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

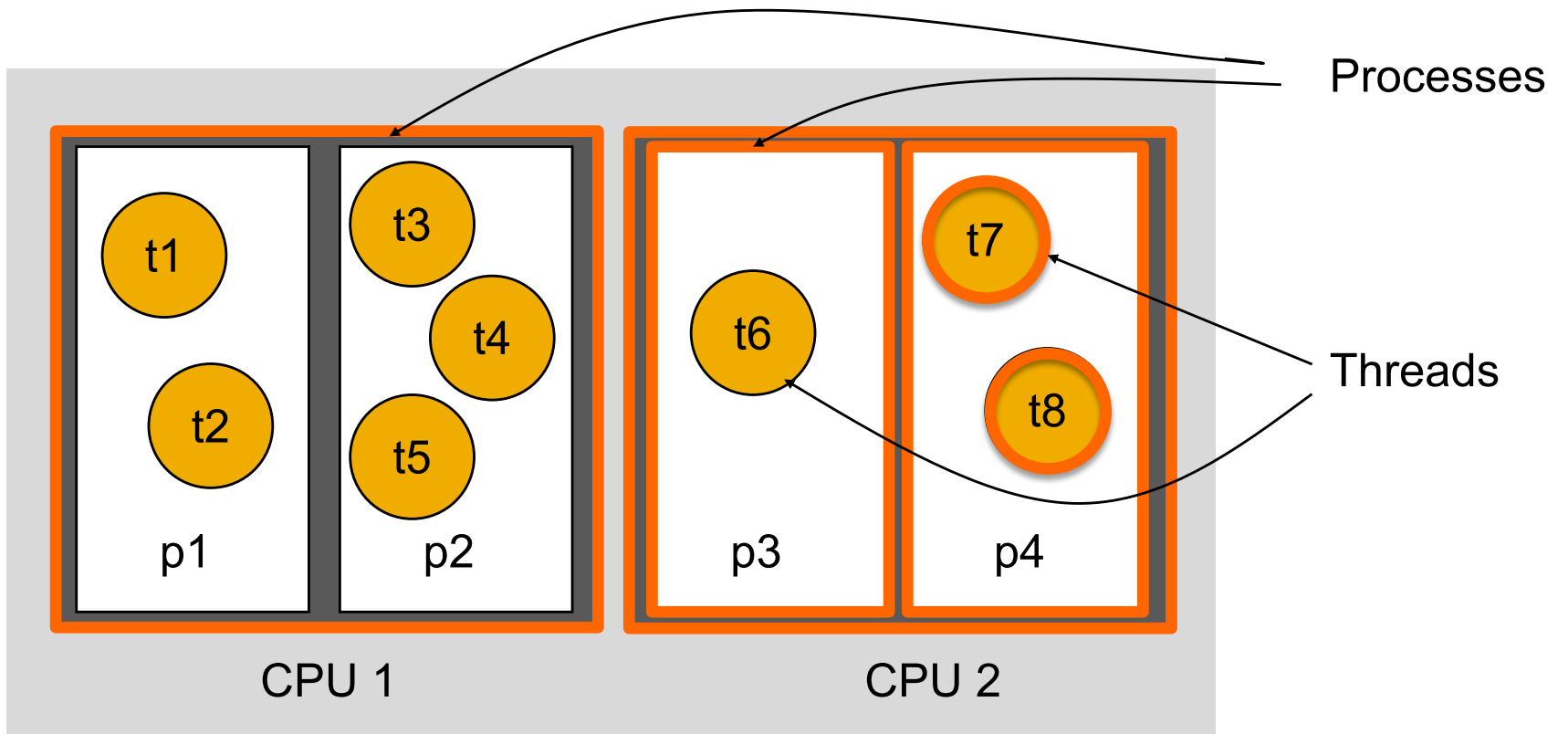# What is a Thread?

Conceptual view

    Parallel computation running in a process

Implementation view

    A program counter and a stack

    Heap and static areas shared with other threads

Processes

t1

t3

t4

t2

t5

t6

t7

t8

Threads

p1

p2

p3

p4

CPU 1

CPU 2

Computing Device

Common Thread Model

Java represents by an Object of type Java.lang.Thread

Threads implement the Runnable interface

    public void run()

See:

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

Some Thread Methods

void start()

    Starts the Thread

void sleep(long time)

    Sleeps for the given period

Some Object Methods

void wait()

Current thread waits until another thread invokes notify() on this object

void notify()

Wakes up a single thread that is waiting on this object
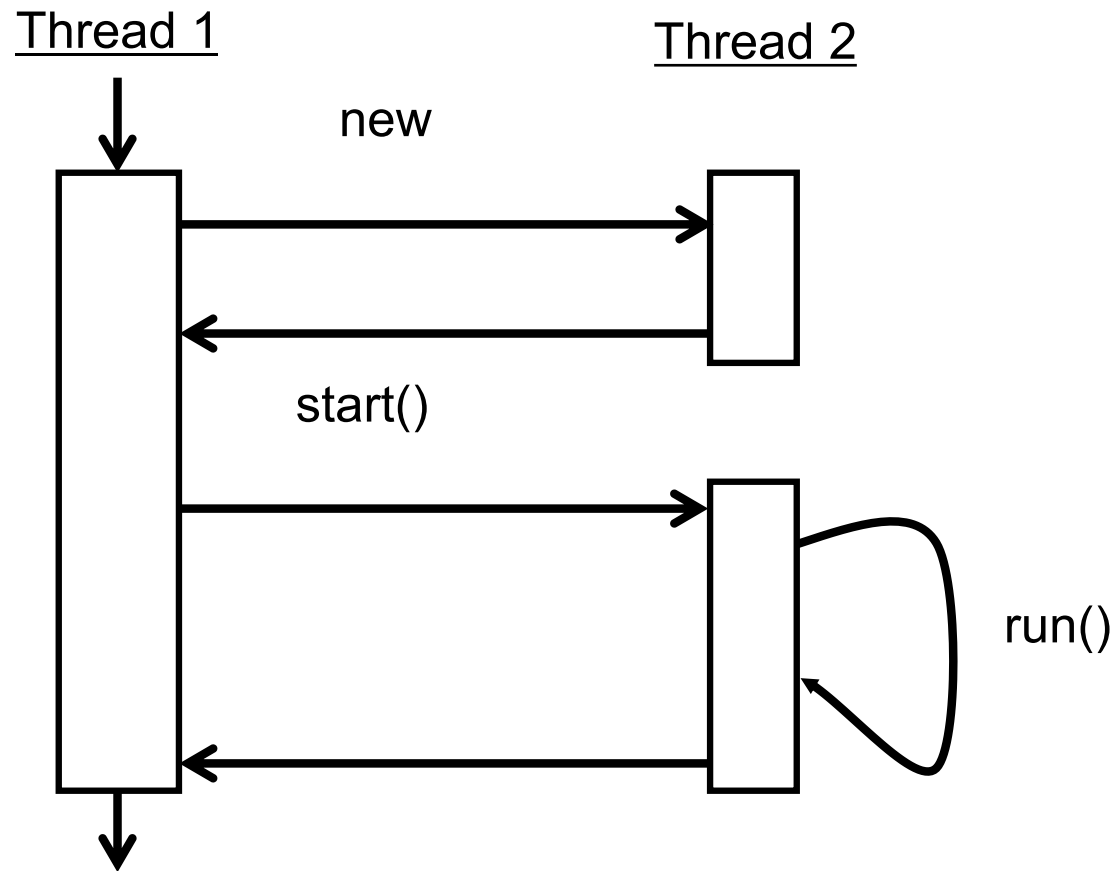
Basic Thread Use Case

