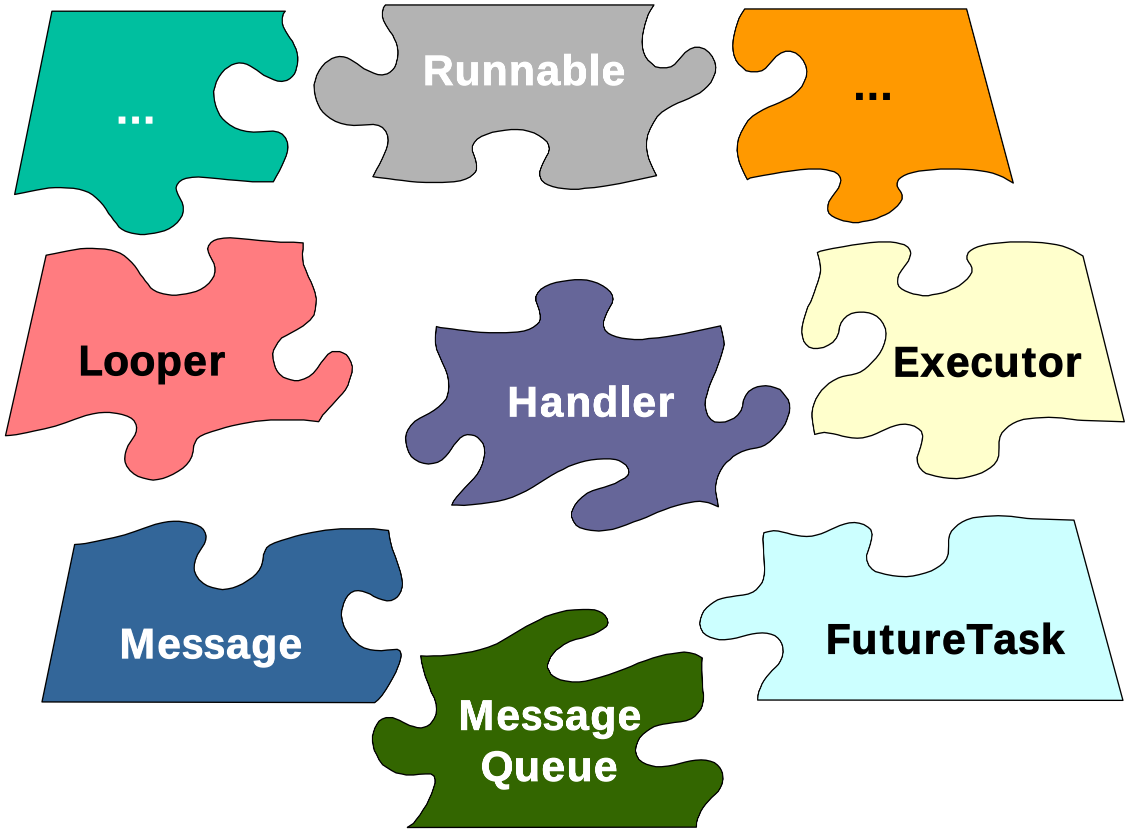
# Android Concurrency: Overview of Android Concurrency Frameworks

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

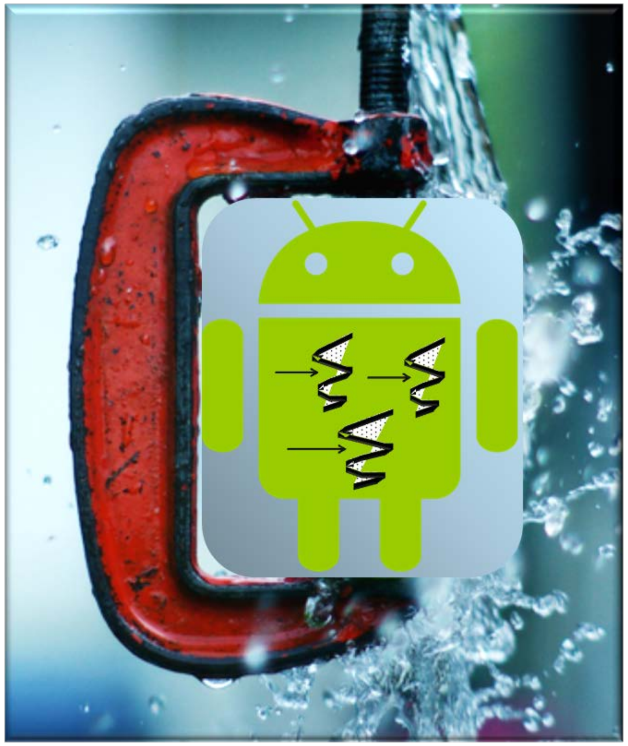
- Identify key characteristics of a software framework

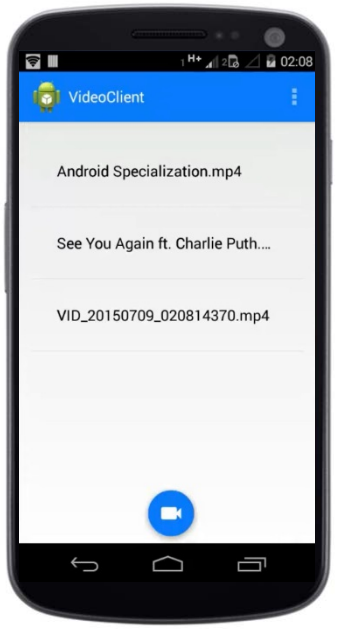


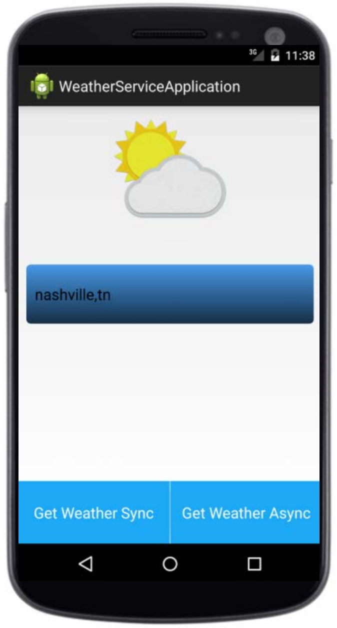
A framework provides an integrated set of classes that collaborate to provide a reusable software architecture for a family of related apps

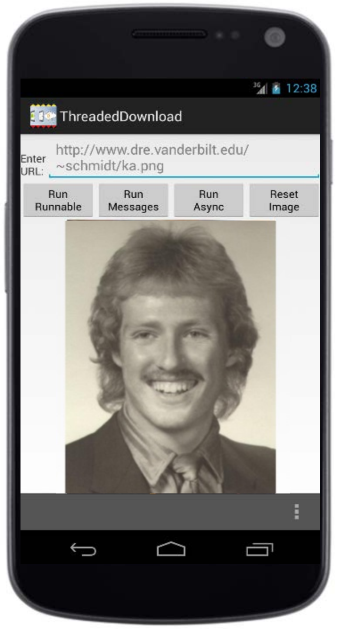
# Learning Objectives in this Part of the Module

- Identify key characteristics of a software framework
- Understand motivations for Android concurrency & concurrency frameworks




02:08
VideoClient
Android Specialization.mp4
See You Again ft. Charlie Puth....
VID_20150709_020814370.mp4

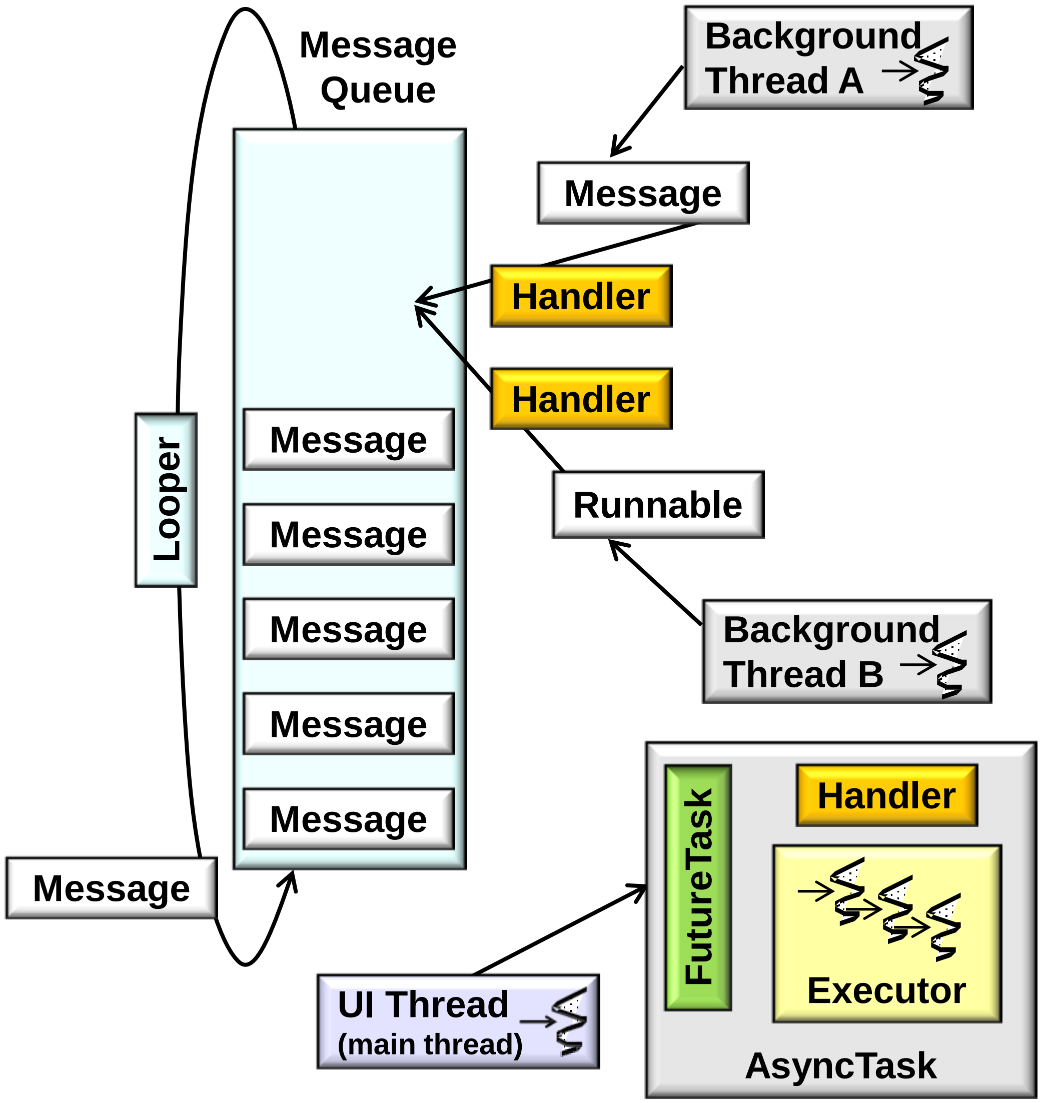




A concurrency framework provides integrated classes that collaborate to provide a resuable software architecture for concurrent apps

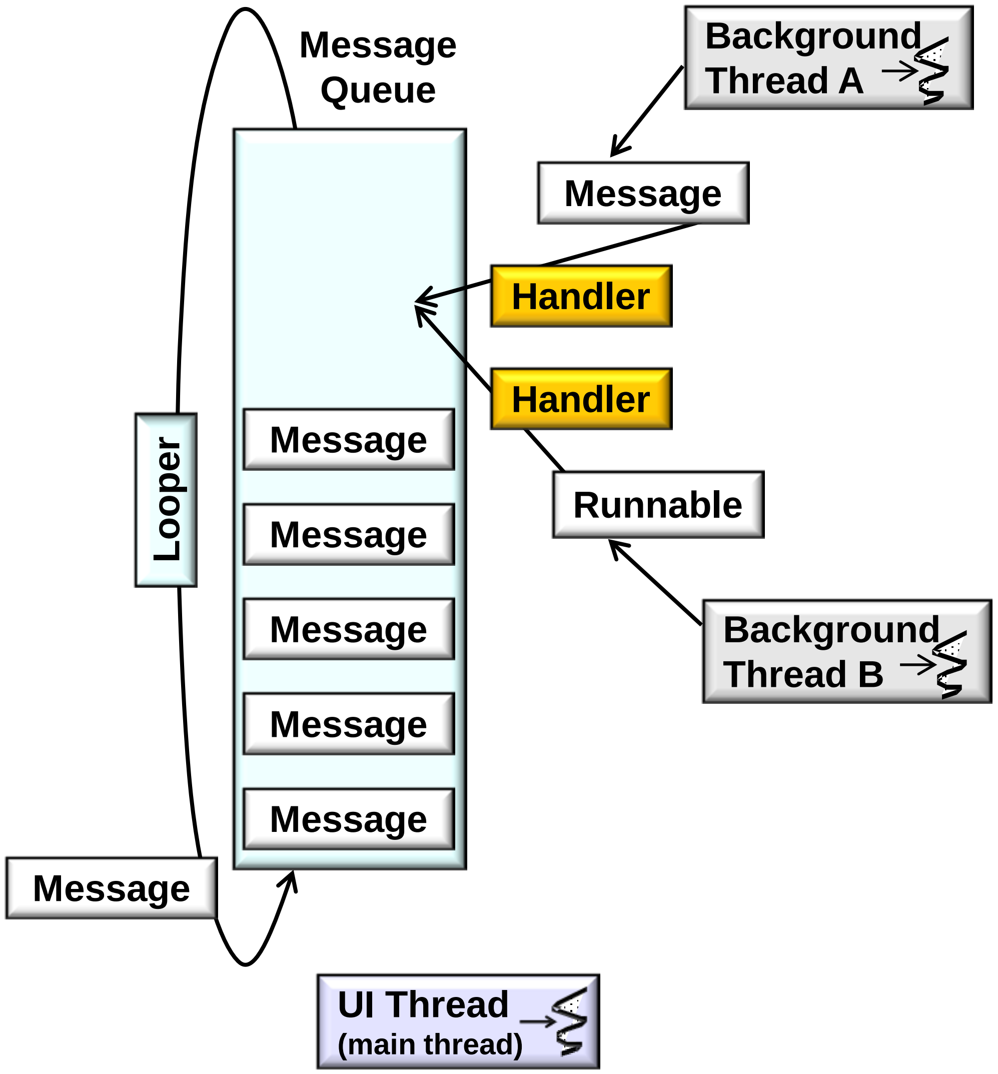
# Learning Objectives in this Part of the Module

- Identify key characteristics of a software framework
- Understand motivations for Android concurrency & concurrency frameworks
- Recognize the structure & functionality of Android's concurrency frameworks



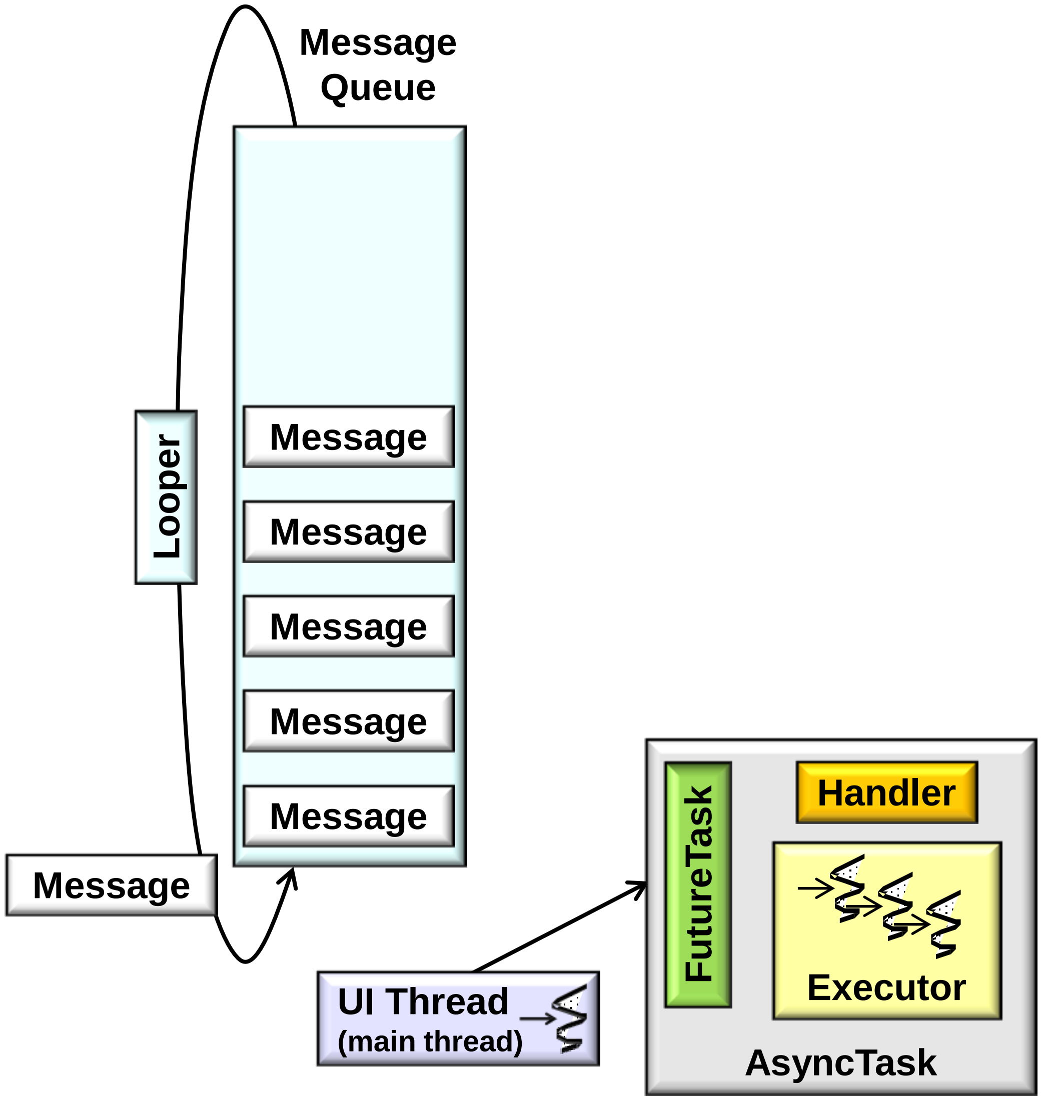
# Learning Objectives in this Part of the Module

- Identify key characteristics of a software framework
- Understand motivations for Android concurrency & concurrency frameworks
- Recognize the structure & functionality of Android's concurrency frameworks
  - Handler, Messages, & Runnables (HaMeR) framework



# Learning Objectives in this Part of the Module

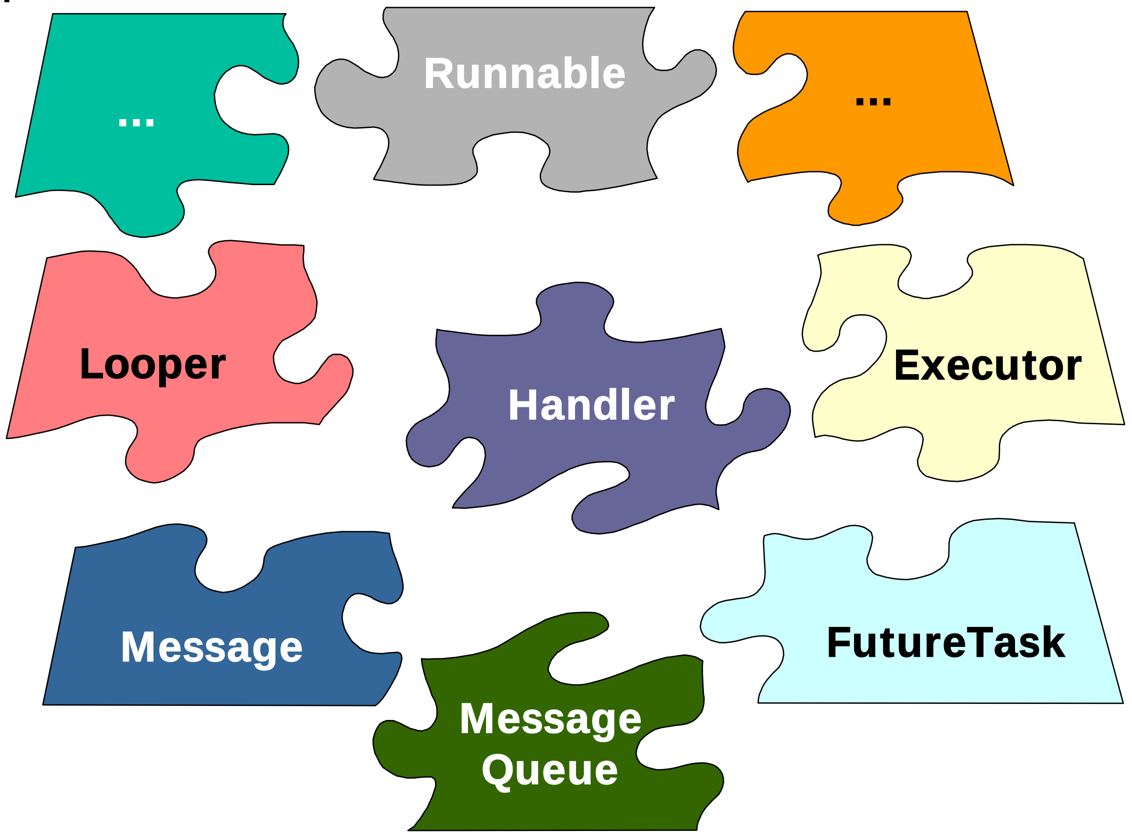
- Identify key characteristics of a software framework
- Understand motivations for Android concurrency & concurrency frameworks
- Recognize the structure & functionality of Android's concurrency frameworks
  - Handler, Messages, & Runnables (HaMeR) framework
  - AsyncTask framework



## Overview of Frameworks (Part 1)

# Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps

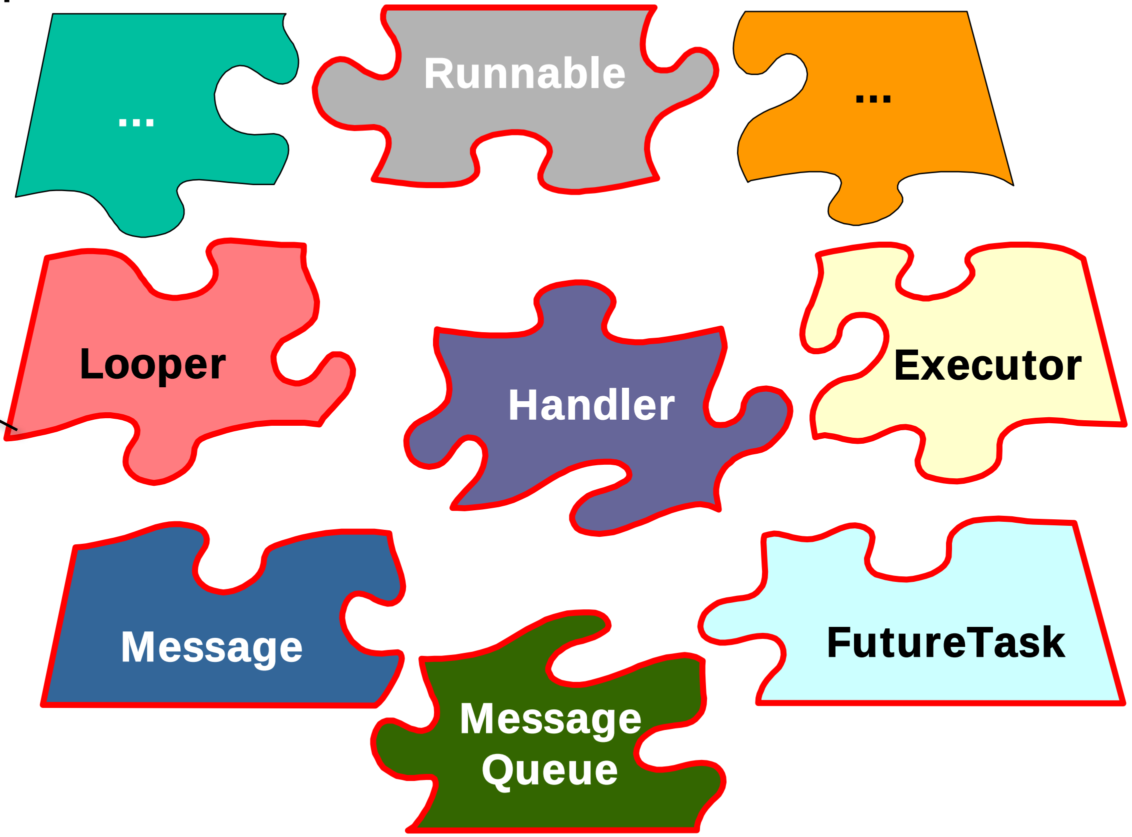


See www.dre.vanderbilt.edu/~schmidt/frameworks.html

# Overview of Frameworks

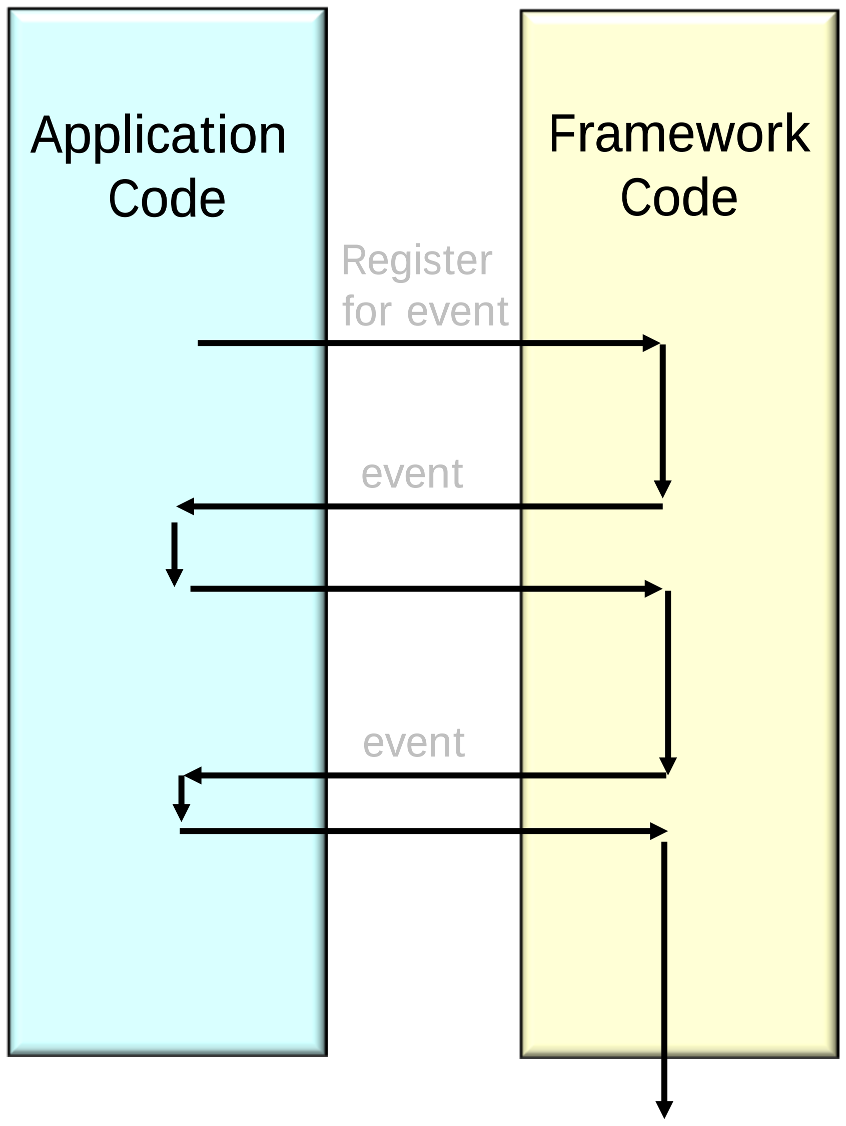
- A framework is an integrated set of components that provide a reusable architecture for a family of apps

*We'll analyze all these classes in this course*



# Overview of Frameworks

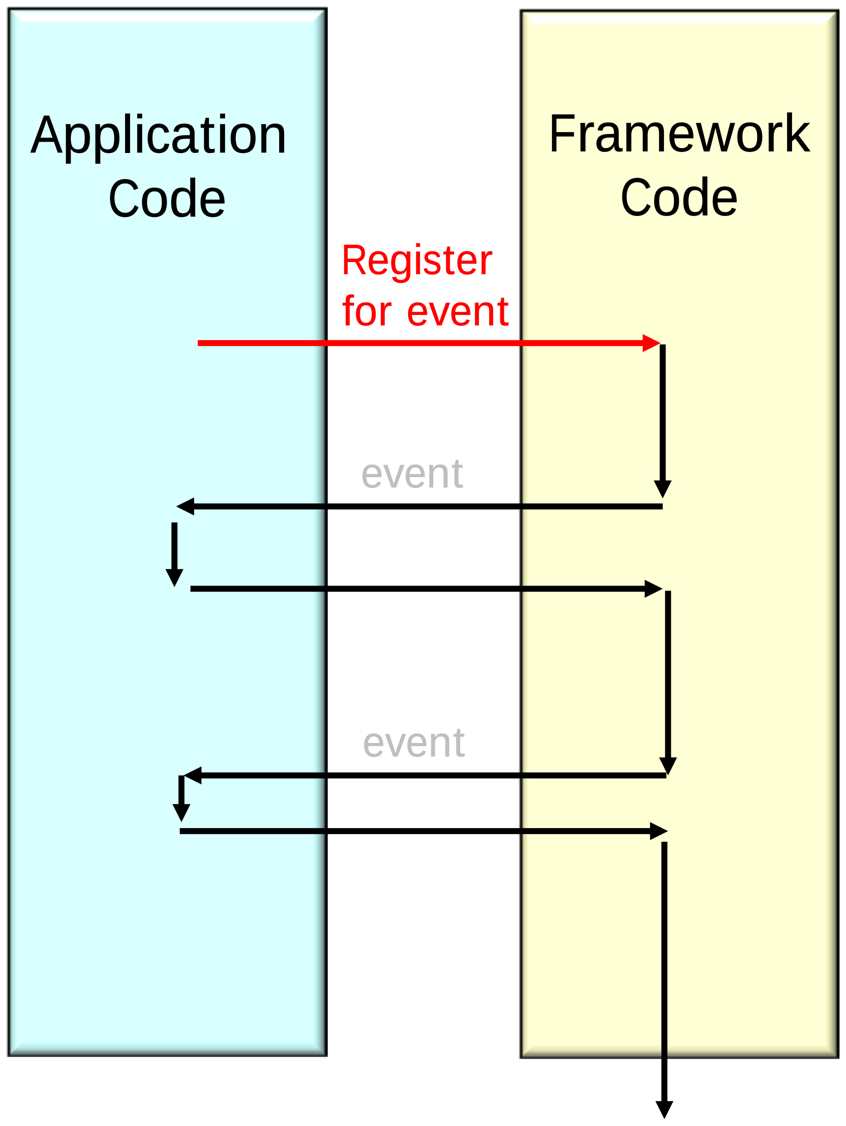
- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them



See [en.wikipedia.org/wiki/Event-driven_programming](en.wikipedia.org/wiki/Event-driven_programming)

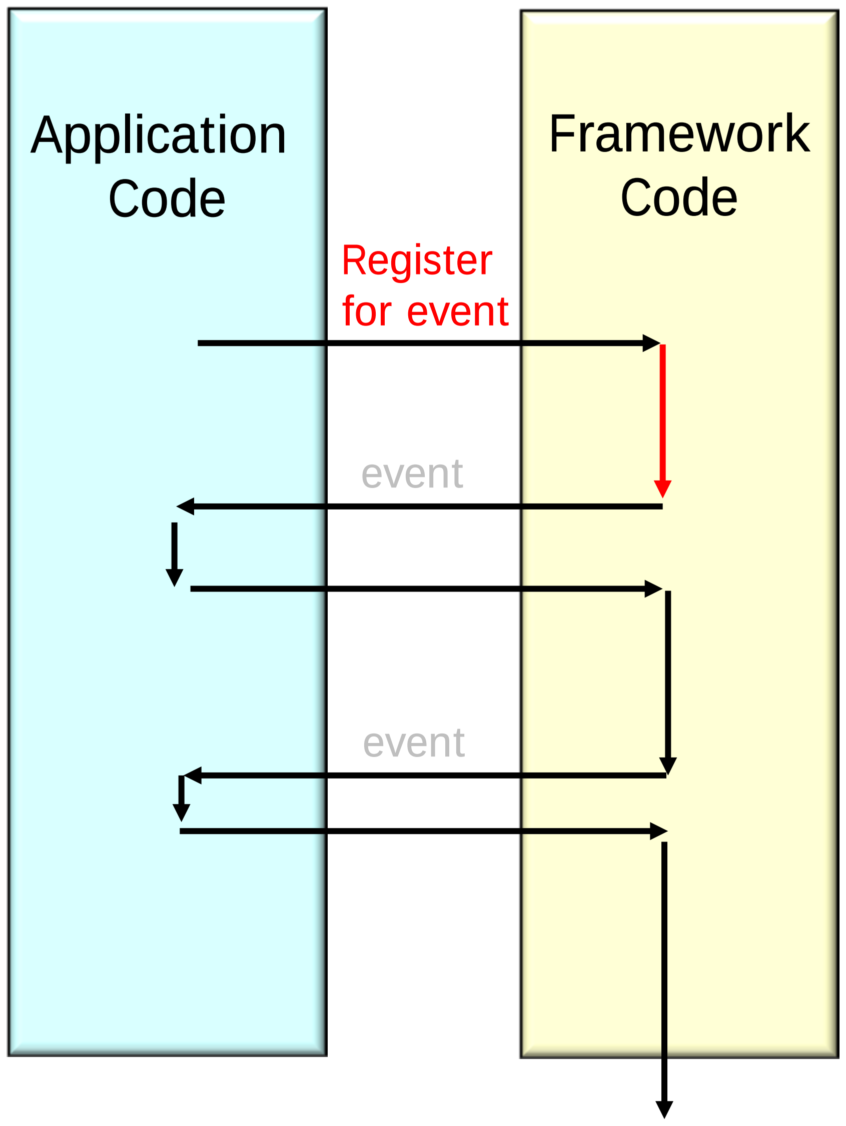
## Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework



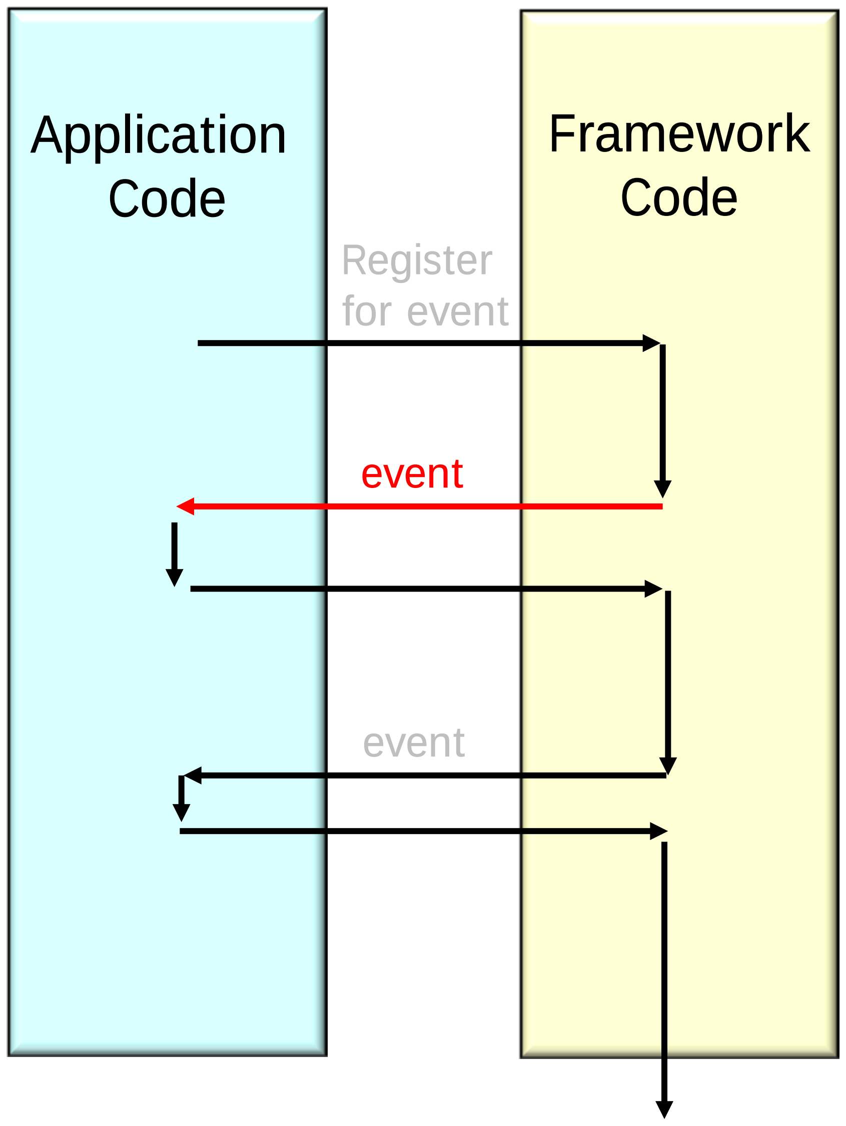
## Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework
    - e.g., arrival of messages from remote servers, gestures on GUI elements, etc.



