Evaluating Java's Concurrency & Parallelism Mechanisms & Frameworks

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University

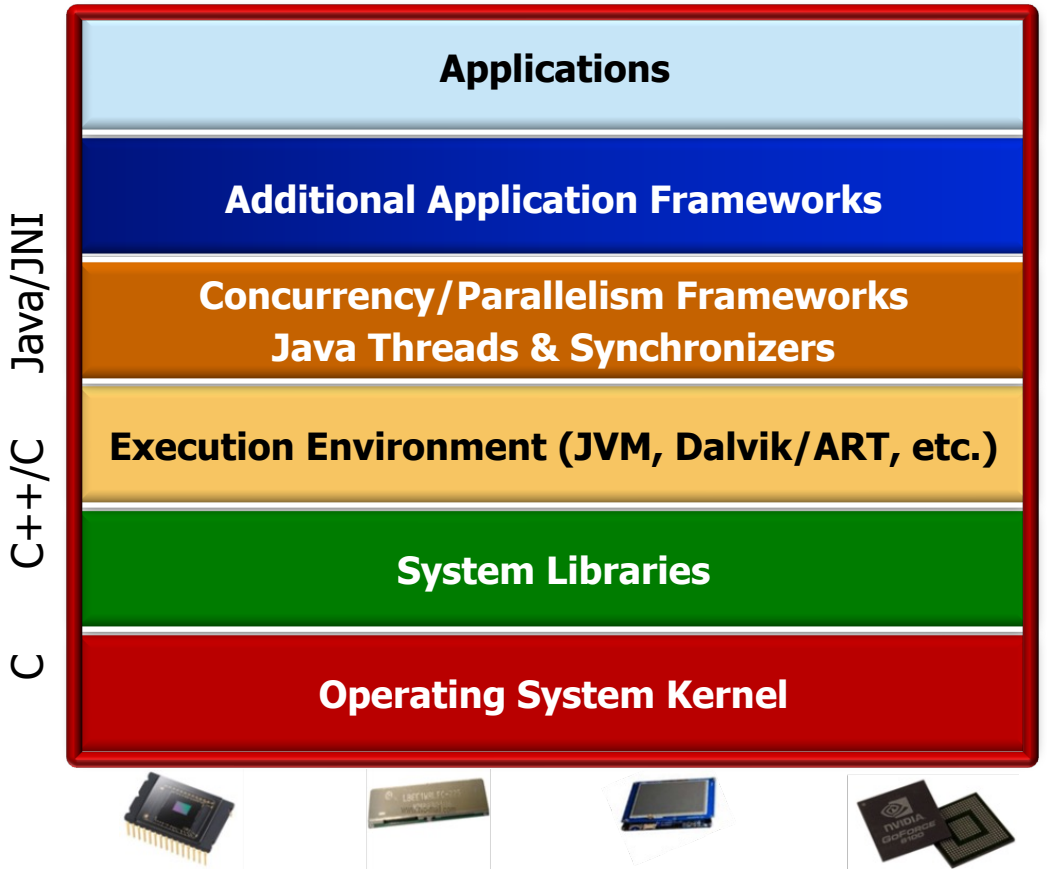
Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Know which Java concurrency & parallelism mechanism(s) to understand & apply based on the context in which they are being considered

Learning Objectives in this Lesson

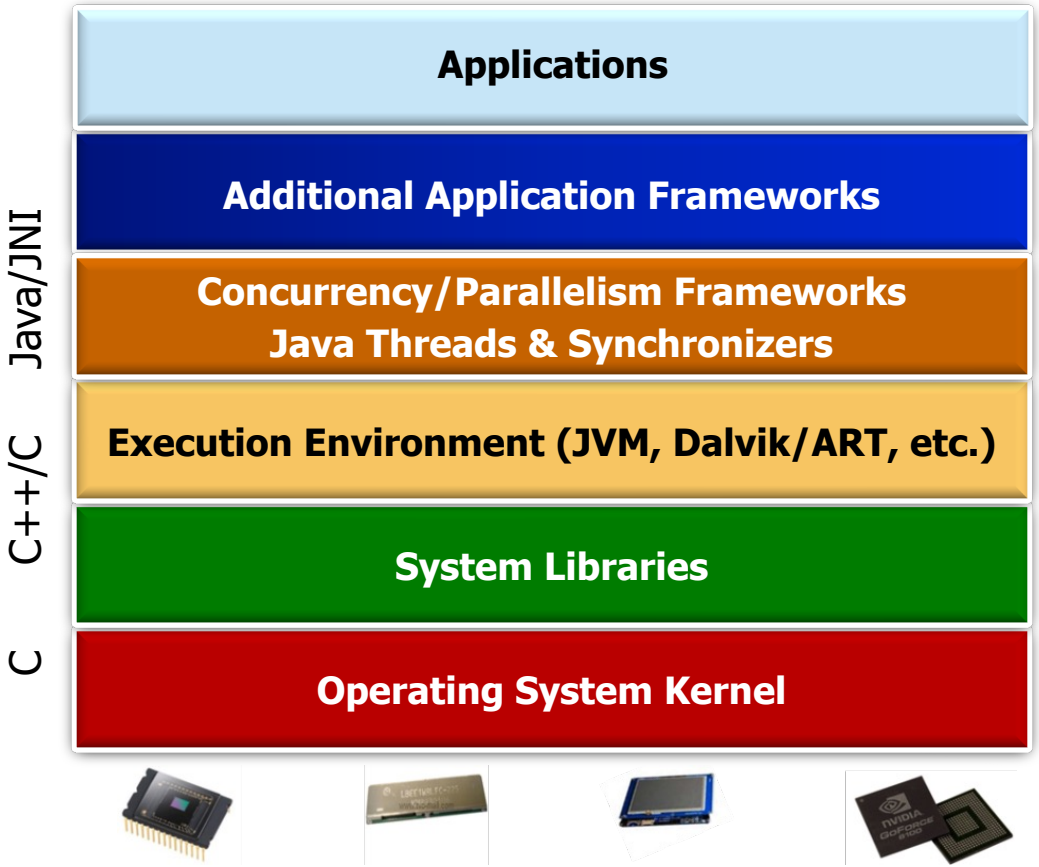
- Know which Java concurrency & parallelism mechanism(s) to understand & apply based on the context in which they are being considered
- Strive to be a full-stack developer!