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
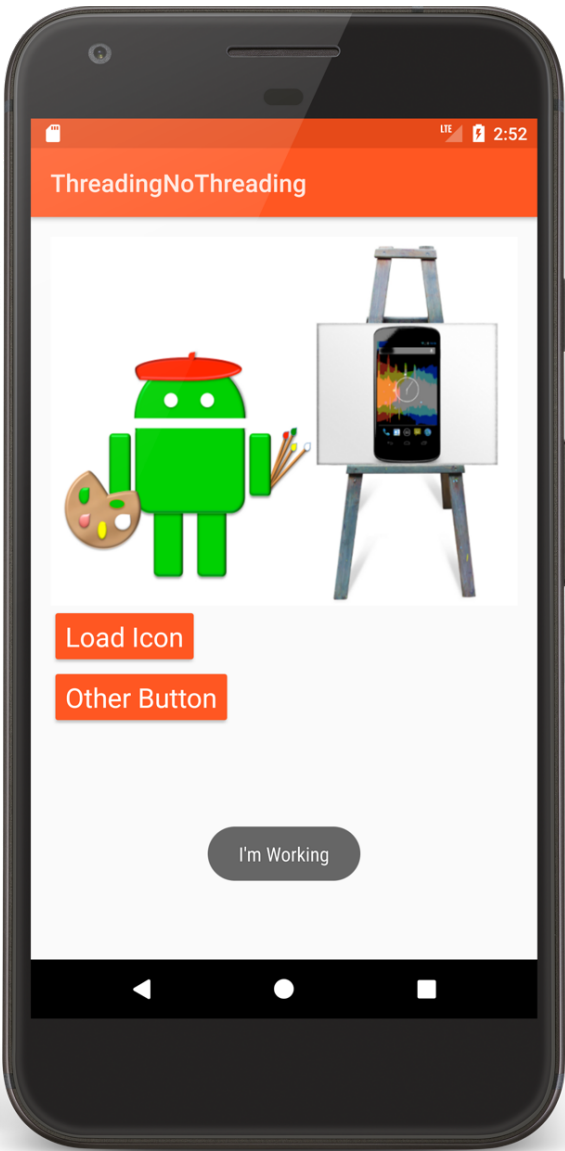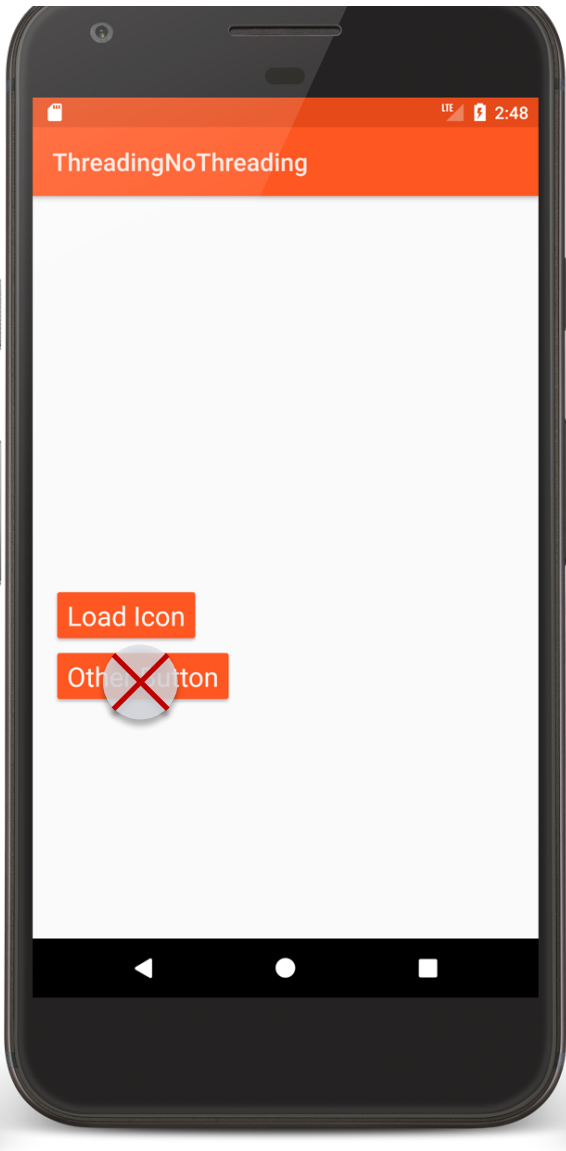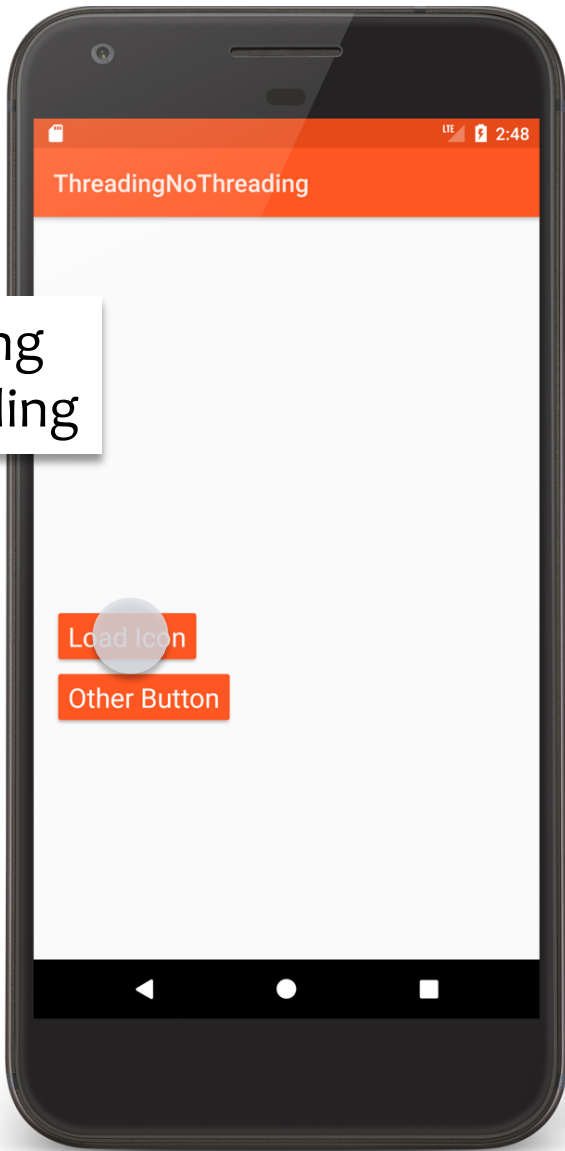
Application displays two buttons