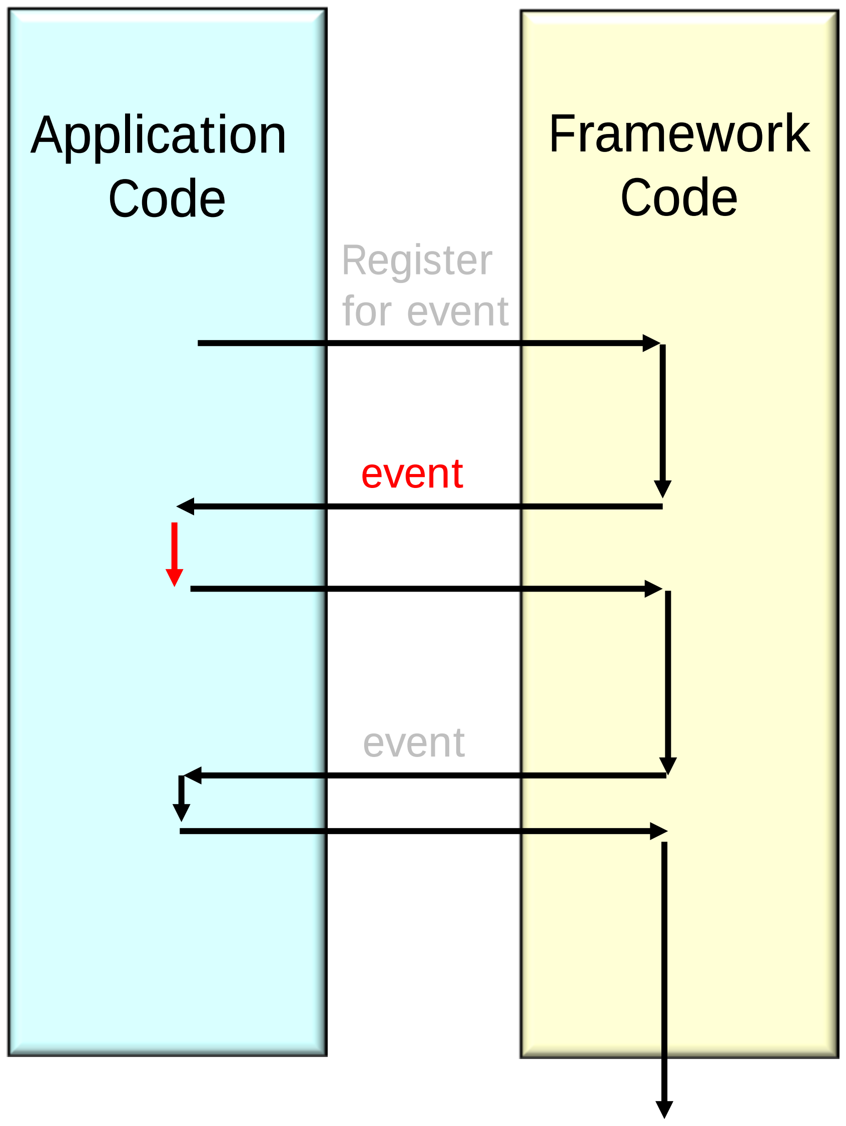
# Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework
  - Framework calls back to app code when an event occurs



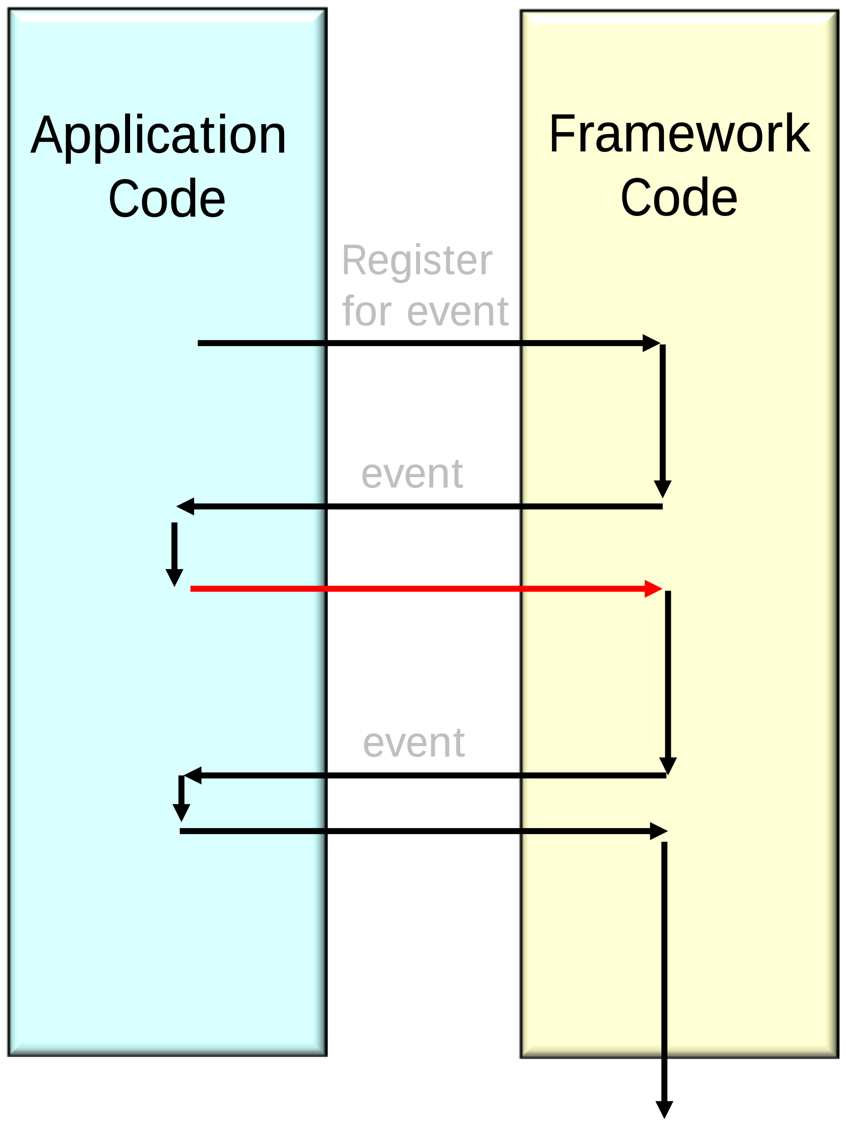
# Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework
  - Framework calls back to app code when an event occurs
    - The app performs its processing in context of framework's thread



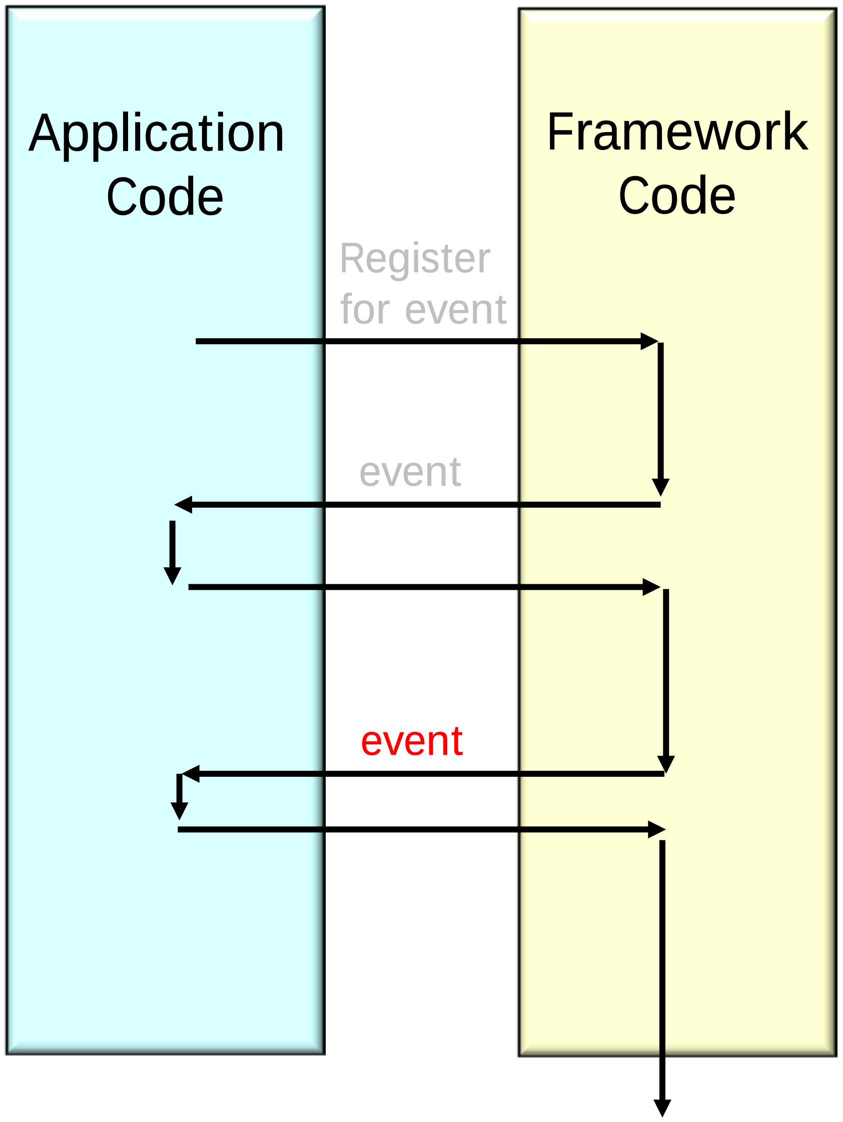
# Overview of Frameworks

- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework
  - Framework calls back to app code when an event occurs
  - When app code is done, control returns to the framework



# Overview of Frameworks

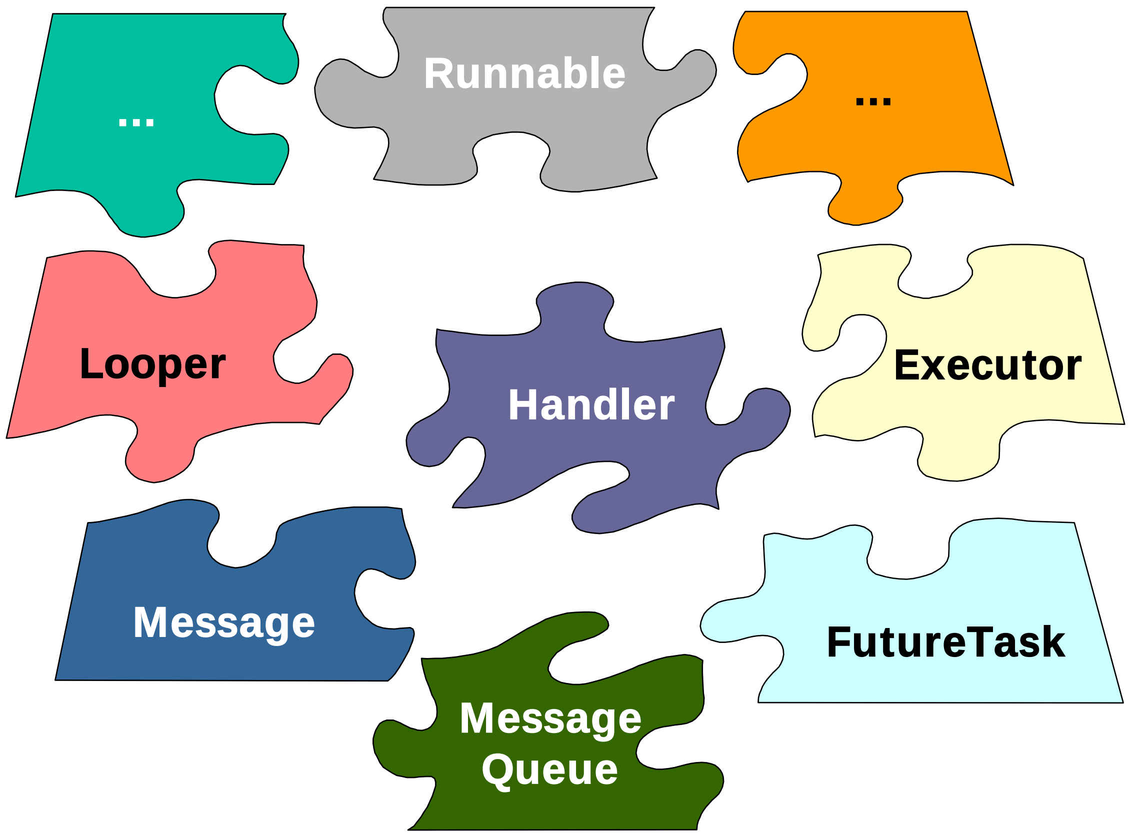
- A framework is an integrated set of components that provide a reusable architecture for a family of apps
  - Often use an event-driven programming model to plug app code into them
  - Apps register callbacks for specific types of events that can occur within the framework
  - Framework calls back to app code when an event occurs
  - When app code is done, control returns to the framework
  - Lather, rinse, repeat until app is done …



## Overview of Frameworks (Part 2)

# Overview of Frameworks

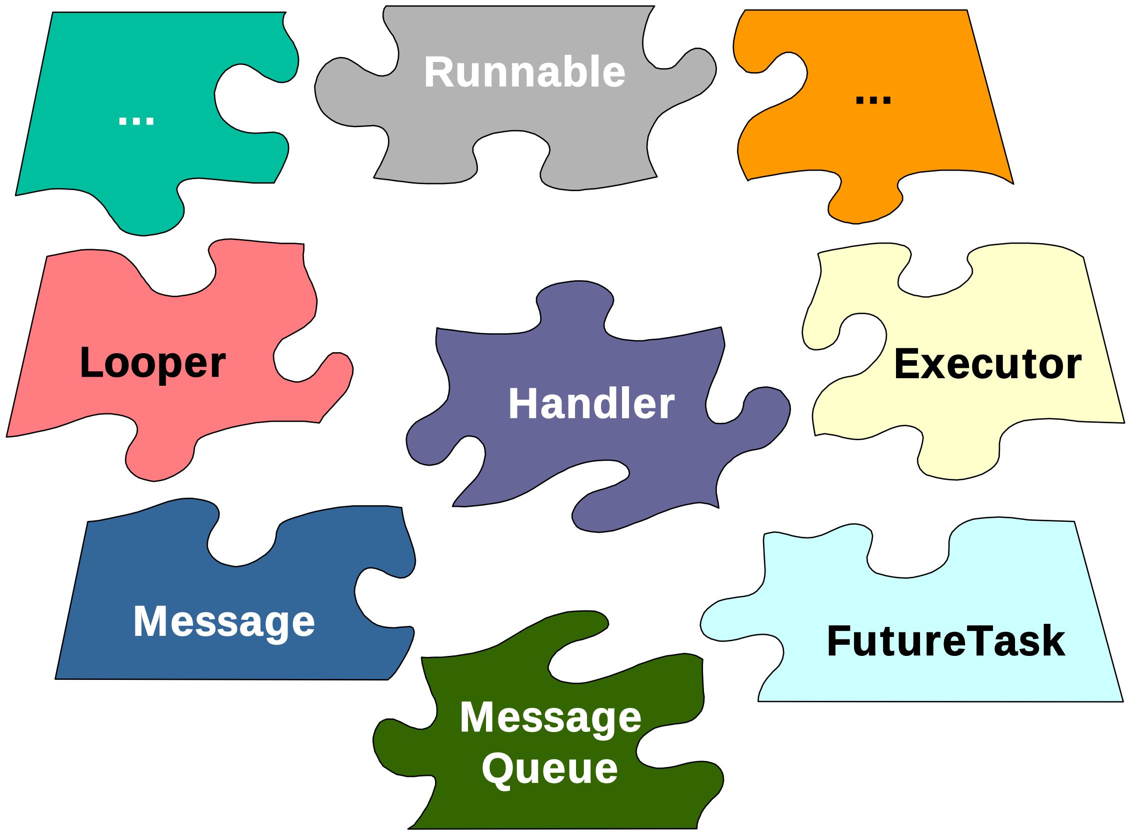
- Key frameworks characteristics



See en.wikipedia.org/wiki/Software_framework

# Overview of Frameworks

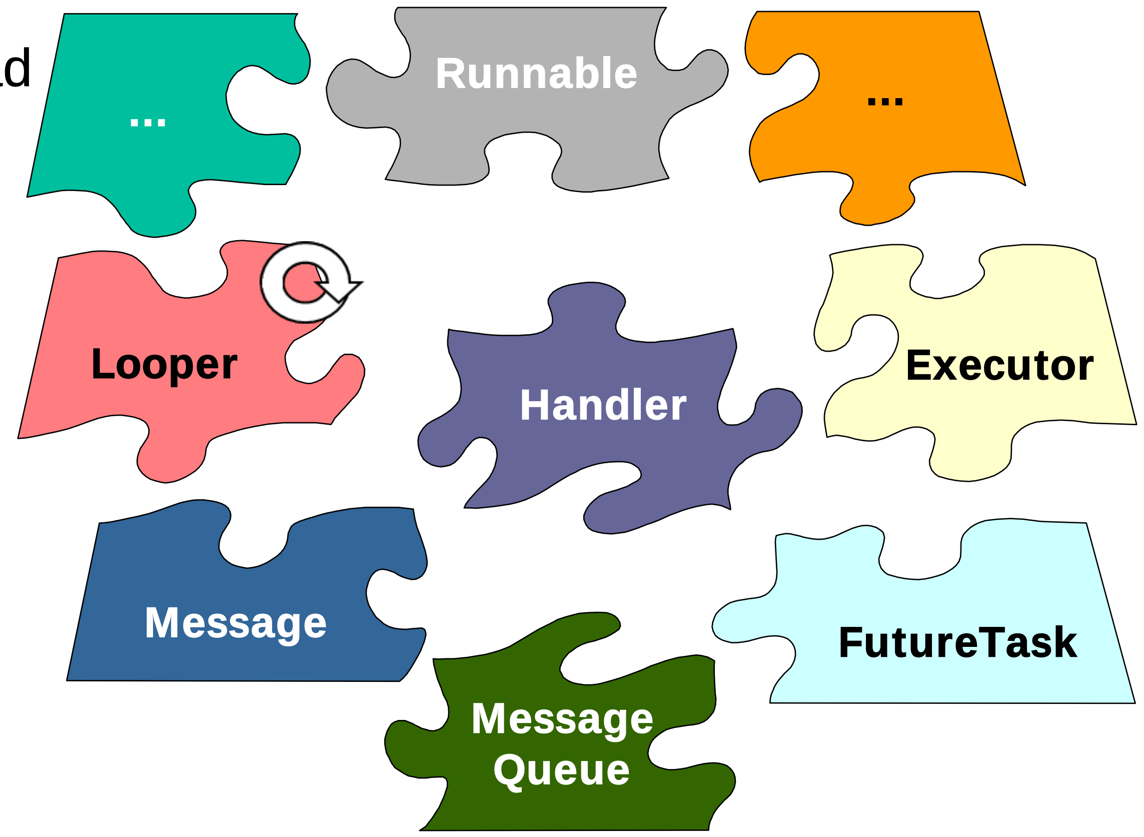
- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks



See www.dre.vanderbilt.edu/~schmidt/Coursera/articles/hollywood-principle.txt

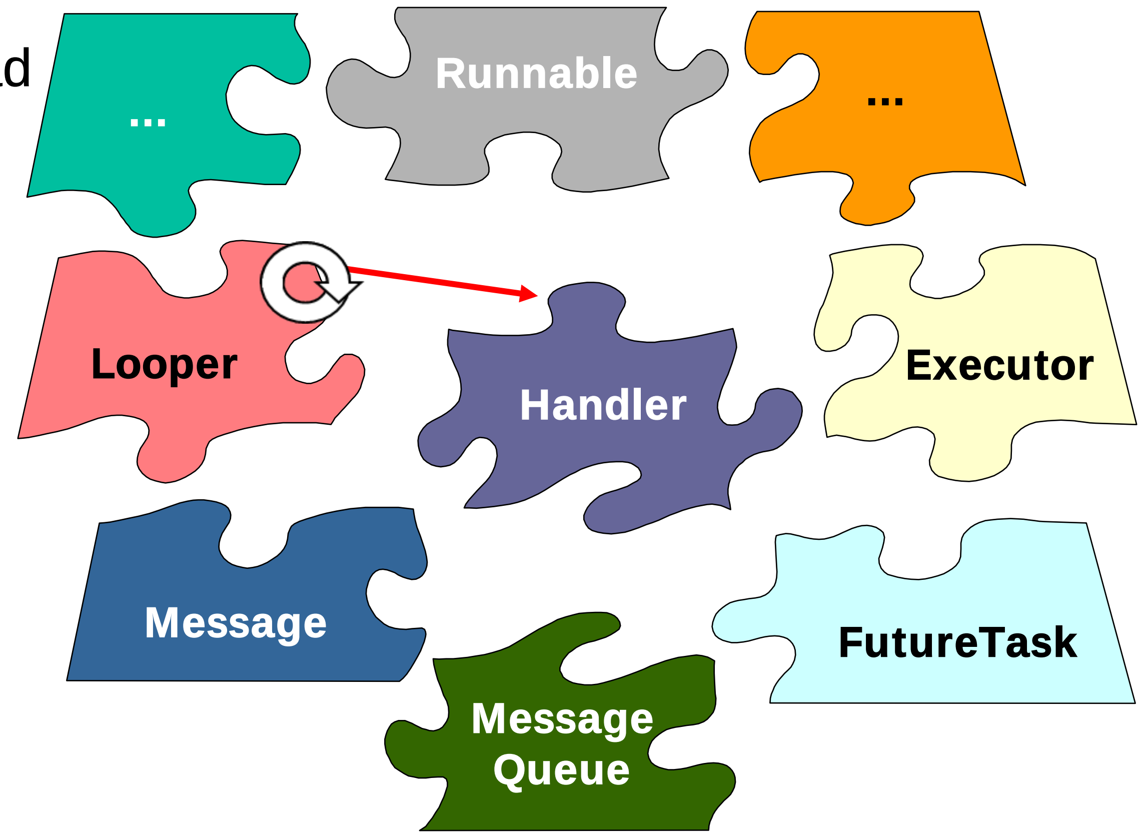
# Overview of Frameworks

- Key frameworks characteristics
  - Exhibit “inversion of control” via callbacks
    - i.e., it controls main thread of execution & decides when to run app code



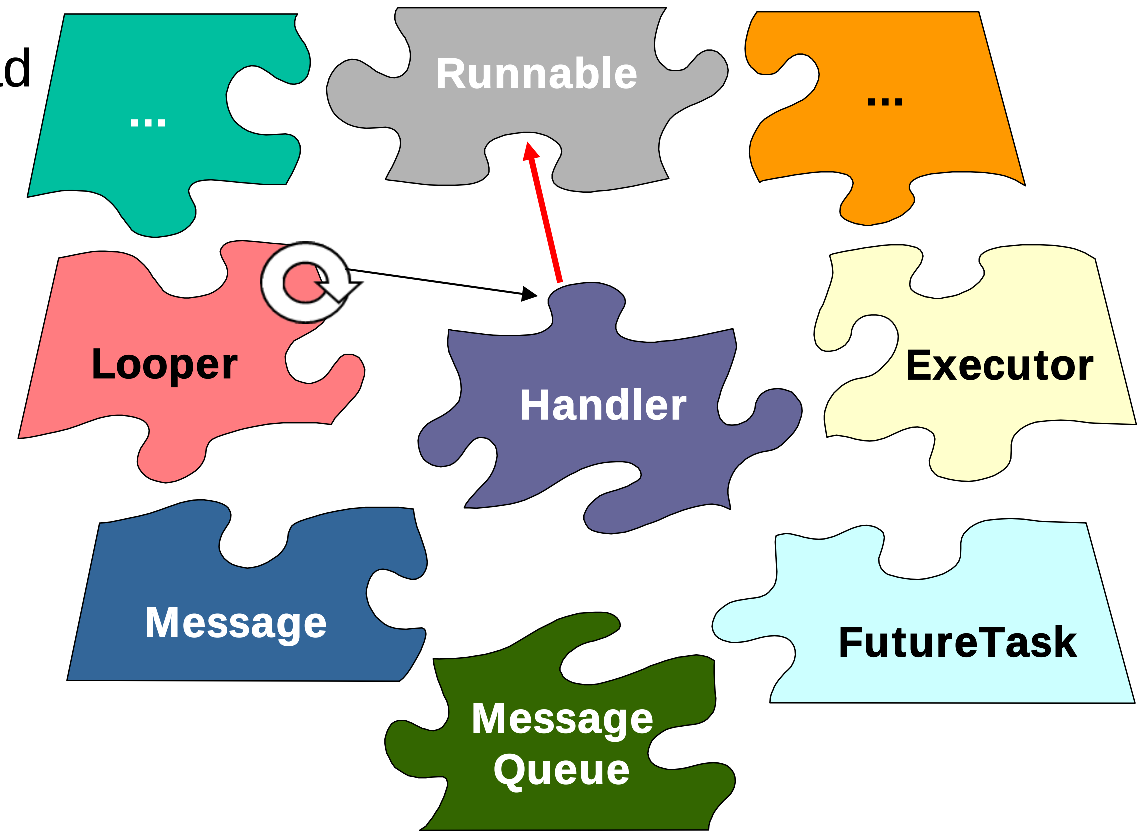
# Overview of Frameworks

- Key frameworks characteristics
  - Exhibit “inversion of control” via callbacks
    - i.e., it controls main thread of execution & decides when to run app code



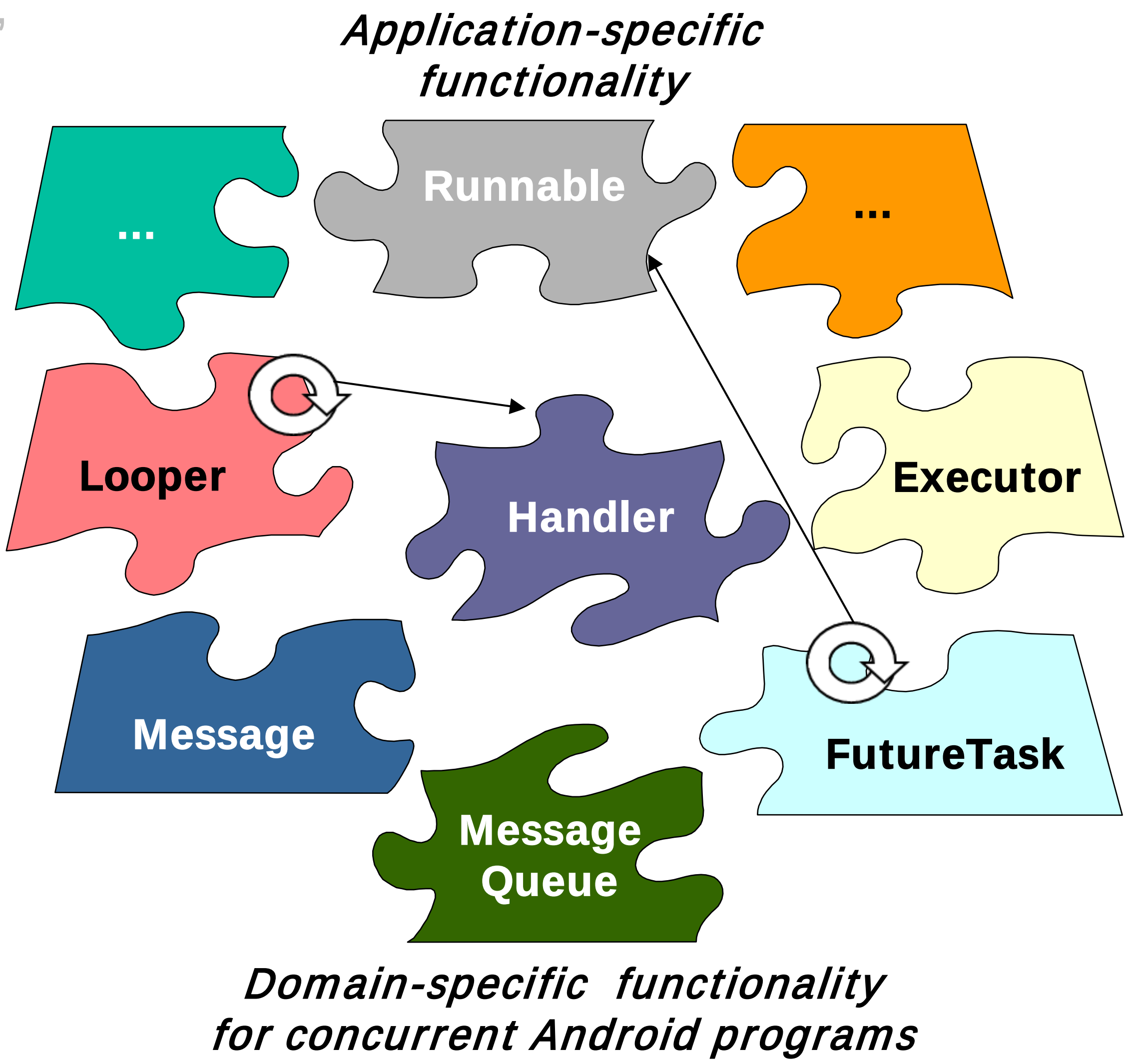
## Overview of Frameworks

- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
    - i.e., it controls main thread of execution & decides when to run app code



# Overview of Frameworks

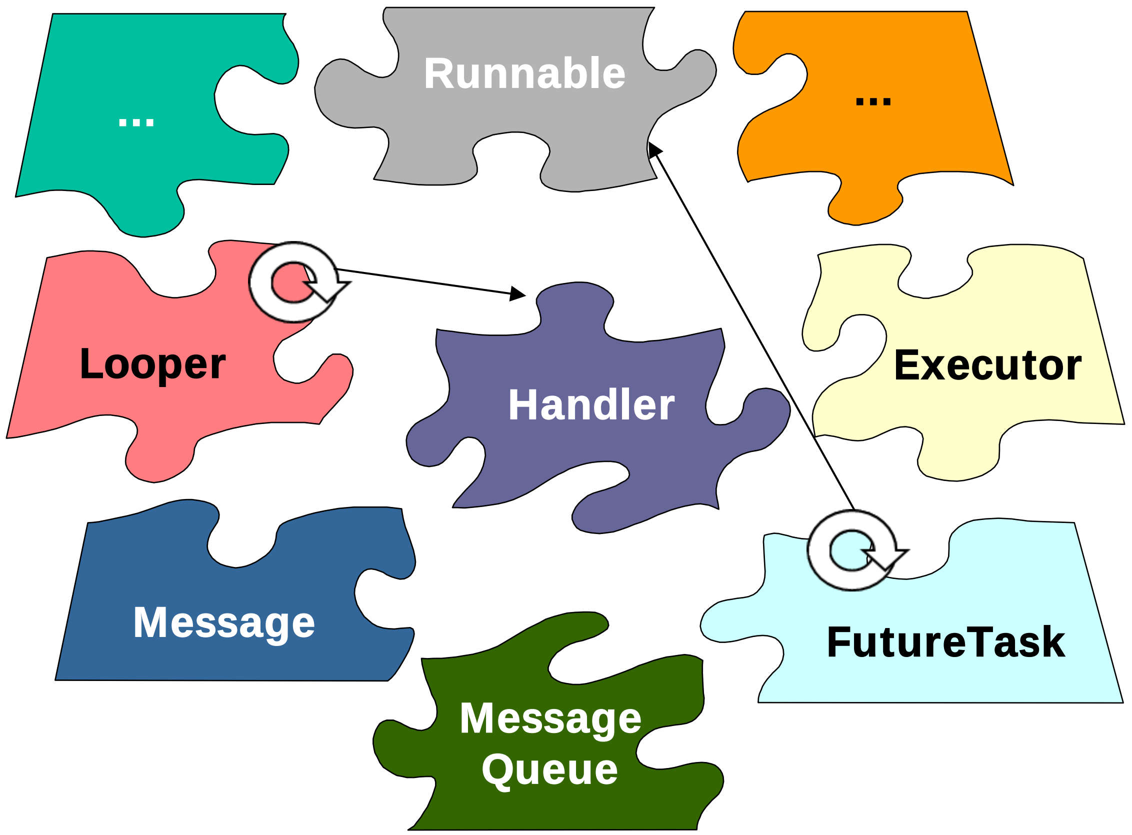
- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality


*Application-specific functionality*
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
*Domain-specific functionality for concurrent Android programs*

# Overview of Frameworks

- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality
    - e.g., provide default capabilities useful to some domain(s)