Which Java Mechanism(s) to Understand & Apply

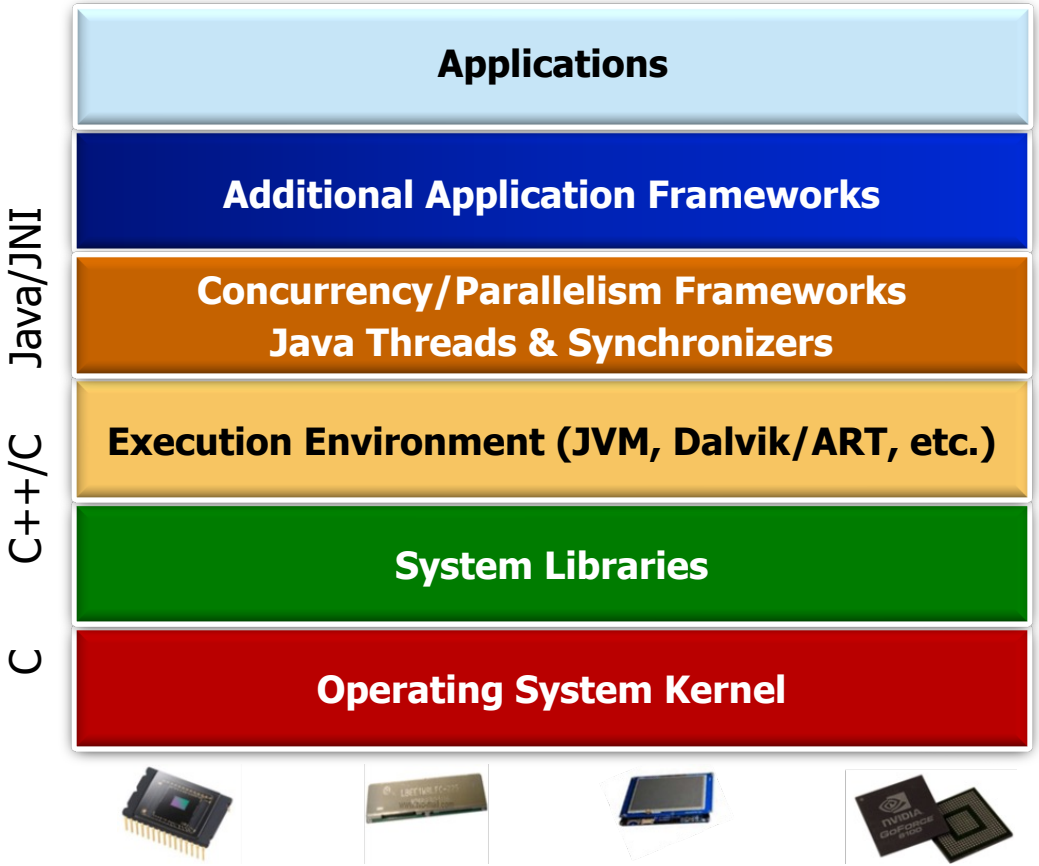
Which Java Mechanism(s) to Understand & Apply

- Java's concurrency & parallelism mechanisms span multiple layers in the software stack



Which Java Mechanism(s) to Understand & Apply

- Java's concurrency & parallelism mechanisms span multiple layers in the software stack
  - Choosing best mechanism(s) depend on various factors



Which Java Mechanism(s) to Understand & Apply

- Developers of low-level classes & performance-sensitive apps may prefer shared object mechanisms

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.

va/JNI

+/C

C

Applications

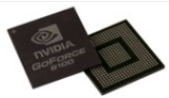
Additional Application Frameworks

Concurrency/Parallelism Frameworks
Java Threads & Synchronizers

Execution Environment (JVM, Dalvik/ART, etc.)