LoadIcon

    Load a bitmap from a resource file & display

    Show loaded bitmap

Other Button

    Display some text

Threading
NoThreading

NoThreadingExample.kt

```kotlin
fun onClickOtherButton(v: View) {
    Toast.makeText(this@NoThreadingExample, "I'm Working",
                                    Toast.LENGTH_SHORT).show()
}

fun onClickLoadButton(view: View) {
    try {
        // Accentuates pretend slow operation
        Thread.sleep(5000)
        mIView.setImageBitmap(
            BitmapFactory.decodeResource(resources, R.drawable.painter))
    } catch (e: InterruptedException) {
        e.printStackTrace()
    }
}
```
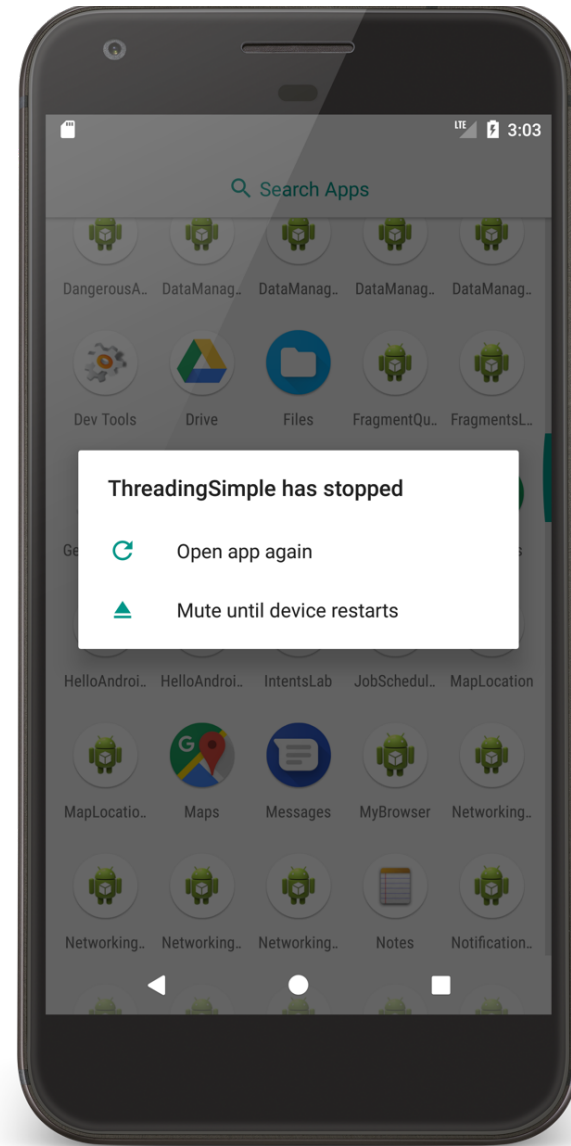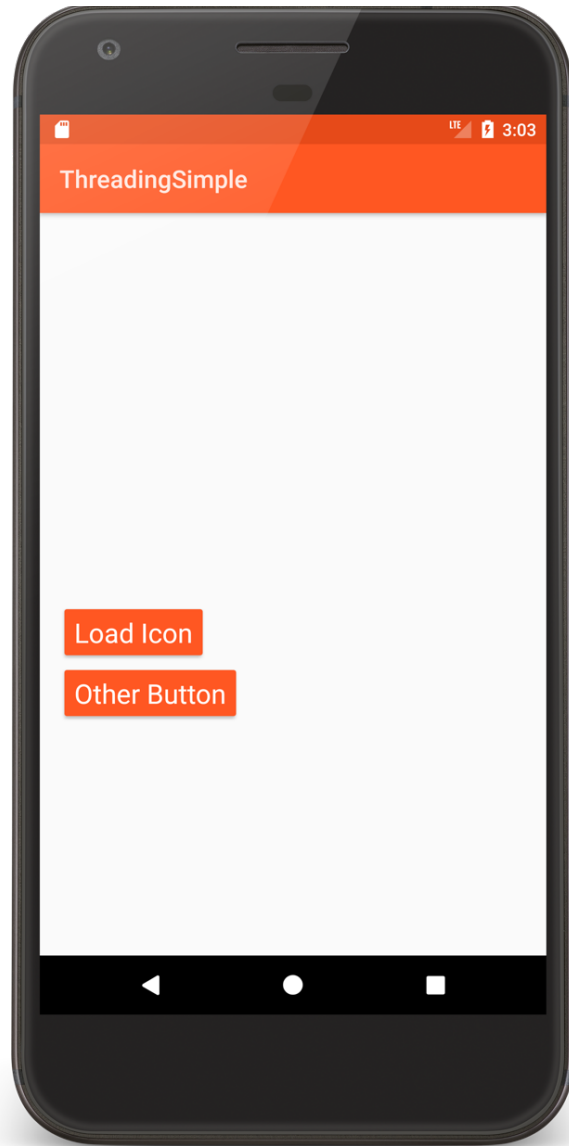
ThreadingSimple

Seemingly obvious, but incorrect, solution:

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

Logcat

Emulator Pixel_XL_API_26 Android 8.0.0, API 26   No Debuggable Processes   Verbose   Regex Show only selected application

```
:03:11.897 18760:18760 D/        ]
tion::get() New Host Connection established 0xb42af800, tid 18760


:03:11.983 18760:18760 W/        ]
ed GLES max version string in extensions: ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_dma_v1
glCreateSyncKHR(1884): error 0x3004 (EGL_BAD_ATTRIBUTE)
aScannerReceiver: action: android.intent.action.MEDIA_SCANNER_SCAN_FILE path: /data/local/tmp/screen.png
loc: Creating ashmem region of size 4096
lying enough data to HAL, expected position 7760265 , only wrote 7760160
lying enough data to HAL, expected position 8066579 , only wrote 7914240
readingsimple E/AndroidRuntime: FATAL EXCEPTION: Thread-4
                                Process: course.examples.threading.threadingsimple, PID: 18730
                                android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
                                    at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7286)
                                    at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1155)
                                    at android.view.View.requestLayout(View.java:21922)
                                    at android.view.View.requestLayout(View.java:21922)
                                    at android.view.View.requestLayout(View.java:21922)
                                    at android.view.View.requestLayout(View.java:21922)
                                    at android.widget.RelativeLayout.requestLayout(RelativeLayout.java:360)
                                    at android.view.View.requestLayout(View.java:21922)
                                    at android.widget.ImageView.setImageDrawable(ImageView.java:570)
                                    at android.widget.ImageView.setImageBitmap(ImageView.java:704)
                                    at course.examples.threading.threadingsimple.SimpleThreadingExample$1.run(SimpleThreadingExample.java:44)
                                    at java.lang.Thread.run(Thread.java:764)
er:    Force finishing activity course.examples.threading.threadingsimple/.SimpleThreadingExample
er: Showing crash dialog for package course.examples.threading.threadingsimple u0
loc: Creating ashmem region of size 4096
loc: Creating ashmem region of size 4096


:03:19.926  2022: 2521 D/        ]
erface::setAsyncMode: set async mode 1
layer name: changing com.google.android.apps.nexuslauncher/com.google.android.apps.nexuslauncher.NexusLauncherActivity to com.google.android.apps.nexuslauncher/com.goo
loc: Creating ashmem region of size 4096
loc: Creating ashmem region of size 4096
searchbox:search D/EGL_emulation: eglMakeCurrent: 0xa3a067a0: ver 2 0 (tinfo 0xa3a03610)
loc: Creating ashmem region of size 4096
```

TODO   6: Logcat   Android Profiler   9: Version Control   Terminal

Event Log   Gradle Console

Gradle sync finished in 880ms (from cached state) (6 minutes ago)   828 chars, 25 line breaks 35:2 LF÷ UTF-8÷ Git: master÷ Context: <no context>

SimpleThreadingExample.kt

```kotlin
fun onClickLoadButton(v: View) {
    GlobalScope.launch (Default){
        delay(mDelay)
        Log.i(TAG, "In onClickLoadButton")

        // This doesn't work in Android
        mIView.setImageBitmap(
            BitmapFactory.decodeResource(resources,R.drawable.painter))
    }
}
```

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

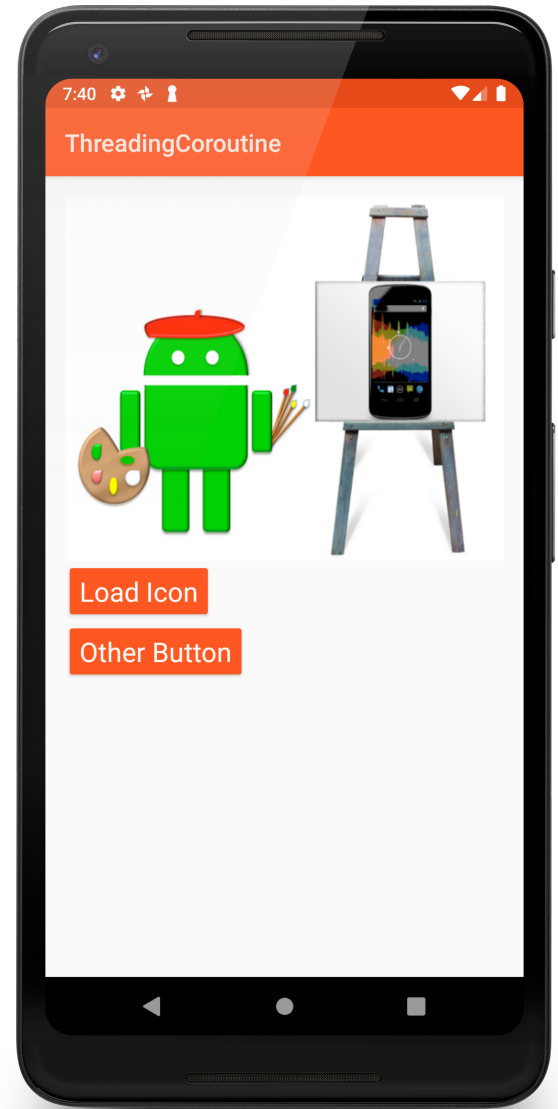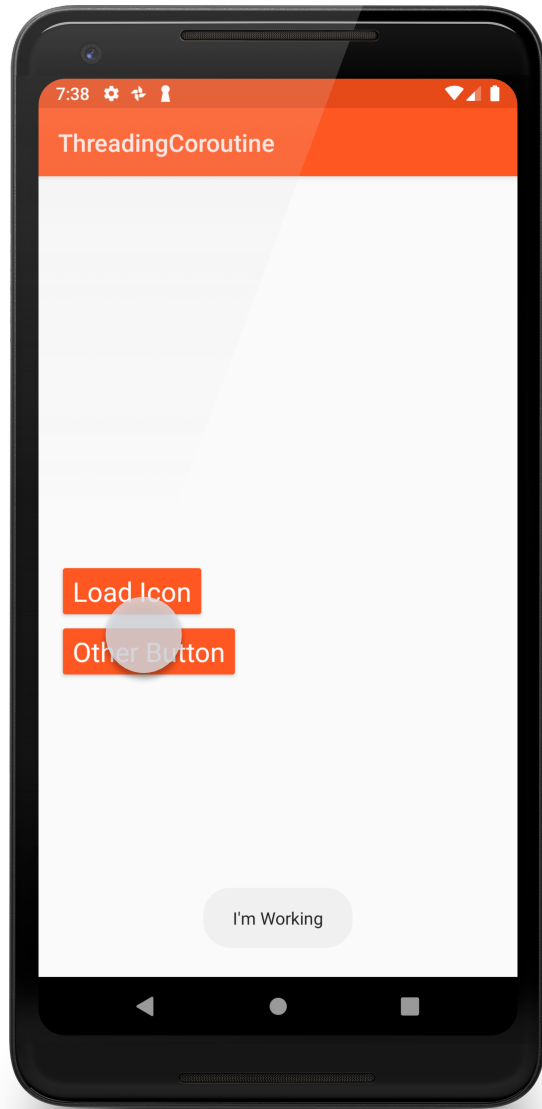Don't access the UI toolkit from a non-UI thread