*Application-specific functionality*

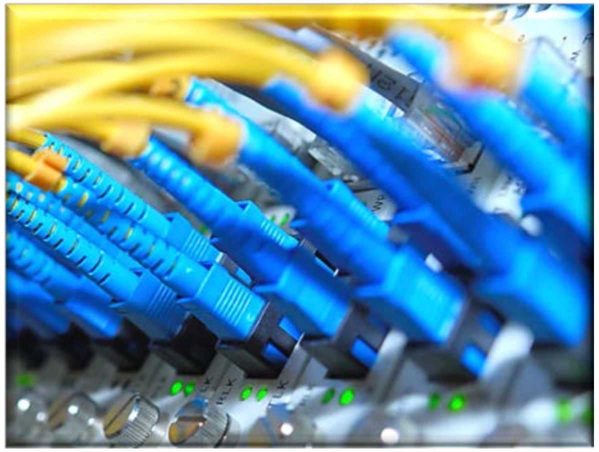


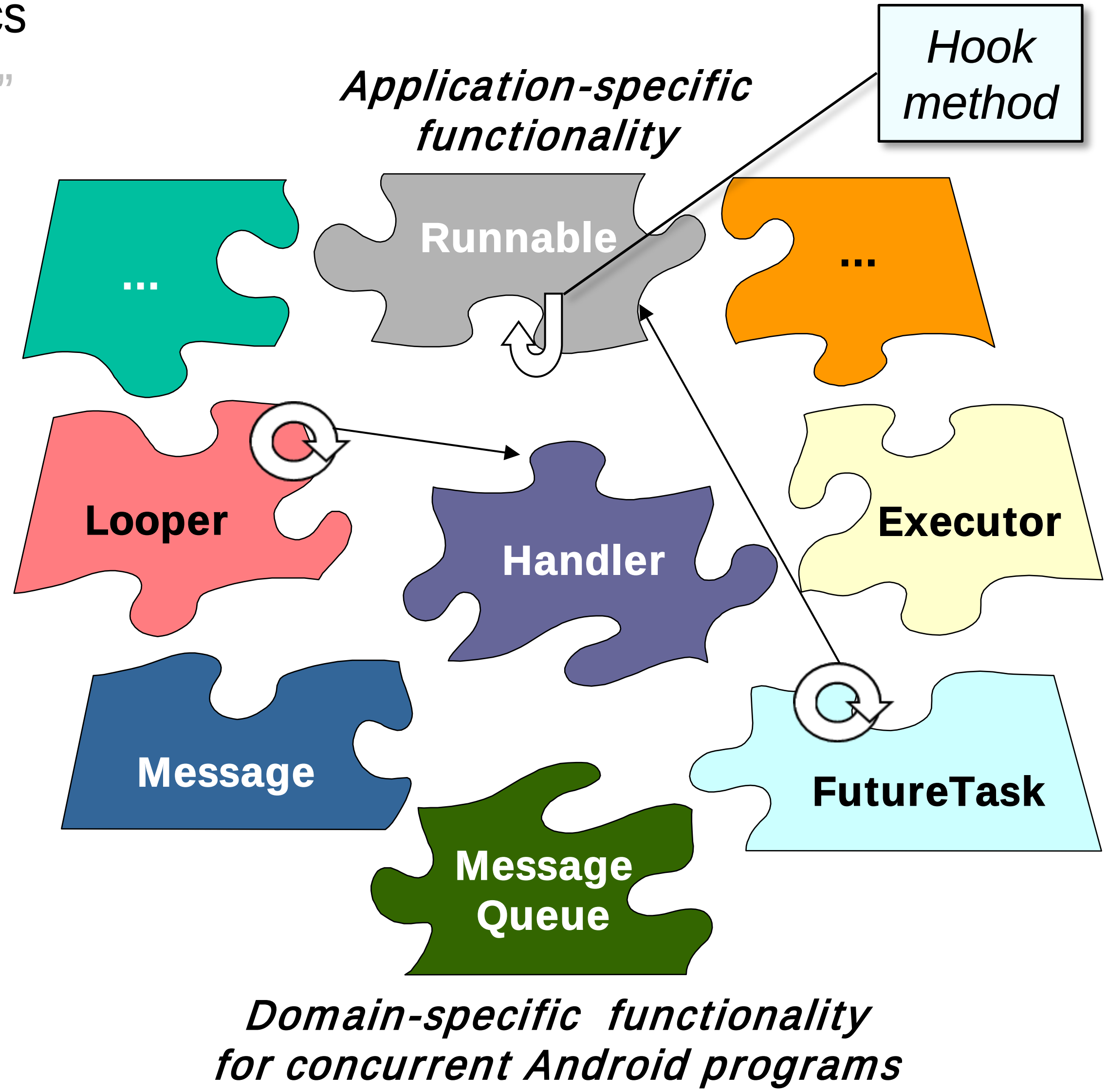
*Domain-specific functionality for concurrent Android programs*

Android's frameworks focus on domains associated with mobile apps & services

## Overview of Frameworks

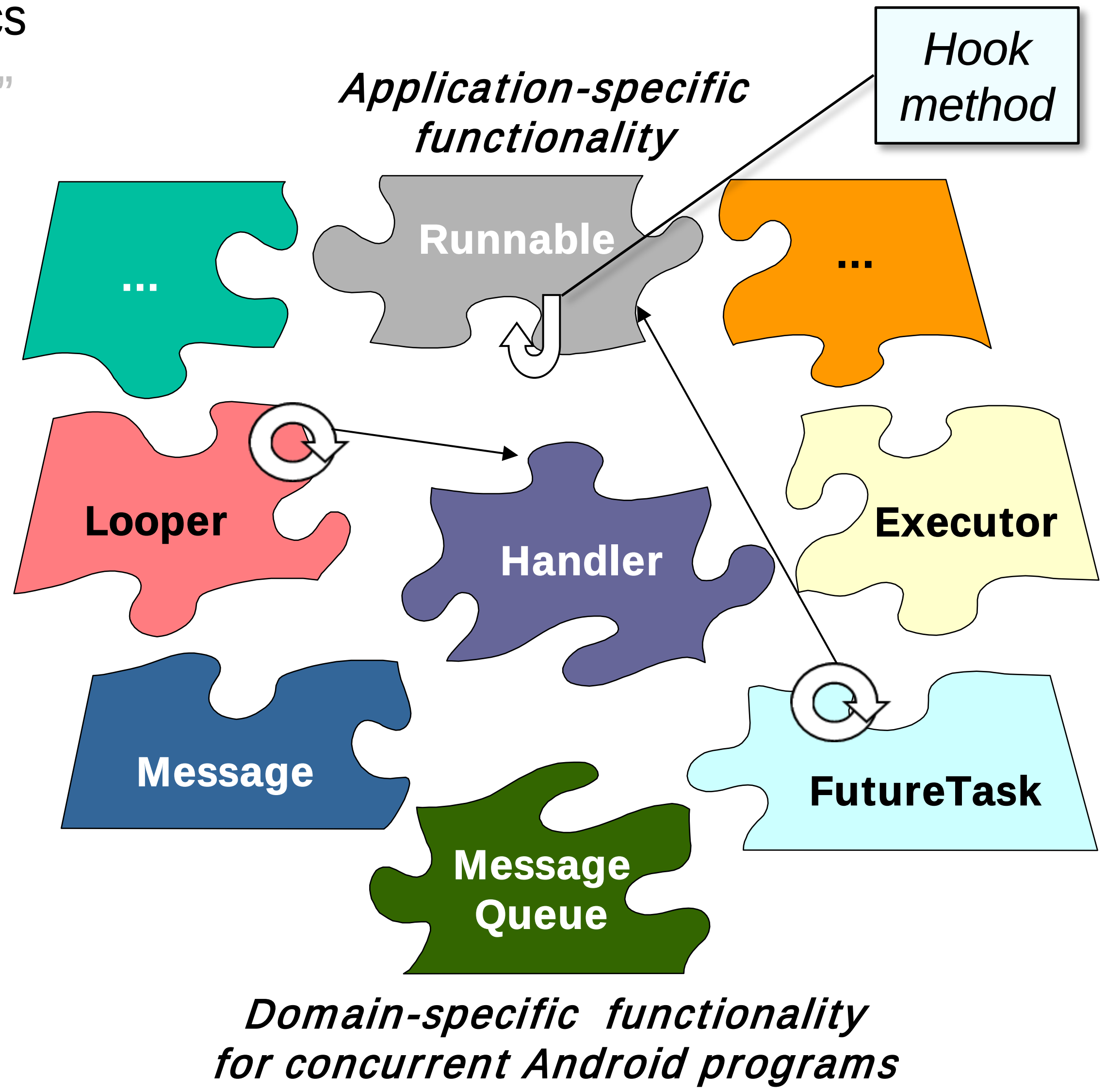
- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality
  - Provide semi-complete (portions of) apps




Hook method
Application-specific functionality
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
Domain-specific functionality for concurrent Android programs

# Overview of Frameworks

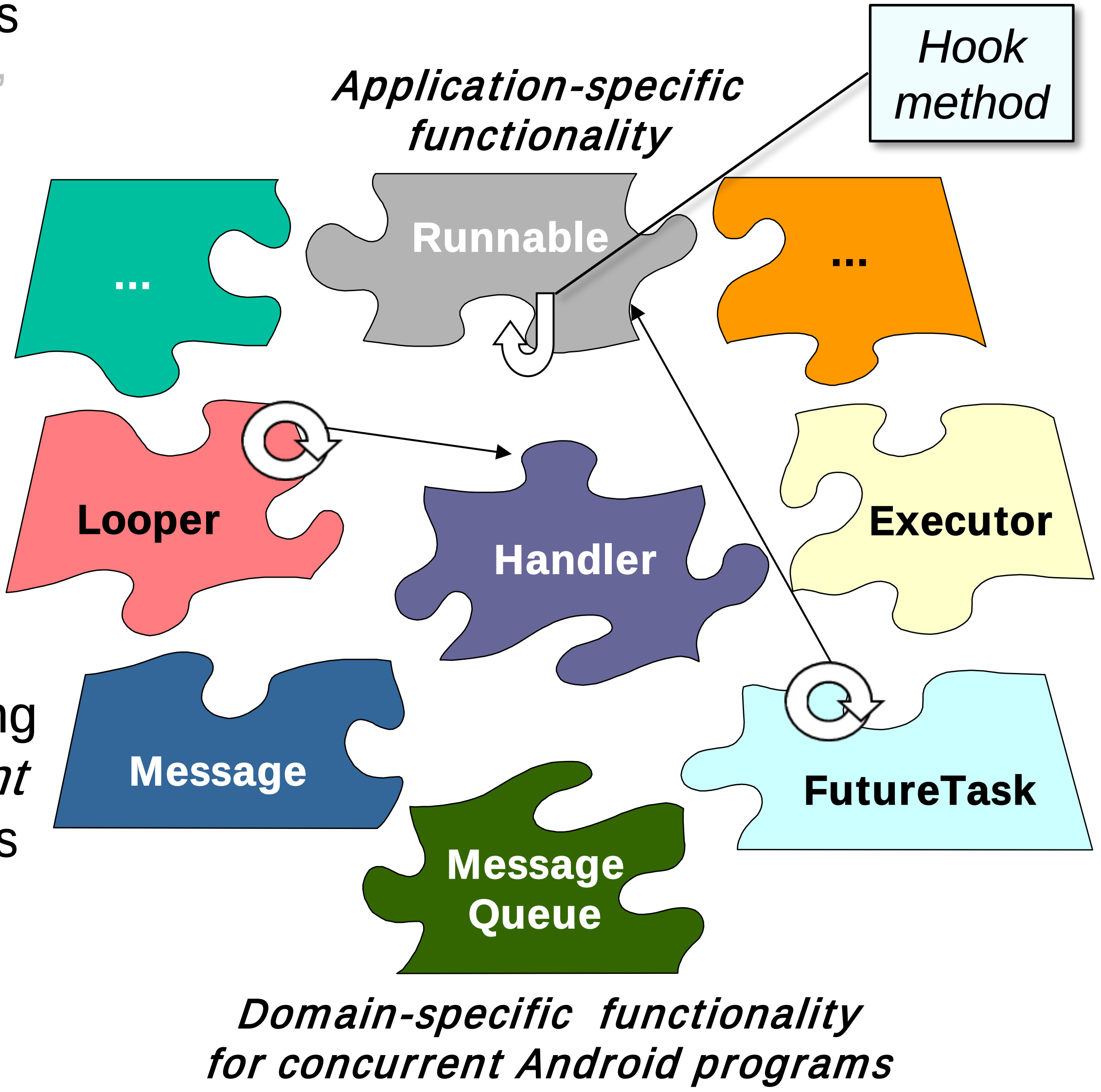
- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality
  - Provide semi-complete (portions of) app
    - Extensible *hook methods* plug app logic into the framework


Hook method
Application-specific functionality
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
Domain-specific functionality for concurrent Android programs

These hook methods customize reusable framework classes to run app-specific logic

# Overview of Frameworks

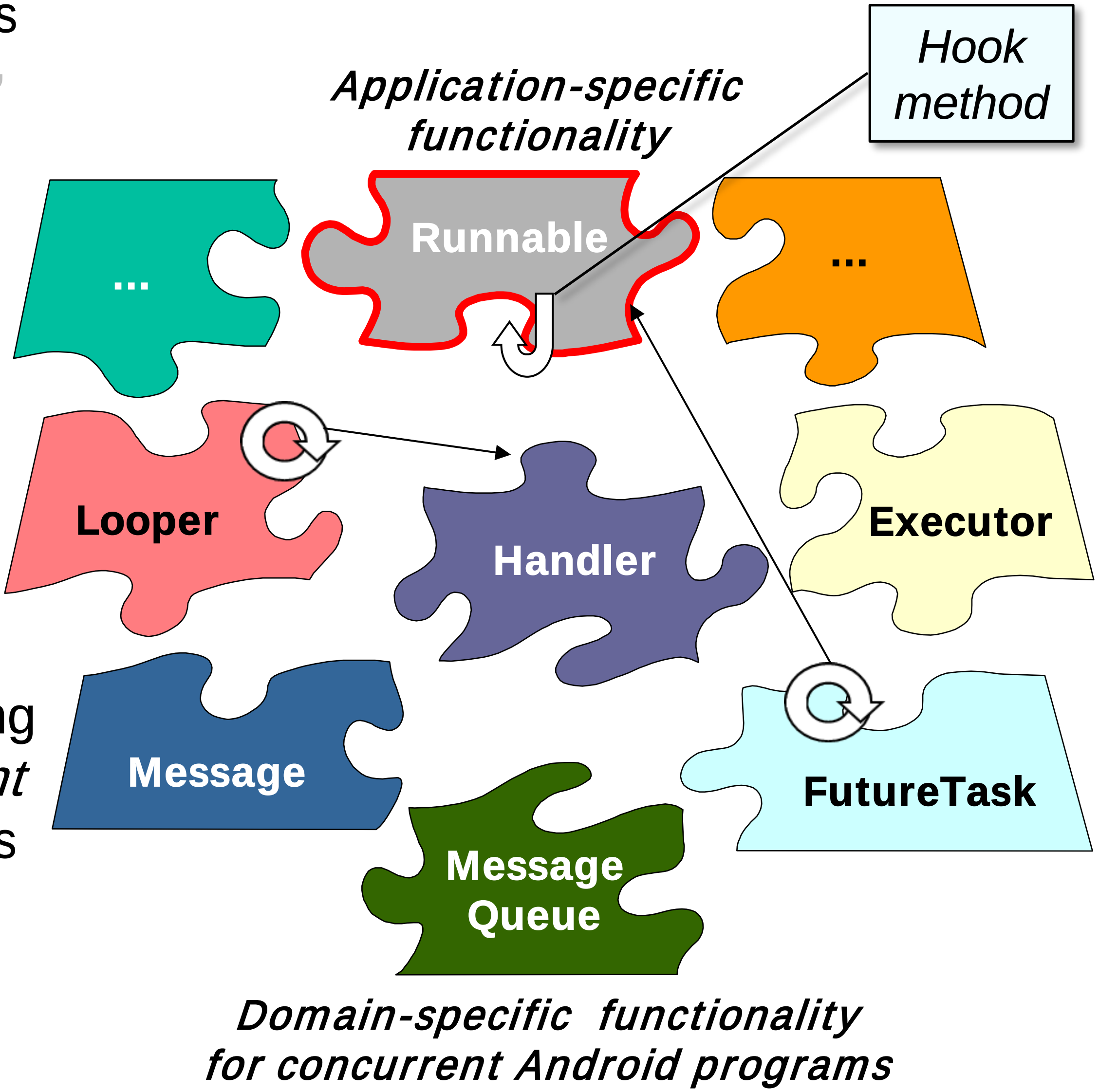
- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality
  - Provide semi-complete (portions of) apps
    - Extensible *hook methods* plug app logic into the framework
    - Mediate interactions among *common* abstract & *variant* concrete classes/interfaces



Hook method
Application-specific functionality
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
Domain-specific functionality for concurrent Android programs

# Overview of Frameworks

- Key frameworks characteristics
  - Exhibit "inversion of control" via callbacks
  - Integrated domain-specific structure & functionality
  - Provide semi-complete (portions of) apps
    - Extensible *hook methods* plug app logic into the framework
    - Mediate interactions among *common* abstract & *variant* concrete classes/interfaces



Hook method
Application-specific functionality
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
Domain-specific functionality for concurrent Android programs

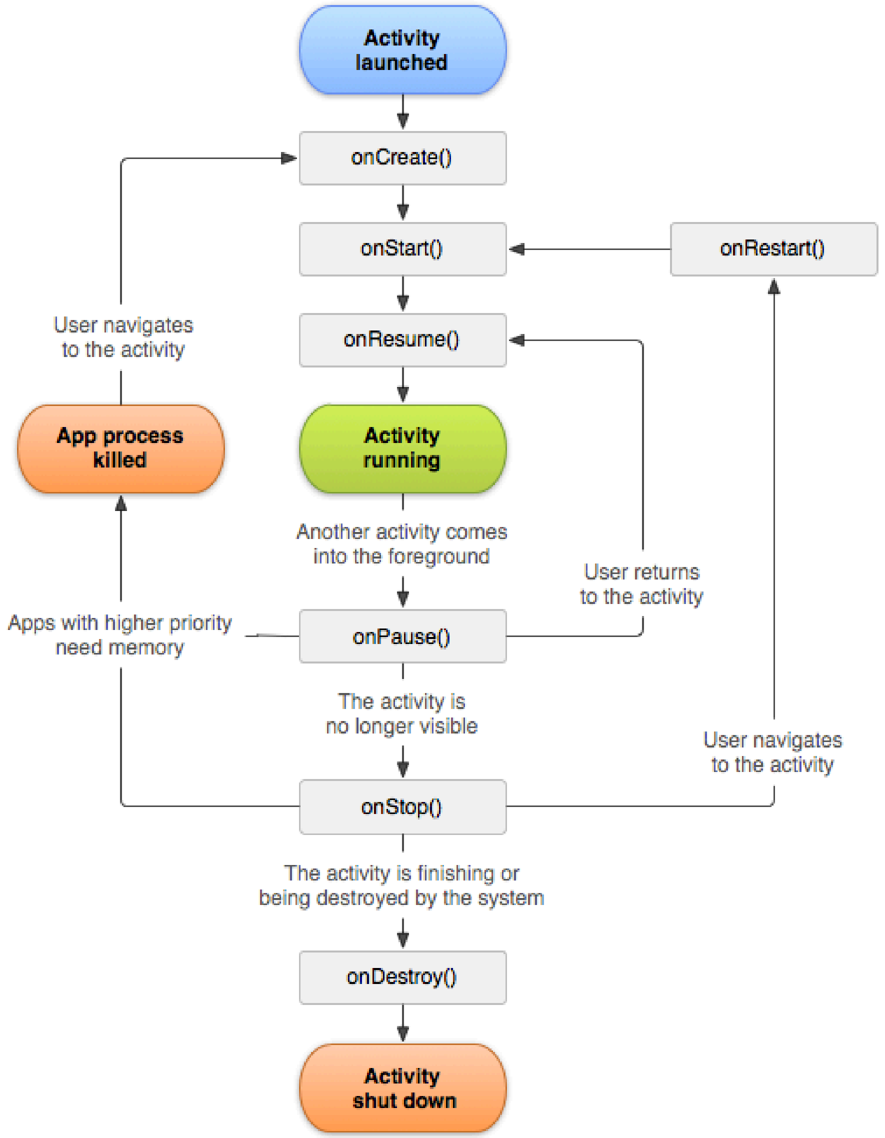
Runnable is a common abstract interface that provides the basis for concrete variants

# Overview of Frameworks

- Key frameworks characteristics
- Android & Java provide many frameworks

# Overview of Frameworks

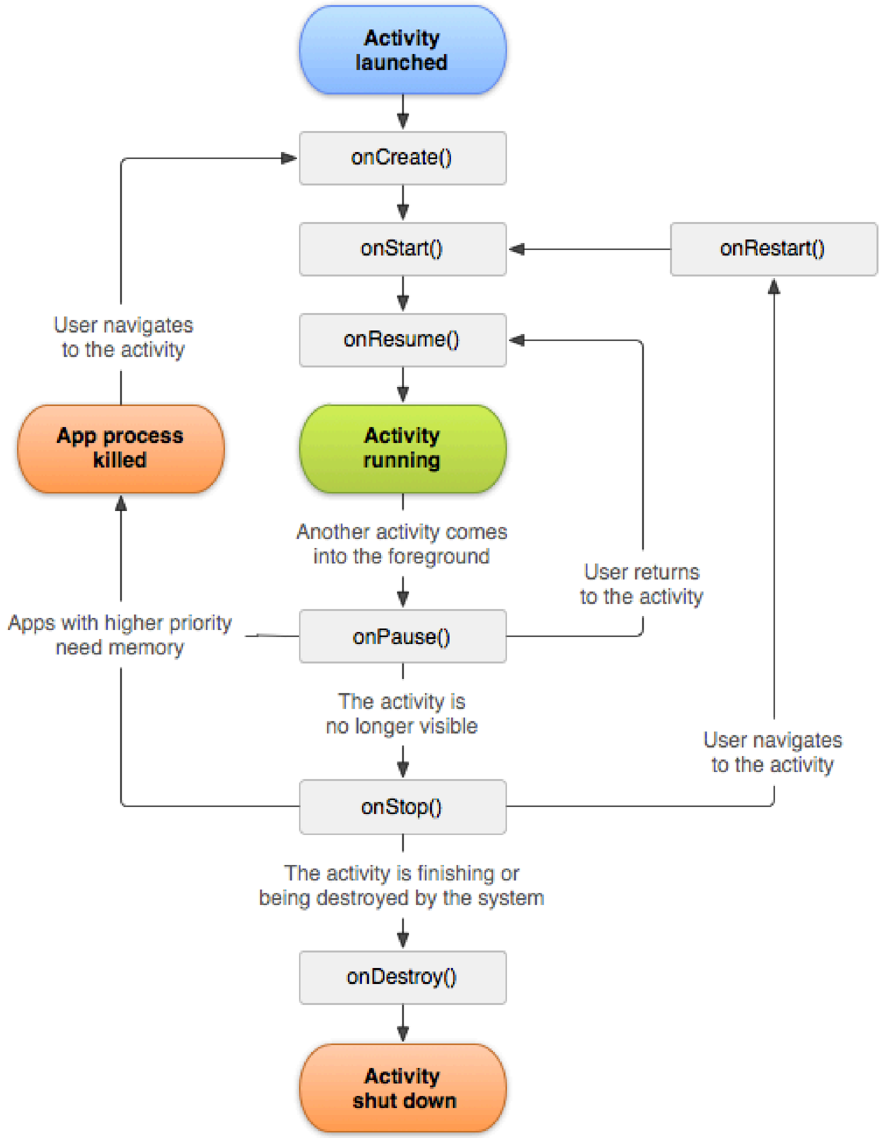
- Key frameworks characteristics
- Android & Java provide many frameworks
  - **Android**
    - Android Activity framework controls the main thread


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

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

# Overview of Frameworks

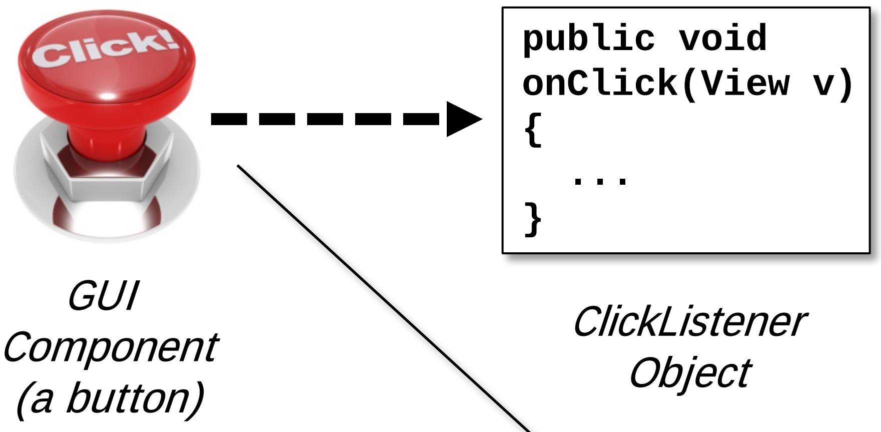
- Key frameworks characteristics
- Android & Java provide many frameworks
  - **Android**
    - Android Activity framework controls the main thread
    - App lifecycle methods are called back by the Activity framework
      - e.g., onCreate(), onStart(), onStop(), onDestroy(), etc.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

See developer.android.com/training/basics/activity-lifecycle

# Overview of Frameworks

- Key frameworks characteristics
- Android & Java provide many frameworks
  - **Android**
    - Android Activity framework controls the main thread
    - App lifecycle methods are called back by the Activity framework
    - A listener for button clicks is called back by Android's GUI framework

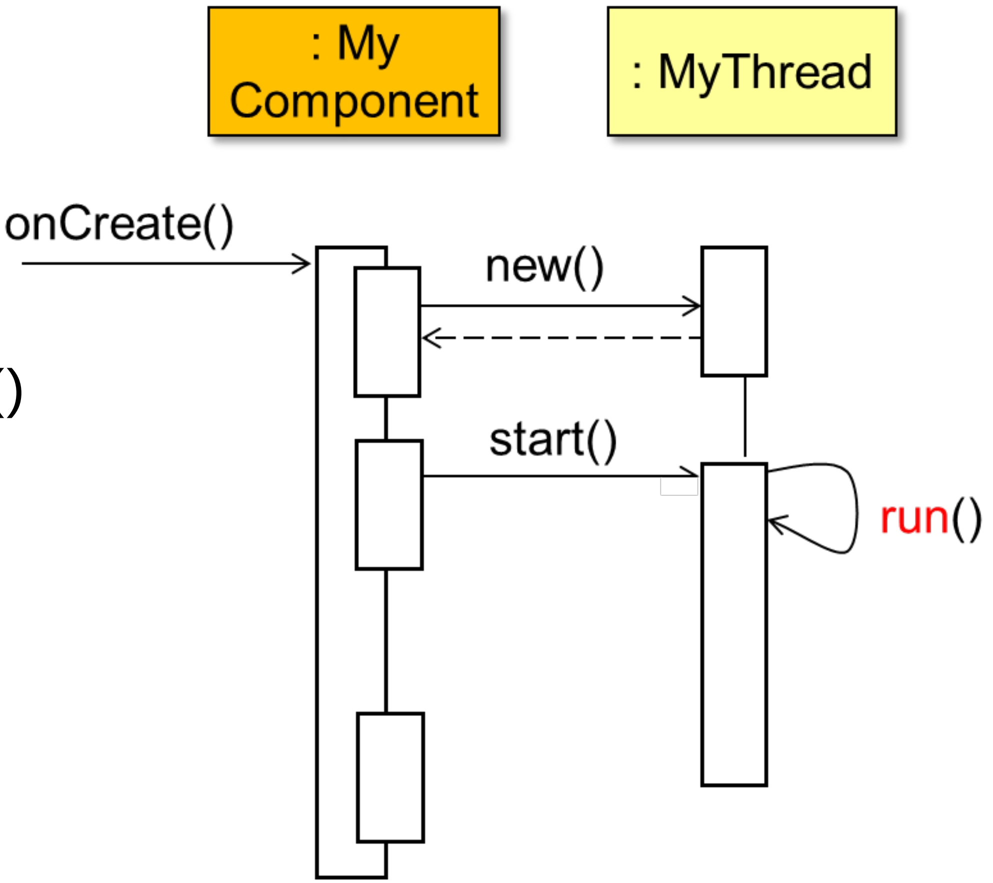


```
public void
onClick(View v)
{
   ...
}
```

*A GUI component sending an event to its registered listener*

# Overview of Frameworks

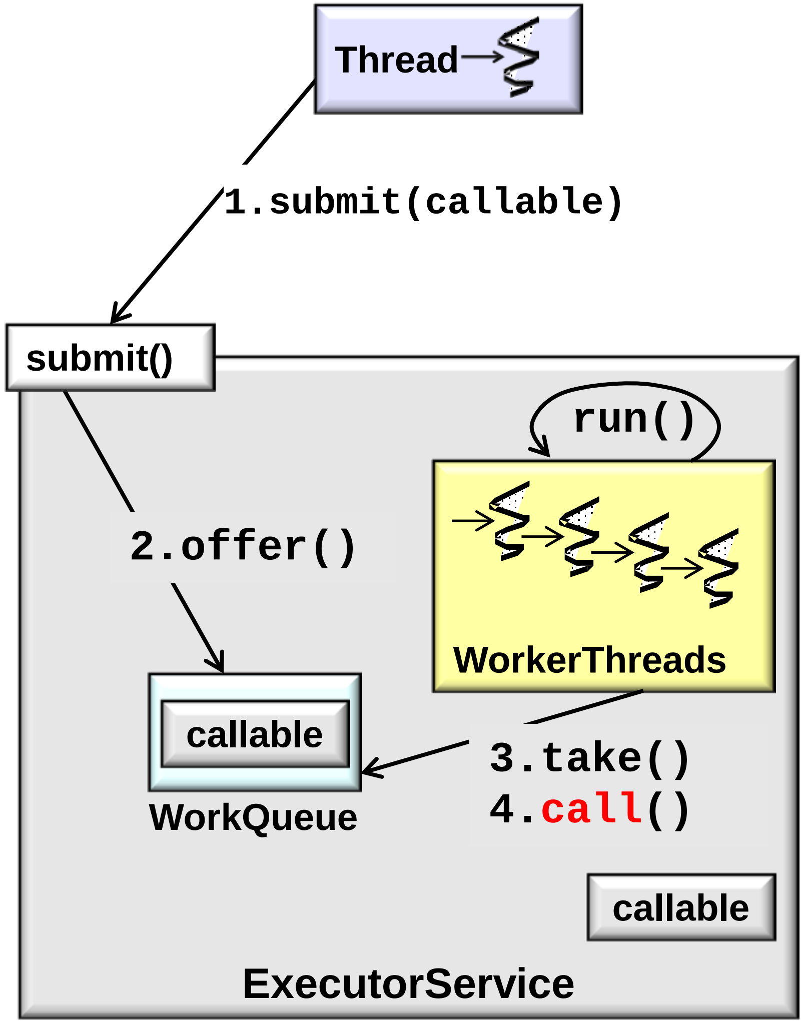
- Key frameworks characteristics
- Android & Java provide many frameworks
  - **Android**
  - **Java**
    - A Thread calls back on the run() hook method of a Runnable



See [docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html](docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html)

# Overview of Frameworks

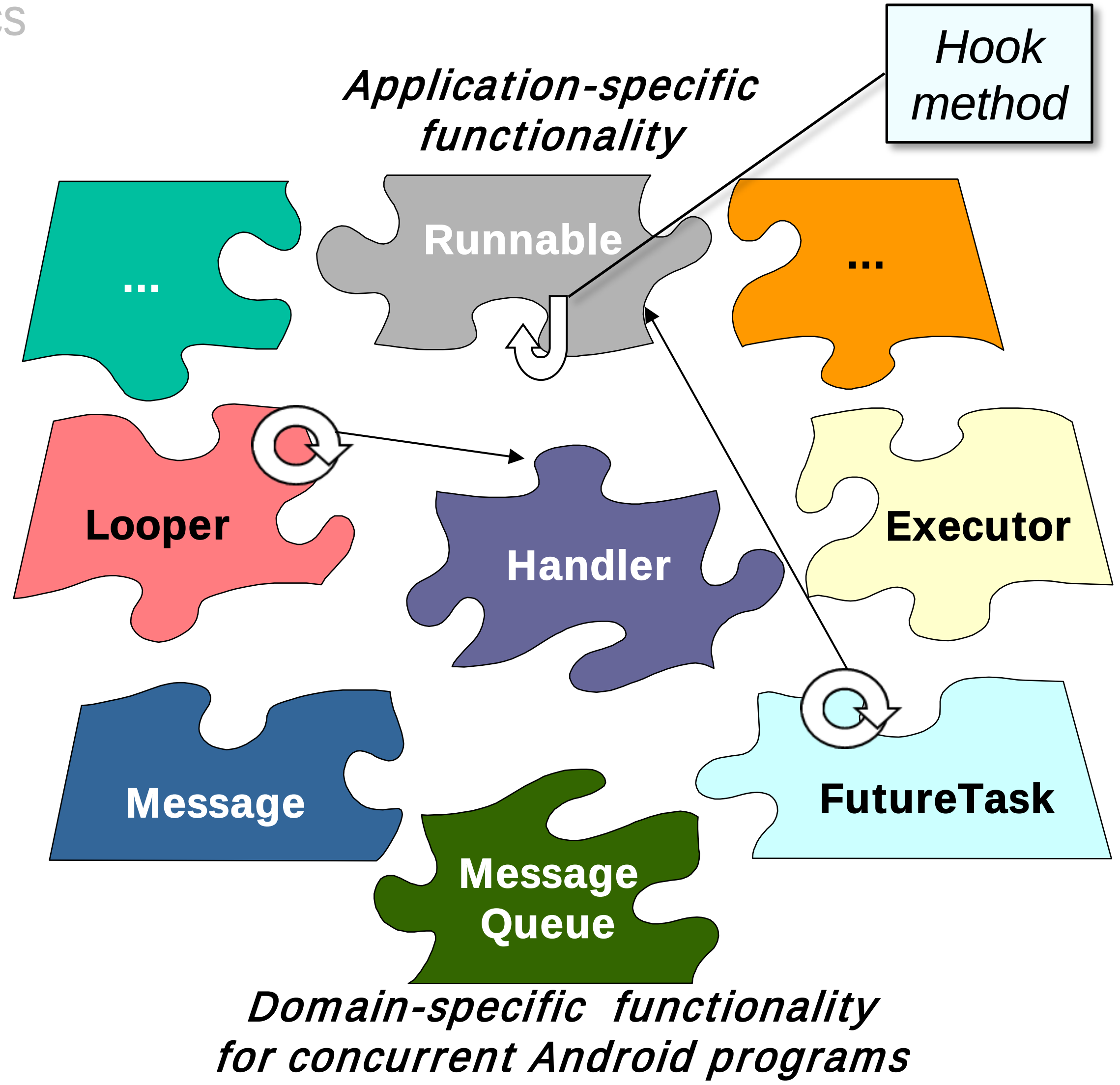
- Key frameworks characteristics
- Android & Java provide many frameworks
  - **Android**
  - **Java**
    - A Thread calls back on the run() hook method of a Runnable
    - The ExecutorService calls back to the call() hook method of a Callable



See docs.oracle.com/javase/tutorial/essential/concurrency/executors.html

# Overview of Frameworks

- Key frameworks characteristics
- Android & Java provide many frameworks
- Your apps in these MOOCs use one or more frameworks