System Libraries

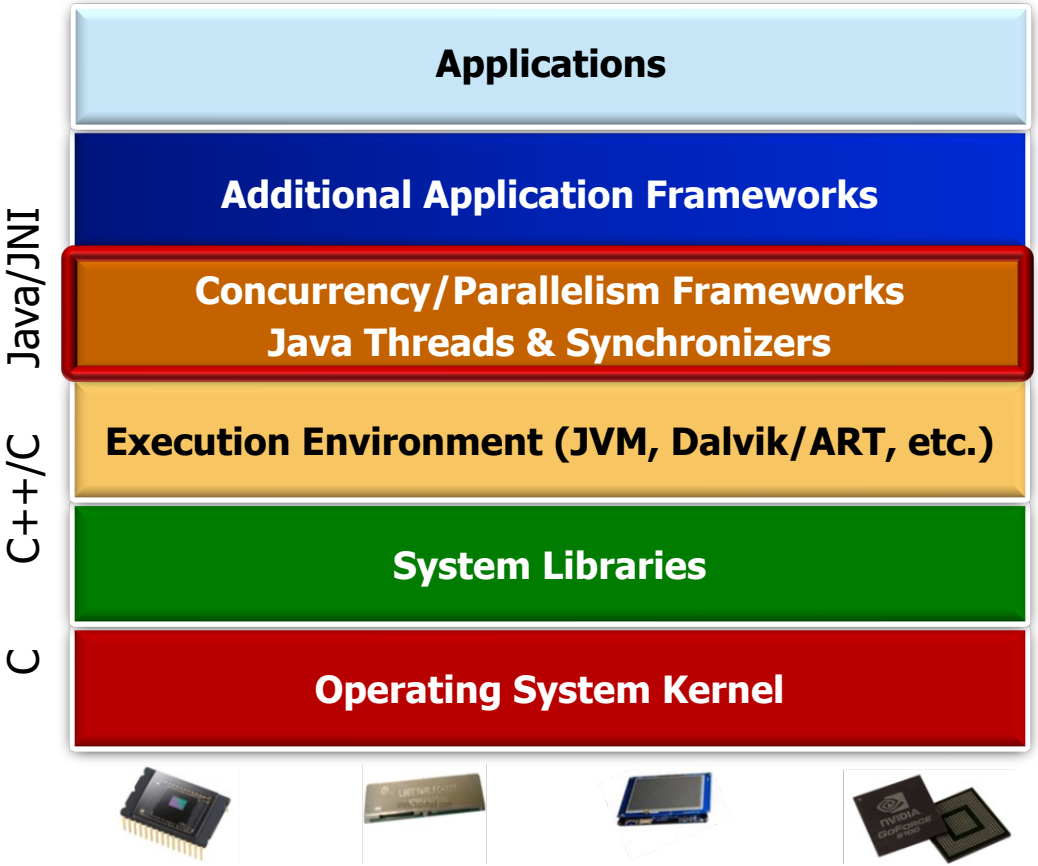
Operating System Kernel



e.g., java.util.concurrent as per www.youtube.com/watch?v=sq0MX3fHkro

Which Java Mechanism(s) to Understand & Apply

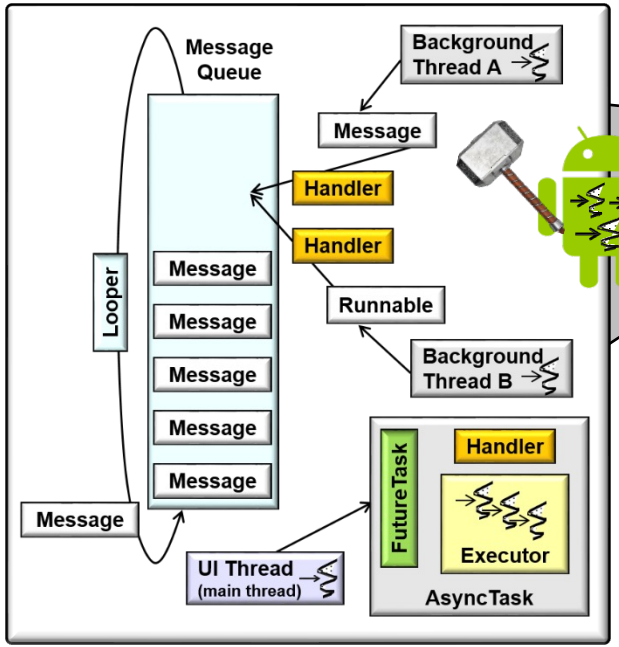
- Developers of low-level classes & performance-sensitive apps may prefer shared object mechanisms
  - **Pros**: Efficient & lightweight
  - **Cons**: Tedious & error-prone