Improved Solution

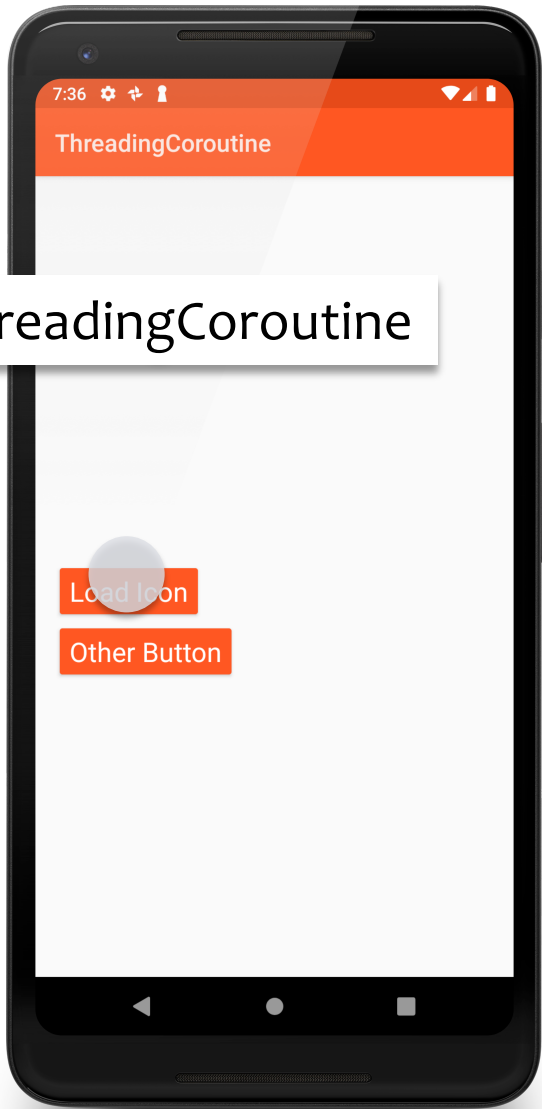Do work on a background thread, but update the UI on the UI Thread

Android provides several methods that are guaranteed to run in the UI Thread

    open fun View.post (action: Runnable!): Boolean

    fun Activity.runOnUiThread(action: Runnable!): Unit

CoroutineExampleActivity.kt

```kotlin
fun onClickLoadButton(v: View) {

    mIView.isEnabled = false

    GlobalScope.launch(Dispatchers.Main) {
        val bitmap = withContext(Dispatchers.Default) {
            // public suspend fun delay(timeMillis: Long)
            delay(mDelay)
            BitmapFactory.decodeResource(resources, R.drawable.painter)
        }
        bitmap?.apply { mIView.setImageBitmap(this) }
    }
}
```

See also:

ThreadingViewPost

ThreadingRunOnUiThread

AsyncTask

Provides a structured way to manage work involving background & UI Threads

Note: AsyncTask is deprecated in API 30

AsyncTask

Background Thread

- Performs work
- Indicates progress

UI Thread

- Does setup
- Publishes intermediate progress
- Uses results

AsyncTask

Generic class

```
class AsyncTask<Params, Progress, Result> {

   …

}
```

Generic type parameters

Params – Type used in background work

Progress – Type used when indicating progress

Result – Type of result

AsyncTask

void onPreExecute()

  Runs on UI Thread

Result doInBackground (Params…params)

  Runs on background Thread

void publishProgress(Progress... values)

  Can be called by doInBackground

  Runs on background Thread

AsyncTask

void onProgressUpdate (Progress... values)
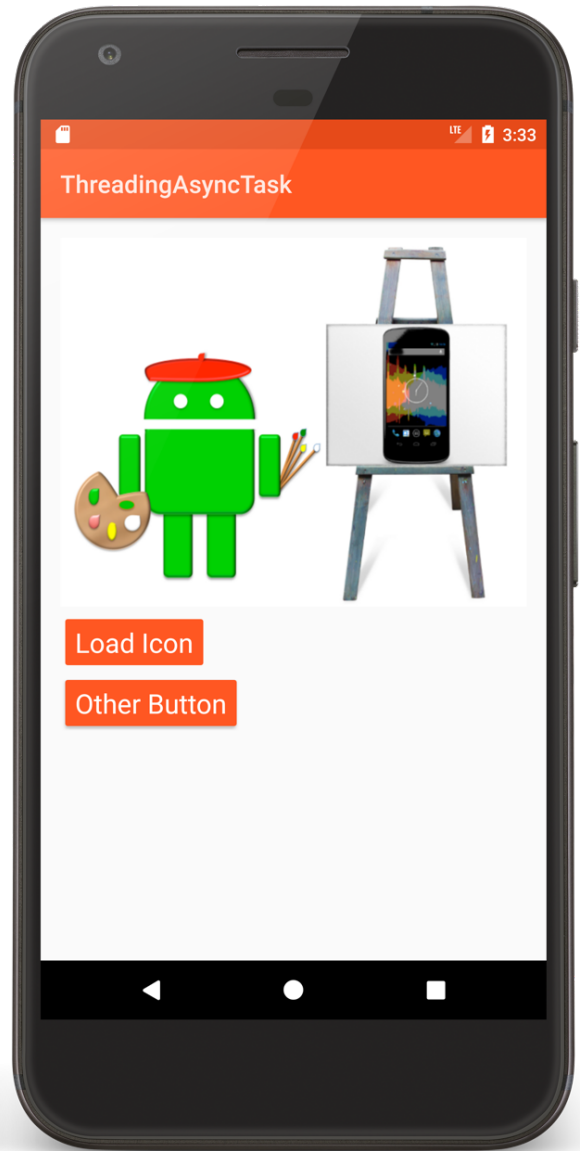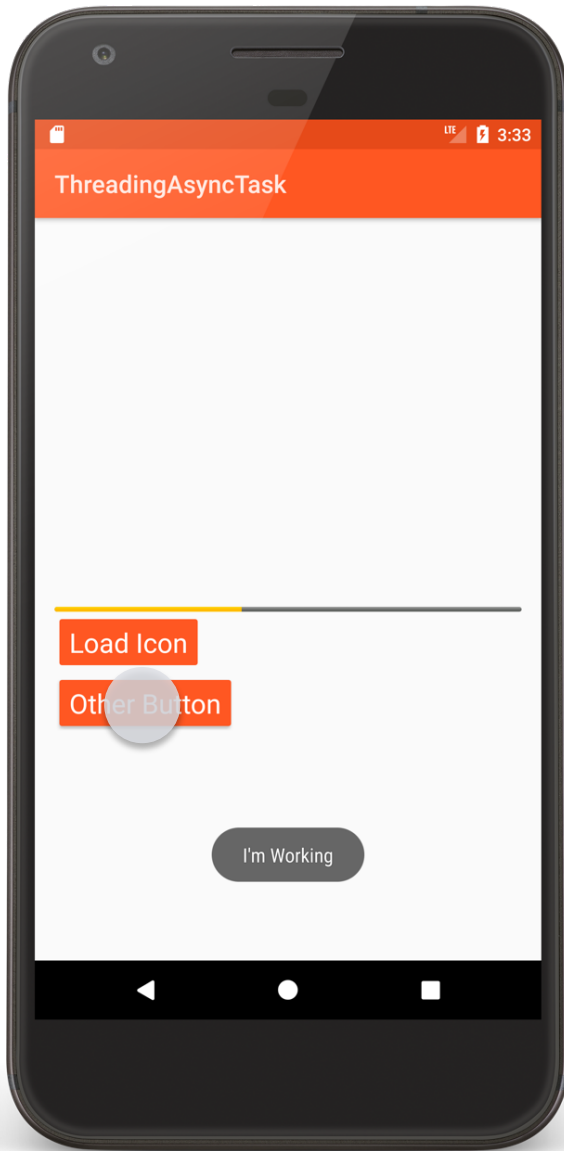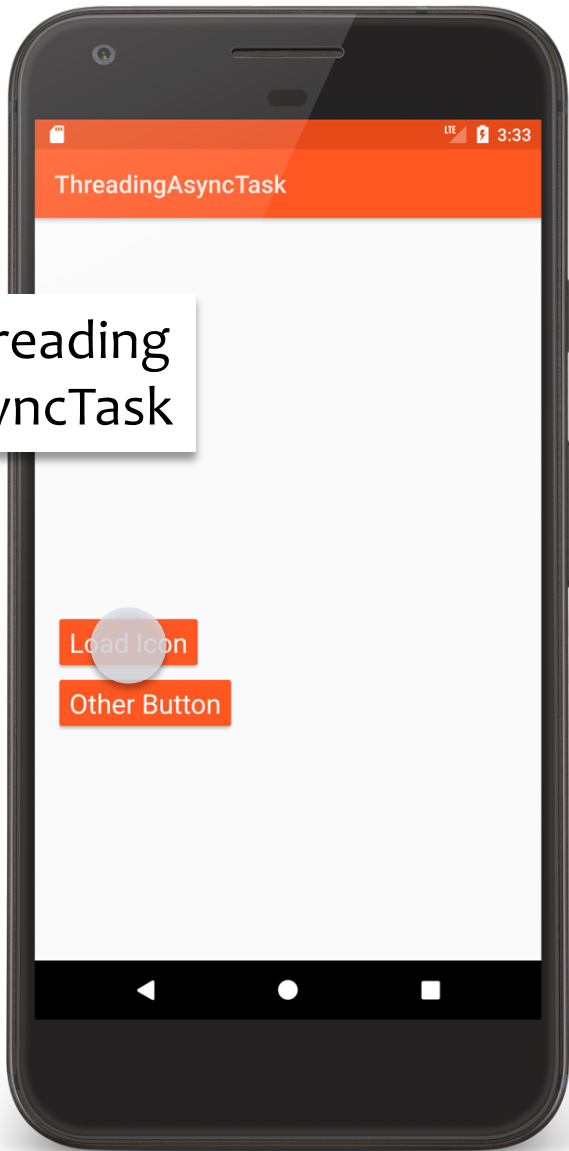
    Invoked in response to publishProgress()