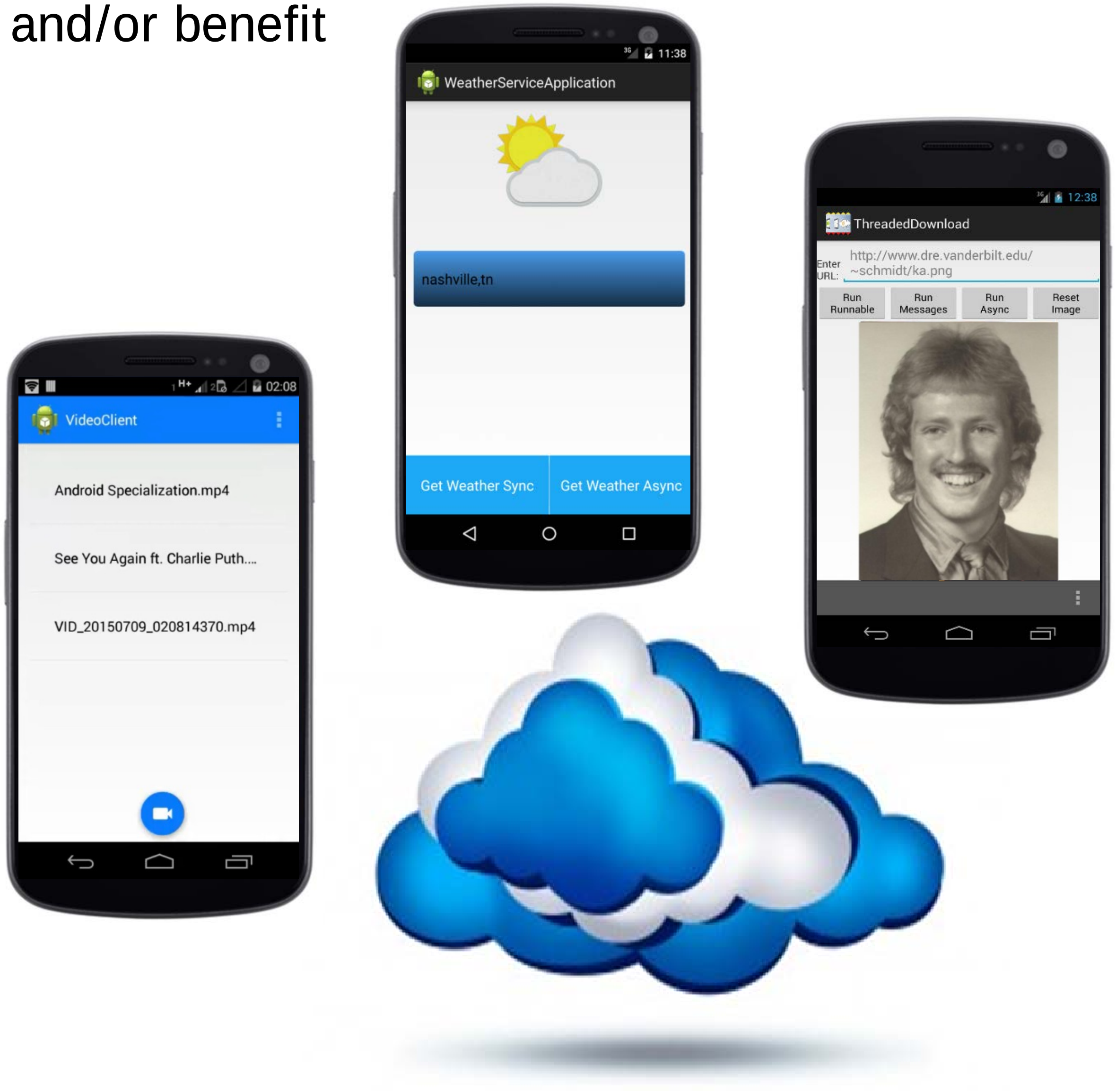

Hook method
Application-specific functionality
...
Runnable
...
Looper
Handler
Executor
Message
FutureTask
Message Queue
Domain-specific functionality for concurrent Android programs

*All* Android apps run inside one or more frameworks

## Motivation for Android Concurrency

# Motivation for Android Concurrency

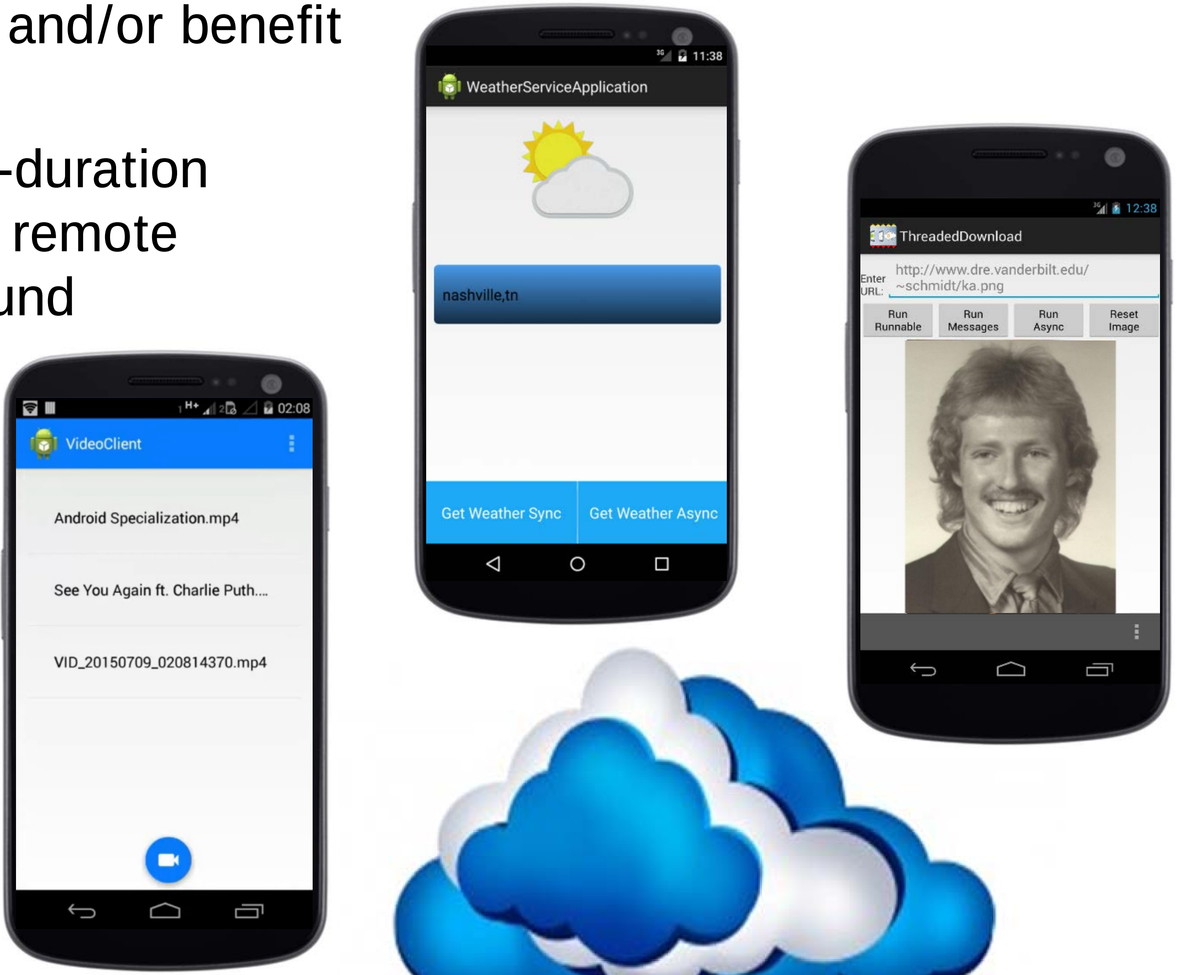
- Many Android apps require and/or benefit from concurrency



VideoClient
Android Specialization.mp4
See You Again ft. Charlie Puth....
VID_20150709_020814370.mp4
WeatherServiceApplication
nashville,tn
Get Weather Sync
Get Weather Async
ThreadedDownload
Enter URL:
http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image

# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
  - These apps perform long-duration operations and/or access remote resources in the background


VideoClient
Android Specialization.mp4
See You Again ft. Charlie Puth....
VID_20150709_020814370.mp4
WeatherServiceApplication
nashville,tn
Get Weather Sync
Get Weather Async
ThreadedDownload
Enter URL:
http://www.dre.vanderbilt.edu/~schmidt/ka.png
Run Runnable
Run Messages
Run Async
Reset Image

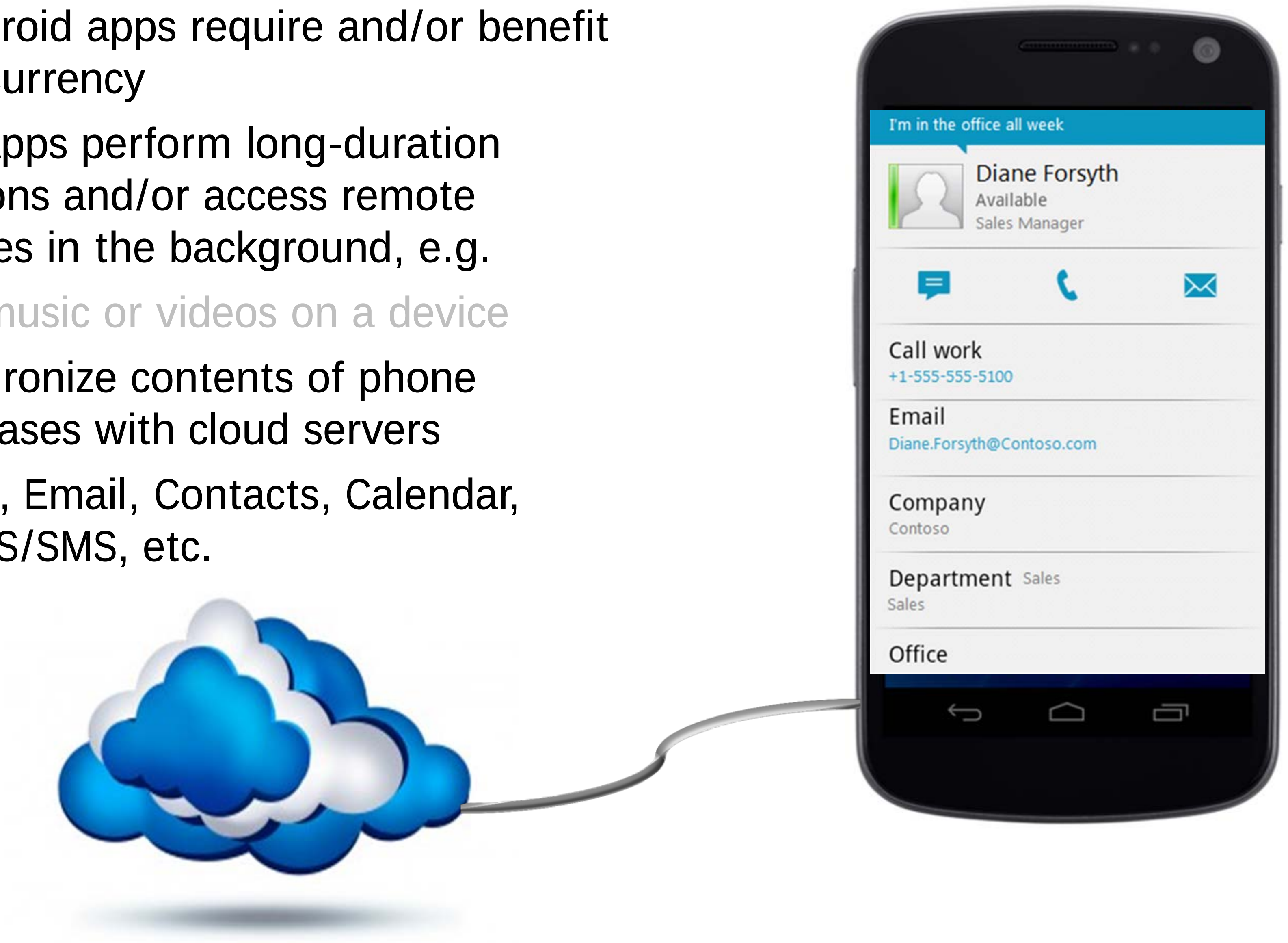
Often (but not always), apps interact with servers that reside in the cloud

# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
  - These apps perform long-duration operations and/or access remote resources in the background, e.g.
    - Play music or videos on a device

# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
  - These apps perform long-duration operations and/or access remote resources in the background, e.g.
    - Play music or videos on a device
    - Synchronize contents of phone databases with cloud servers
      - e.g., Email, Contacts, Calendar, MMS/SMS, etc.



I'm in the office all week

Diane Forsyth
Available
Sales Manager

Call work
+1-555-555-5100

Email
Diane.Forsyth@Contoso.com

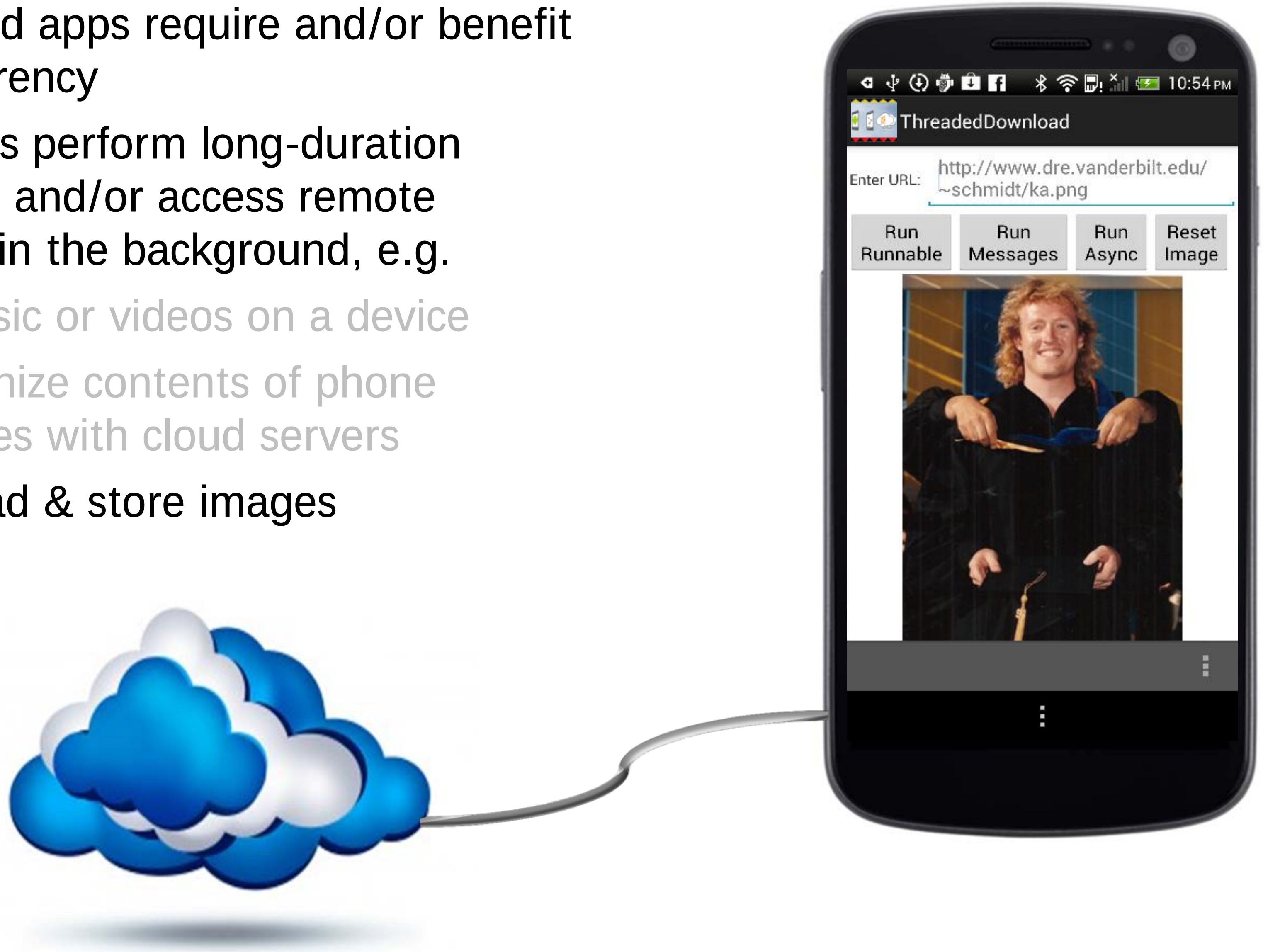
Company
Contoso

Department Sales
Sales

Office

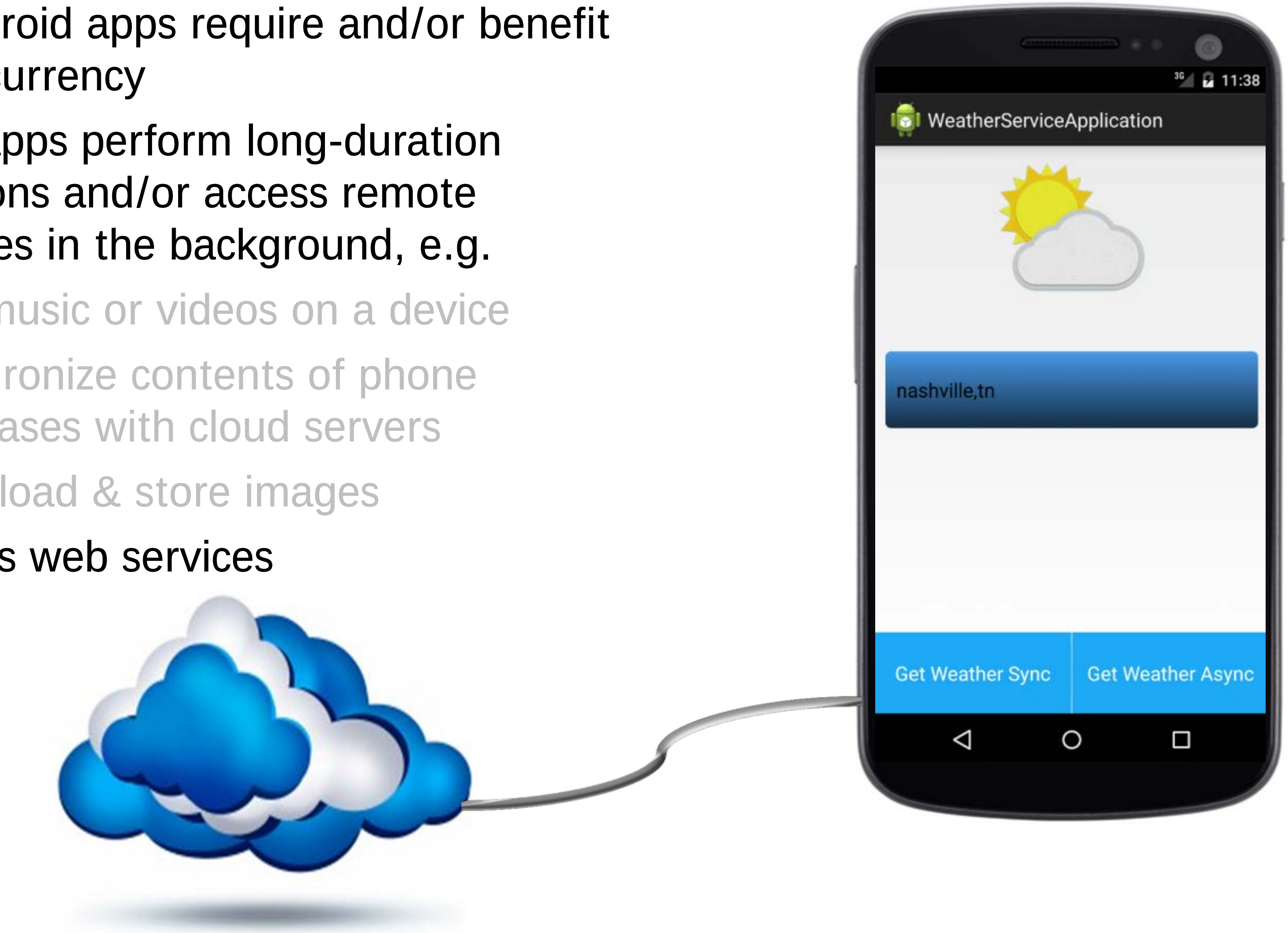
# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
  - These apps perform long-duration operations and/or access remote resources in the background, e.g.
    - Play music or videos on a device
    - Synchronize contents of phone databases with cloud servers
    - Download & store images



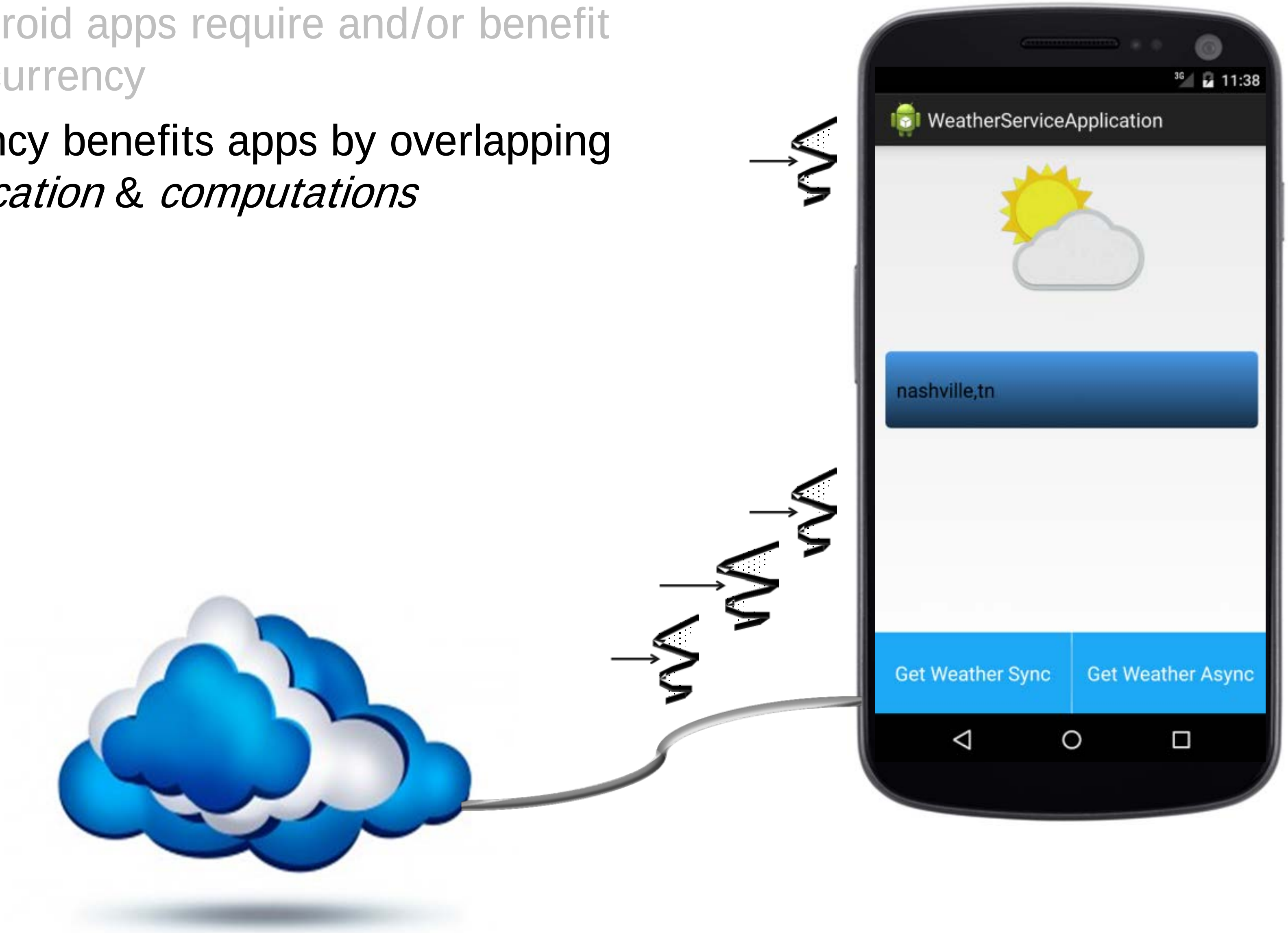
# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
  - These apps perform long-duration operations and/or access remote resources in the background, e.g.
    - Play music or videos on a device
    - Synchronize contents of phone databases with cloud servers
    - Download & store images
    - Access web services



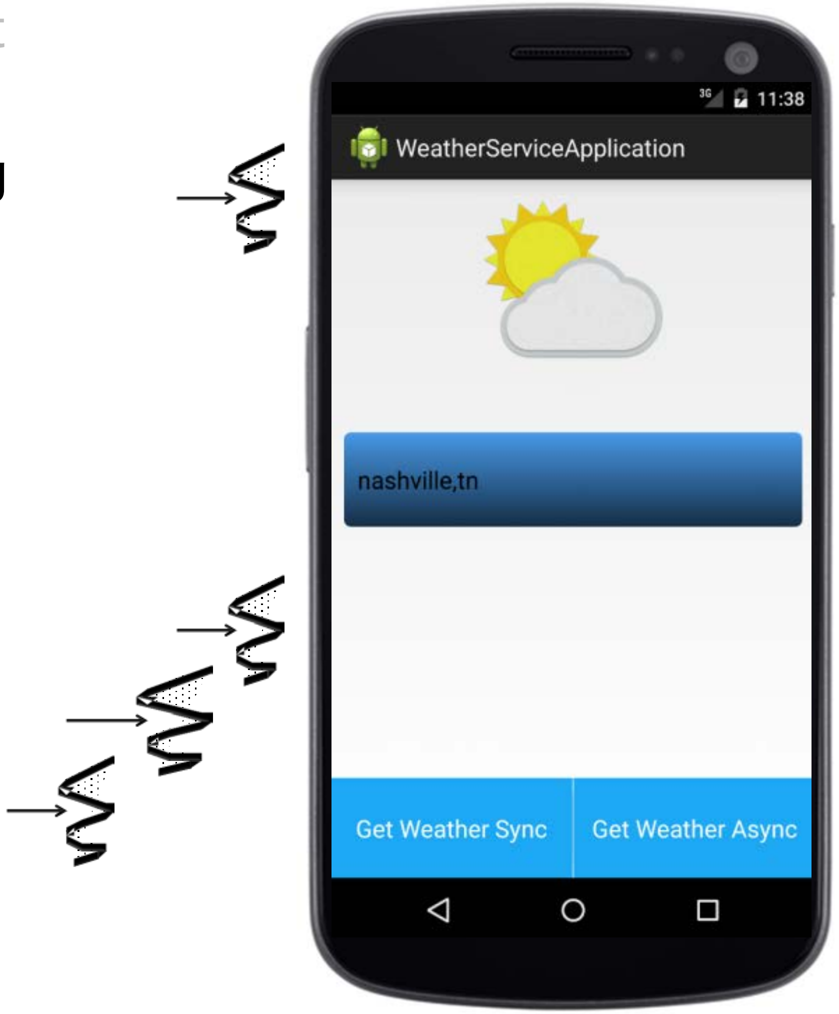
# Motivation for Android Concurrency

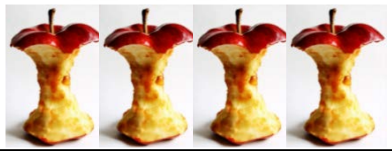
- Many Android apps require and/or benefit from concurrency
- Concurrency benefits apps by overlapping *communication* & *computations*



# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
- Concurrency benefits apps by overlapping *communication* & *computations*, e.g.
  - Increase performance via multi-core parallelism

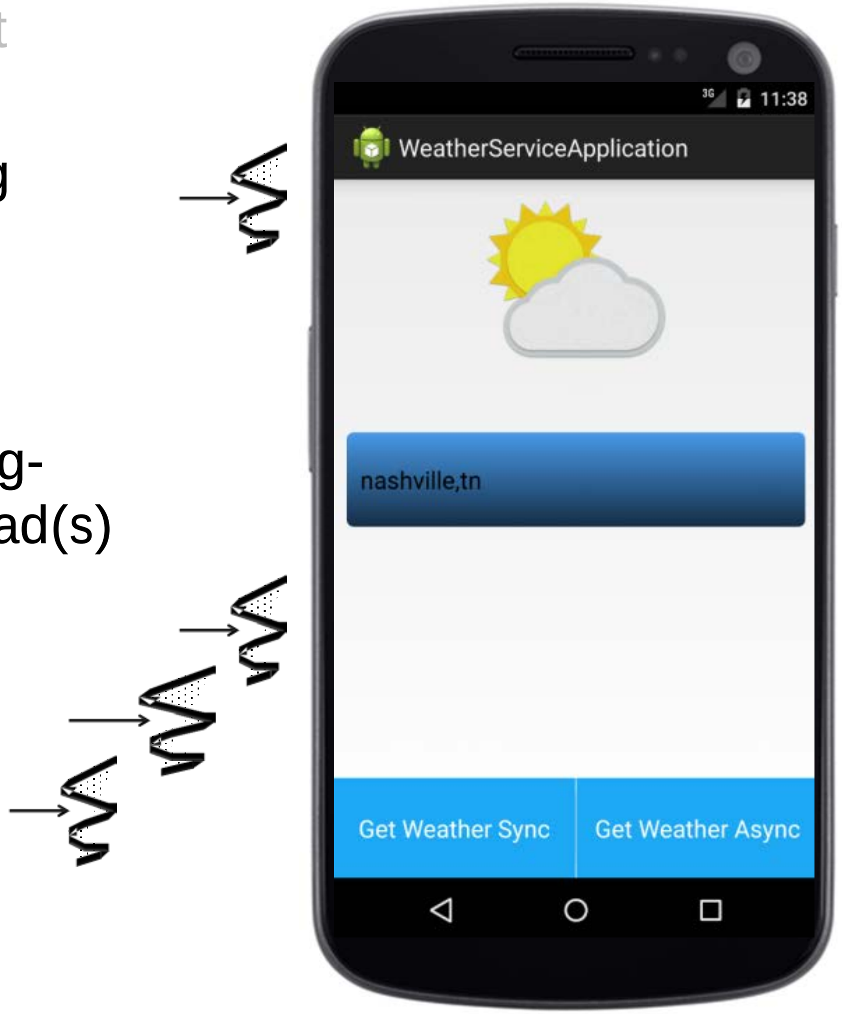


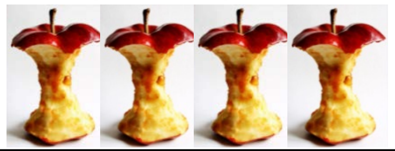


See developer.qualcomm.com/blog/multi-threading-android-apps-multi-core-processors-part-1-2

# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
- Concurrency benefits apps by overlapping *communication* & *computations*, e.g.
  - Increase performance via multi-core parallelism
  - Improve responsiveness by running long-duration operations in background thread(s)

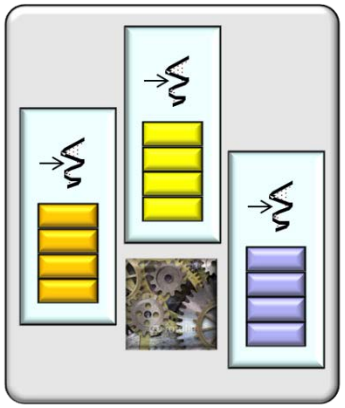


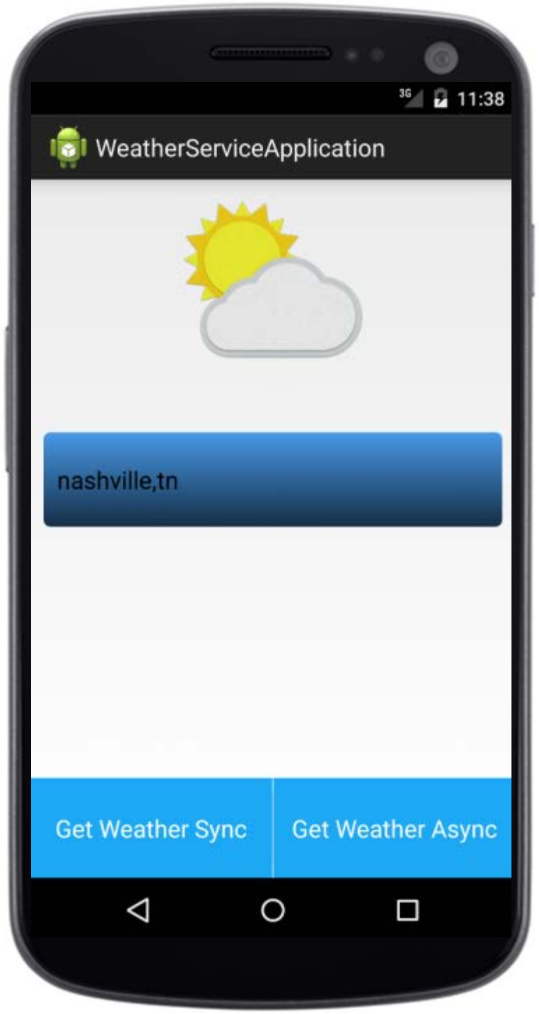


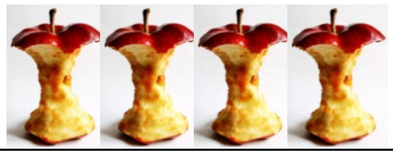
See [developer.android.com/training/articles/perf-anr.html](developer.android.com/training/articles/perf-anr.html)

## Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
- Concurrency benefits apps by overlapping *communication* & *computations*, e.g.
  - Increase performance via multi-core parallelism
  - Improve responsiveness by running long-duration operations in background thread(s)
  - Simplify program structure by allowing threads to block synchronously

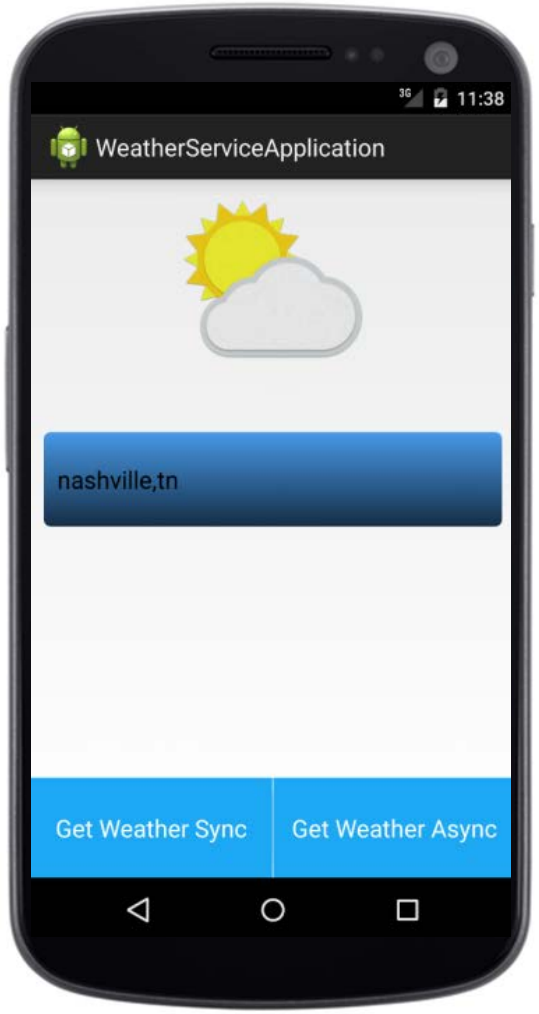


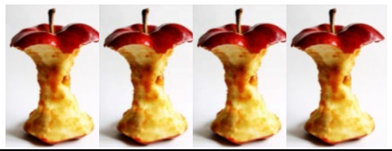




# Motivation for Android Concurrency

- Many Android apps require and/or benefit from concurrency
- Concurrency benefits apps by overlapping *communication* & *computations*, e.g.
  - Increase performance via multi-core parallelism
  - Improve responsiveness by running long-duration operations in background thread(s)
  - Simplify program structure by allowing threads to block synchronously
    - Can yield more natural control flow & collaboration within an app



