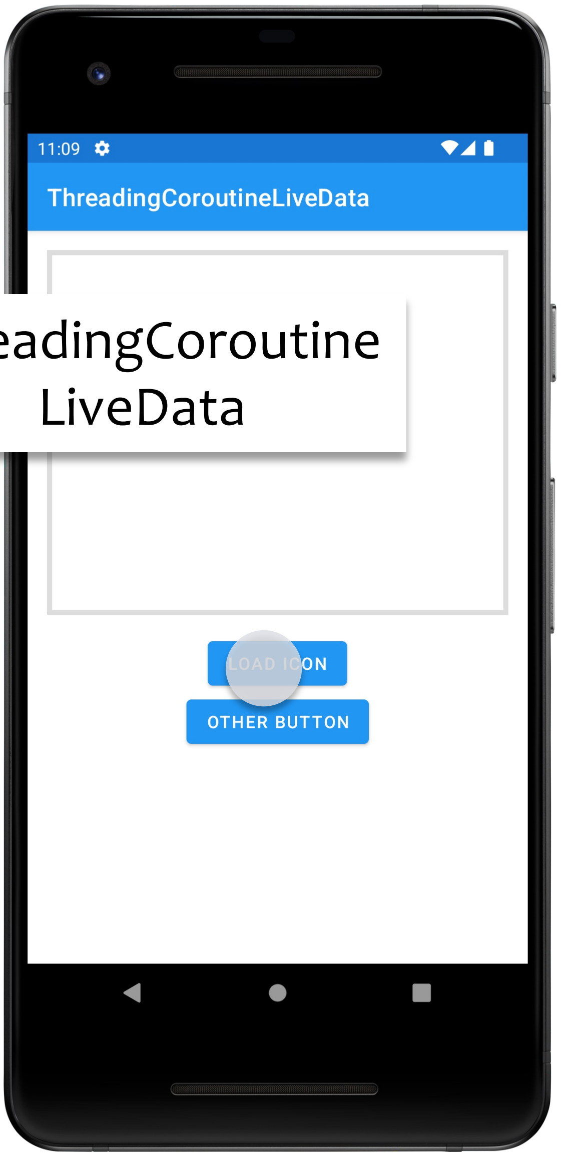


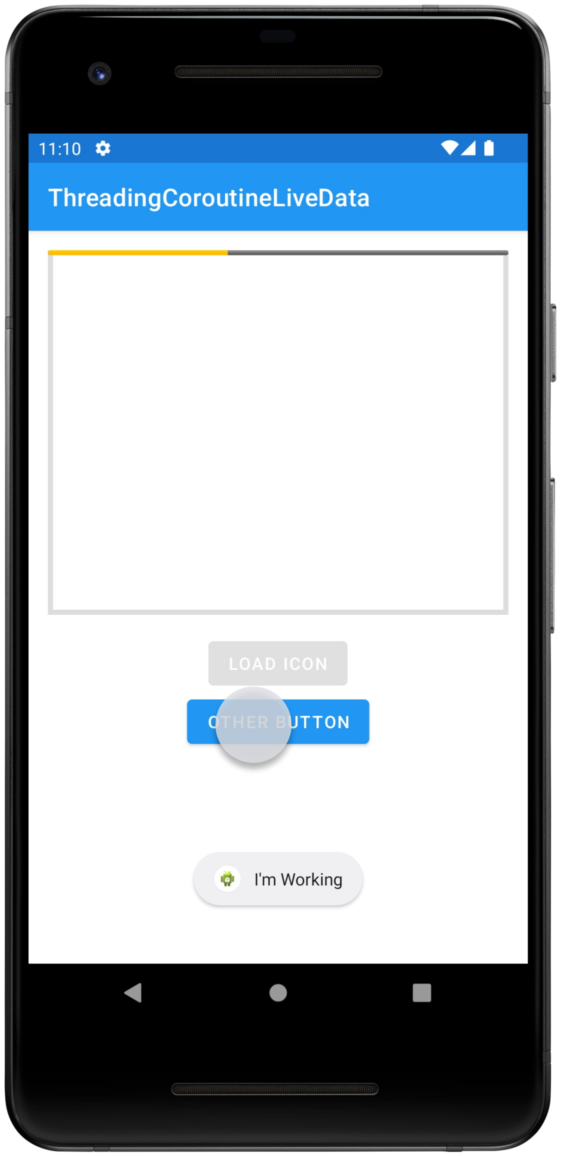
ThreadingCoroutine
MainThread

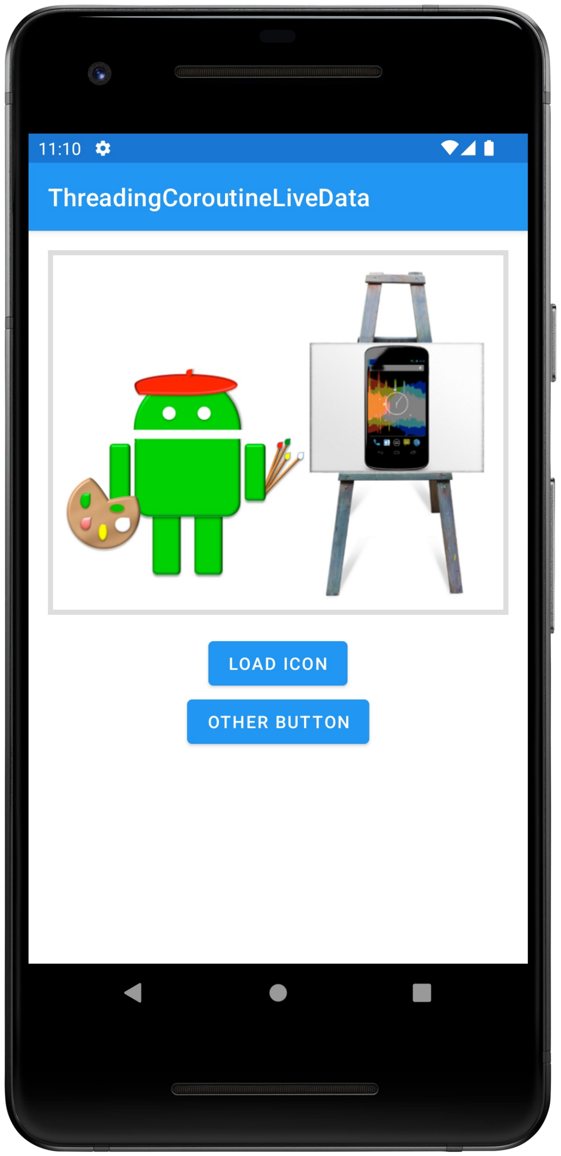
ThreadingCoroutine BackgroundThread


11:05
ThreadingCoroutineBackgroundThread
LOAD ICON
OTHER BUTTON
11:05
ThreadingCoroutineBackgroundThread
LOAD ICON
OTHER BUTTON
I'm Working
11:06
ThreadingCoroutineBackgroundThread
LOAD ICON
OTHER BUTTON

ThreadingCoroutine LiveData







See also:

ThreadingViewPost

ThreadingRunOnUiThread

Handler

Handler lets you enqueue and process Messages and Runnables to/on a Thread’s Message queue

Each Handler is bound to the Thread in which it was created

Main uses

- Schedule Messages and Runnables to be executed at some point in the future
- Enqueue an action to be performed on a different thread