    Runs on UI Thread

void onPostExecute (Result result)

    Runs after doInBackground()

    Runs in UI Thread

Threading
AsyncTask

AsyncTaskActivity.kt

```kotlin
// In AsyncTaskActivity.java
fun onClickLoadButton(v: View) {
    mLoadButton.isEnabled = false
    mAsyncTaskFragment.onButtonPressed()
}


// In AsyncTaskFragment.java
fun onButtonPressed() {
    LoadIconTask(this).execute(PAINTER)
}
```

AsyncTaskFragment.kt

```kotlin
private class LoadIconTask(fragment: AsyncTaskFragment) :
                                    AsyncTask<Int, Int, Bitmap>() {
    // GC can reclaim weakly referenced variables.
    private val mAsyncTaskFragment:
        WeakReference<AsyncTaskFragment> = WeakReference(fragment)

    override fun onPreExecute() {
      mAsyncTaskFragment.get()?.setProgressBarVisibility(ProgressBar.VISIBLE)
    }

    override fun doInBackground(vararg resId: Int?): Bitmap {
    // simulating long-running operation
      for (i in 1..10) {
        sleep()
        publishProgress(i * 10)
      }
      return BitmapFactory.
            decodeResource(mAsyncTaskFragment.get()?.resources, resId[0]!!)
    }
```

AsyncTaskFragment.kt

```kotlin
override fun onProgressUpdate(vararg values: Int?) {
    mAsyncTaskFragment.get()?.setProgress(values[0])
}

override fun onPostExecute(result: Bitmap) {

mAsyncTaskFragment.get()?.setProgressBarVisibility(ProgressBar.INVISIBLE)
    mAsyncTaskFragment.get()?.imageBitmap = result
}
```

AsyncTask Threading Rules

The AsyncTask class must be loaded on the UI thread

The AsyncTask instance must be created on the UI thread

execute(Params...) must be invoked on the UI thread

Do not invoke onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...)

The task can be executed only once. An exception will be thrown on violation

Dealing with Reconfiguration

Remember that Android kills and restarts Activities on reconfiguration

ThreadingAsyncTask gracefully handles reconfiguration

Runs AsyncTask in Headless Fragment

Saves and restores other Activity state

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
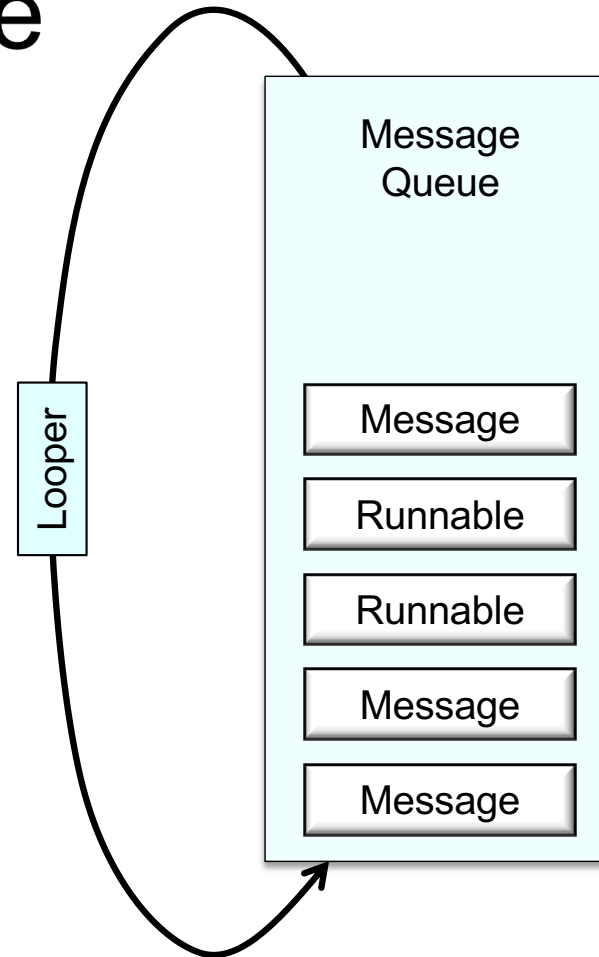
Enqueuer implements response