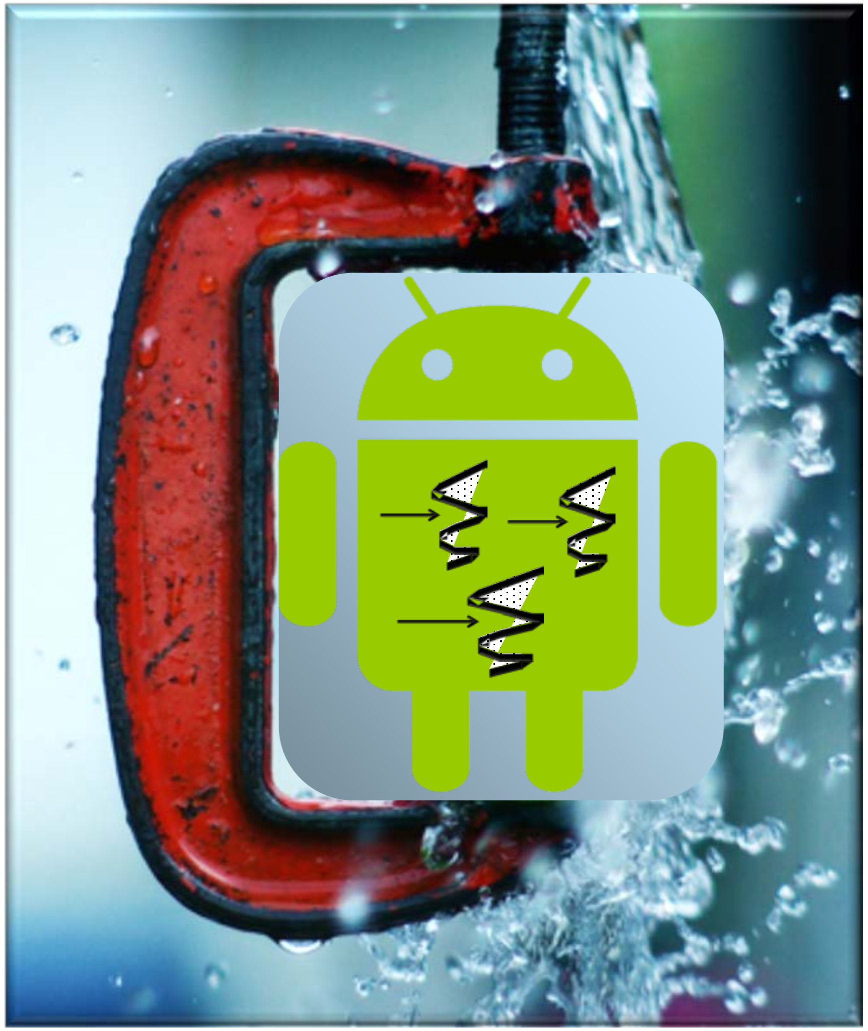


See [en.wikipedia.org/wiki/Control_flow](https://en.wikipedia.org/wiki/Control_flow)

## Motivating Android's Concurrency Frameworks

# Motivating Android Concurrency Frameworks

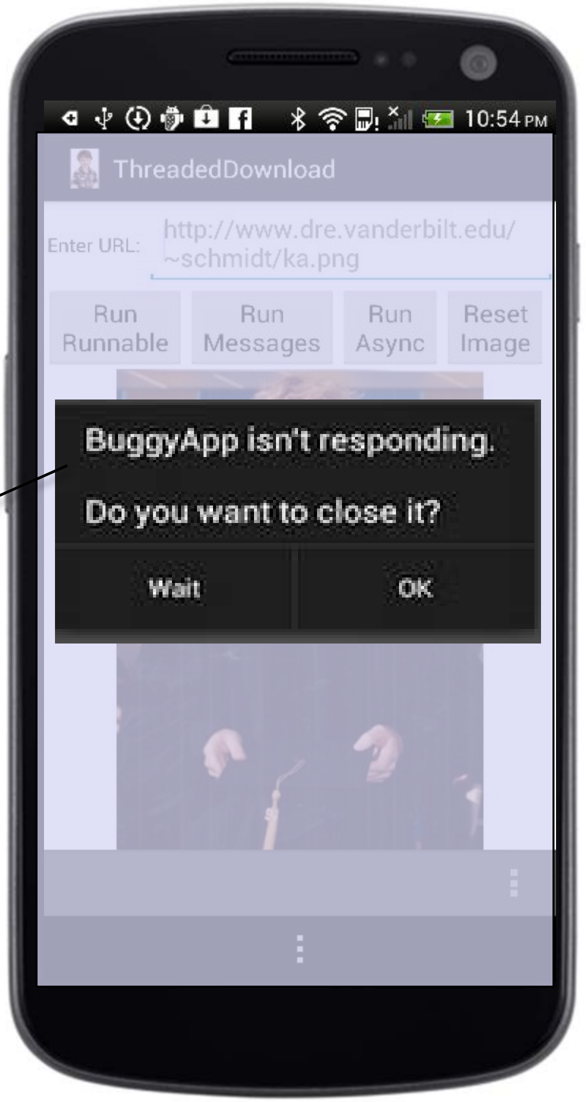
- These frameworks also address Android design constraints



See developer.android.com/guide/components/processes-and-threads.html#Threads

# Motivating Android Concurrency Frameworks

- These frameworks also address Android design constraints, e.g.
  - "ANR" dialog is generated if the UI thread blocks too long



BuggyApp isn't responding.

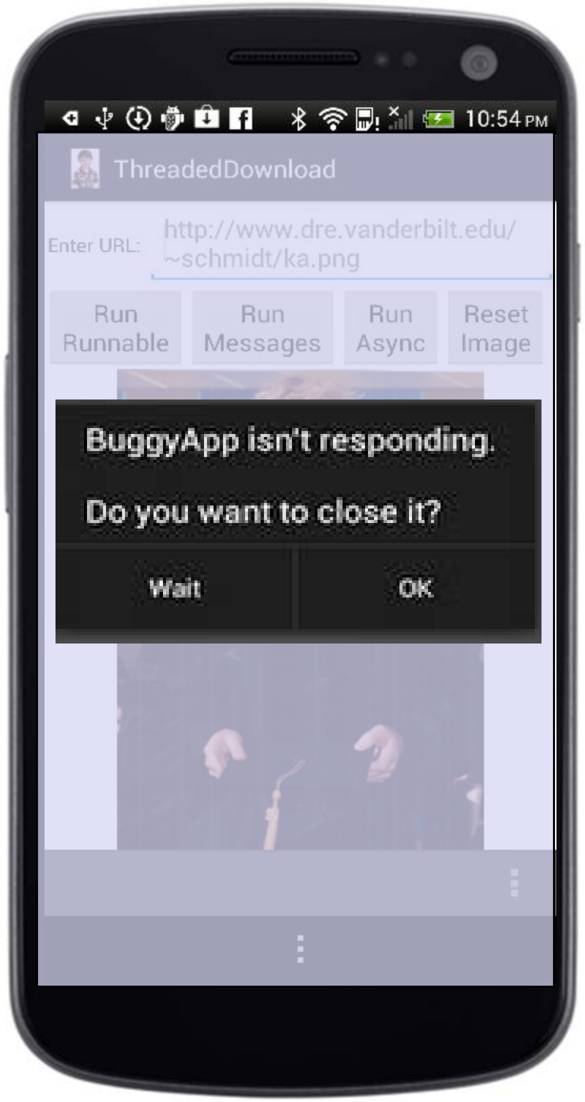
Do you want to close it?

*The UI thread can't block on long-duration operations for more than several seconds*

See [developer.android.com/training/articles/perf-anr.html](developer.android.com/training/articles/perf-anr.html)

# Motivating Android Concurrency Frameworks

- These frameworks also address Android design constraints, e.g.
  - "ANR" dialog is generated if the UI thread blocks too long
    - Network operations are disallowed on the UI thread by default



See [developer.android.com/reference/android/os/NetworkOnMainThreadException.html](developer.android.com/reference/android/os/NetworkOnMainThreadException.html)

## Motivating Android Concurrency Frameworks

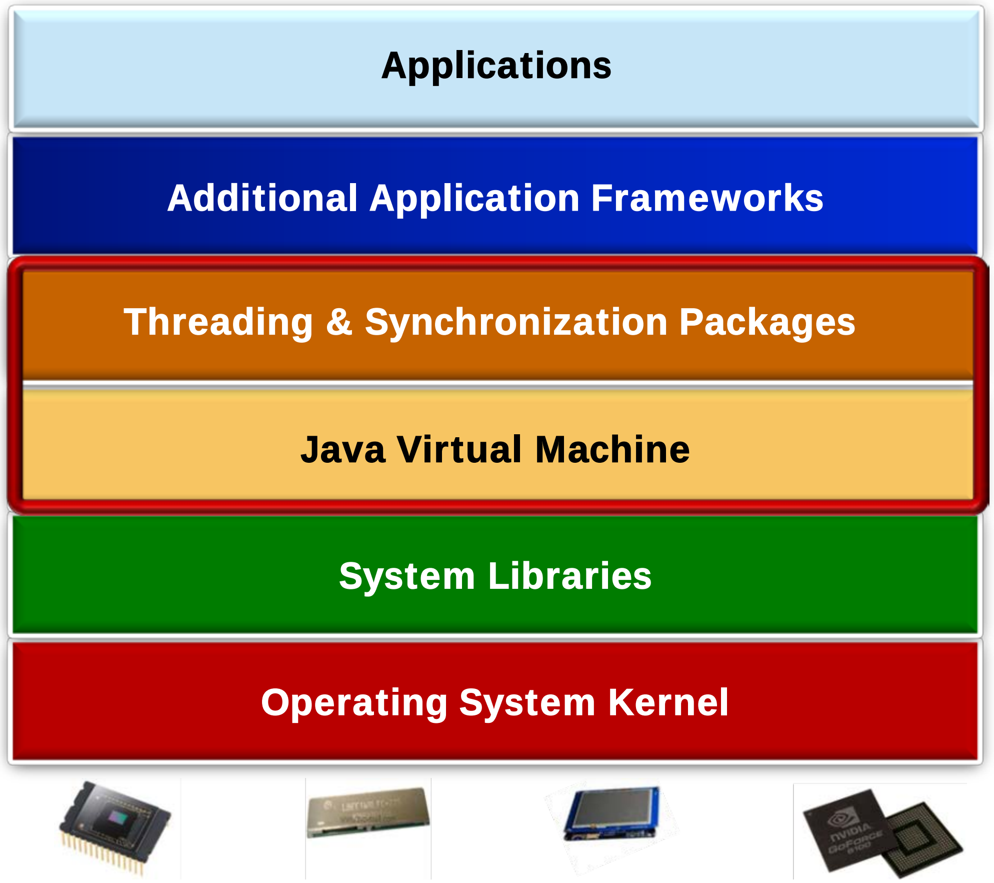
- These frameworks also address Android design constraints, e.g.
  - "ANR" dialog is generated if the UI thread blocks too long
  - Non-UI threads can't access UI toolkit components directly

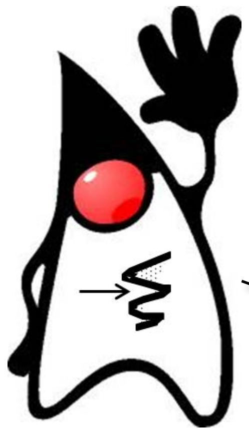
*UI toolkit components aren't thread-safe*

See android-developers.blogspot.com/2009/05/painless-threading.html

# Motivating Android Concurrency Frameworks

- These frameworks also address Android design constraints, e.g.
  - "ANR" dialog is generated if the UI thread blocks too long
  - Non-UI threads can't access UI toolkit components directly





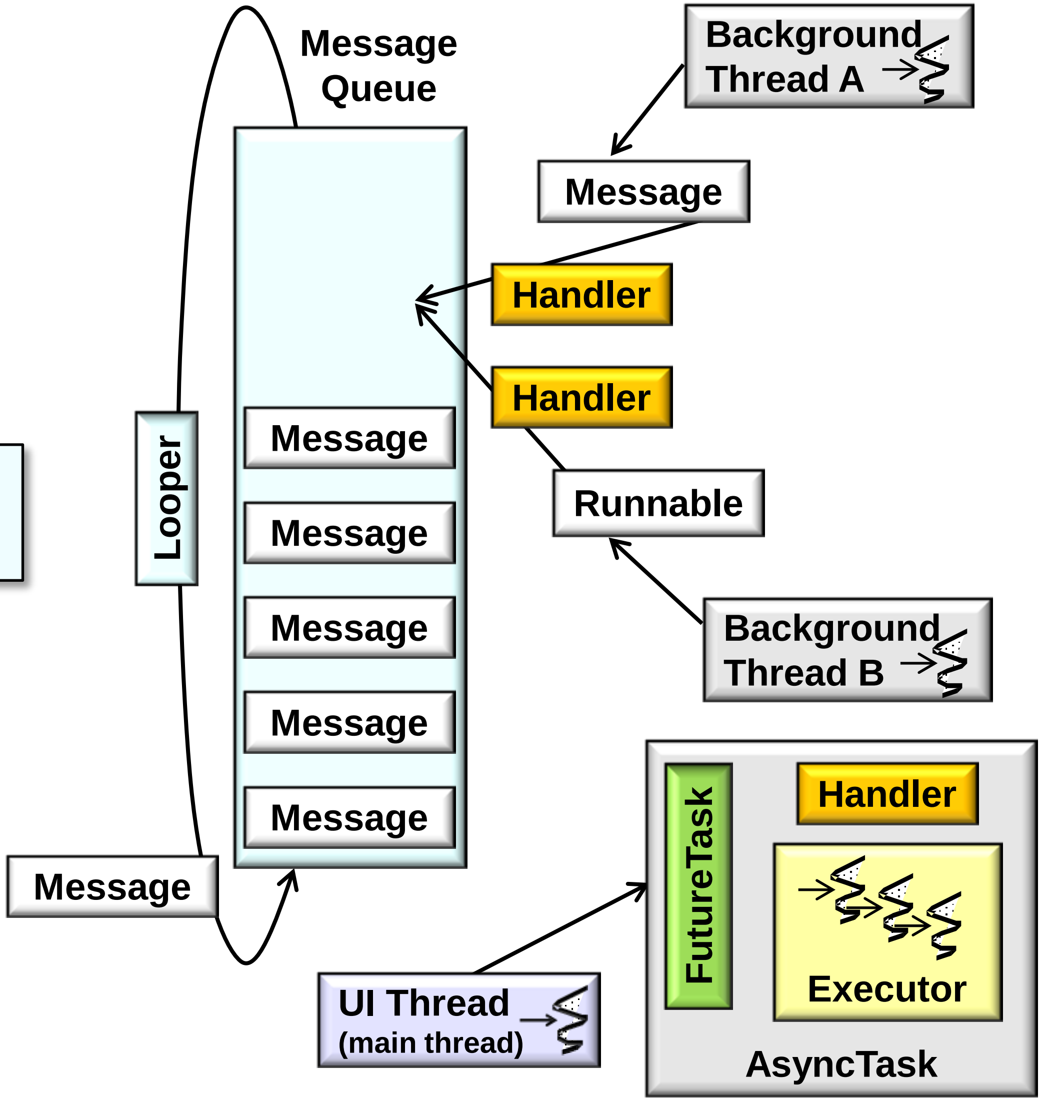
*Java concurrency mechanisms alone don't address these constraints*

See www.dre.vanderbilt.edu/~schmidt/LiveLessons/CPiJava

# Motivating Android Concurrency Frameworks

- These frameworks also address Android design constraints, e.g.
  - "ANR" dialog is generated if the UI thread blocks too long
  - Non-UI threads can't access UI toolkit components directly

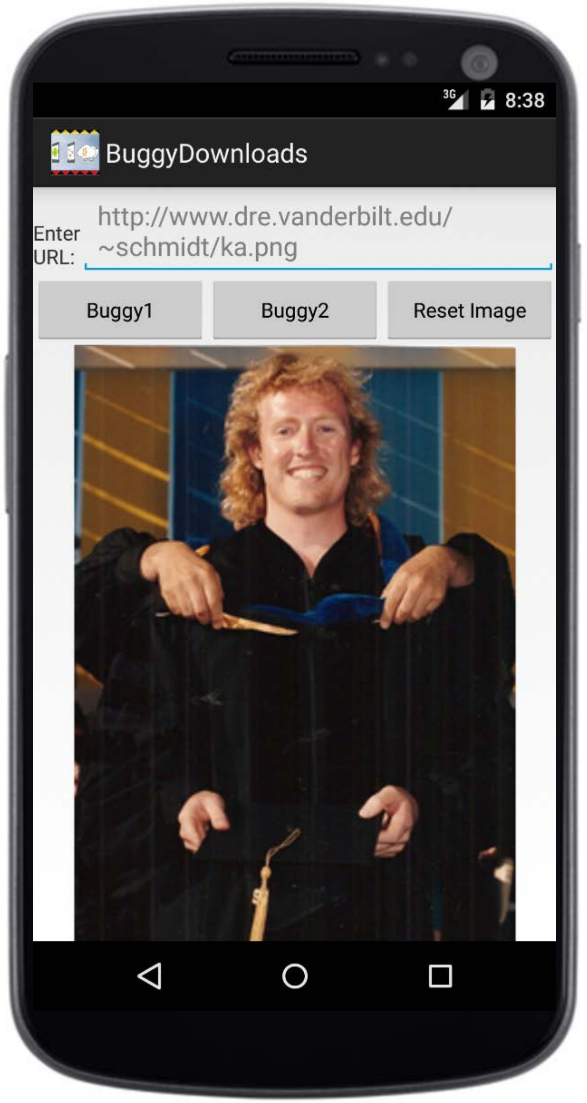
*Android concurrency frameworks address these design constraints*



See developer.android.com/guide/components/processes-and-threads.html#WorkerThreads

# Motivating Android Concurrency Frameworks

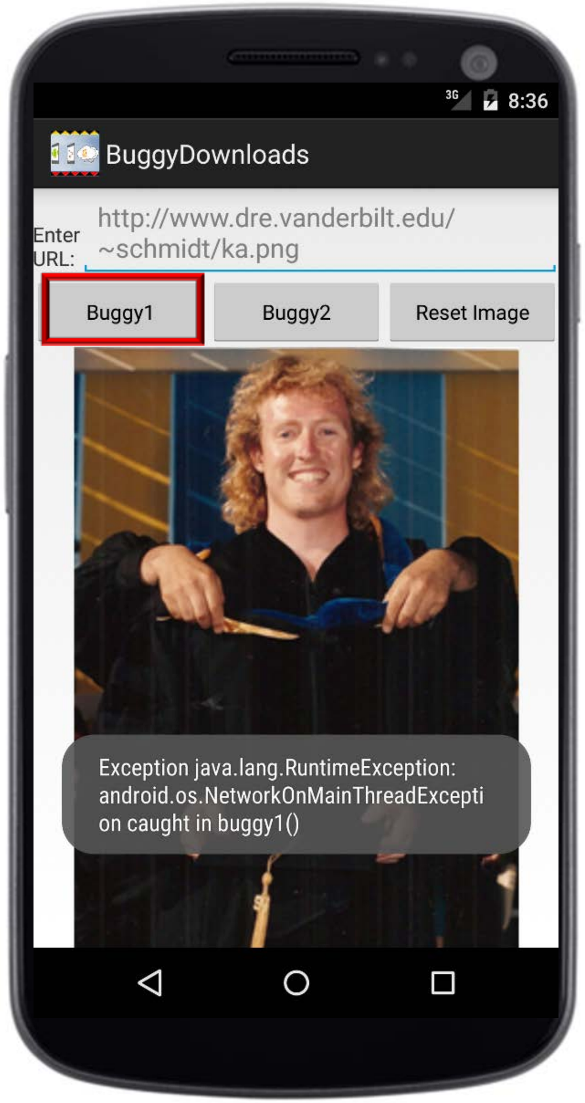
- These frameworks also address Android design constraints
- The “Buggy Downloads” app motivates the need for the Android concurrency frameworks



See github.com/douglascraigschmidt/POSA-15/tree/master/ex/BuggyDownloads

# Motivating Android Concurrency Frameworks

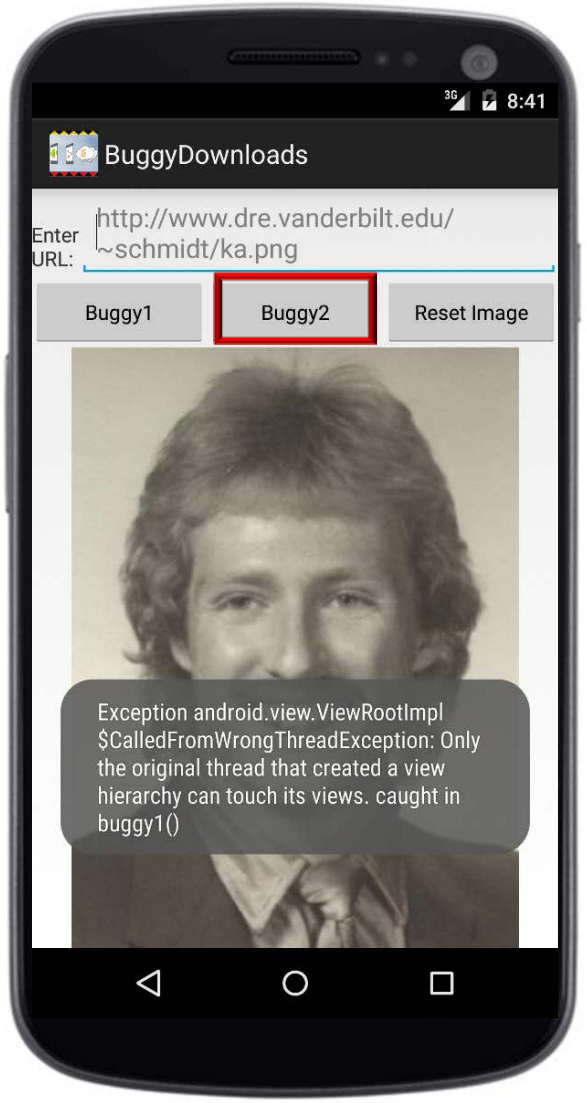
- These frameworks also address Android design constraints
- The “Buggy Downloads” app motivates the need for the Android concurrency frameworks
  - “Buggy1” throws an exception since the image is downloaded in the UI thread



Exception java.lang.RuntimeException: android.os.NetworkOnMainThreadException caught in buggy1()

# Motivating Android Concurrency Frameworks

- These frameworks also address Android design constraints
- The "Buggy Downloads" app motivates the need for the Android concurrency frameworks
  - "Buggy1" throws an exception since the image is downloaded in the UI thread
  - "Buggy2" throws an exception since a UI component is accessed via a background thread



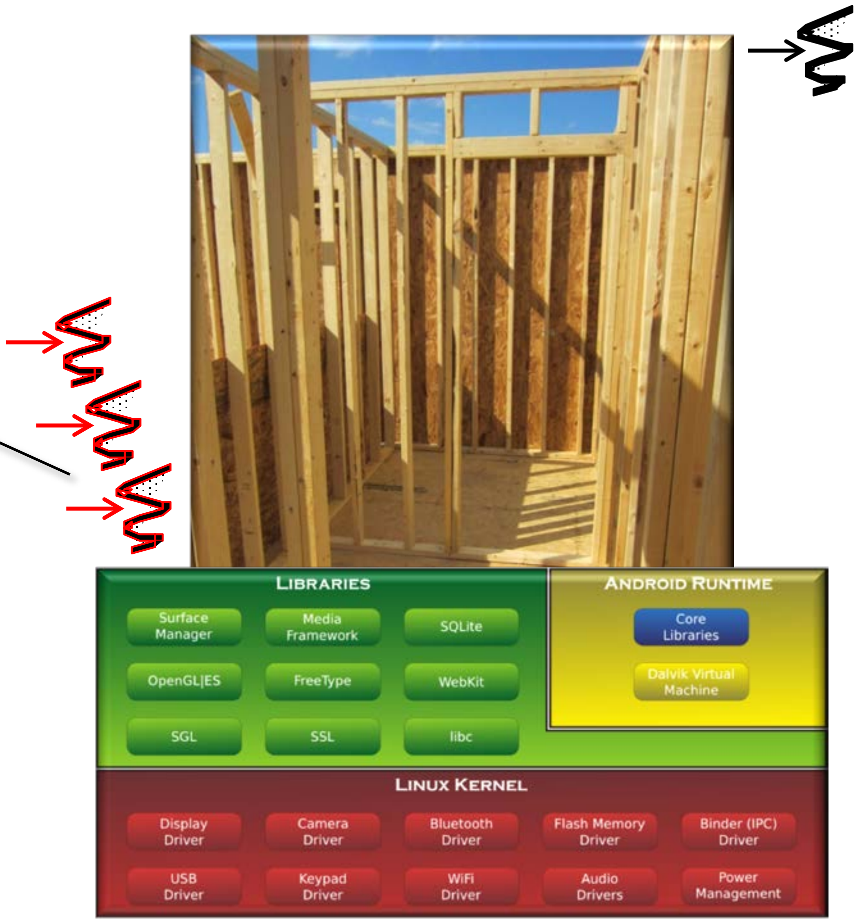
Exception android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views. caught in buggy1()

## Overview of Android Concurrency Frameworks

# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication

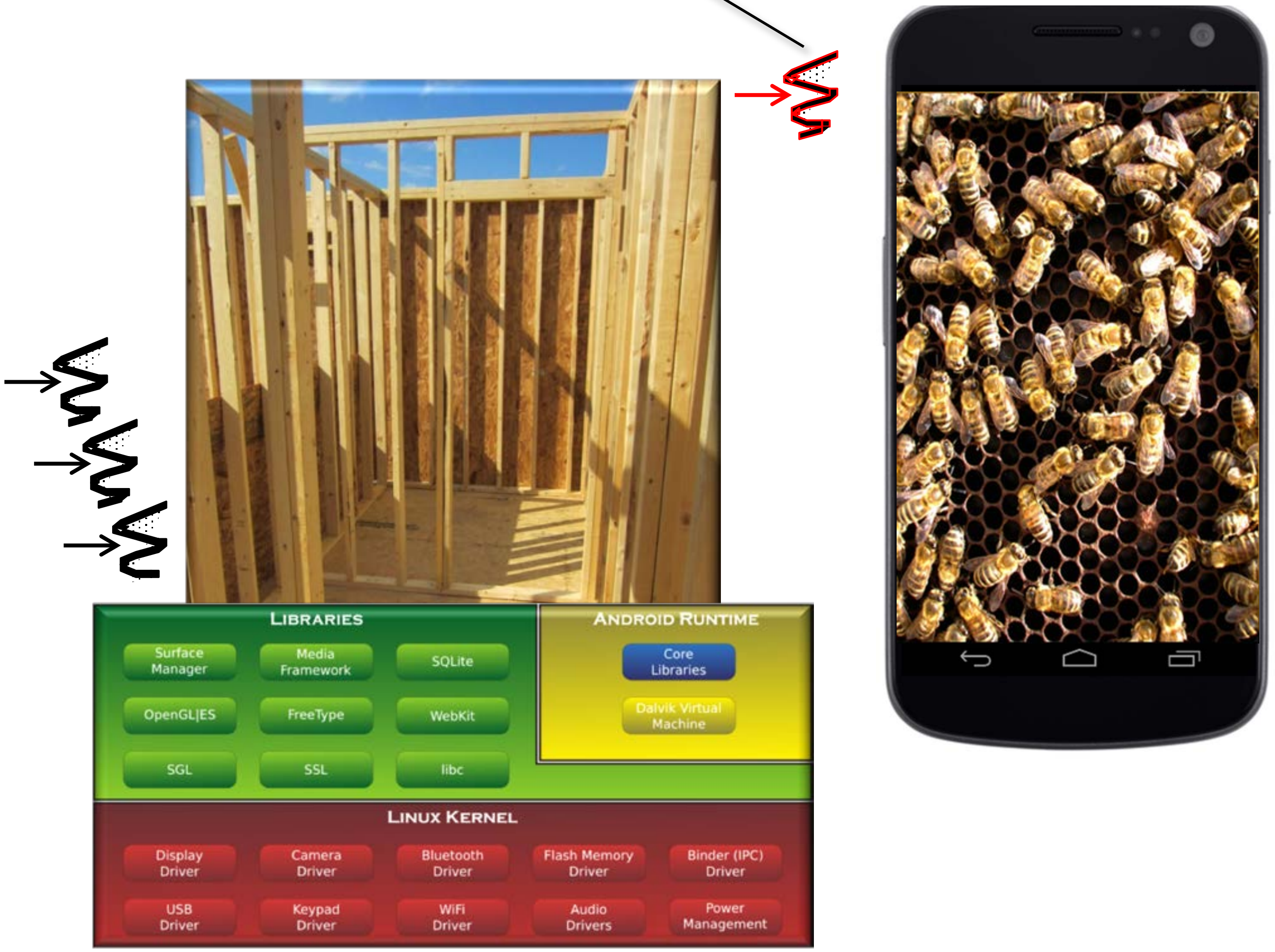
*Long-duration & (potentially) blocking operations run in background Thread(s)*



# Overview of Android Concurrency Frameworks

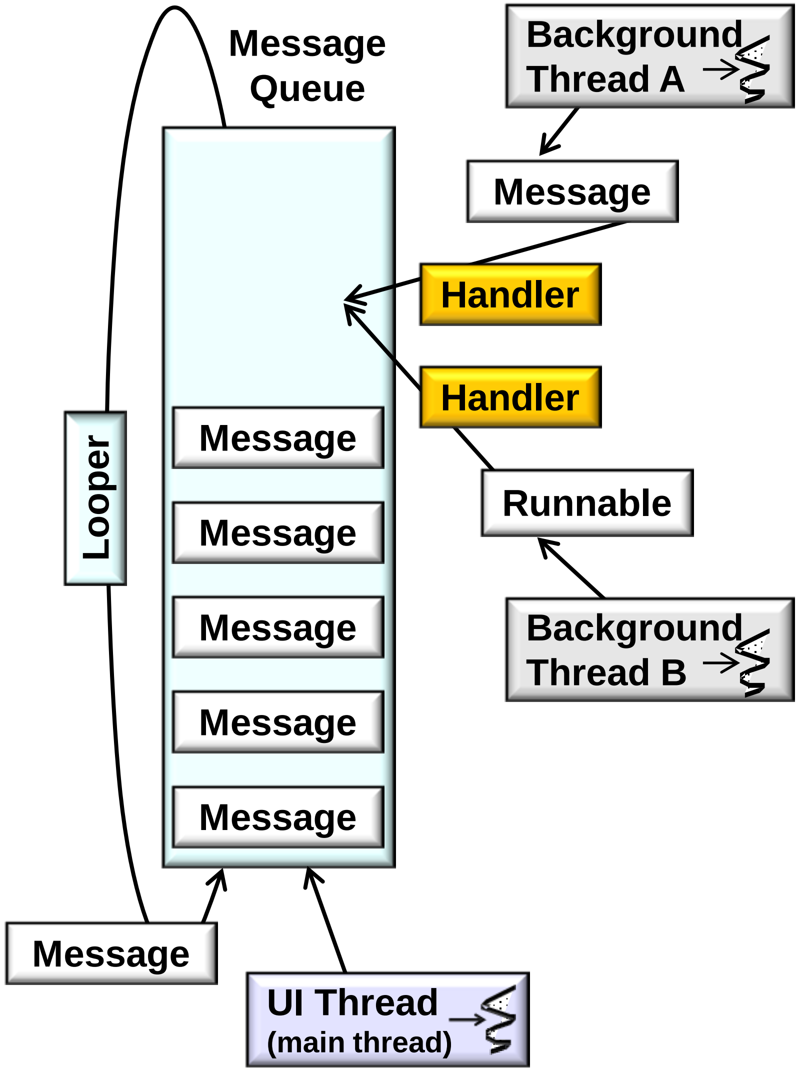
- Decouple computation(s) & communication

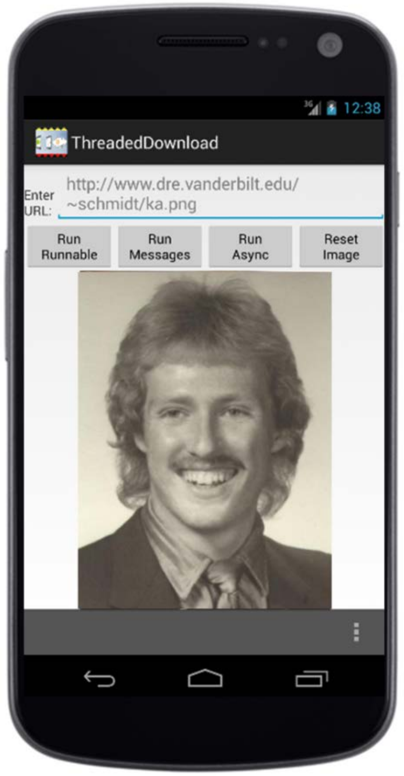
*Short-duration, user-facing operations run in the UI Thread*



# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication

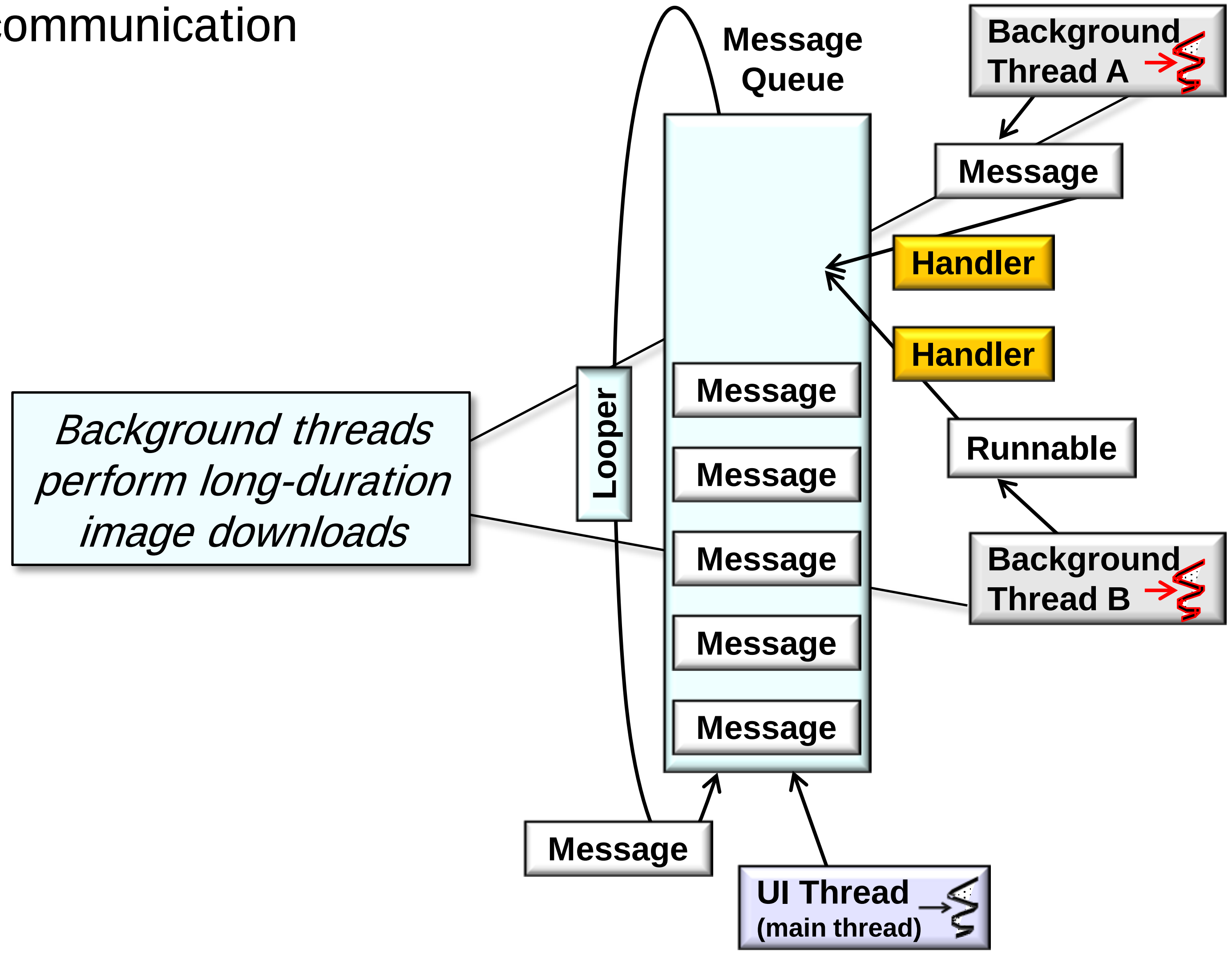


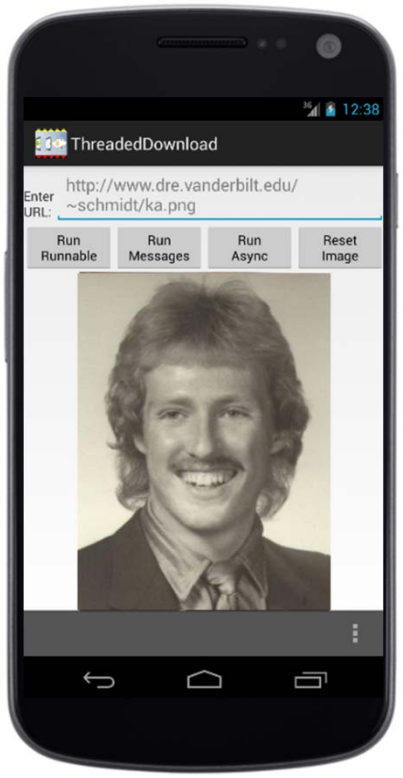


See [github.com/douglascraigschmidt/POSA-15/tree/master/ex/SimpleImageDownloads](github.com/douglascraigschmidt/POSA-15/tree/master/ex/SimpleImageDownloads)

# Overview of Android Concurrency Frameworks

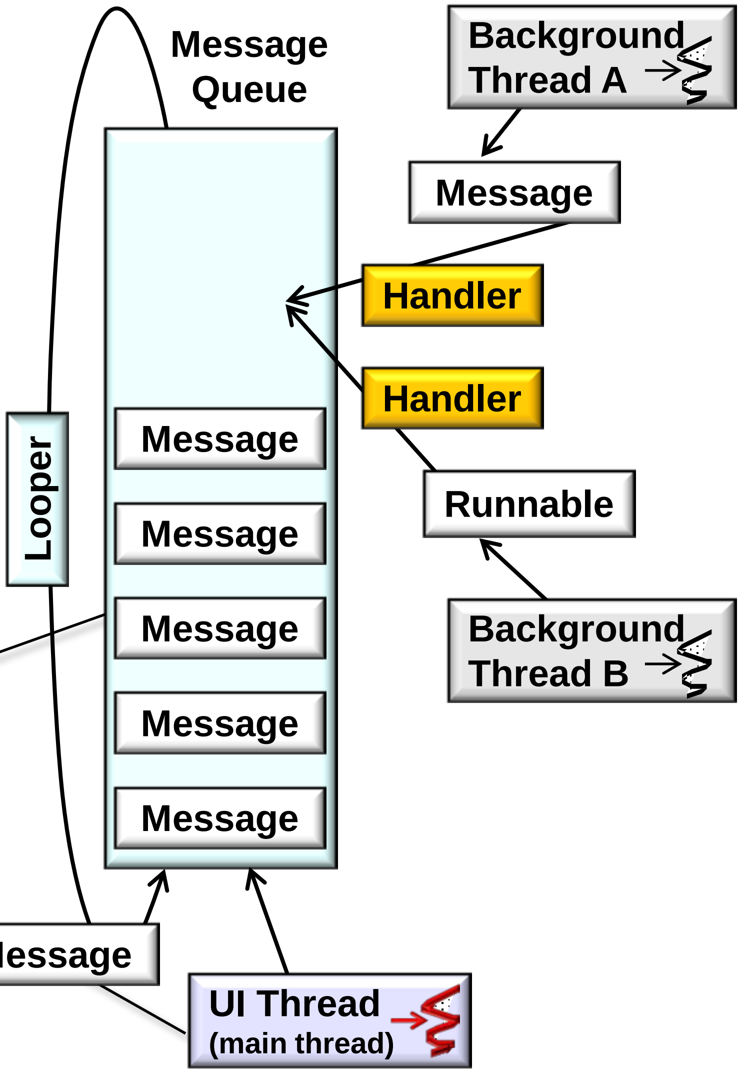
- Decouple computation(s) & communication



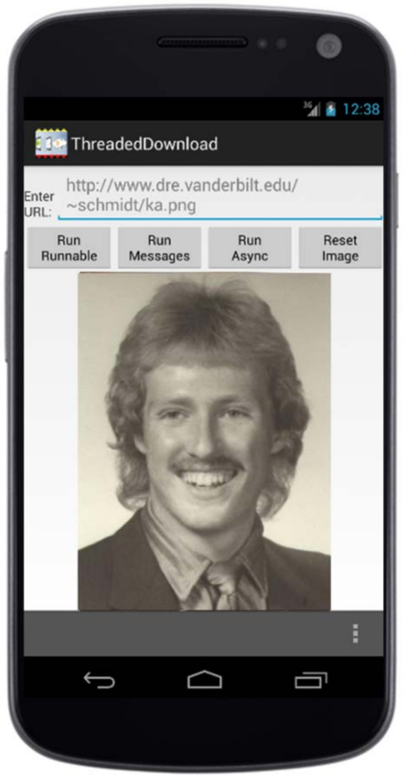


# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication

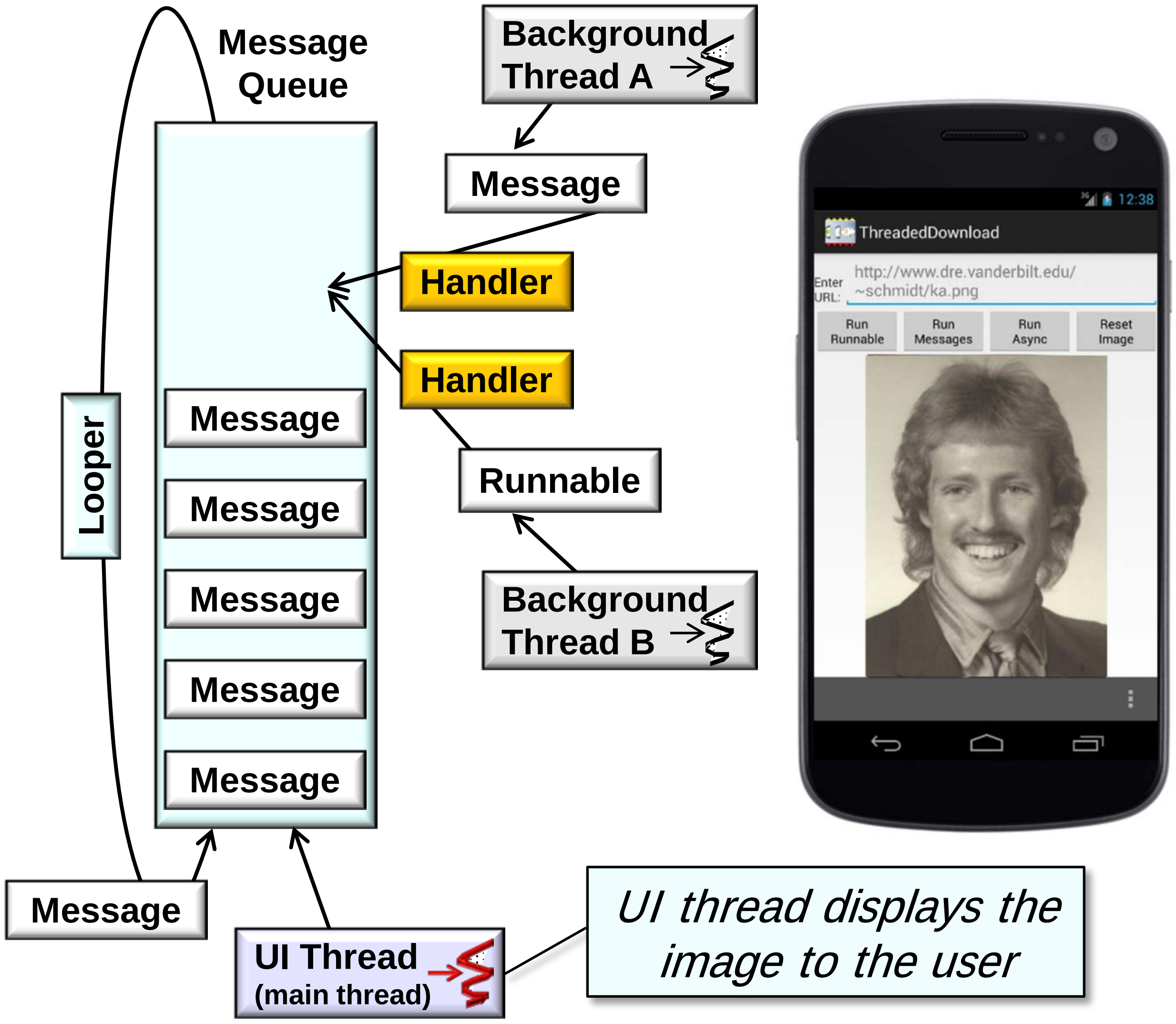


*Synchronized message queue passes results from background Thread(s) to UI thread*



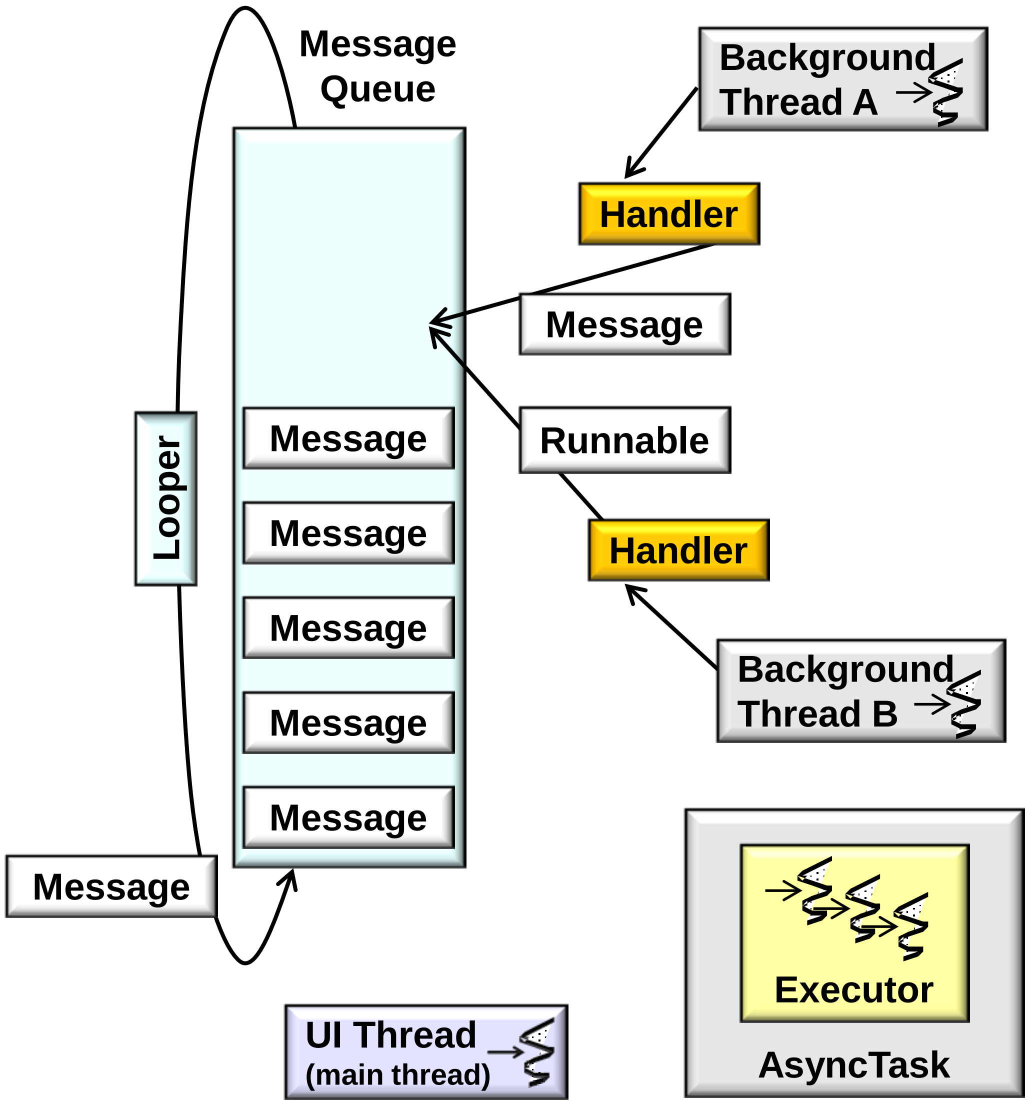
# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication



# Overview of Android Concurrency Frameworks

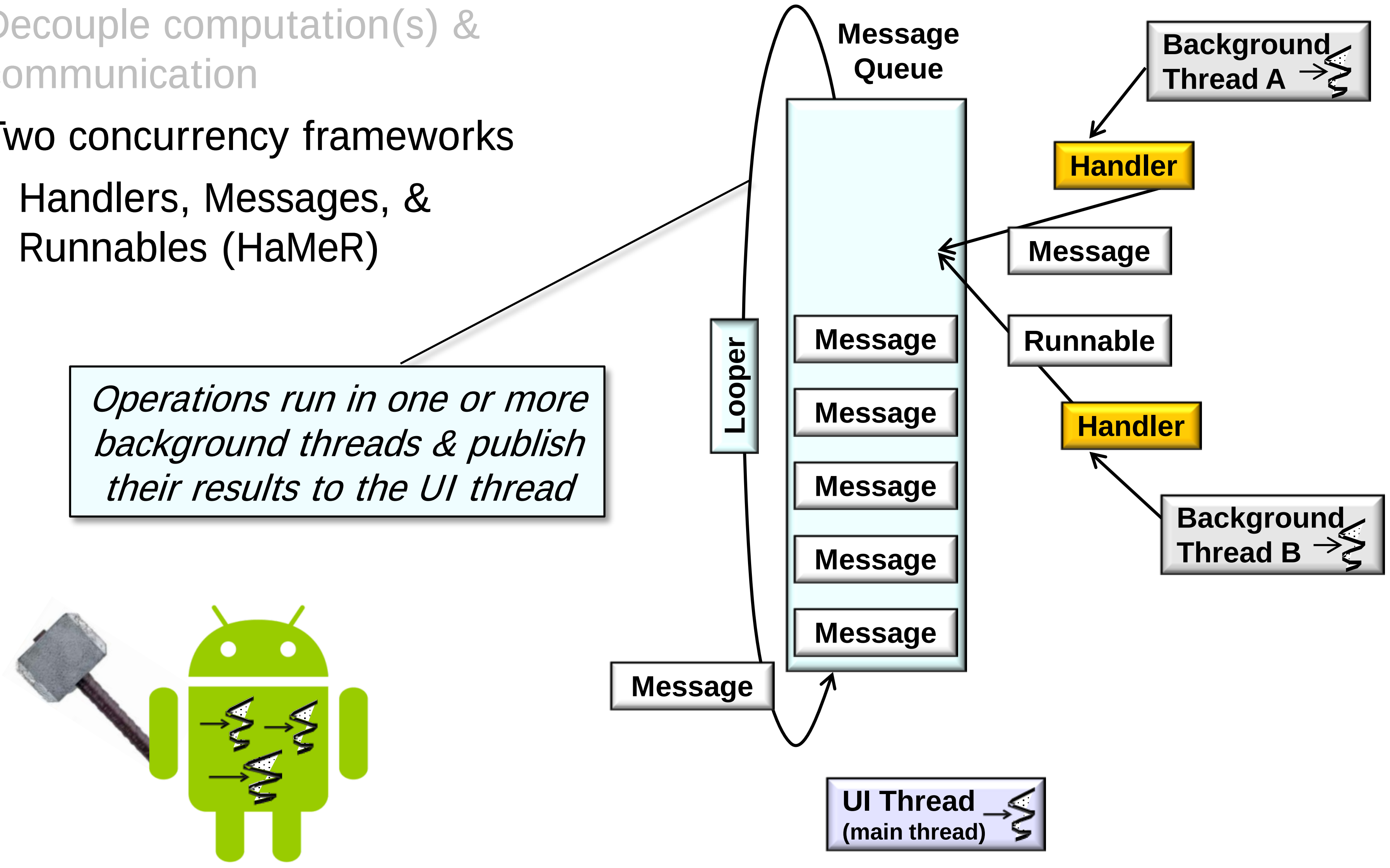
- Decouple computation(s) & communication
- Two concurrency frameworks



# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
  - Handlers, Messages, & Runnables (HaMeR)

*Operations run in one or more background threads & publish their results to the UI thread*

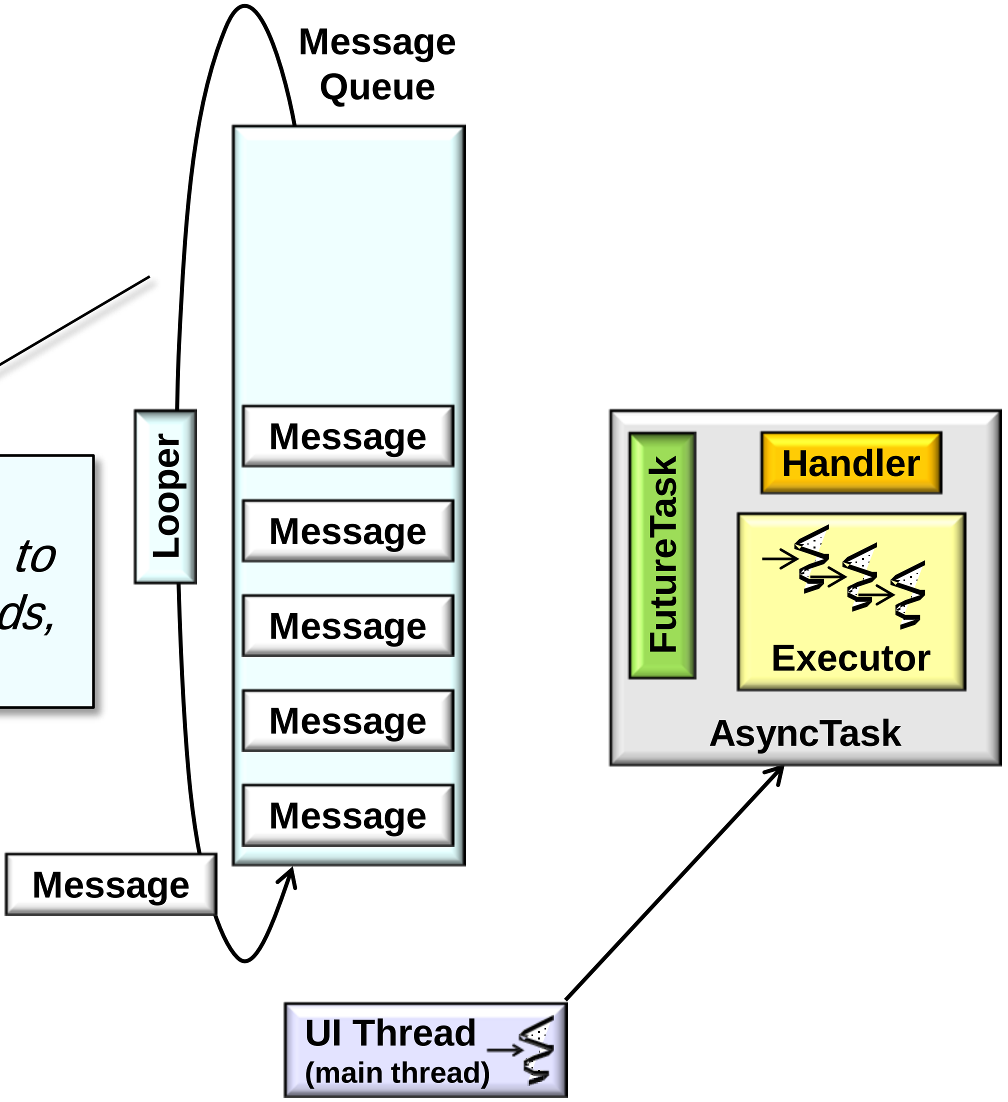


See [developer.android.com/training/multiple-threads/communicate-ui.html](developer.android.com/training/multiple-threads/communicate-ui.html)

# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
  - Handlers, Messages, & Runnables (HaMeR)
  - AsyncTask

*Operations run in one or more background threads & publish results to UI thread without manipulating threads, handlers, messages, or runnables*

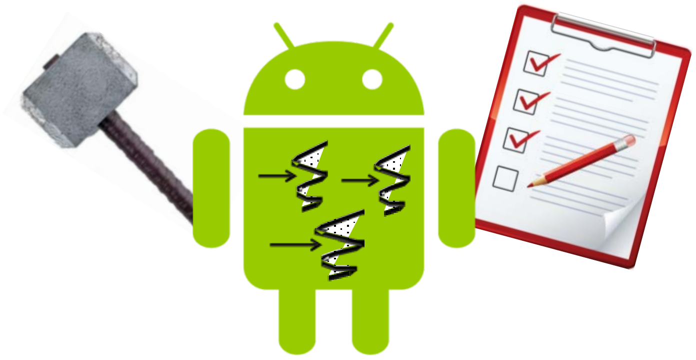


See [developer.android.com/reference/android/os/AsyncTask.html](developer.android.com/reference/android/os/AsyncTask.html)

# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android

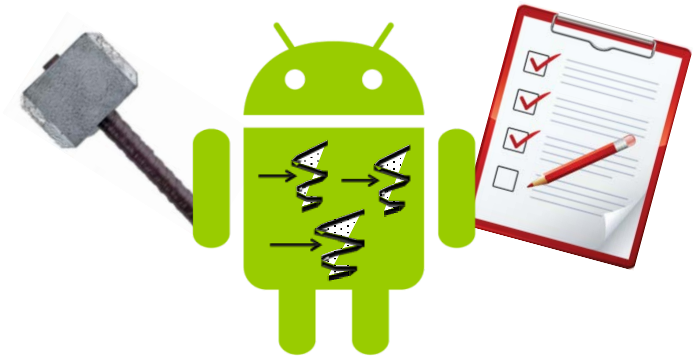
| | Async Task | Posting Runnables | Sending Messages |
|---|---|---|---|
| **Usability (Simple)** | +++ | +++ | ++ |
| **Usability (Complex)** | +++ | + | ++ |
| **Scalability** | +++ | + | + |
| **Flexibility** | ++ | + | +++ |
| **Efficiency** | ++ | +++ | +++ |

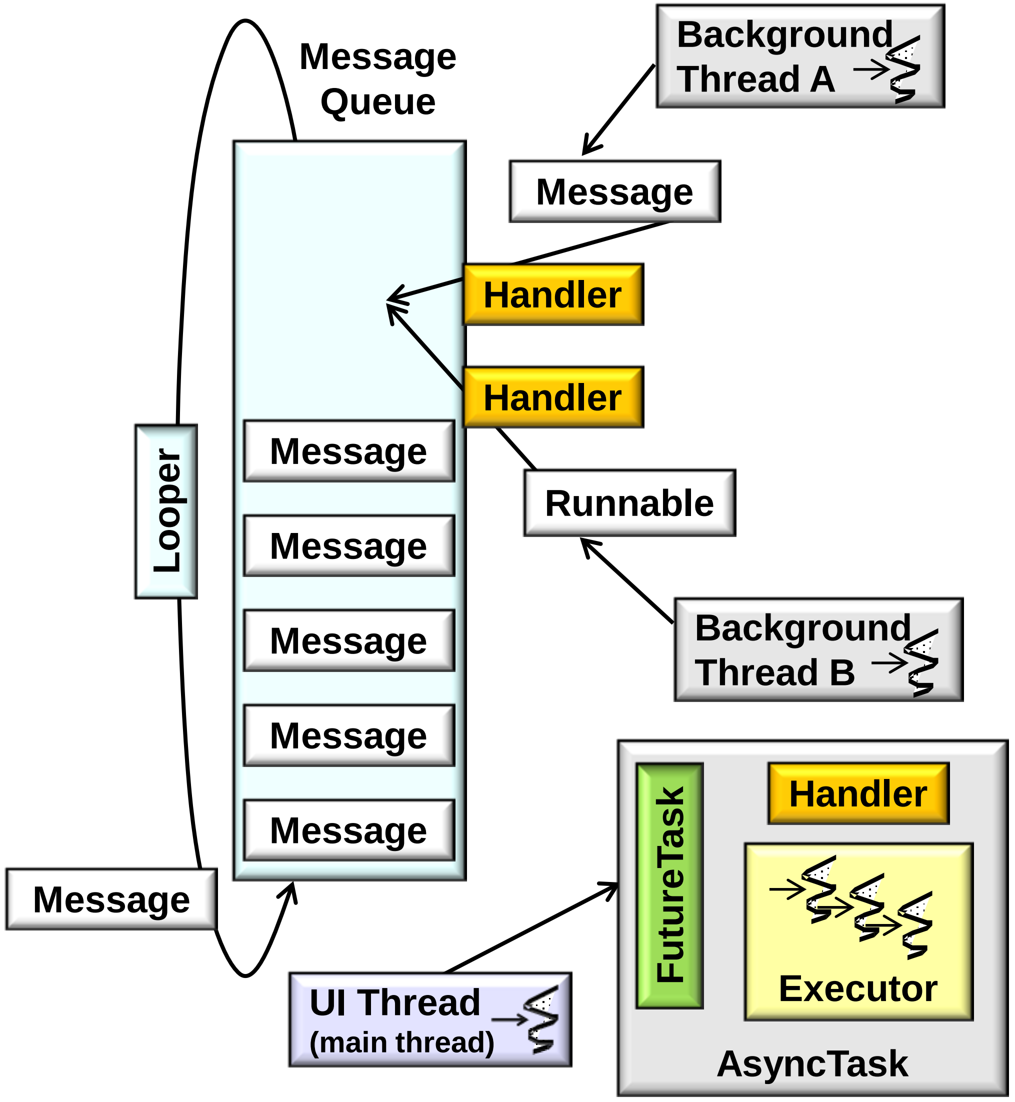


See upcoming part on "Evaluating Android's Concurrency Frameworks"

# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*

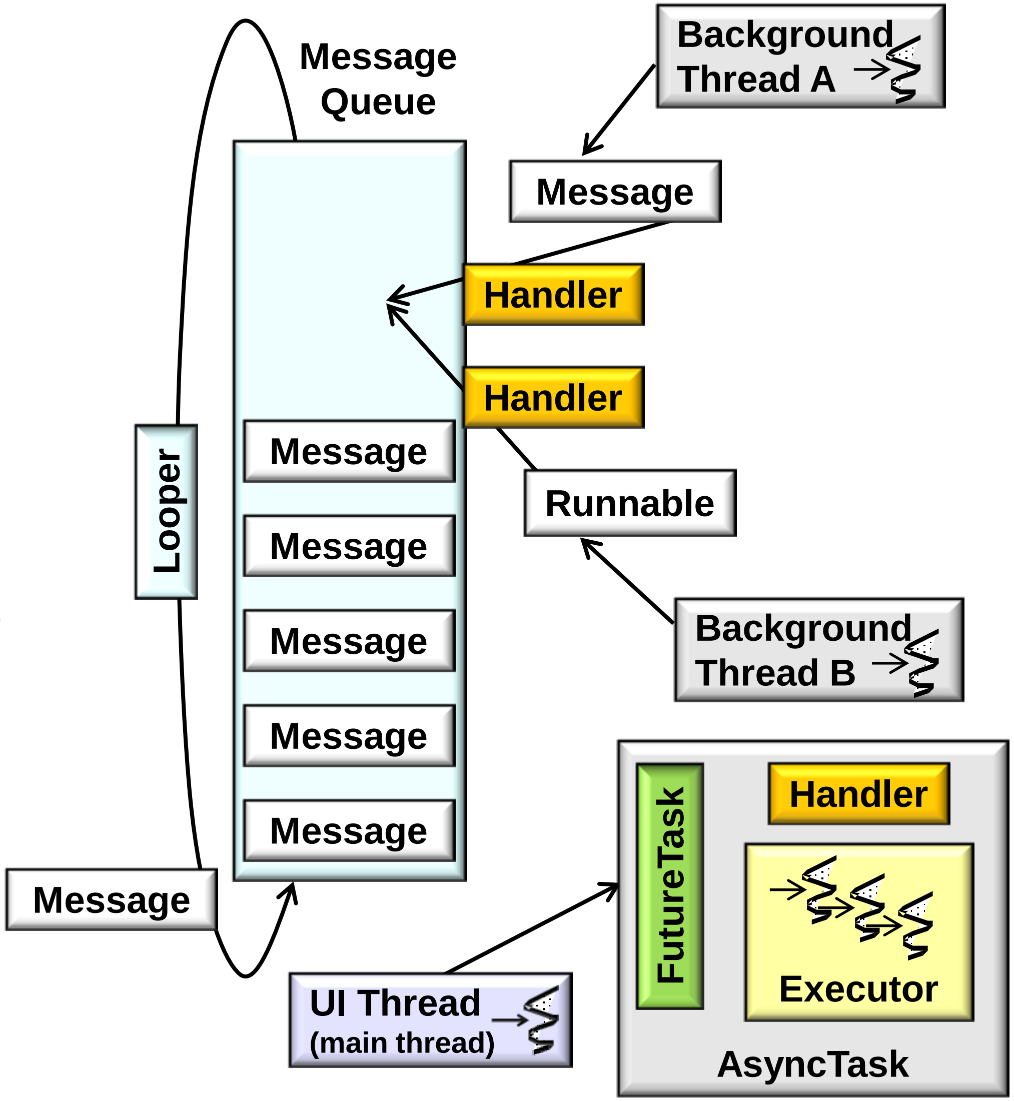




See [en.wikipedia.org/wiki/Programming_idiom](en.wikipedia.org/wiki/Programming_idiom)

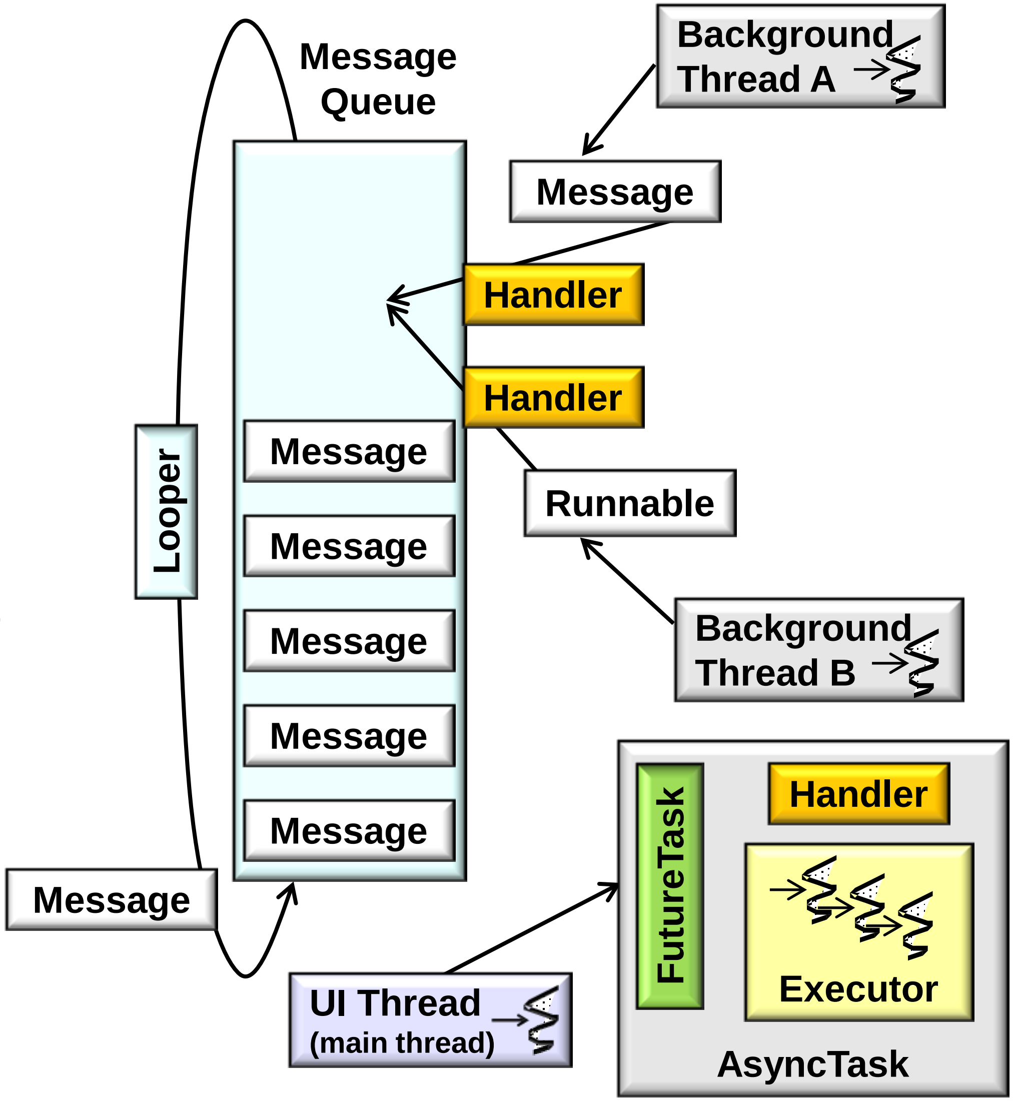
# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*
  - An idiom is a pattern specific to a certain context, such as a design method, platform, or language