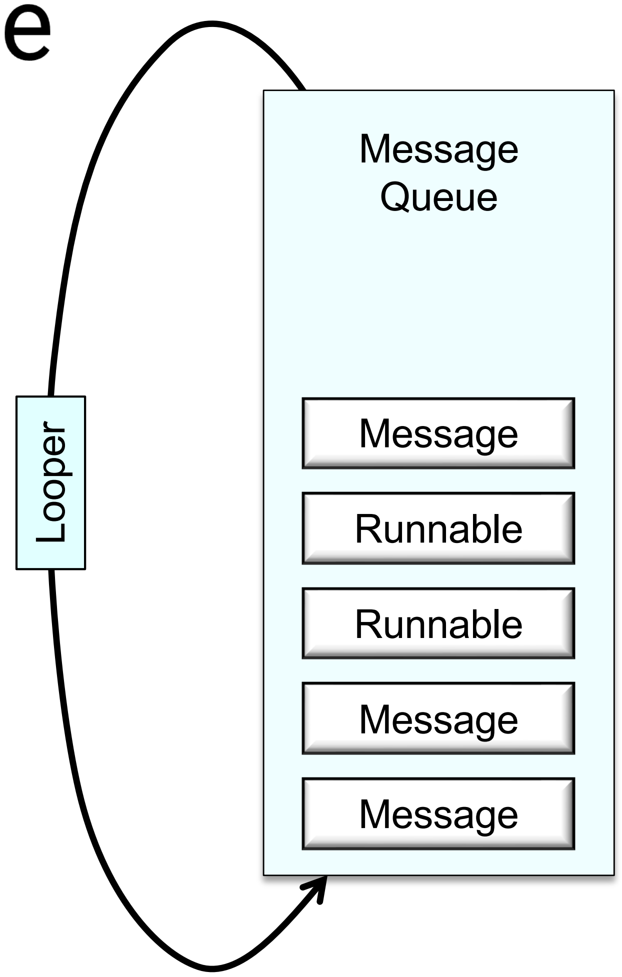
Handler Message Types

- Runnable
  - Contains an instance of the Runnable interface
  - Enqueuer implements response
- Message
  - Can contain a message code, an object & integer arguments
  - Handler implements response

Handler Architecture

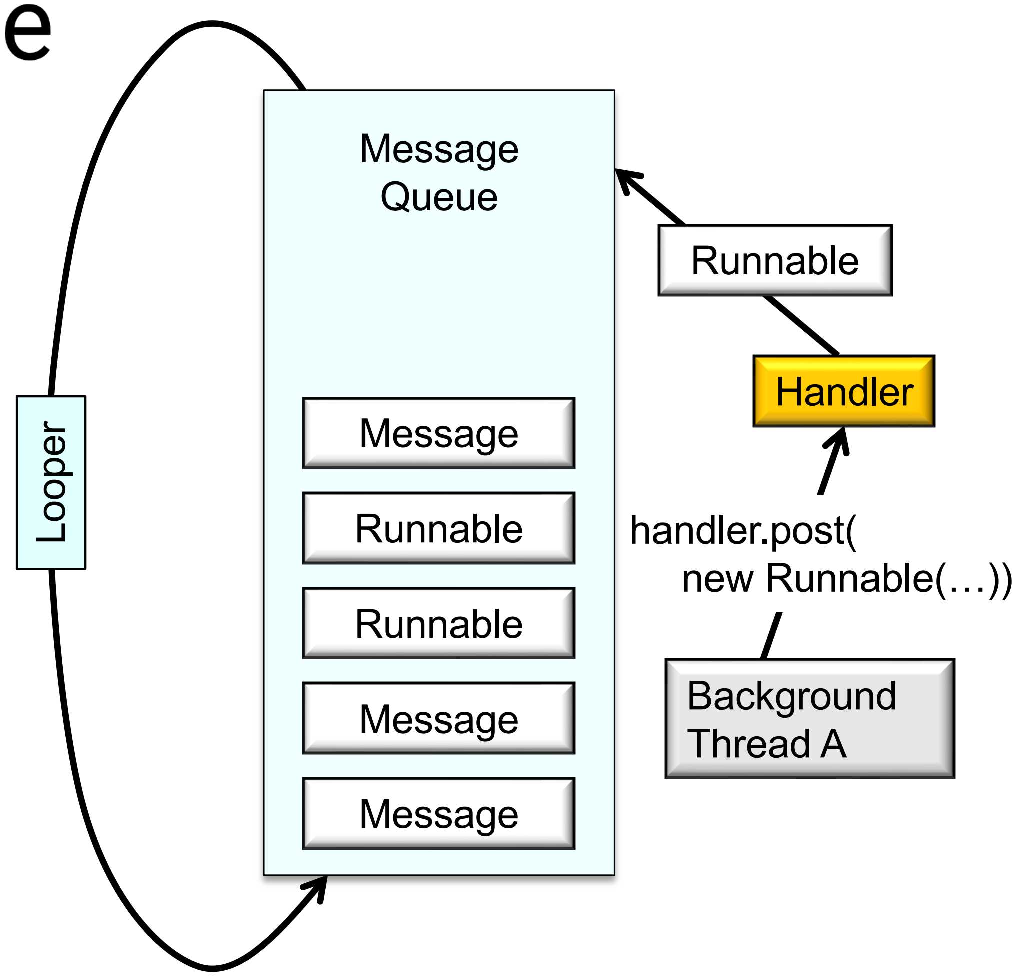
Each Android Thread is associated with a messageQueue and a Looper