Message

Can contain a message code, an object & integer arguments
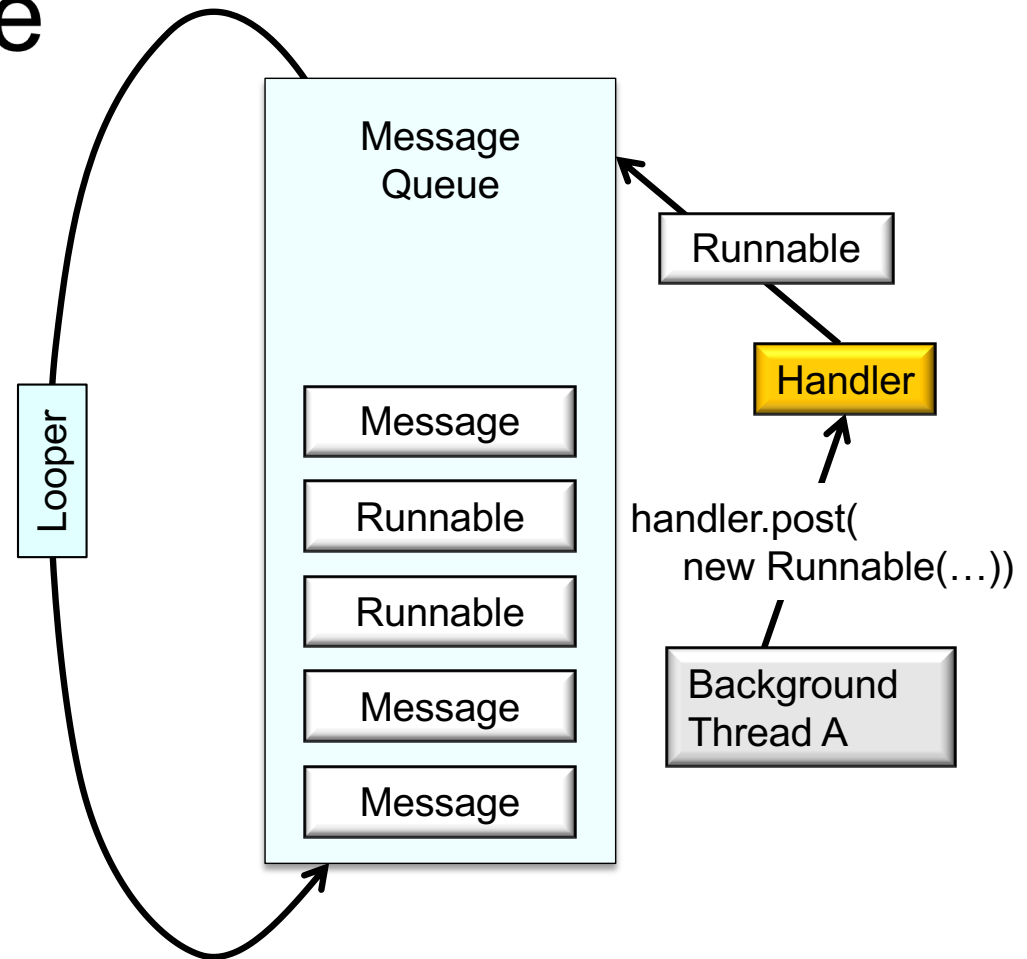
Handler implements response

Handler Architecture

Each Android Thread is associated with a messageQueue and a Looper

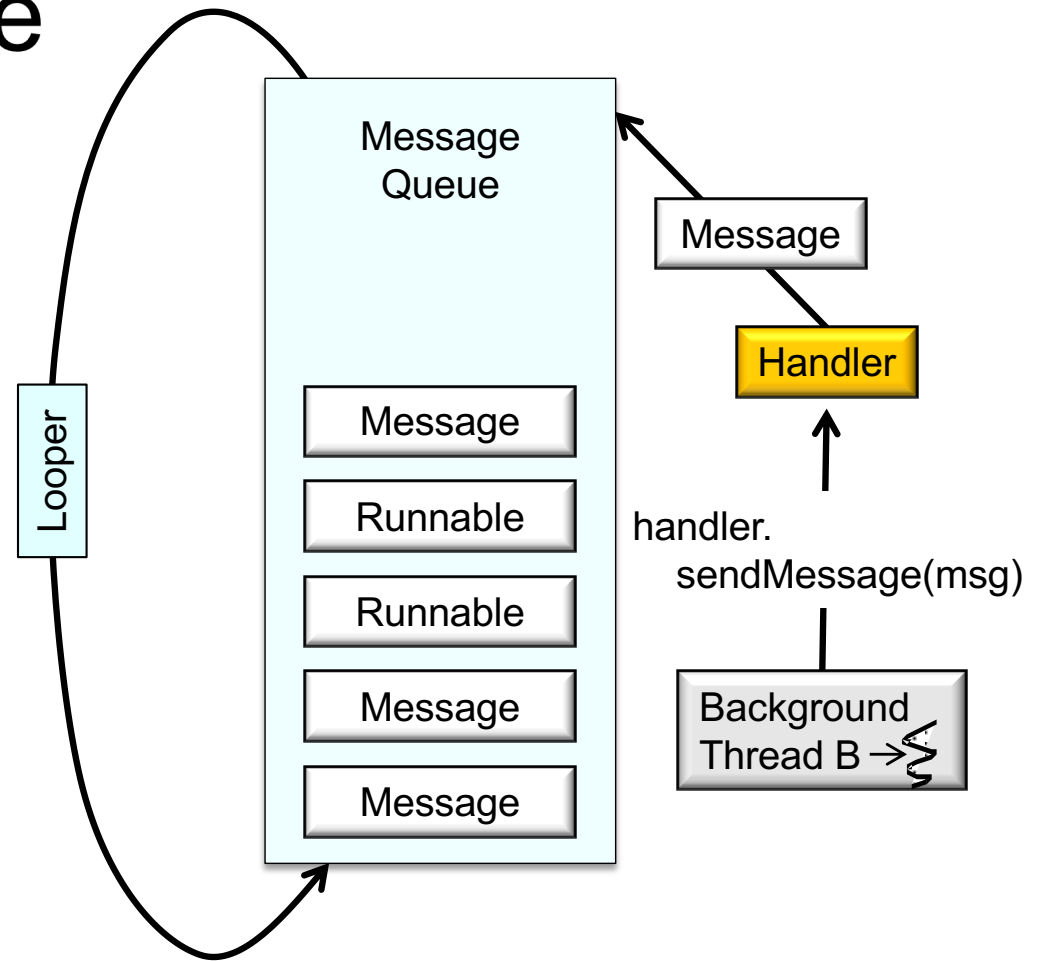A MessageQueue holds Messages and Runnables to be dispatched by the Looper

Looper

Message Queue

| Message |

| Runnable |

| Runnable |

| Message |

| Message |

Handler Architecture

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

Looper

Message Queue

| Message |
| Runnable |
| Runnable |
| Message |
| Message |

Runnable

Handler

handler.post(
new Runnable(…))

Background
Thread A

Handler Architecture

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

Message Queue

Message

Runnable

Runnable

Message

Message

Looper

Message

Handler

handler.
sendMessage(msg)

Background
Thread B →

Handler Architecture

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

Looper

Message
Queue

Message

Runnable

Runnable

Message

Message

handleMessage(
Message)
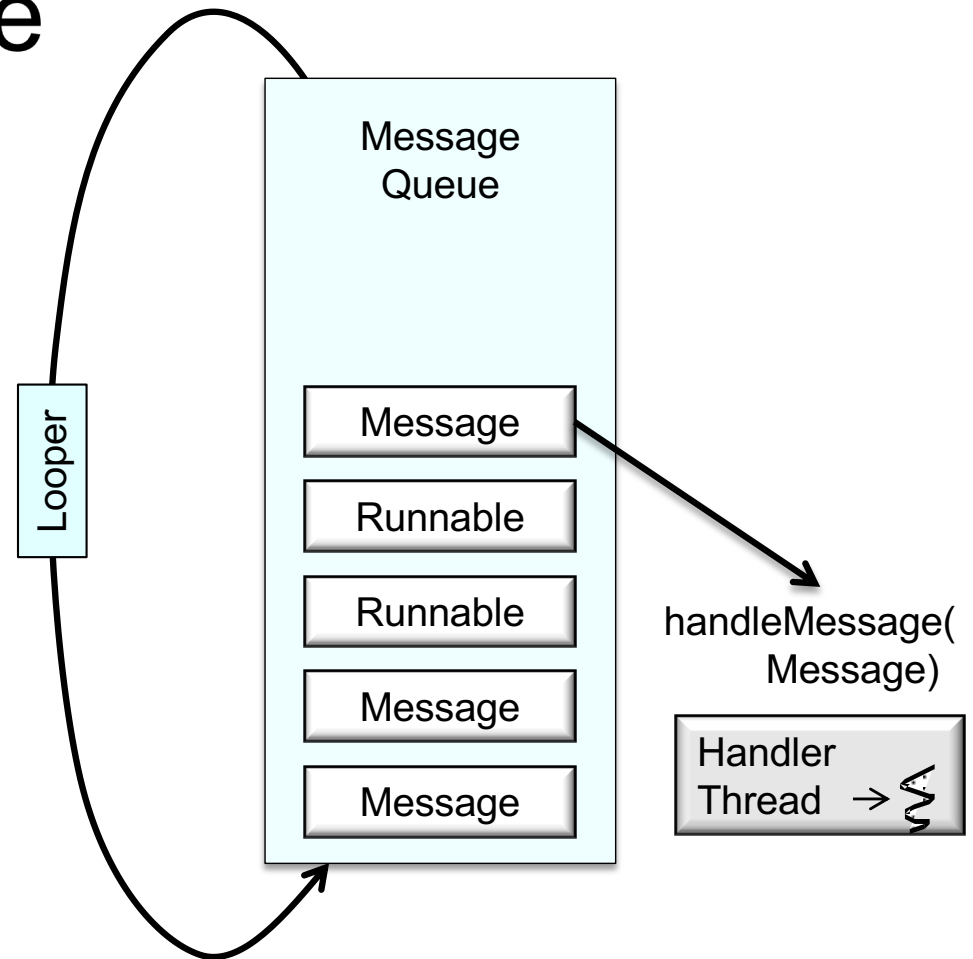
Handler
Thread →