# Overview of Android Concurrency Frameworks

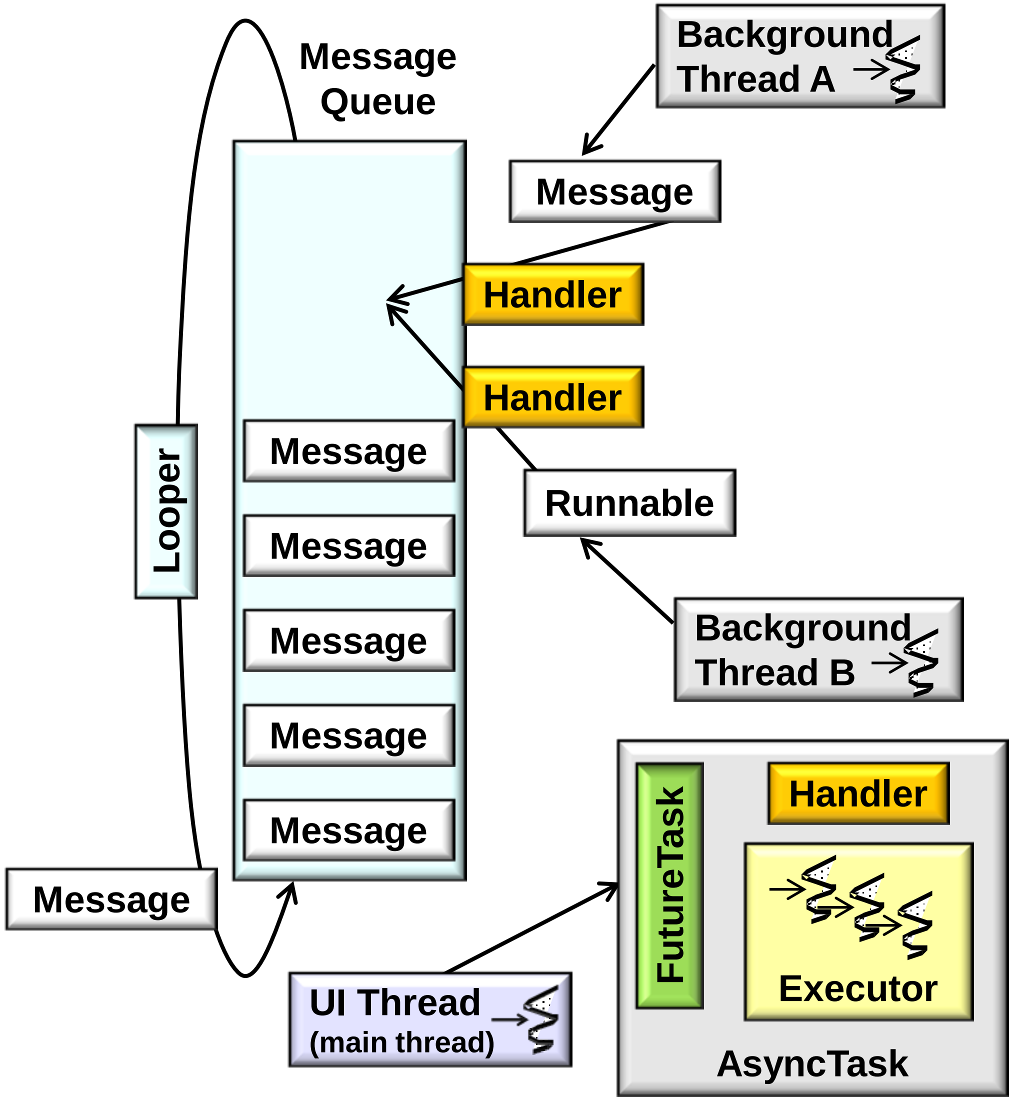
- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*
  - An idiom is a pattern specific to a certain context, such as a design method, platform, or language
    - e.g., Messages passed between threads via sendToTarget()



See developer.android.com/reference/android/os/Message.html#sendToTarget()

# Overview of Android Concurrency Frameworks

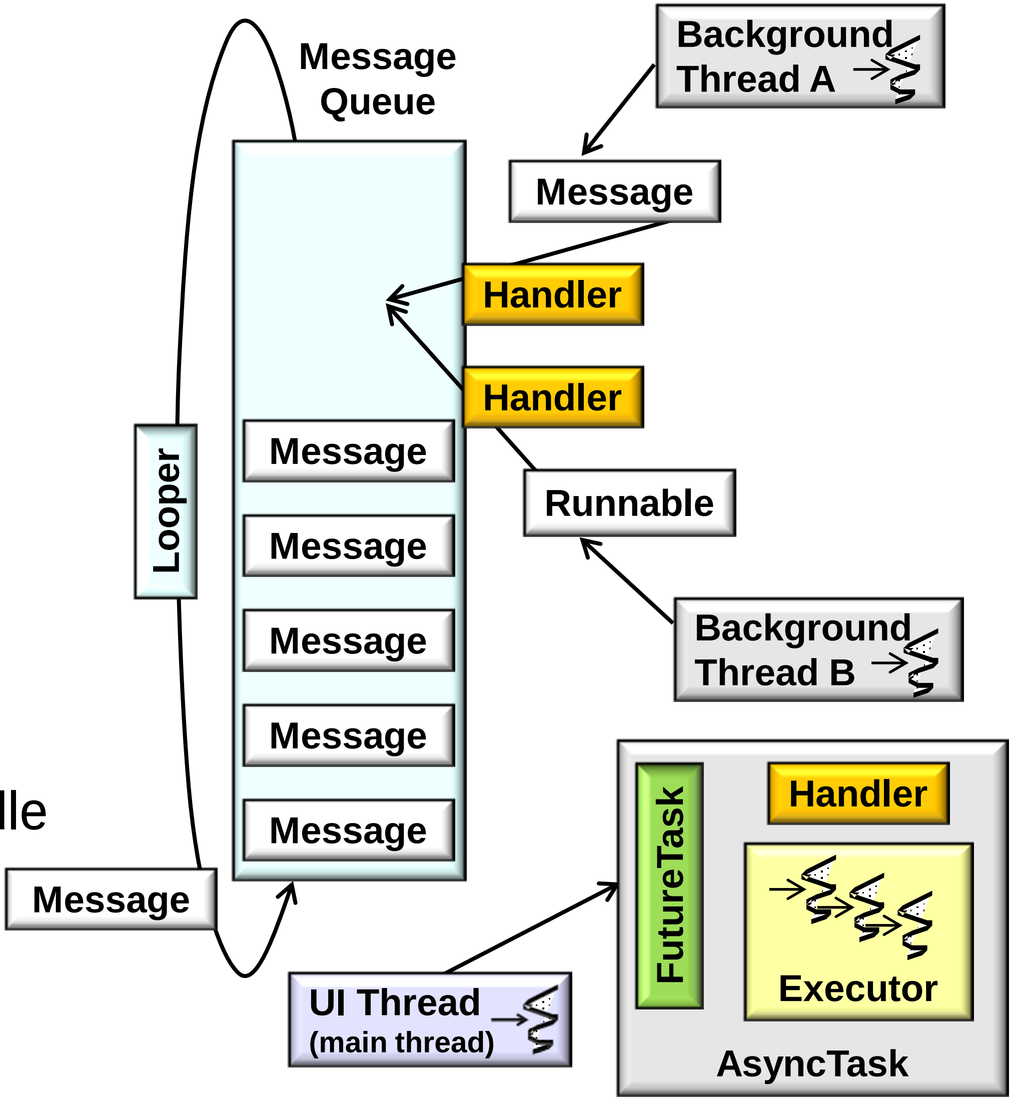
- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*
- Patterns/idioms also needed to program with Android's concurrency frameworks



See [developer.android.com/guide/topics/resources/runtime-changes.html](developer.android.com/guide/topics/resources/runtime-changes.html)

# Overview of Android Concurrency Frameworks

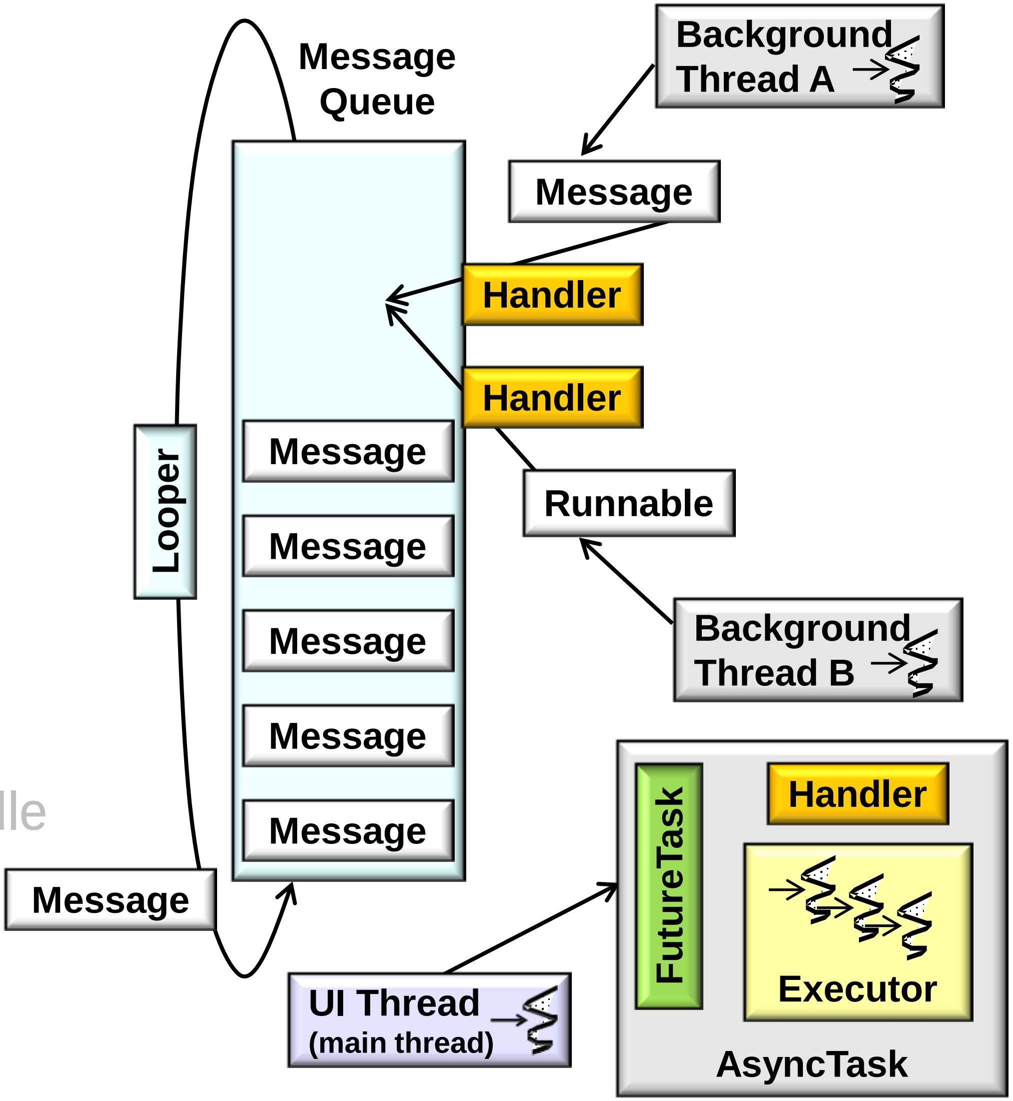
- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*
- Patterns/idioms also needed to program with Android's concurrency frameworks, e.g.
  - Use *MVP* pattern to robustly handle runtime configuration changes



See [developer.android.com/guide/topics/resources/runtime-changes.html](developer.android.com/guide/topics/resources/runtime-changes.html)

# Overview of Android Concurrency Frameworks

- Decouple computation(s) & communication
- Two concurrency frameworks
- Each frameworks has pros & cons & are used heavily throughout Android
- Both frameworks implement Android concurrency *idioms*
- Patterns/idioms also needed to program with Android's concurrency frameworks, e.g.
  - Use *MVP* pattern to robustly handle runtime configuration changes
  - "don't store a message passed to handleMessage() without copying it first"

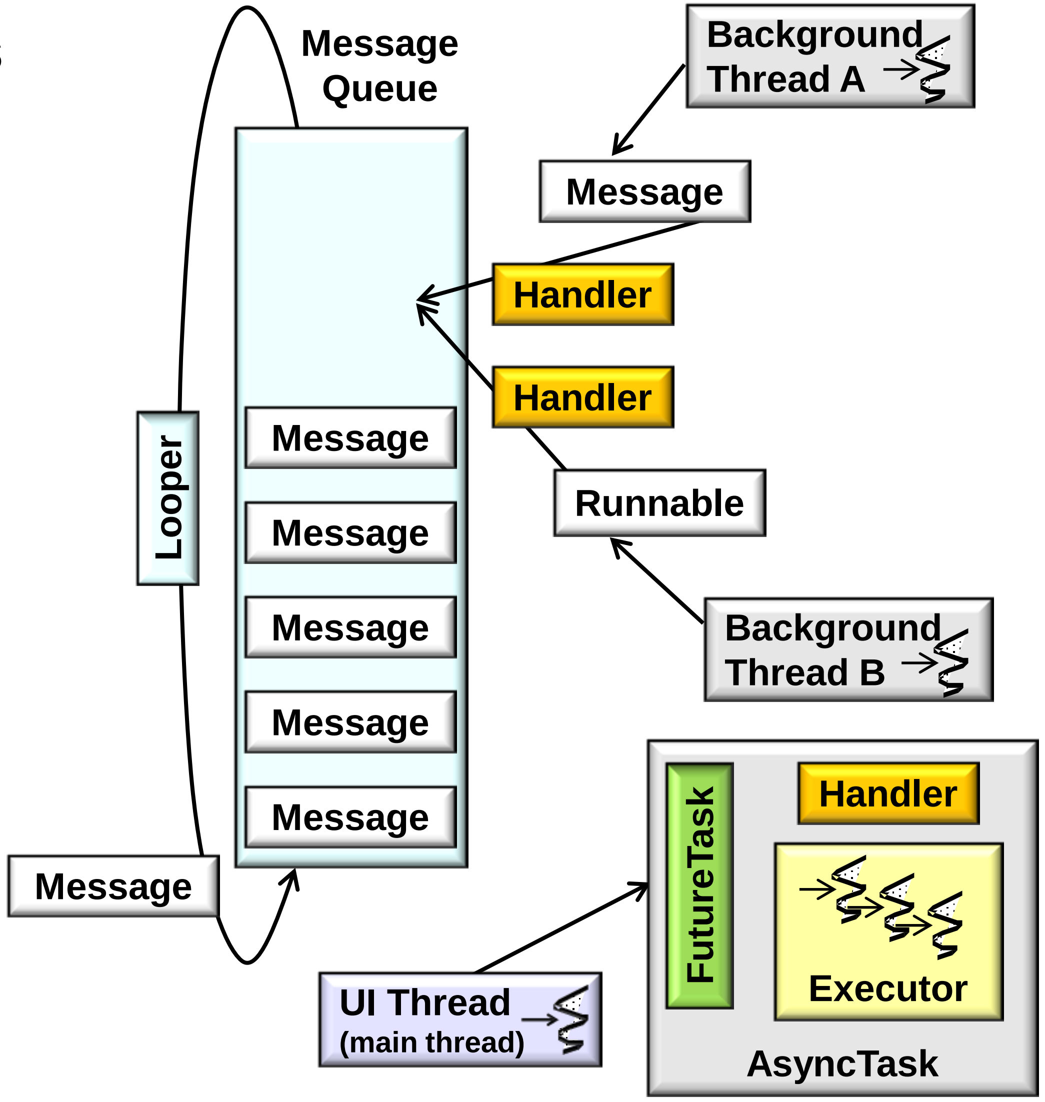


See groups.google.com/forum/#!topic/android-developers/9pHuc7lGunY

## Elements of Android Concurrency Frameworks

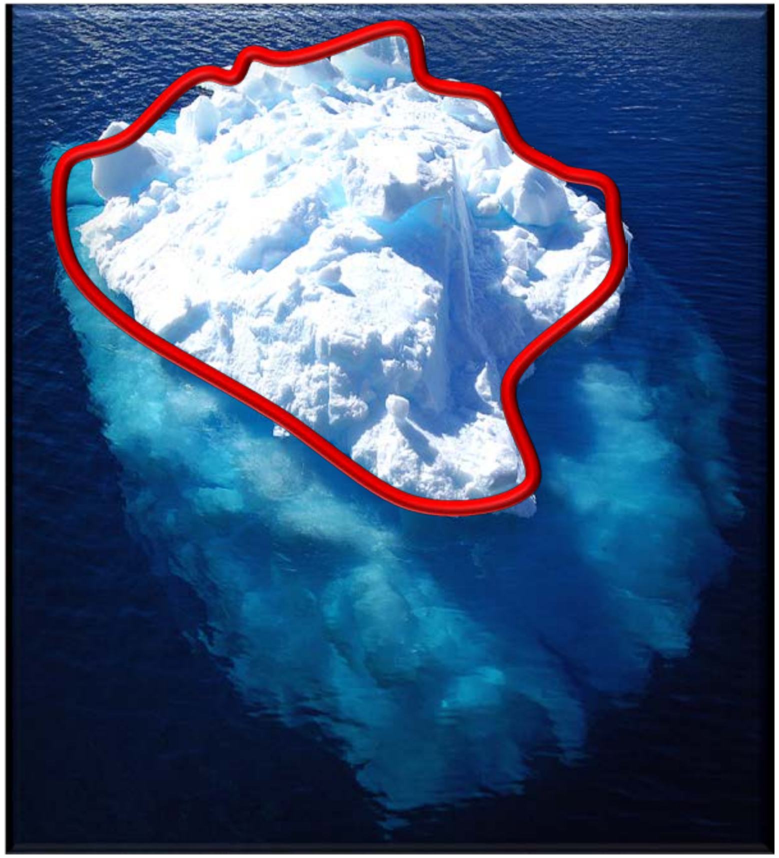
# Elements of Android Concurrency Frameworks

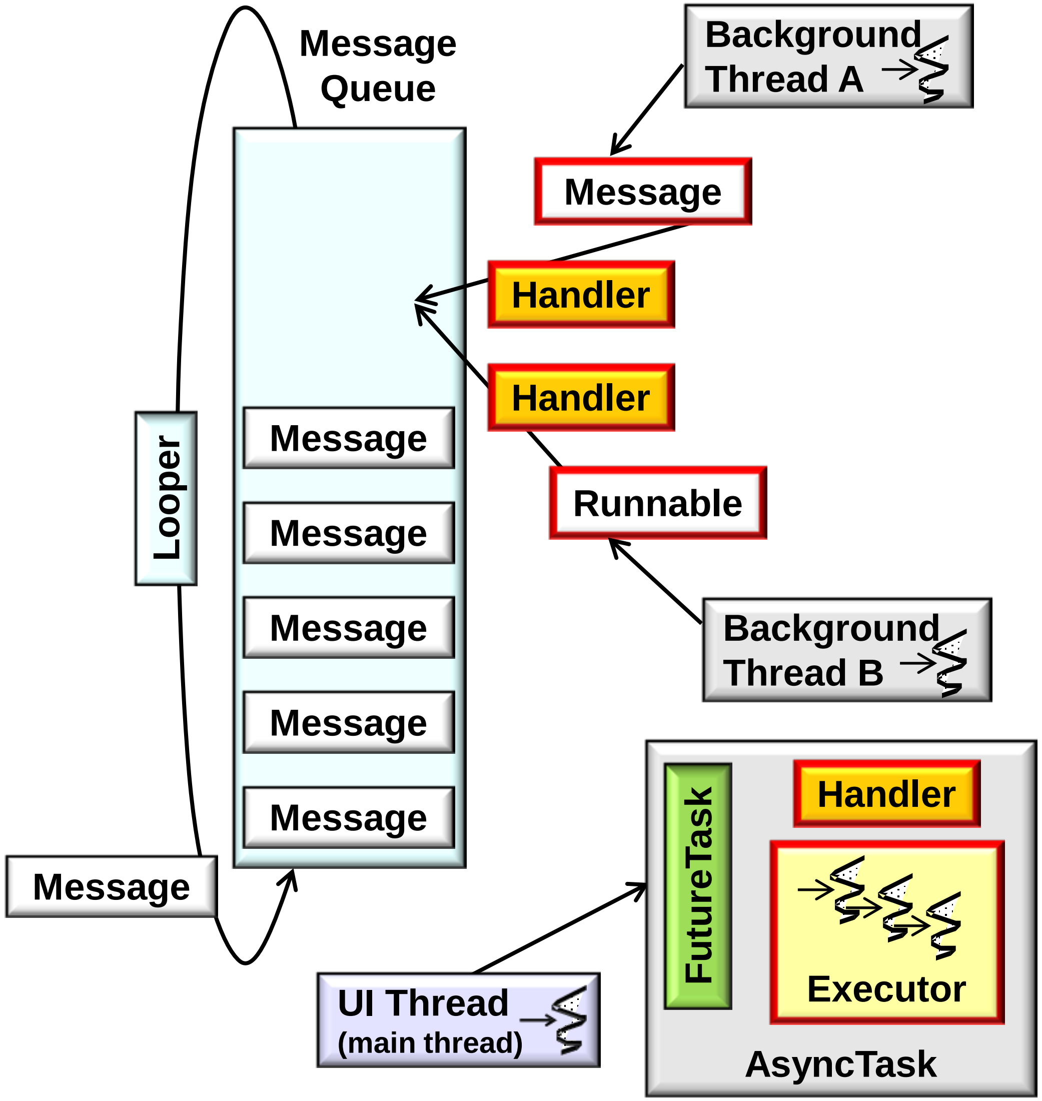
- Android's concurrency frameworks are built using reusable classes



# Elements of Android Concurrency Frameworks

- Android's concurrency frameworks are built using reusable classes

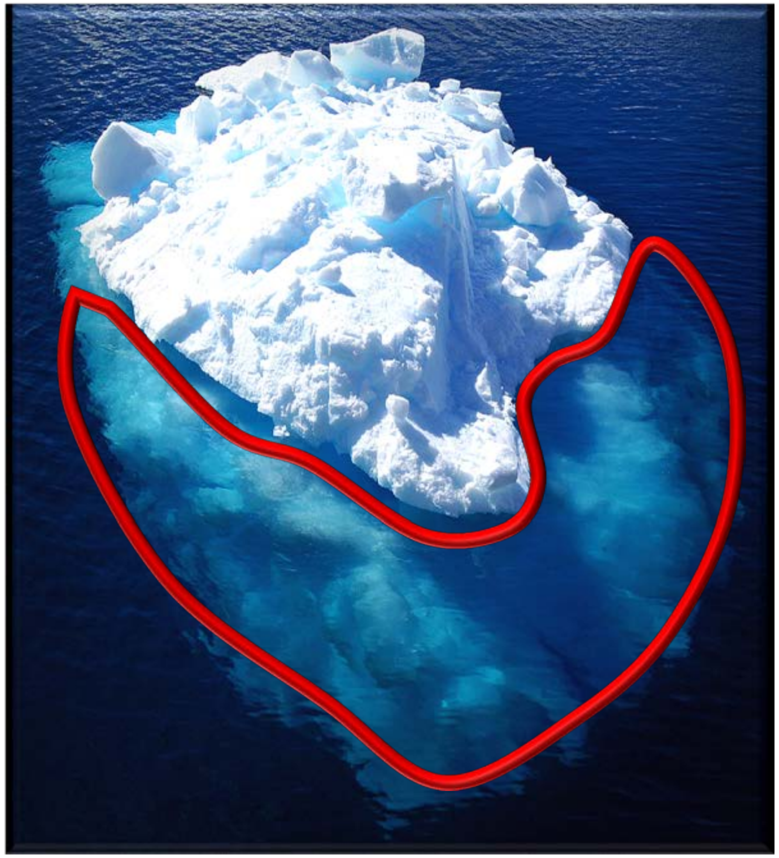


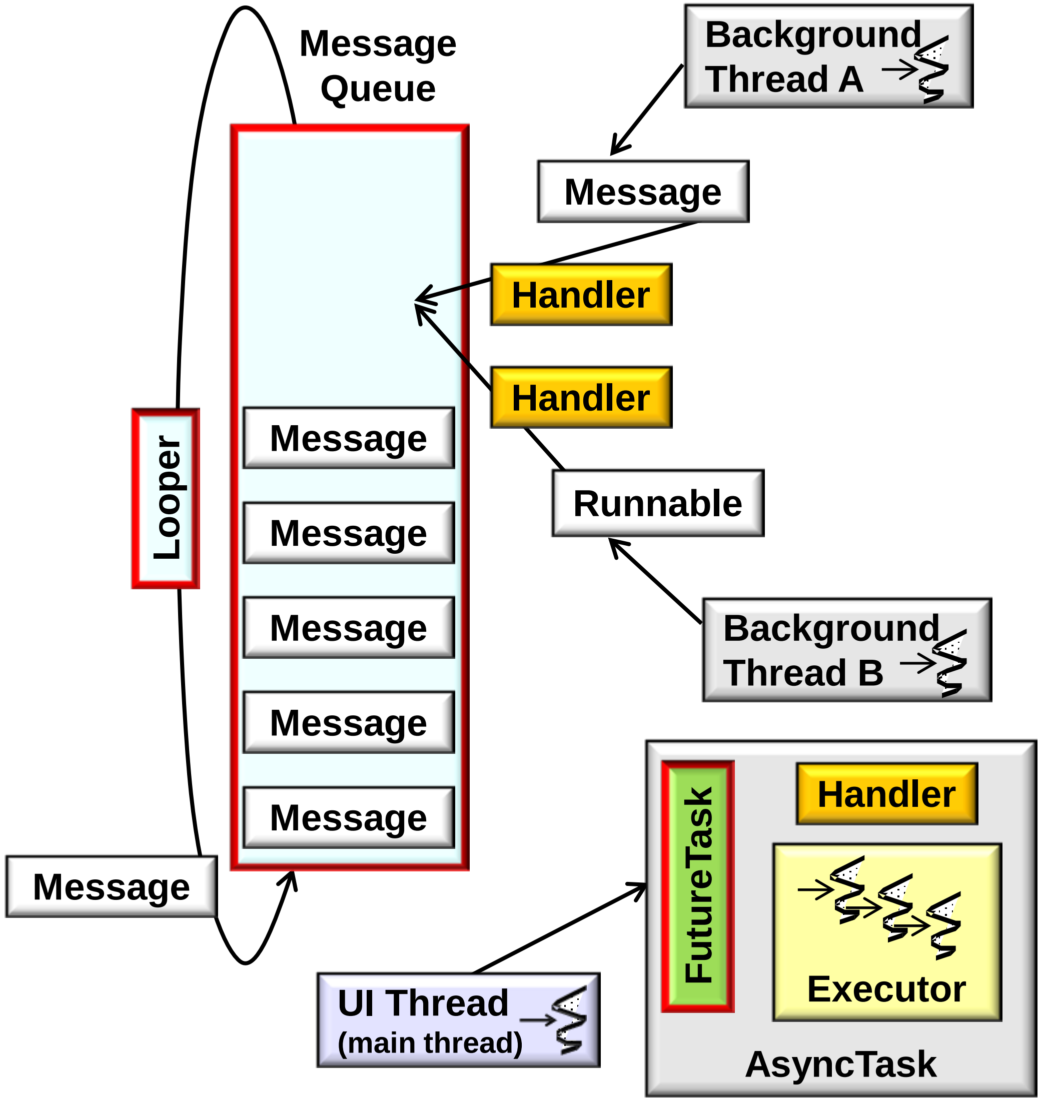


We first cover the classes used to write concurrent Android programs

# Elements of Android Concurrency Frameworks

- Android's concurrency frameworks are built using reusable classes

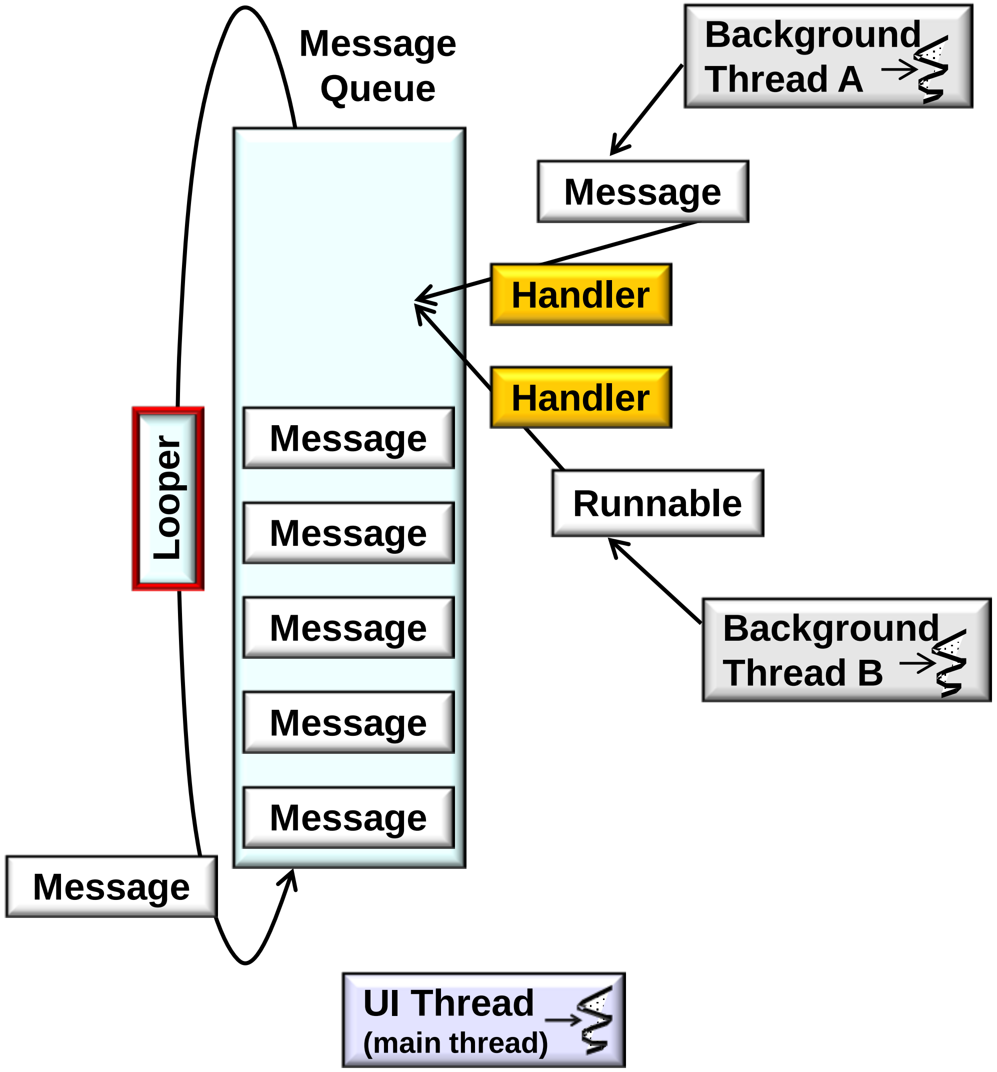




We next explore the implementation of Android's concurrency frameworks

# Elements of Android Concurrency Frameworks

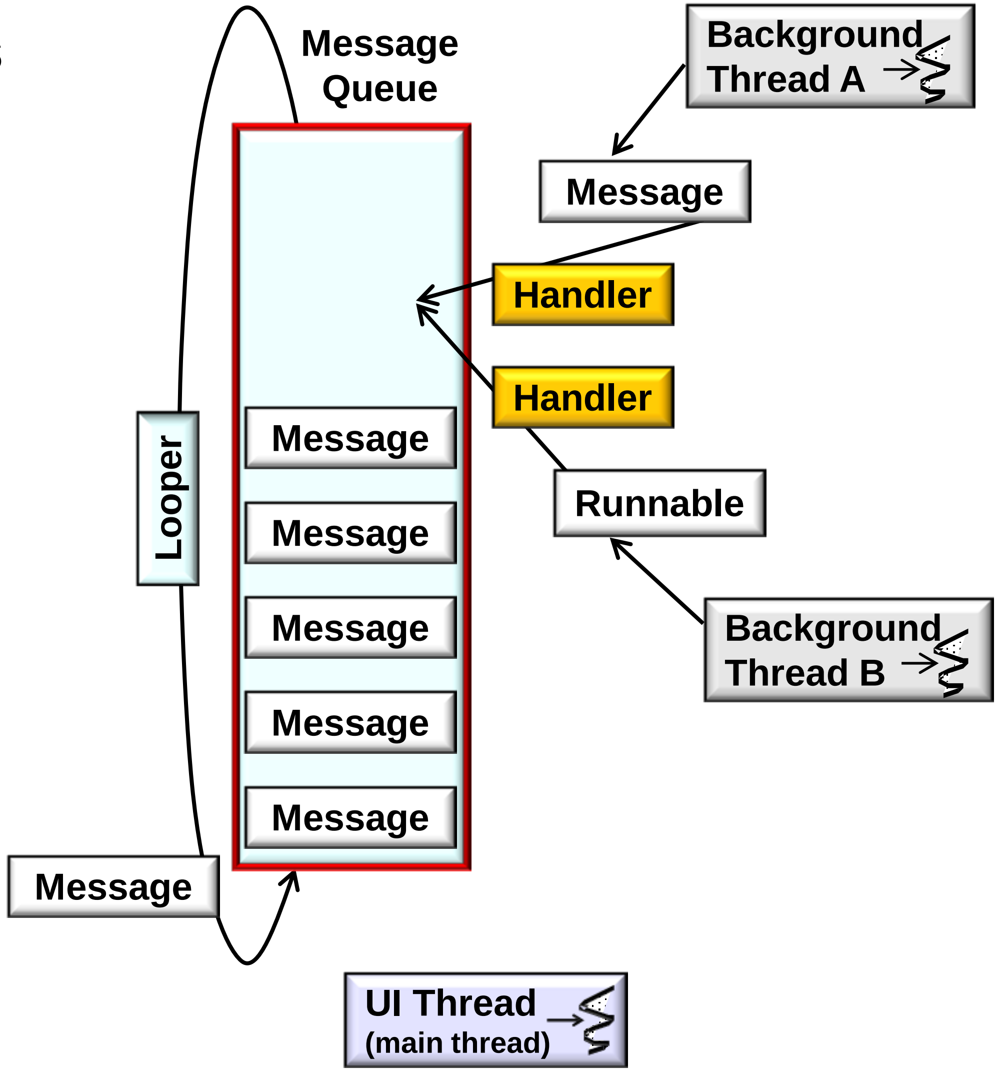
- Android's concurrency frameworks are built using reusable classes
  - Looper – Run a message loop for a thread



See [developer.android.com/reference/android/os/Looper.html](developer.android.com/reference/android/os/Looper.html)