A MessageQueue holds Messages and Runnables to be dispatched by the Looper



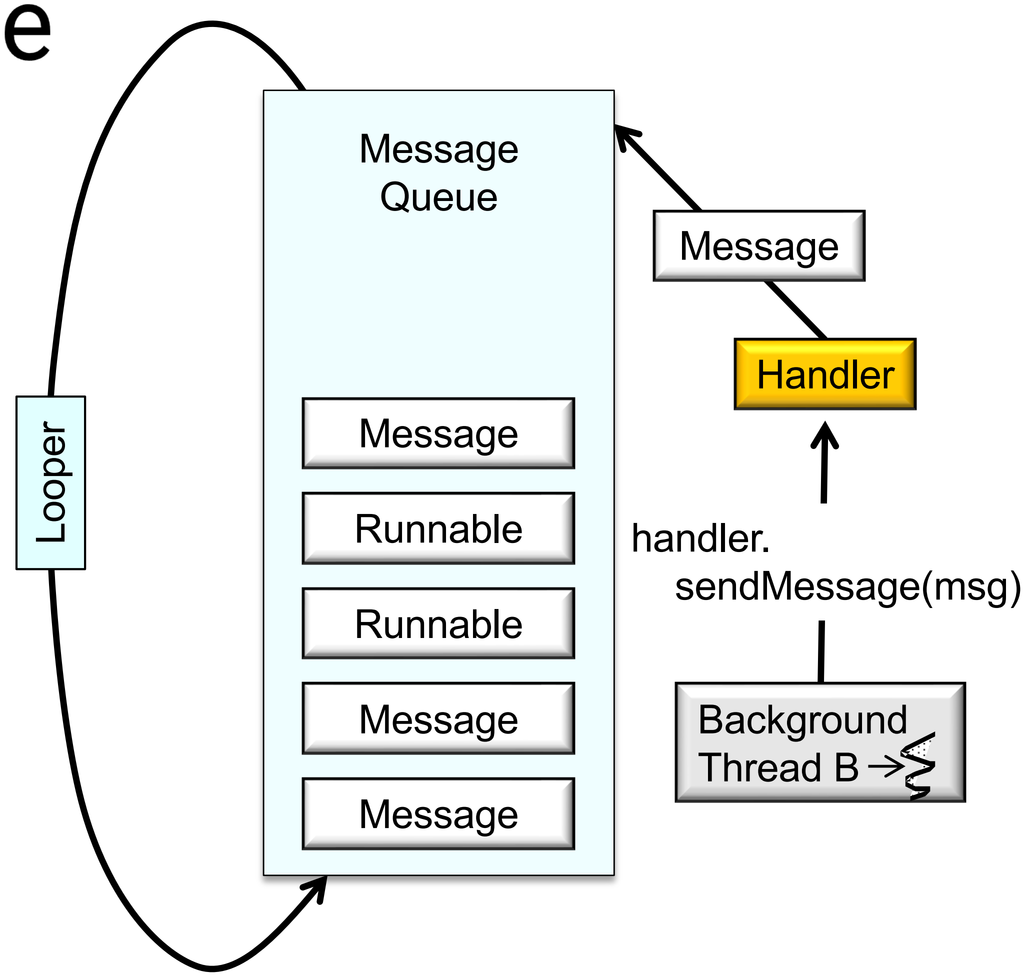
Handler Architecture

Add Runnables to MessageQueue by calling Handler's post() method



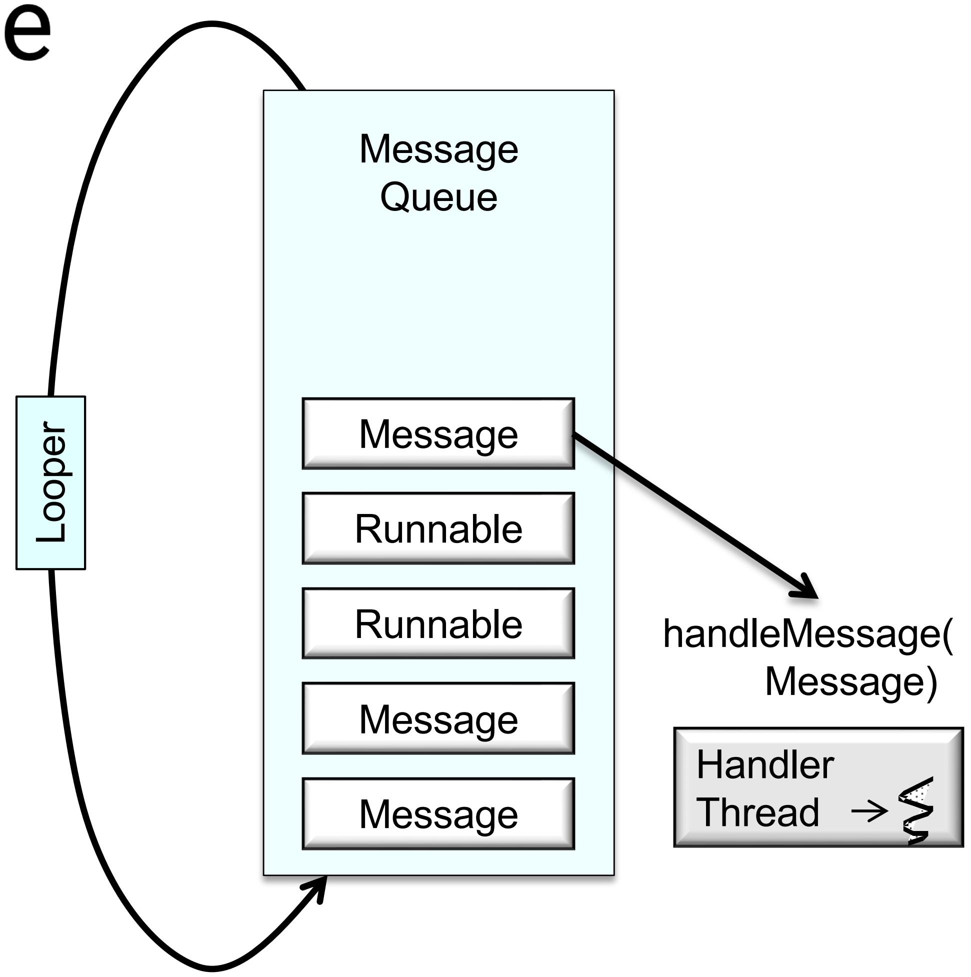
Handler Architecture

Add Messages to MessageQueue by calling Handler's sendMessage() method



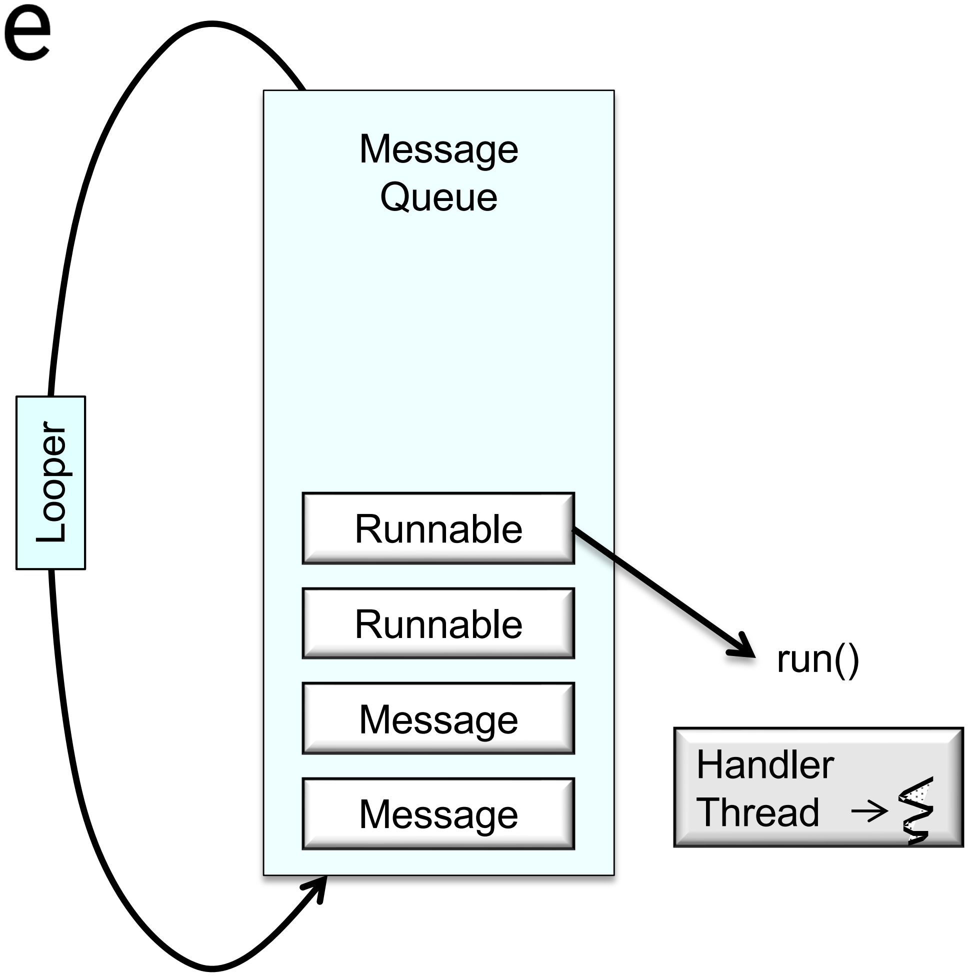
Handler Architecture

Looper dispatches Messages by calling the Handler's handleMessage() method on the Handler's Thread



Handler Architecture

Looper dispatches Runnables by calling their run() method in the Handler's Thread



Handler Methods for Runnables

fun post(r: Runnable): Boolean

Add Runnable to the MessageQueue

fun postAtTime(r: Runnable, uptimeMillis: Long): Boolean

Add Runnable to the MessageQueue. Run at a specific time (based on SystemClock.upTimeMillis())

fun postDelayed(r: Runnable, delayMillis: Long): Boolean

Add Runnable to the message queue. Run after the specified amount of time elapses

Handler Methods for Creating Messages

- Create Message & set Message content
  - Handler.obtainMessage()
  - Message.obtain()
- Message parameters include
  - int arg1, arg2, what
  - Object obj
  - Bundle data
- Many variants. See documentation

Handler Methods for Sending Messages

sendMessage()

- Queue Message now

sendMessageAtFrontOfQueue()

