Managing the Java Thread Lifecycle: Layers Involved in Starting a Thread

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 Lesson

• Understand the layers involved in starting a Java thread
Learning Objectives in this Lesson

• Understand the layers involved in starting a Java thread
• Recognize the steps involved in starting a Java thread
Layers Involved in Starting a Java Thread

- Starting a Java thread involves interesting design & implementation issues
Layers Involved in Starting a Java Thread
Layers Involved in Starting a Java Thread

- Calling `start()` on a thread triggers the execution of its `run()` hook method.
Layers Involved in Starting a Java Thread

- The Java platform provides a stack of layers that define various mechanisms for running concurrent programs on a wide range of computing devices.

Different versions of Android & Java implement these layers differently, though key levels of abstraction are often similar.

See en.wikibooks.org/wiki/Java_Programming/The_Java_Platform
Likewise, the Android platform provides a stack of layers that define various mechanisms for running concurrent programs on mobile computing devices.

See [developer.android.com/guide/platform](http://developer.android.com/guide/platform)
Layers Involved in Starting a Java Thread

• Likewise, the Android platform provides a stack of layers that define various mechanisms for running concurrent programs on mobile computing devices.

The Android Linux kernel controls hardware & manages system resources.
Likewise, the Android platform provides a stack of layers that define various mechanisms for running concurrent programs on mobile computing devices. The Bionic LibC library supports the Pthreads C programming APIs.
Likewise, the Android platform provides a stack of layers that define various mechanisms for running concurrent programs on mobile computing devices. Dalvik & ART provide a managed execution environment for Java apps.
Layers Involved in Starting a Java Thread

• Likewise, the Android platform provides a stack of layers that define various mechanisms for running concurrent programs on mobile computing devices.

Package java.util.concurrent

Description

Utility classes commonly useful in concurrent programming. This package includes a few small standardized extensible frameworks, as well as some classes that provide useful functionality and are otherwise tedious or difficult to implement. Here are brief descriptions of the main components. See also the java.util.concurrent.locks and java.util.concurrent.atomic packages.

Android’s runtime contains the classes in the java.util.concurrent packages
Layers Involved in Starting a Java Thread

• Creating & starting new threads on any Java platform consumes a non-trivial amount of system resources, so use them judiciously!
Layers Involved in Starting a Java Thread

- Creating & starting new threads on any Java platform consumes a non-trivial amount of system resources, so use them judiciously!
- e.g., only create threads for computations that run much longer than the time needed to spawn them!
Steps Involved in Starting a Java Thread
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

See source.android.com
Steps Involved in Starting a Java Thread

• The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()` // Java method

See `libcore/luni/src/main/java/java/lang/Thread.java`
Steps Involved in Starting a Java Thread

• The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()` // Native method

Steps Involved in Starting a Java Thread

1. myThread.start()
2. Thread.start()
3. VMThread.create()
4. Dalvik_java_lang_VMThread_create()
   // JNI method

See dalvik/vm/native/java_lang_VMThread.cpp
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()`
4. `Dalvik_java_lang_VMThread_create()`  // Dalvik method
5. `dvmCreateInterpThread()`  // Dalvik method

See `dalvik/vm/Thread.cpp`
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform:

  1. `myThread.start()`
  2. `Thread.start()`
  3. `VMThread.create()`
  4. `Dalvik_java_lang_VMThread_create()`
  5. `dvmCreateInterpThread()`
  6. `pthread_create(..., interpThreadStart)`
     // Pthreads method

See [bionic/libc/bionic/pthread.c](bionic/libc/bionic/pthread.c)
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()`
4. `Dalvik_java_lang_VMThread_create()`
5. `dvmCreateInterpThread()`
6. `pthread_create(..., interpThreadStart)`
   // Pthreads method

This is the entry point function used to transition between C & Java code
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform:
  1. `myThread.start()`
  2. `Thread.start()`
  3. `VMThread.create()`
  4. `Dalvik_java_lang_VMThread_create()`
  5. `dvmCreateInterpThread()`
  6. `pthread_create(..., interpThreadStart)`
  7. Android Linux kernel...

See [source.android.com/source/building-kernels.html](source.android.com/source/building-kernels.html)
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform:

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()`
4. `Dalvik_java_lang_VMThread_create()`
5. `dvmCreateInterpThread()`
6. `pthread_create(..., interpThreadStart)`
7. `Android Linux kernel...`
8. `interpThreadStart(void* arg) // Adapter`

See `dalvik/vm/Thread.cpp`
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()`
4. `Dalvik_java_lang_VMThread_create()`
5. `dvmCreateInterpThread()`
6. `pthread_create(...)`, `interpThreadStart()`
7. `Android Linux kernel...`
8. `interpThreadStart(void* arg)`
9. `dvmCallMethod(self, run, self->threadObj)`
   // Dalvik method

See [dalvik/vm/interp/Stack.cpp](dalvik/vm/interp/Stack.cpp)
Steps Involved in Starting a Java Thread

- The following steps are involved when starting a Java thread on the Android open-source platform

1. `myThread.start()`
2. `Thread.start()`
3. `VMThread.create()`
4. `Dalvik_java_lang_VMThread_create()`
5. `dvmCreateInterpThread()`
6. `pthread_create(..., interpThreadStart)`
7. `Android Linux kernel...`
8. `interpThreadStart(void* arg)`
9. `dvmCallMethod(self, run, self->threadObj)`
10. `MyThread.run()` // User-defined hook method
End of Managing the Java Thread Lifecycle: Layers Involved in Starting a Thread