Handler Architecture

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

Looper

Message
Queue

Runnable

Runnable

Message

Message

run()
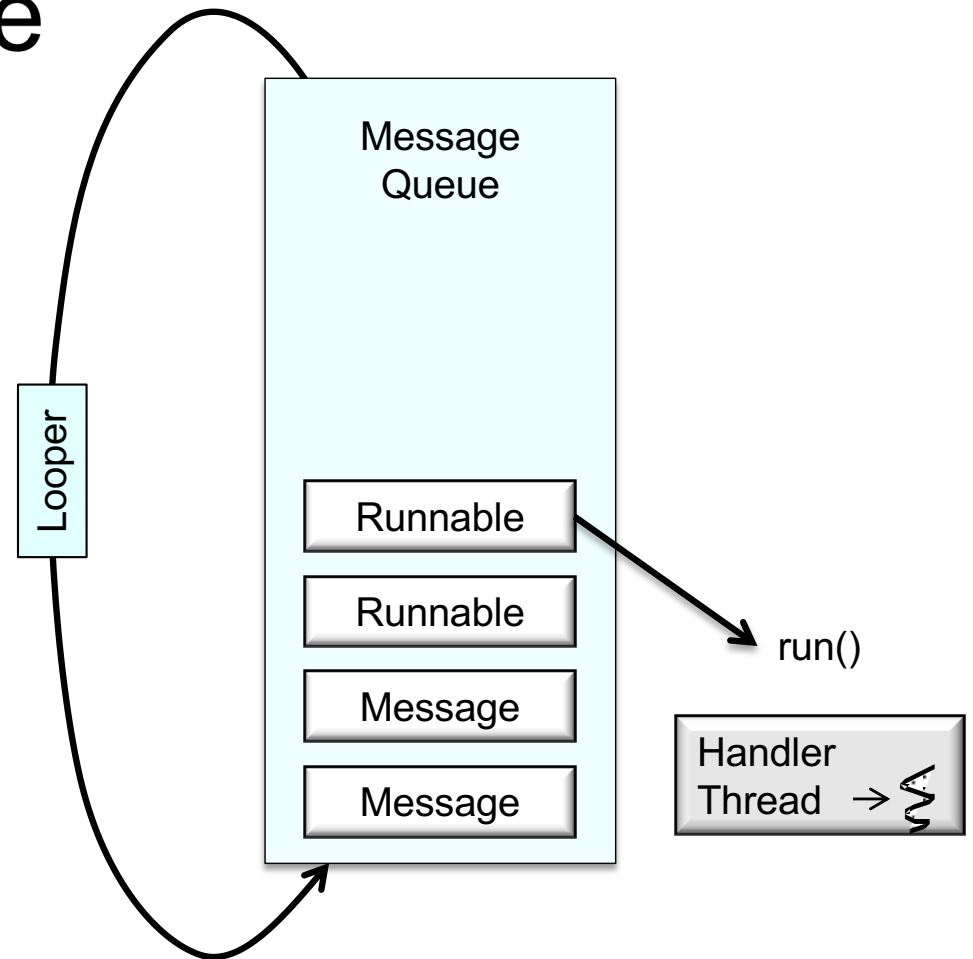
Handler
Thread →

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()
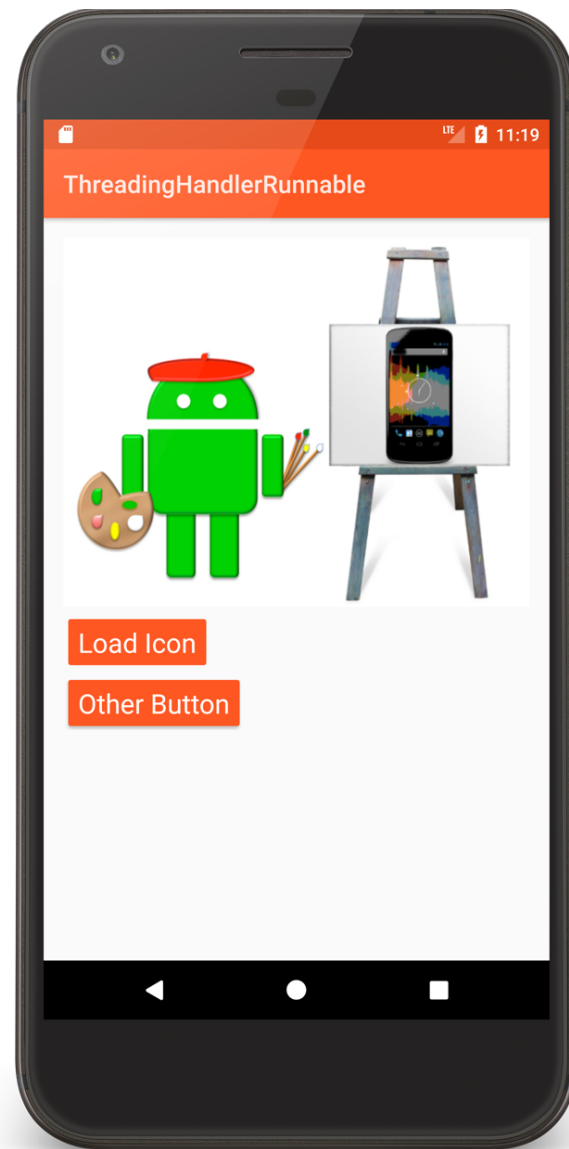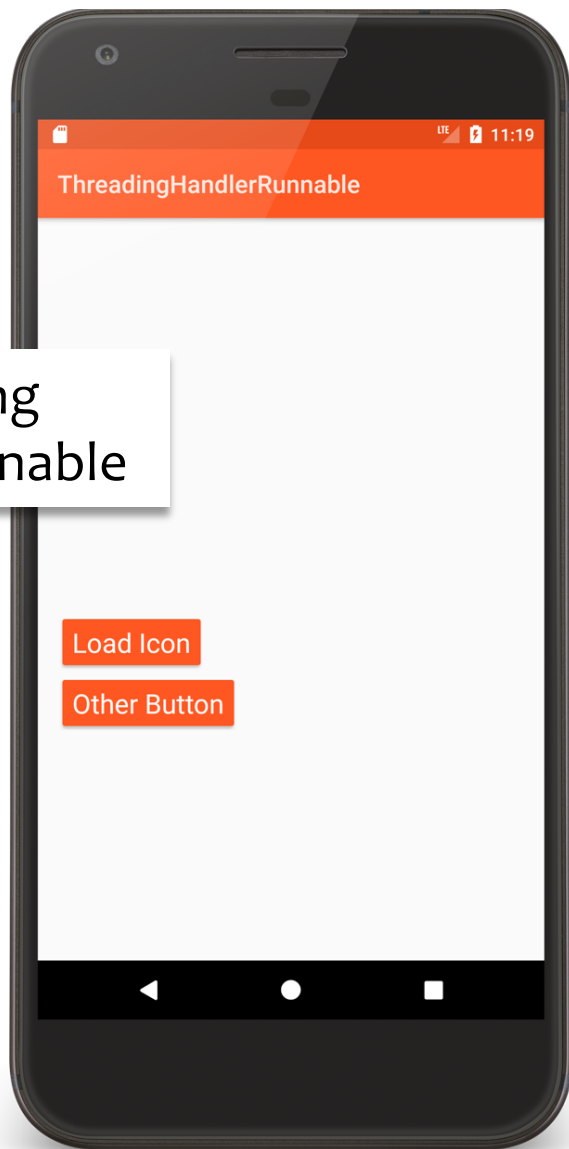
    Insert Message at front of queue

sendMessageAtTime()