# Elements of Android Concurrency Frameworks

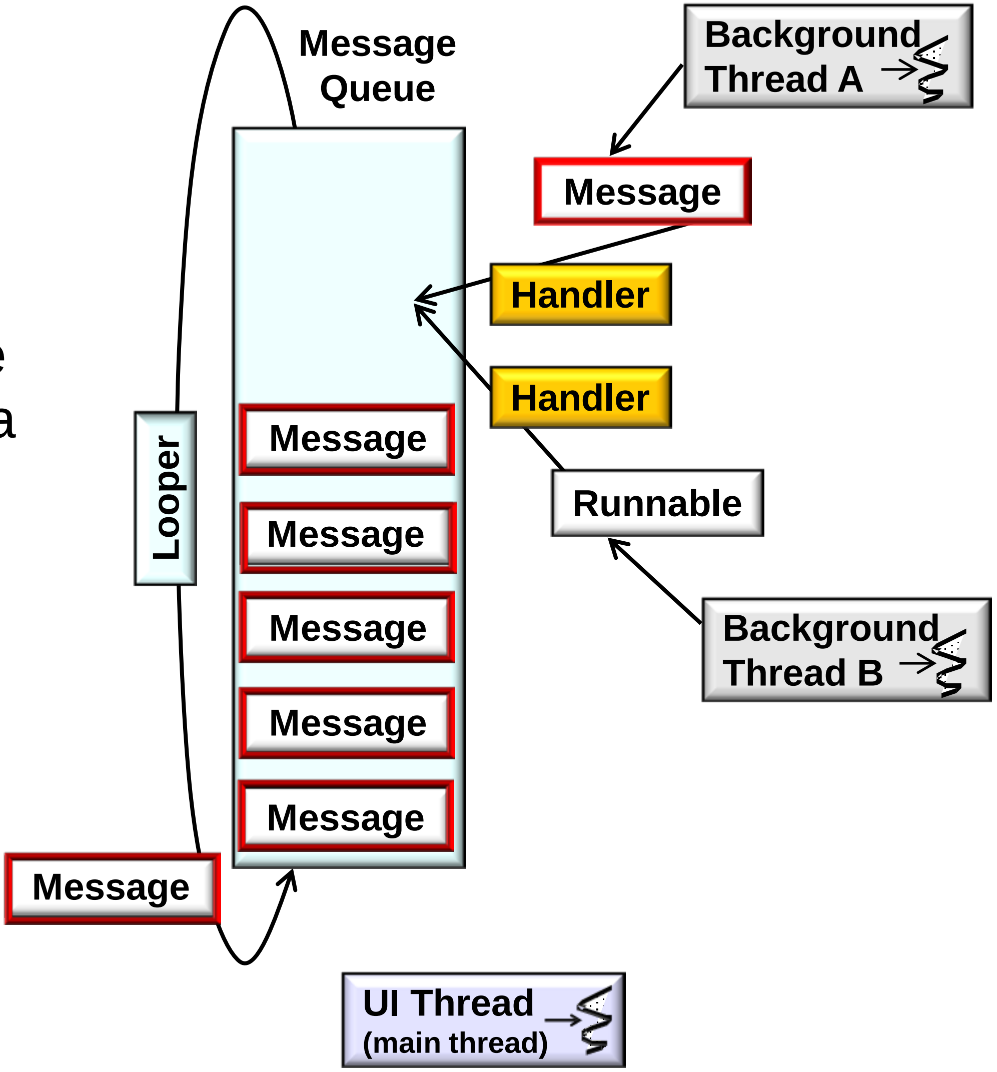
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue – Holds the list of messages to be dispatched by a Looper



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

# Elements of Android Concurrency Frameworks

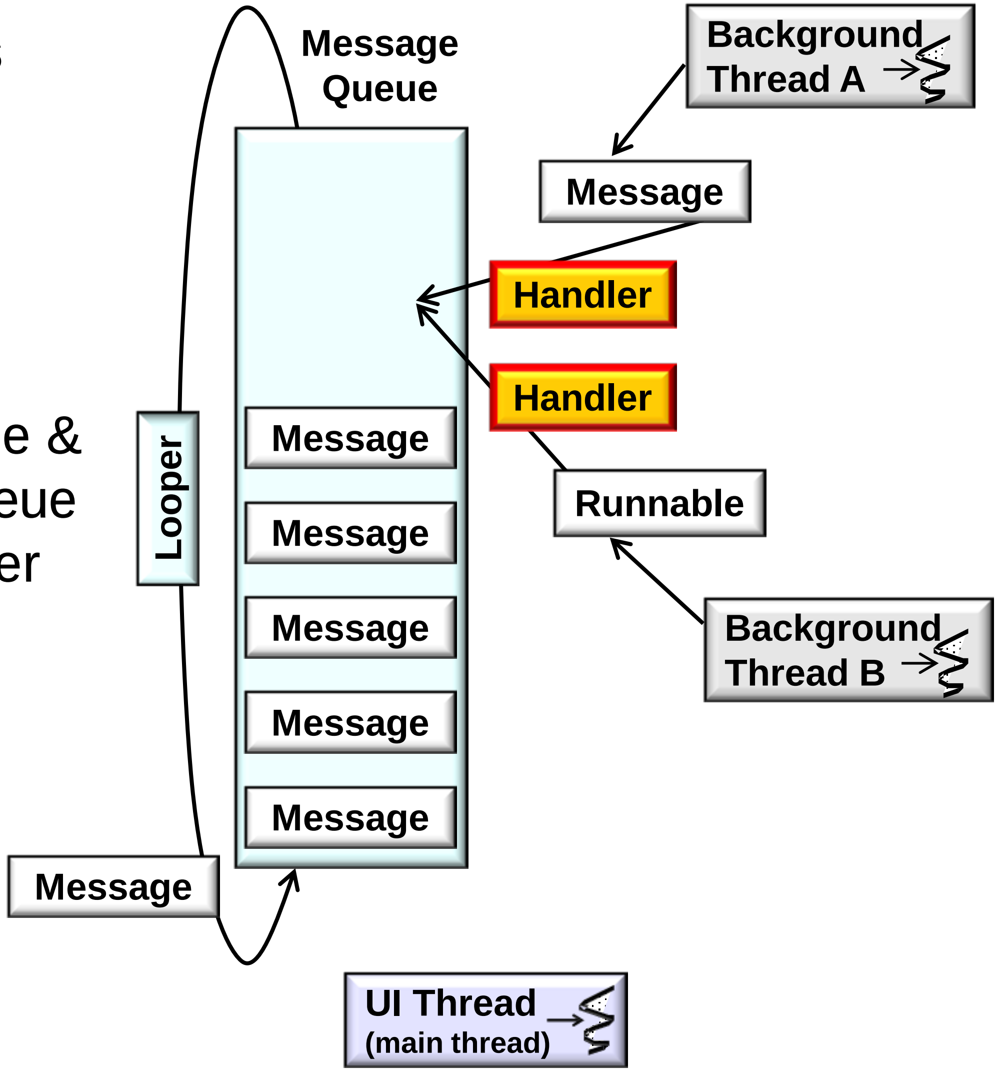
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message – Contains data & type information that can be sent to a Handler via a MessageQueue



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

# Elements of Android Concurrency Frameworks

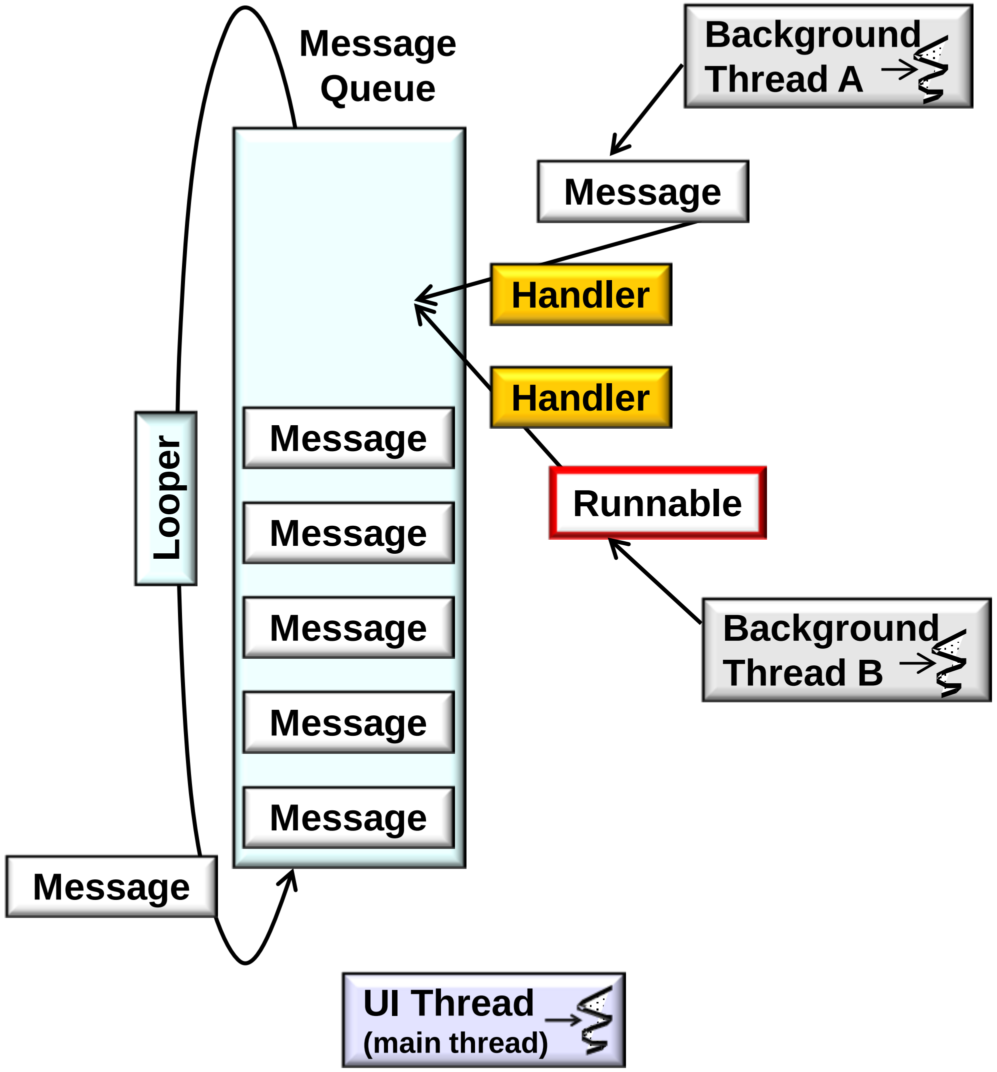
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler – Send/process Message & Runnable objects in MessageQueue associated with a Thread's Looper



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

# Elements of Android Concurrency Frameworks

- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler
  - Runnable – Represents a command that can be executed

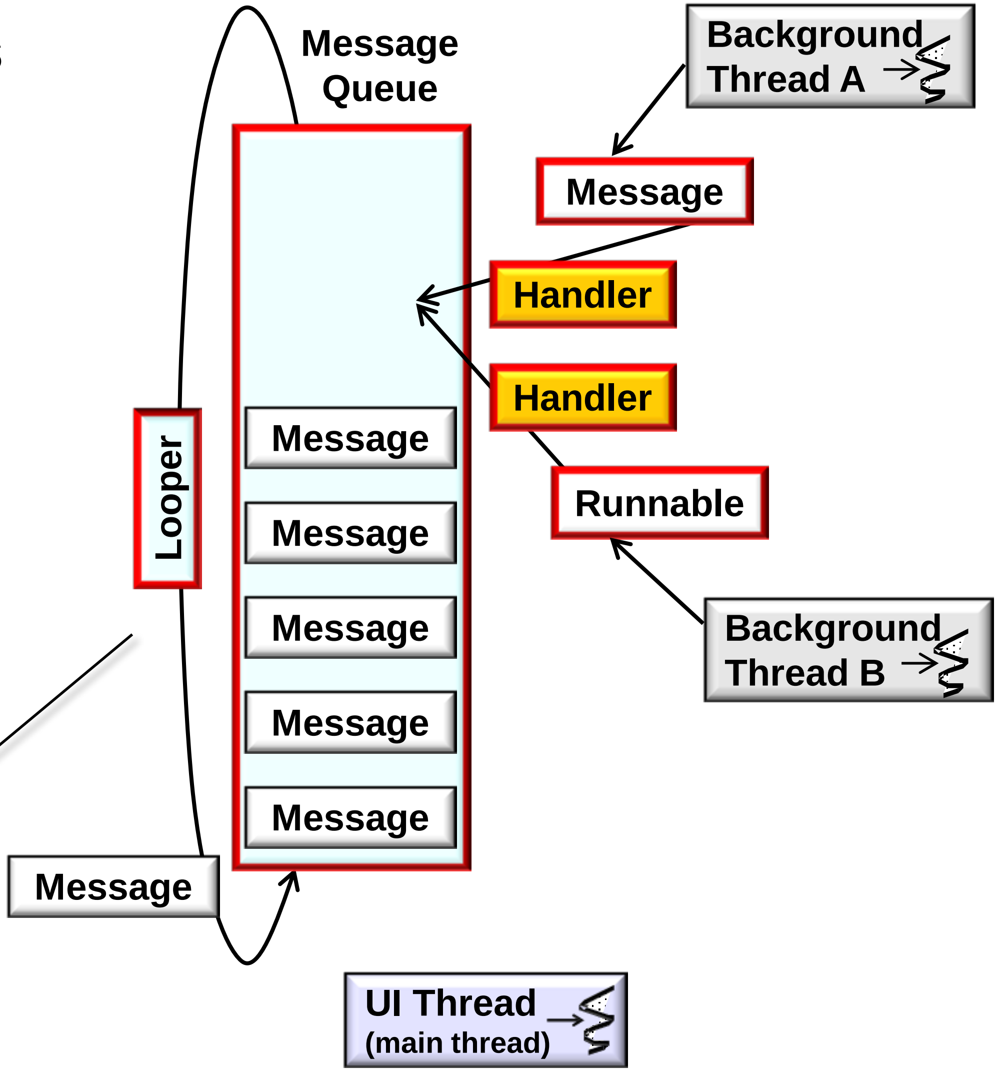


See [developer.android.com/reference/java/lang/Runnable.html](developer.android.com/reference/java/lang/Runnable.html)

# Elements of Android Concurrency Frameworks

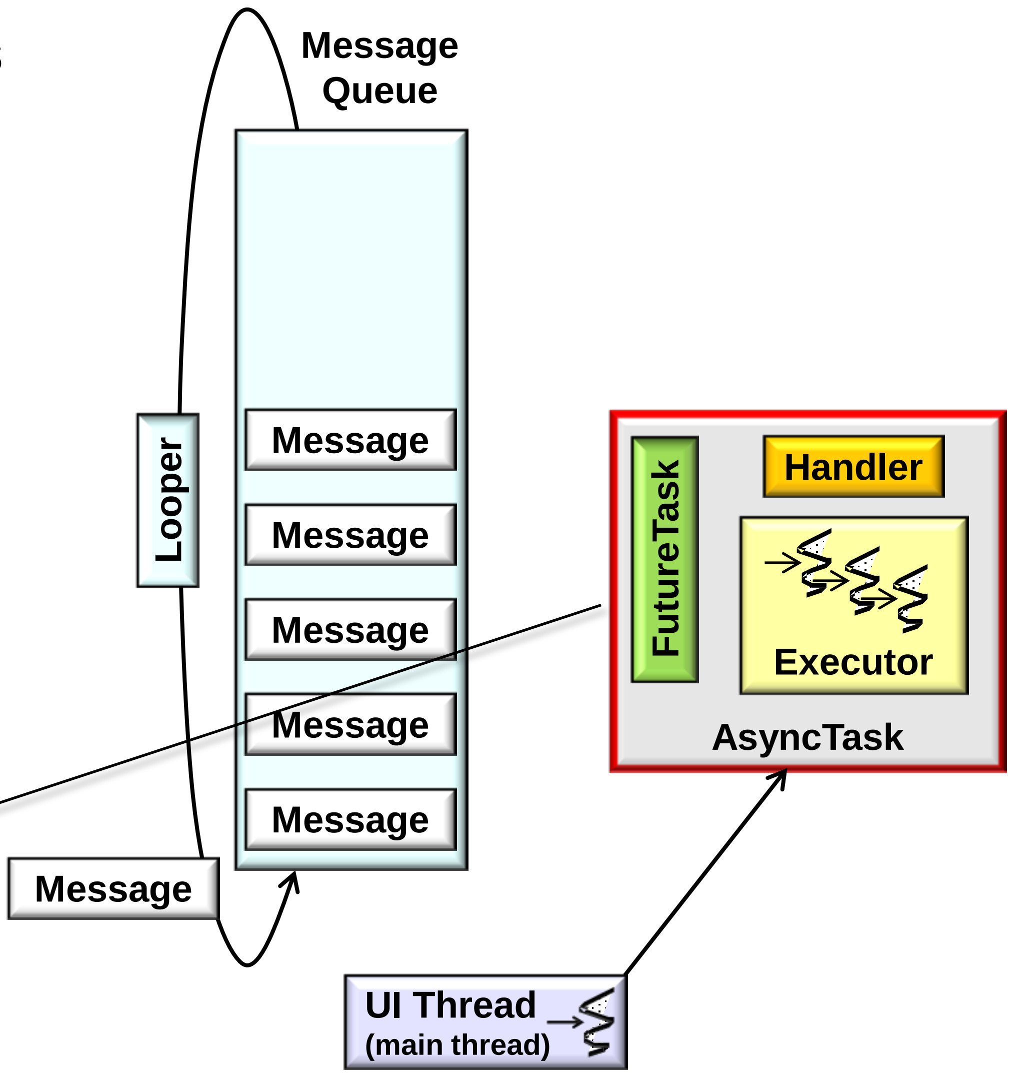
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler
  - Runnable

*These classes are used by both the HaMeR & AsyncTask concurrency frameworks*



# Elements of Android Concurrency Frameworks

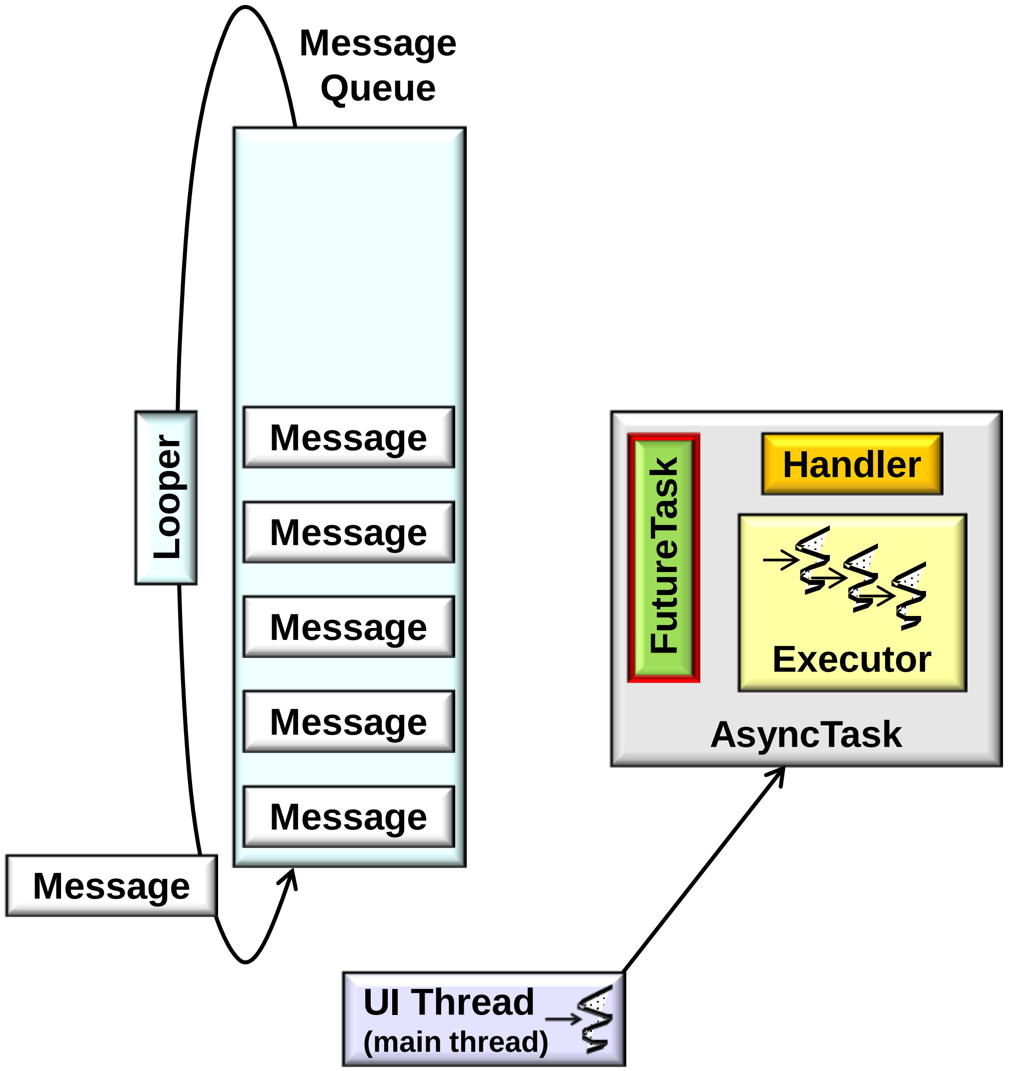
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler
  - Runnable



*Some classes are just used by the AsyncTask concurrency framework*

# Elements of Android Concurrency Frameworks

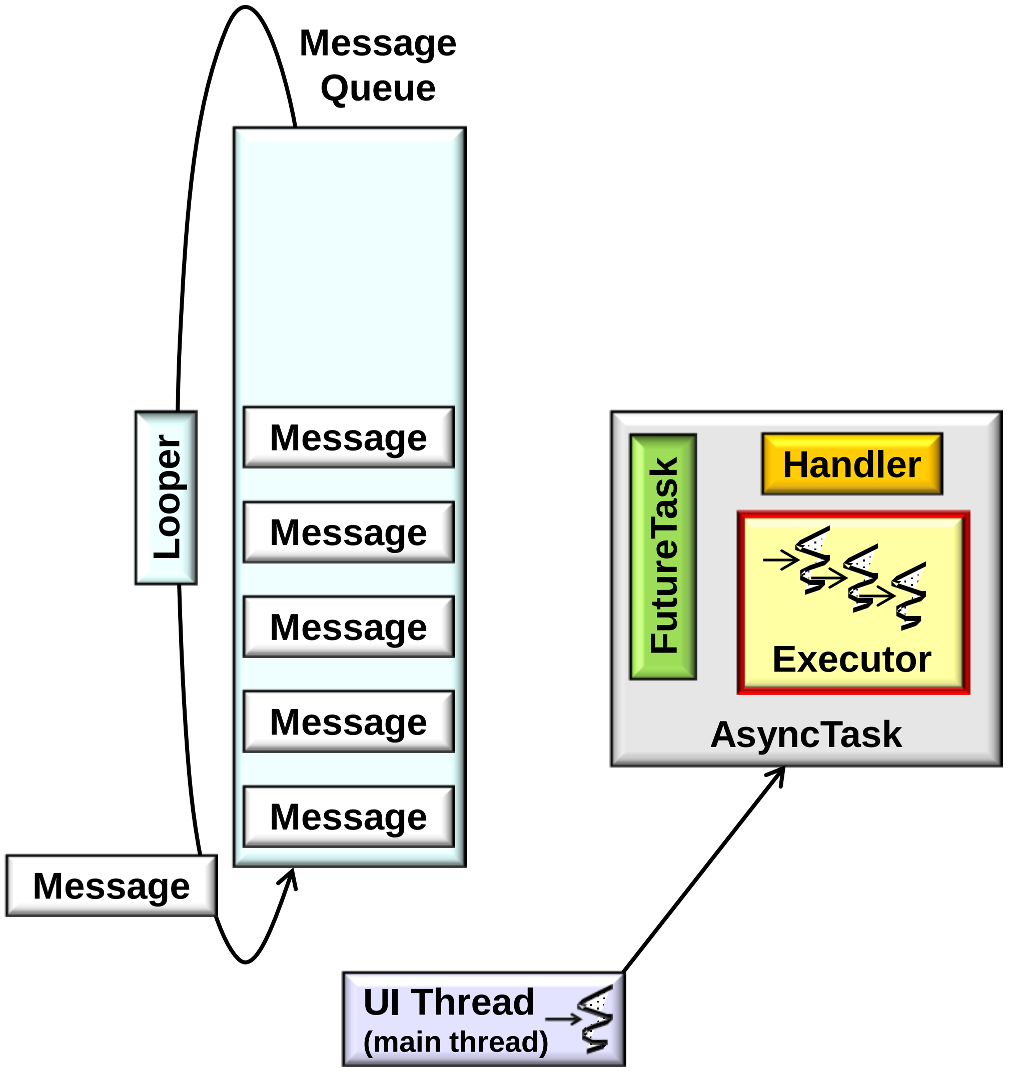
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler
  - Runnable
  - FutureTask – Can be used to
    - Start & cancel a computation that runs asynchronously
    - Query to see if computation is complete
    - Retrieve the result of the computation



See developer.android.com/reference/java/util/concurrent/FutureTask.html

# Elements of Android Concurrency Frameworks

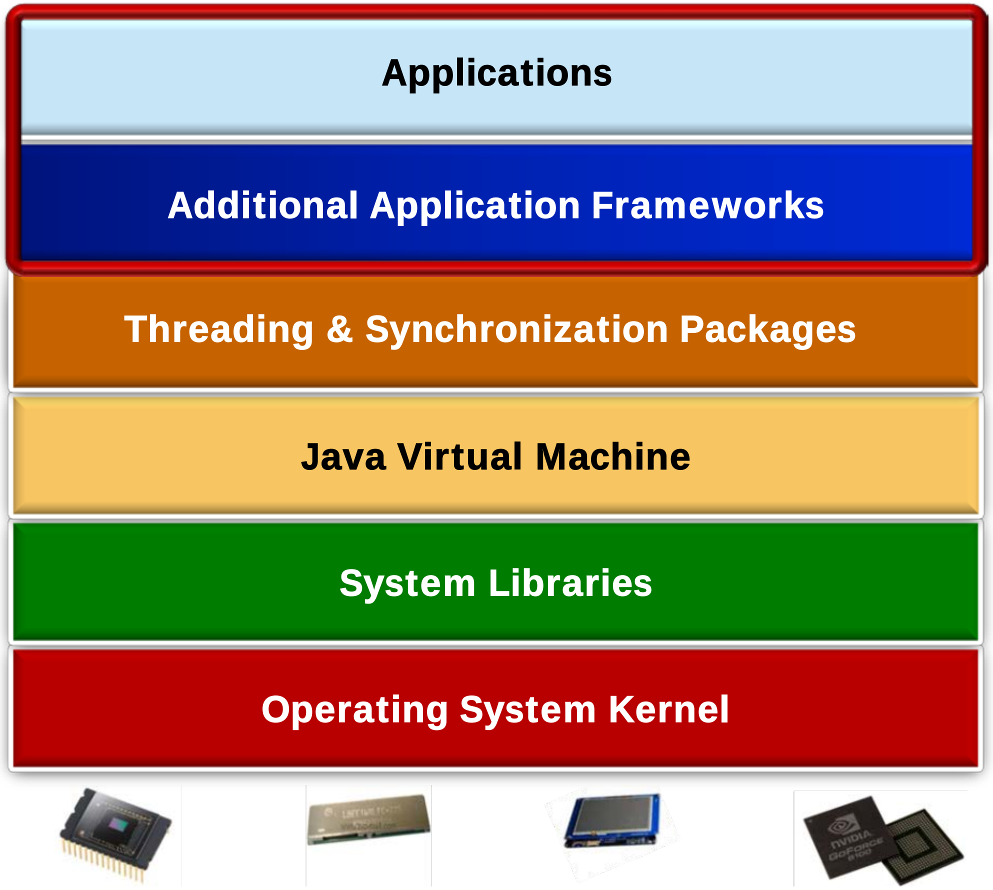
- Android's concurrency frameworks are built using reusable classes
  - Looper
  - MessageQueue
  - Message
  - Handler
  - Runnable
  - FutureTask
  - Executor framework – Execute submitted Runnable tasks either
    - Sequentially in one thread or
    - Concurrently in a thread pool



See [developer.android.com/reference/java/util/concurrent/Executor.html](developer.android.com/reference/java/util/concurrent/Executor.html)

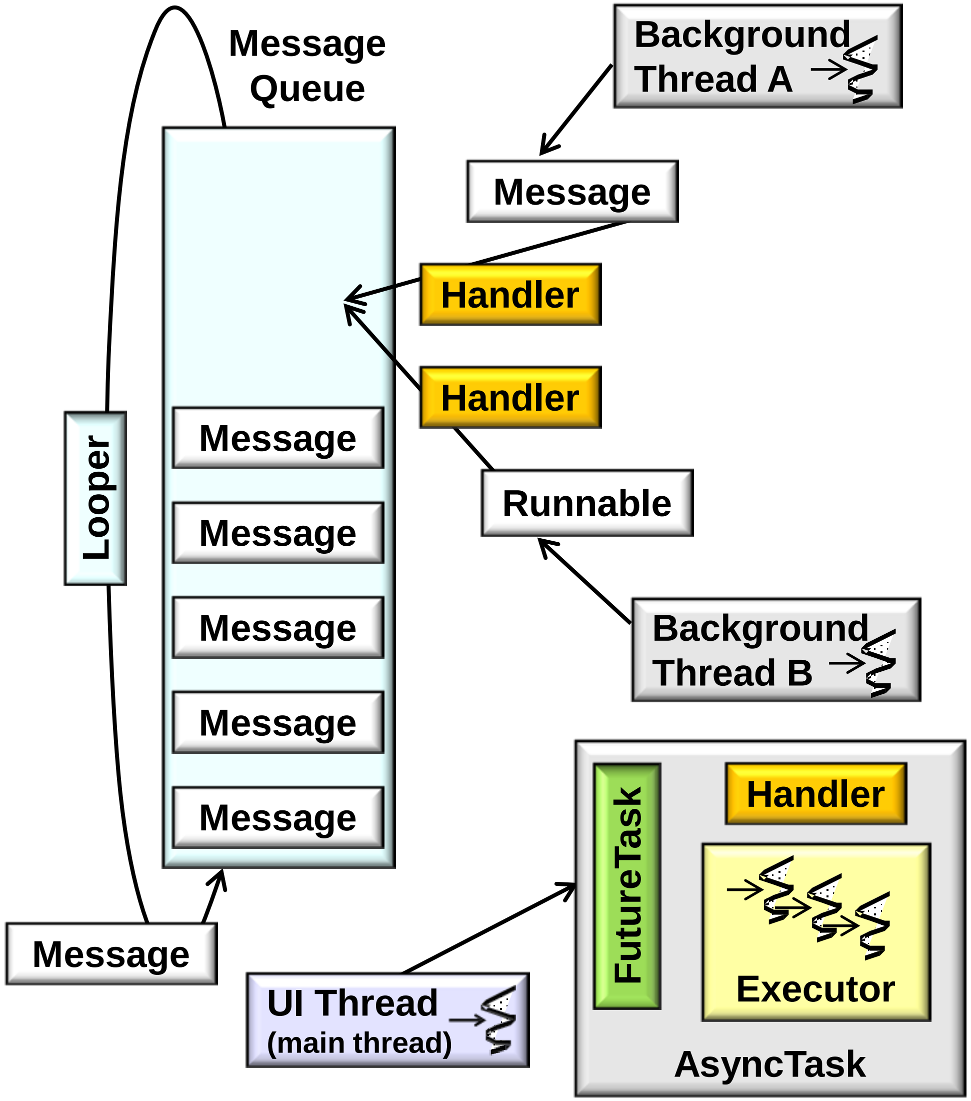
# Elements of Android Concurrency Frameworks

- Android's concurrency frameworks are built using reusable classes
- These frameworks are used by Android's application frameworks & packaged applications



# Elements of Android Concurrency Frameworks

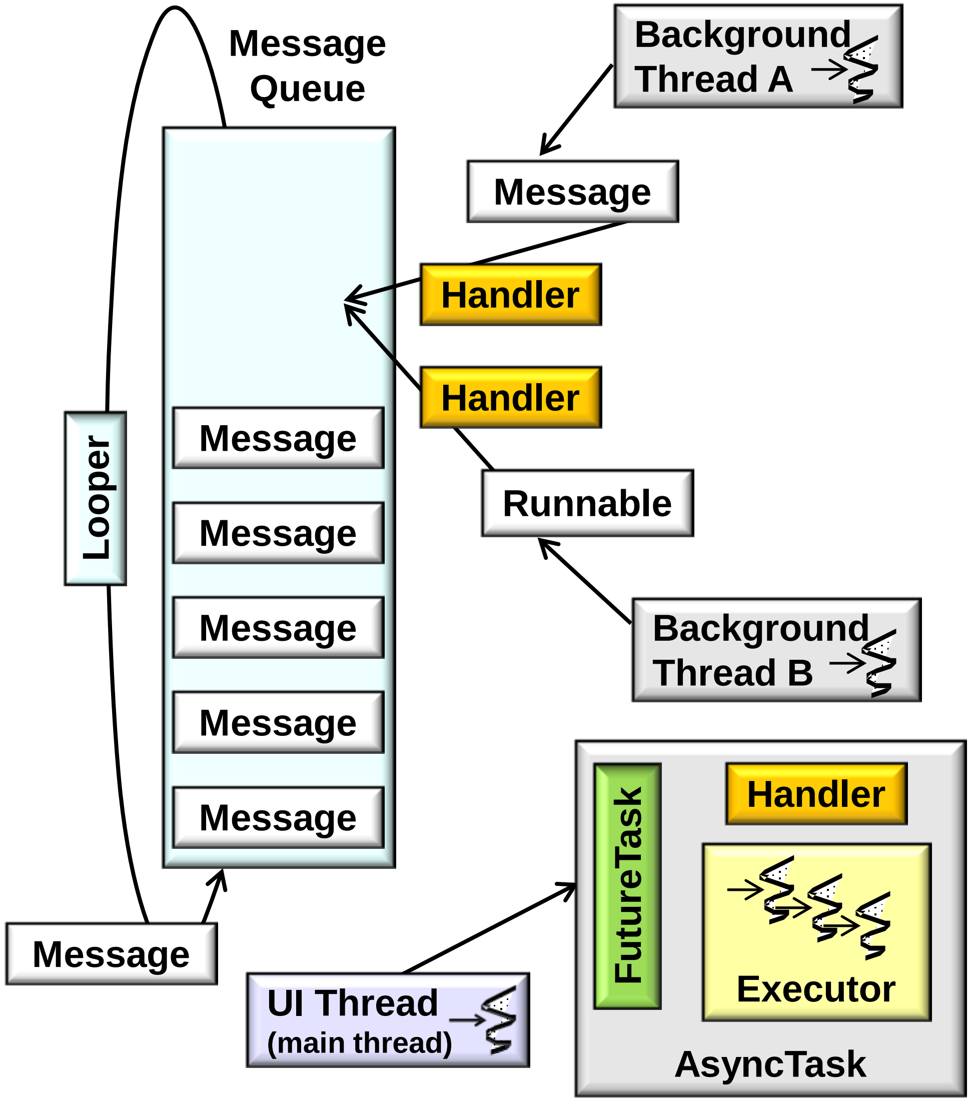
- Android's concurrency frameworks are built using reusable classes
- These frameworks are used by Android's application frameworks & packaged applications
- Android applies may patterns to overcome design constraints & ensure other concurrency benefits

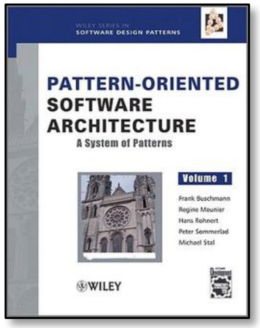


See [en.wikipedia.org/wiki/Concurrency_pattern](https://en.wikipedia.org/wiki/Concurrency_pattern)

# Elements of Android Concurrency Frameworks

- Android's concurrency frameworks are built using reusable classes
- These frameworks are used by Android's application frameworks & packaged applications
- Android applies may patterns to overcome design constraints & ensure other concurrency benefits




WILEY SERIES IN SOFTWARE DESIGN PATTERNS

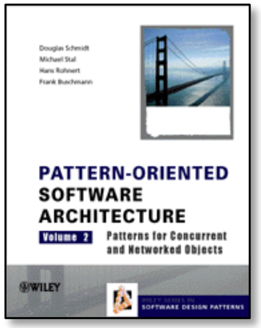
PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A System of Patterns

Volume 1

Frank Buschmann
Regine Meunier
Hans Rohnert
Peter Sommerlad
Michael Stal

WILEY


Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

Volume 2 Patterns for Concurrent and Networked Objects

WILEY

WILEY SERIES IN SOFTWARE DESIGN PATTERNS

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

See upcoming discussions on "Concurrency Patterns in Android"