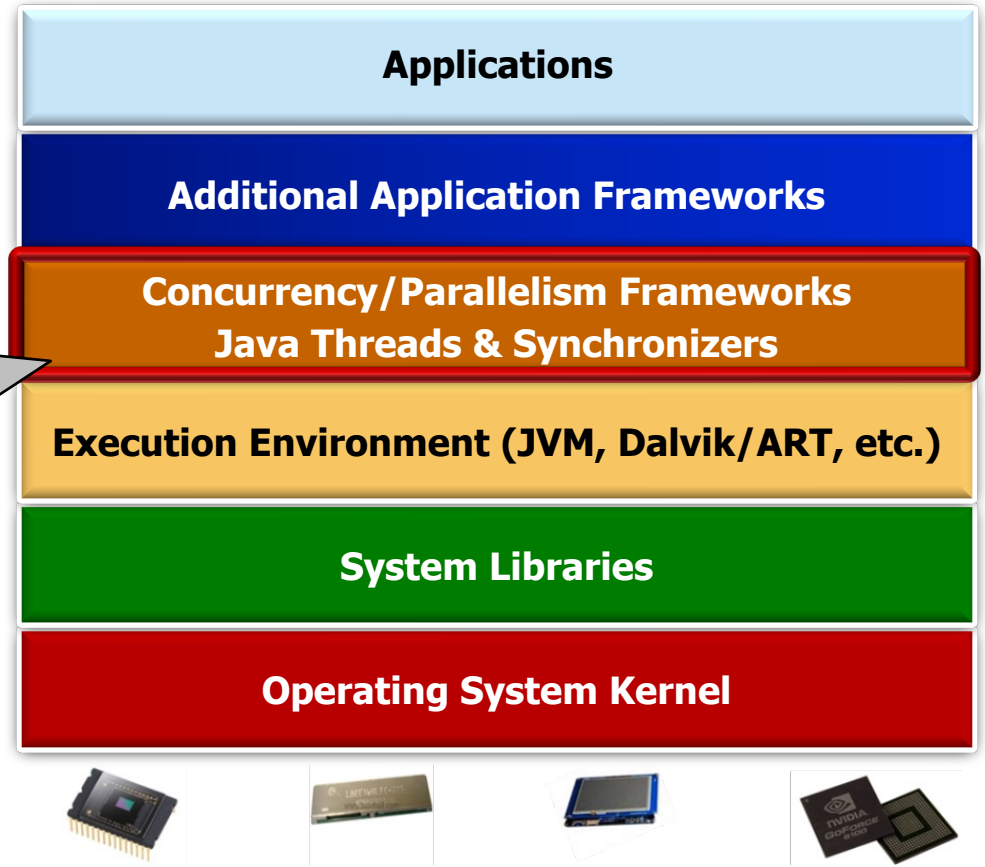


Shared objects are often best used by infrastructure vs. app developers

Which Java Mechanism(s) to Understand & Apply

- Framework developers may want to use the Java message passing mechanisms

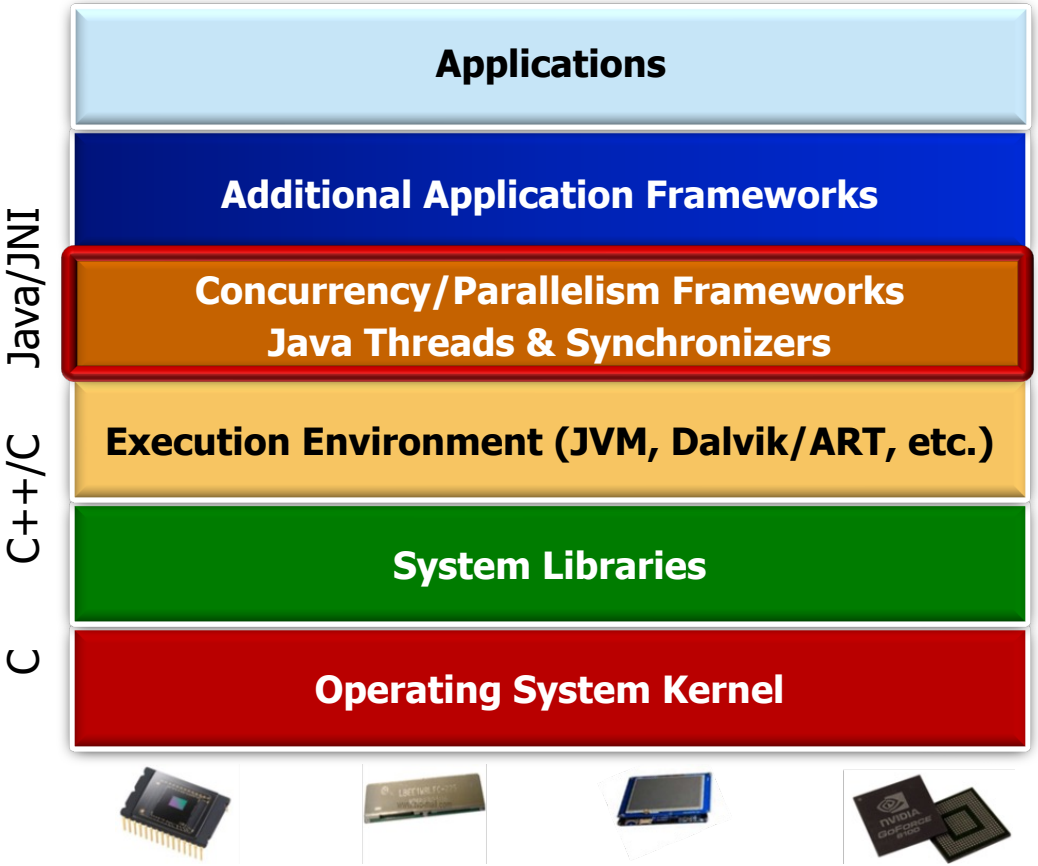




e.g., Android HaMeR or Java ExecutorService & ExecutorCompetionService frameworks