    Queue Message at the stated time

sendMessageDelayed()

    Queue Message after delay

Threading HandlerRunnable

HandlerRunnableActivity.kt

```kotlin
fun onClickLoadButton(v: View) {
    v.isEnabled = false
    mLoadIconTask = LoadIconTask(applicationContext)
        .setImageView(mImageView)
        .setProgressBar(mProgressBar)
    mLoadIconTask.start()
}
```
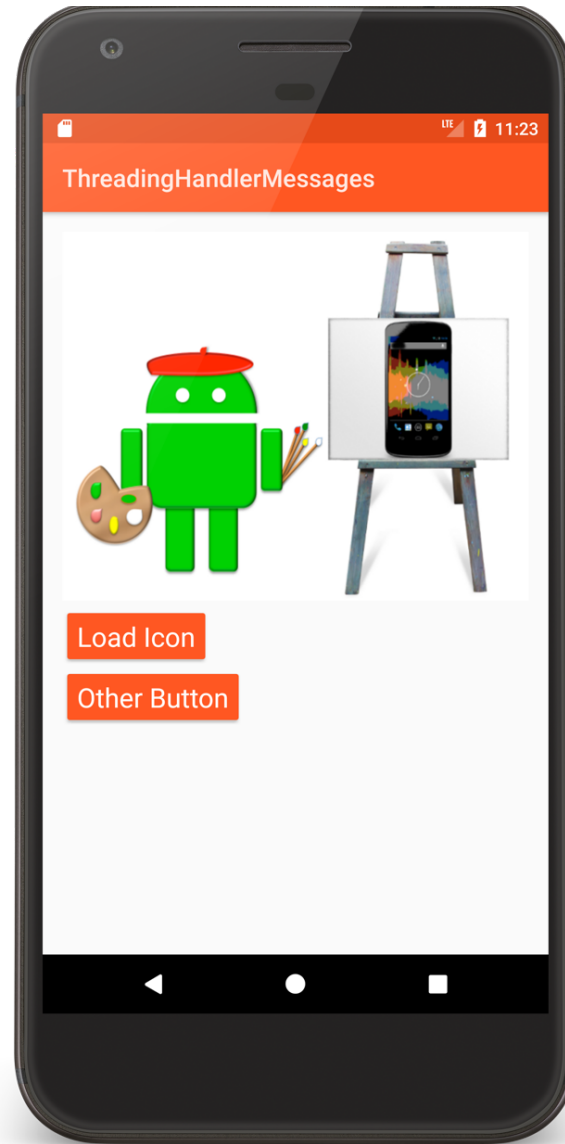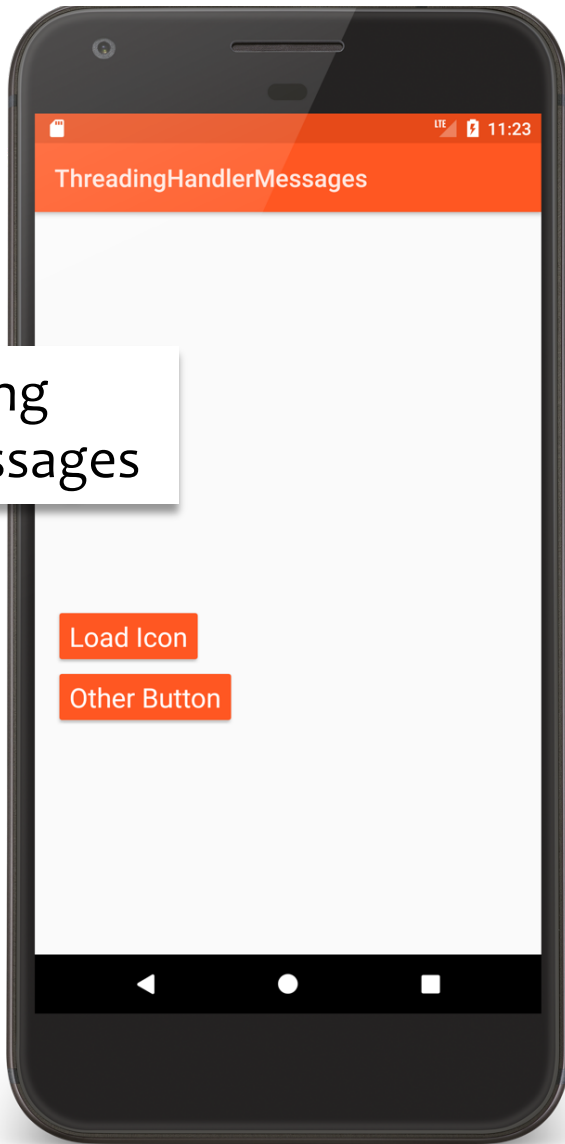
LoadIconTask.kt

```kotlin
class LoadIconTask internal constructor(
                      private val mAppContext: Context):Thread() {
    private val mHandler: Handler = Handler()
    …
    override fun run() {
        mHandler.post { mProgressBar?.visibility = ProgressBar.VISIBLE }

        // Simulating long-running operation
        for (i in 1..10) {
            sleep()
            mHandler.post { mProgressBar?.progress = i * 10 }
        }
```

LoadIconTask.kt

```kotlin
mHandler.post {
  mImageView?.setImageBitmap(
    BitmapFactory.decodeResource(mAppContext.resources, mBitmapResID))
}

  mHandler.post { mProgressBar?.visibility = ProgressBar.INVISIBLE }
}
…
}
```

Threading
HandlerMessages

LoadIconTask.kt

```kotlin
class LoadIconTask internal constructor(
                        private val mContext: Context) : Thread() {
    private val mHandler = UIHandler(Looper.getMainLooper())
    override fun run() {
      var msg = mHandler.obtainMessage(
        HandlerMessagesActivity.SET_PROGRESS_BAR_VISIBILITY,
                                        ProgressBar.VISIBLE)
      mHandler.sendMessage(msg)
      val mResId = R.drawable.painter
      val tmp = BitmapFactory.decodeResource(mContext.resources, mResId)
      for (i in 1..10) {
          sleep()
          msg = mHandler.obtainMessage(
                      HandlerMessagesActivity.PROGRESS_UPDATE, i * 10)
          mHandler.sendMessage(msg)
      }
```

LoadIconTask.kt

```kotlin
    msg = mHandler.obtainMessage(HandlerMessagesActivity.SET_BITMAP, tmp)
    mHandler.sendMessage(msg)

    msg = mHandler.obtainMessage(
        HandlerMessagesActivity.SET_PROGRESS_BAR_VISIBILITY,
                                        ProgressBar.INVISIBLE)

      mHandler.sendMessage(msg)
    }
  }
```

LoadIconTask.kt

```kotlin
private class (mainLooper: Looper) : Handler(mainLooper){
    private var mImageView: ImageView? = null
    private var mProgressBar: ProgressBar? = null

    override fun handleMessage(msg: Message) {
      when (msg.what) {
        HandlerMessagesActivity.SET_PROGRESS_BAR_VISIBILITY -> {
            mProgressBar?.visibility = msg.obj as Int
        }
        HandlerMessagesActivity.PROGRESS_UPDATE -> {
            mProgressBar?.progress = msg.obj as Int
        }
        HandlerMessagesActivity.SET_BITMAP -> {
            mImageView?.setImageBitmap(msg.obj as Bitmap)
        }
      }
    }

    fun setImageView(mImageView: ImageView) {
        this.mImageView = mImageView
    }

    fun setProgressBar(mProgressBar: ProgressBar) {
        this.mProgressBar = mProgressBar
    }
  }
}
```

Next Time

Networking

Example Applications

ThreadingNoThreading

ThreadingSimple

ThreadingCoroutine

ThreadingViewPost

ThreadingRunOnUiThread

ThreadingAsyncTask

ThreadingHandlerRunnable

ThreadingHandlerMessages