- Insert Message at front of queue

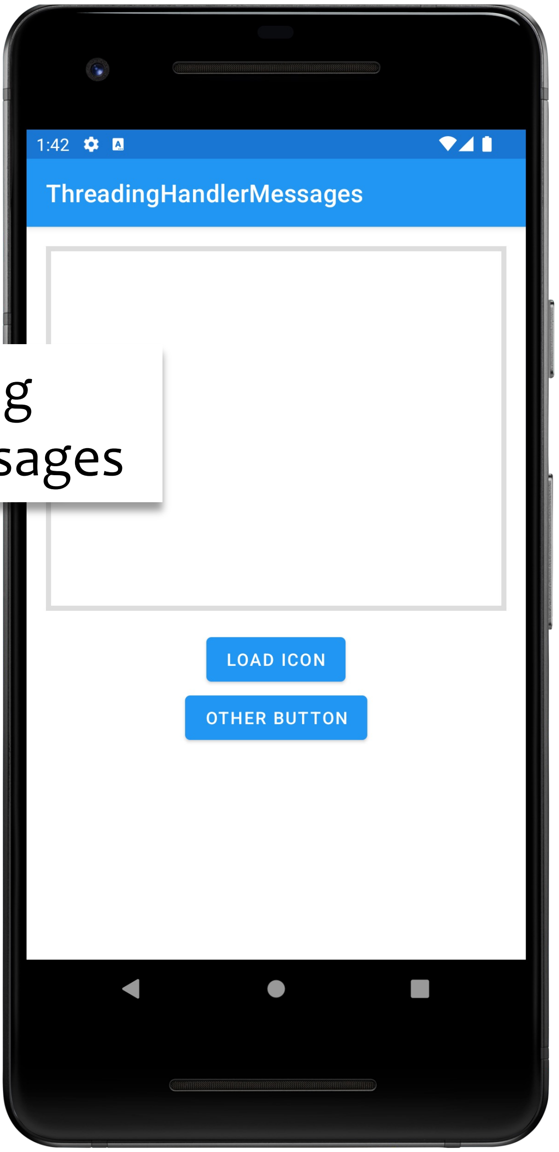
sendMessageAtTime()

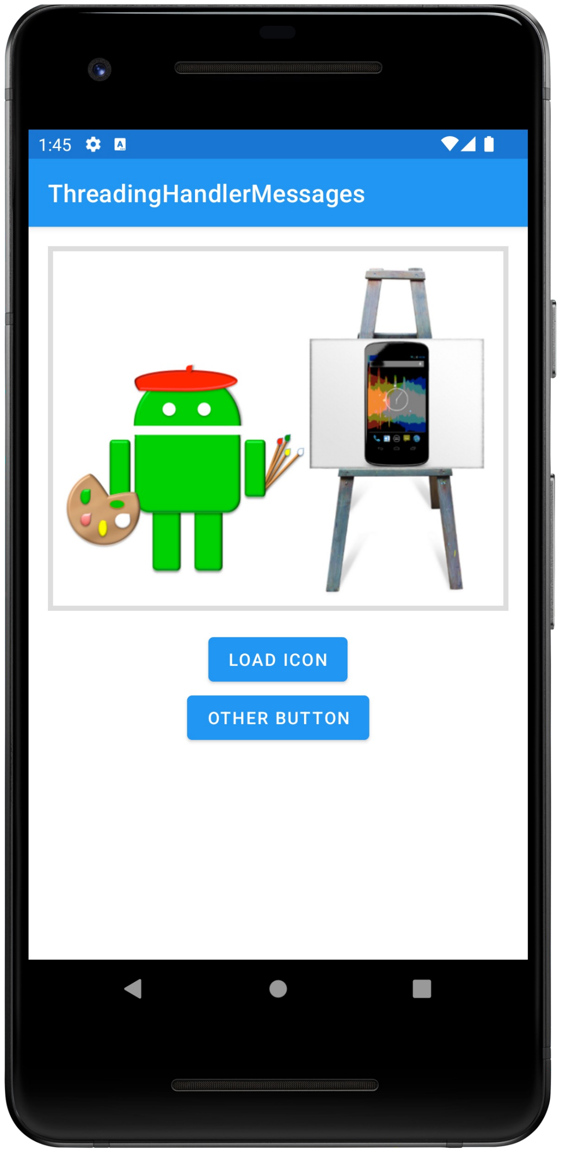
- Queue Message at the stated time

sendMessageDelayed()

- Queue Message after delay

Threading HandlerMessages





Next Time

Networking

Example Applications

ThreadingNoThreading

ThreadingSimple

ThreadingCoroutineMainThread

ThreadingCoroutineBackgroundThread

ThreadingCoroutineLiveData

ThreadingViewPost

ThreadingRunOnUiThread

ThreadingHandlerMessages