Which Java Mechanism(s) to Understand & Apply

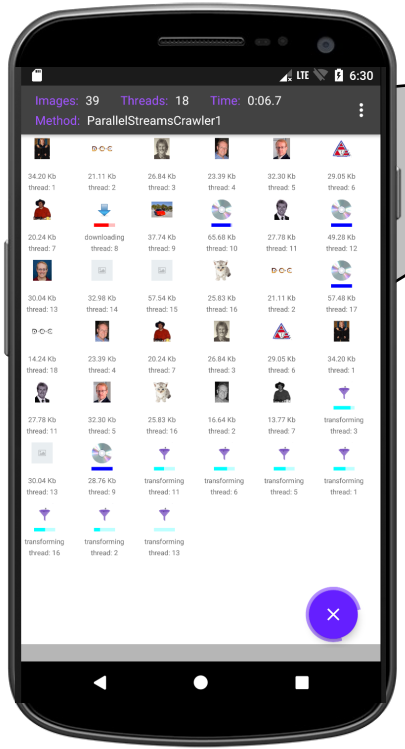
- Framework developers may want to use the Java message passing mechanisms
  - **Pros**: Flexible & decoupled
  - **Cons**: Time/space overhead

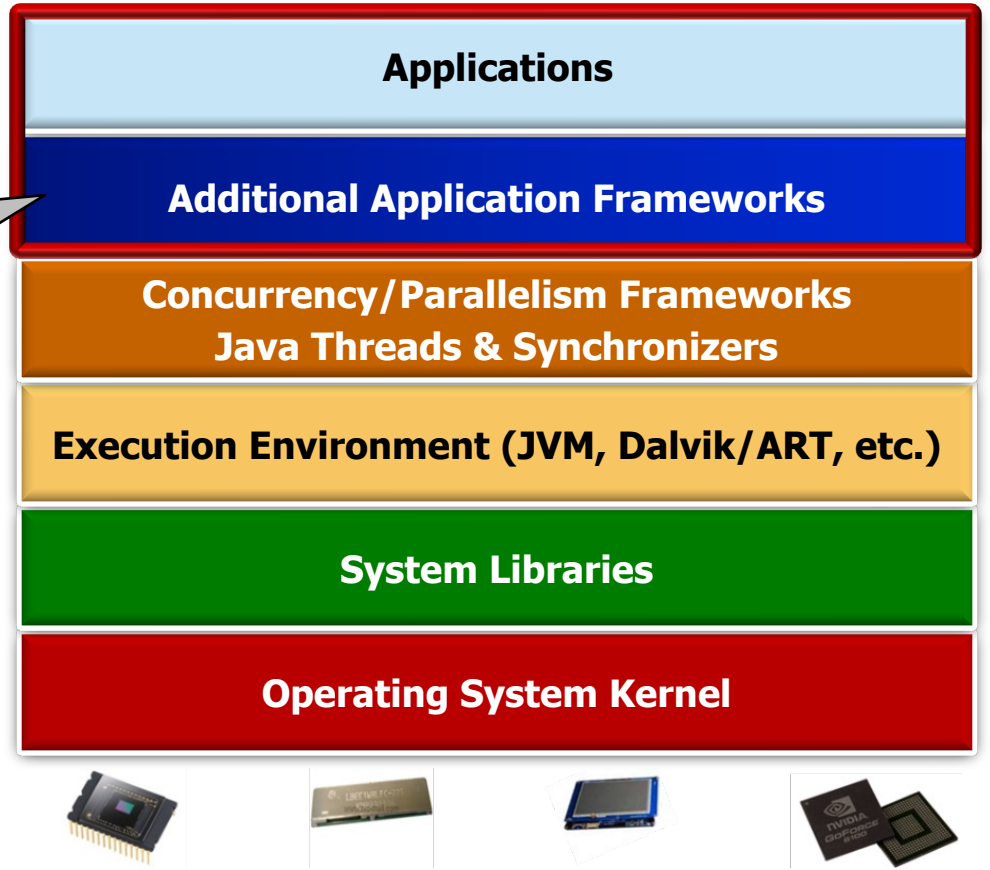


May incur higher context switching, synchronization, & data movement overhead

Which Java Mechanism(s) to Understand & Apply

- App developers most likely want to program w/higher-level frameworks

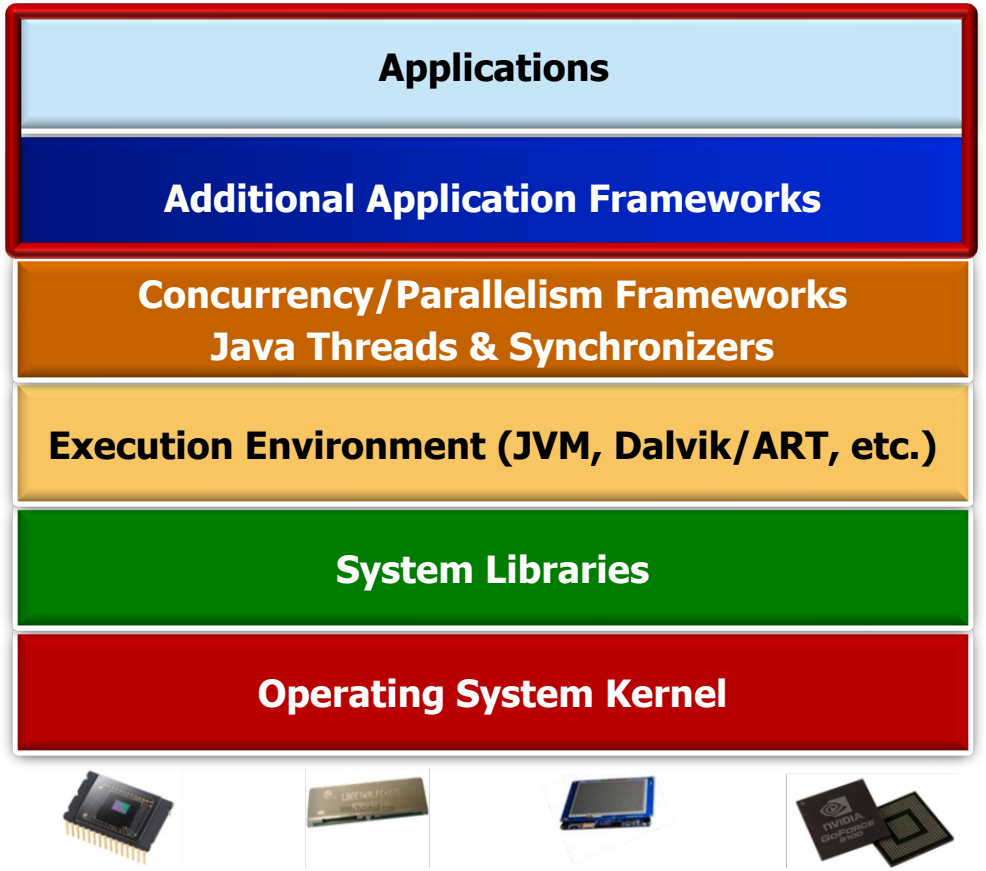




e.g., Java parallel streams, completable futures, RxJava, Project Reactor, etc.

Which Java Mechanism(s) to Understand & Apply

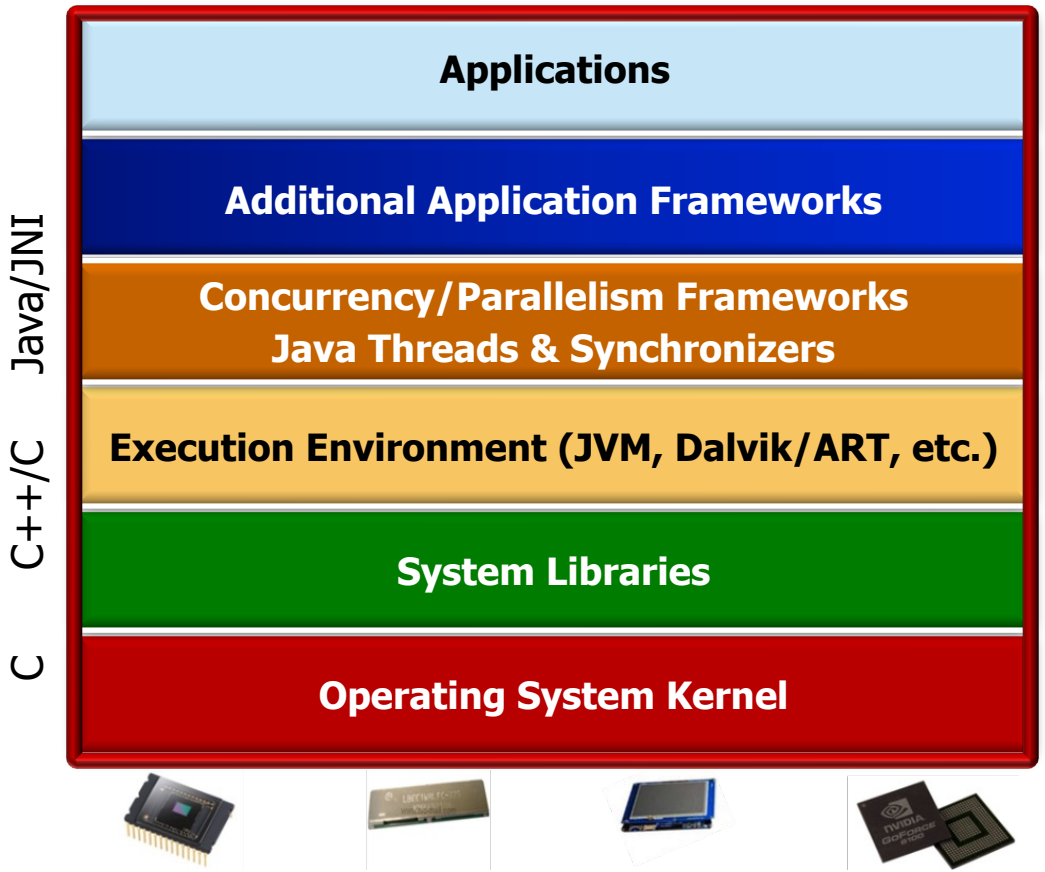
- App developers most likely want to program w/higher-level frameworks
  - **Pros**: Productivity & robustness
  - **Cons**: Time/space overhead & overly prescriptive



Strive to be a Full-Stack Developer!

Strive to be a Full-Stack Developer!

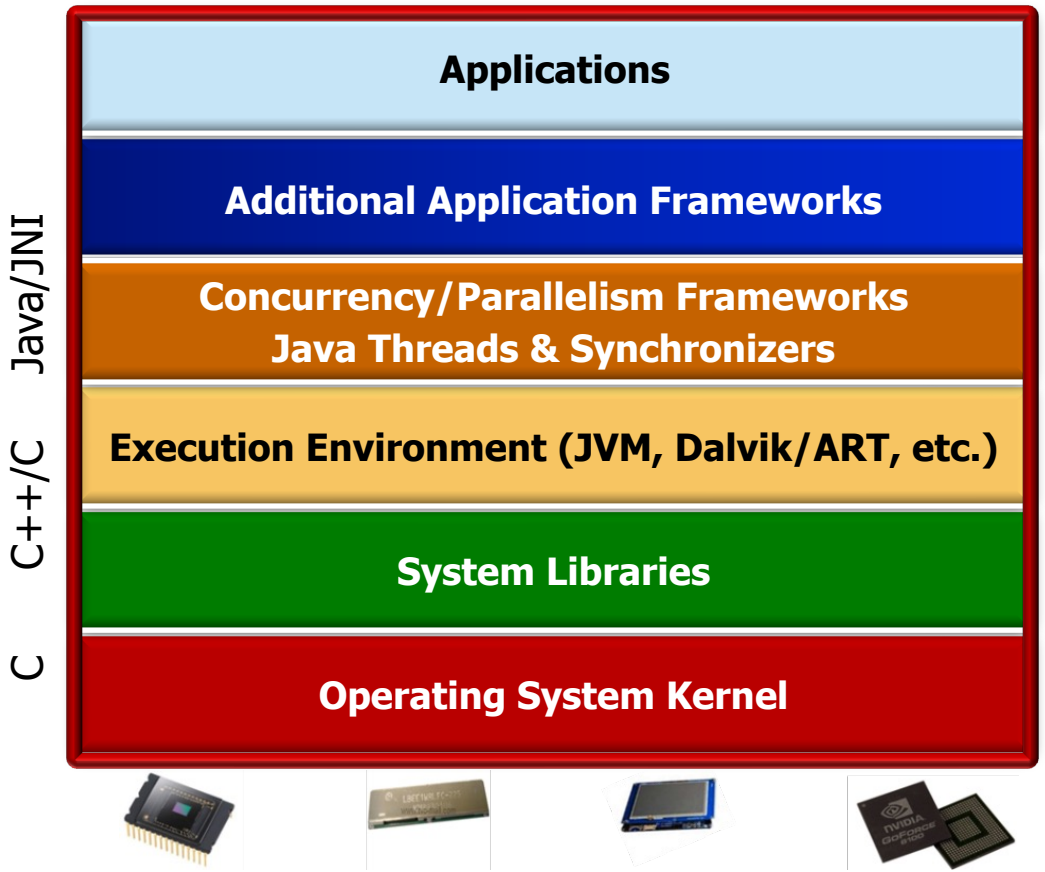
- Full stack" developers should be fluent with every layer!!



See en.everybodywiki.com/Full_Stack_Web_Development

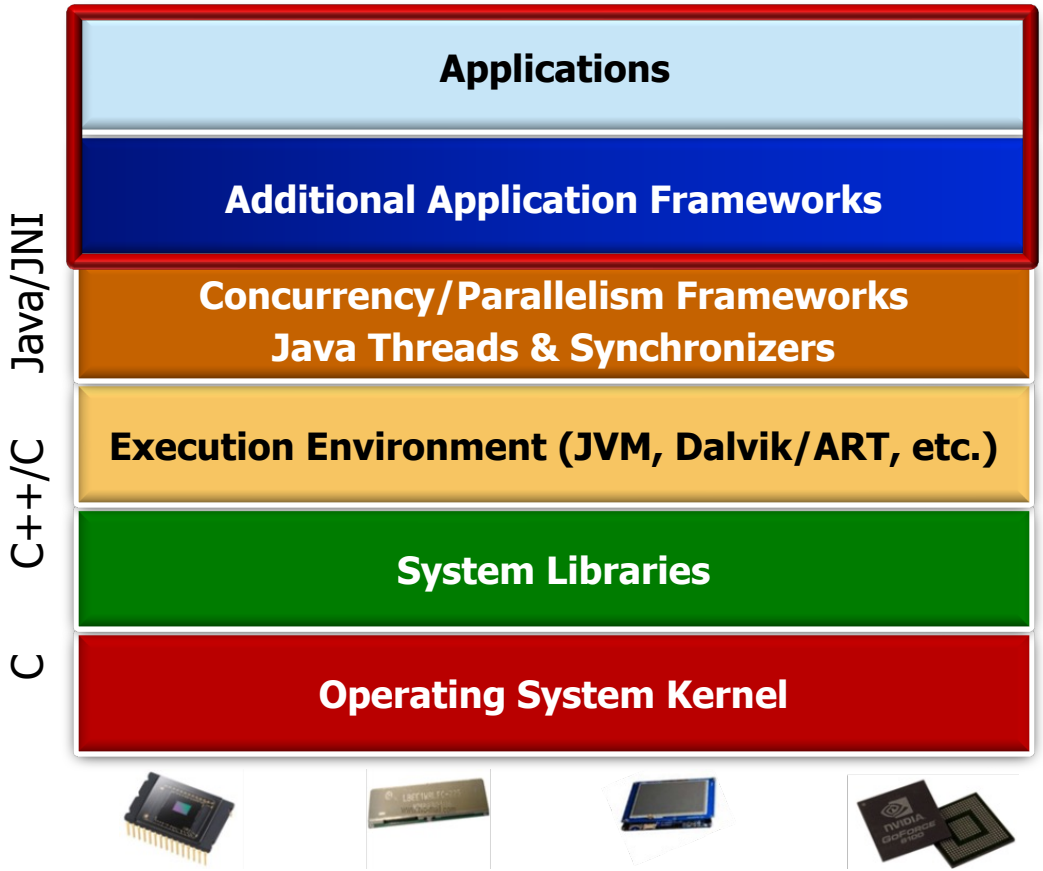
Strive to be a Full-Stack Developer!

- Full stack" developers should be fluent with every layer!!
  - Full-stack developers have expertise in front-end & back -end development



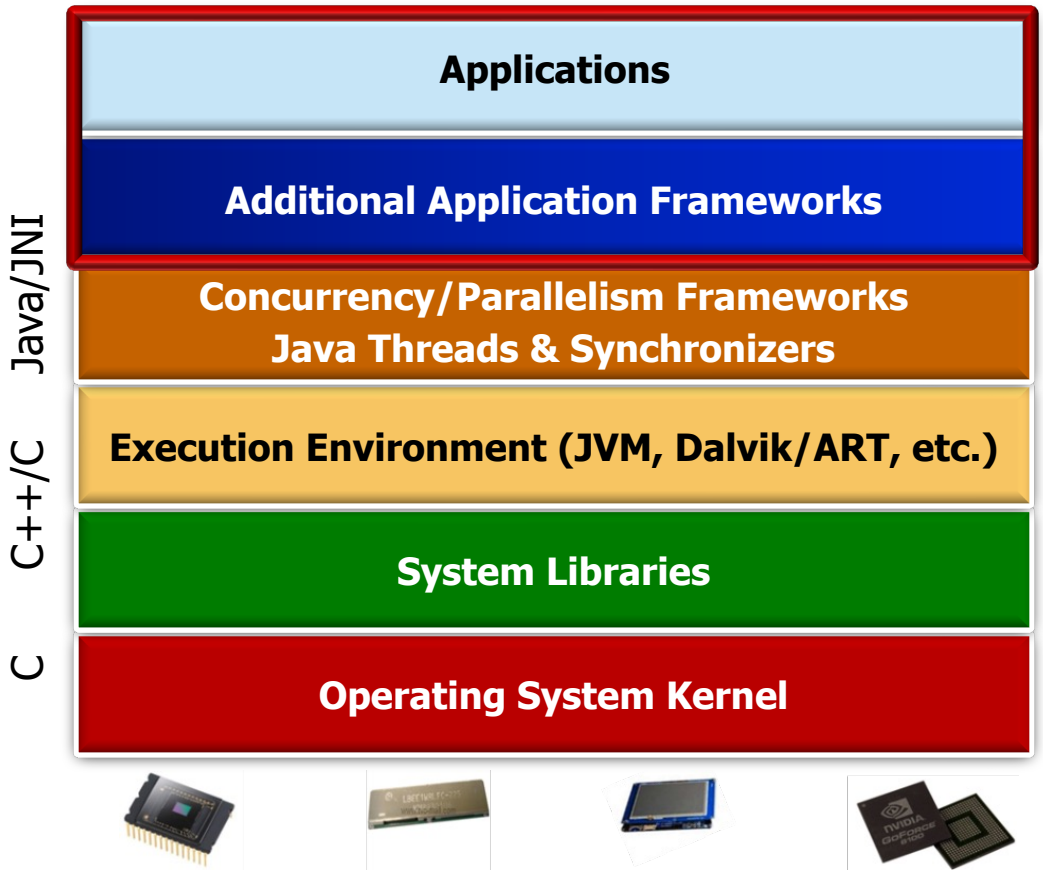
Strive to be a Full-Stack Developer!

- Full stack" developers should be fluent with every layer!!
  - Full-stack developers have expertise in front-end & back -end development, e.g.
    - Can build & maintain all aspects of web apps



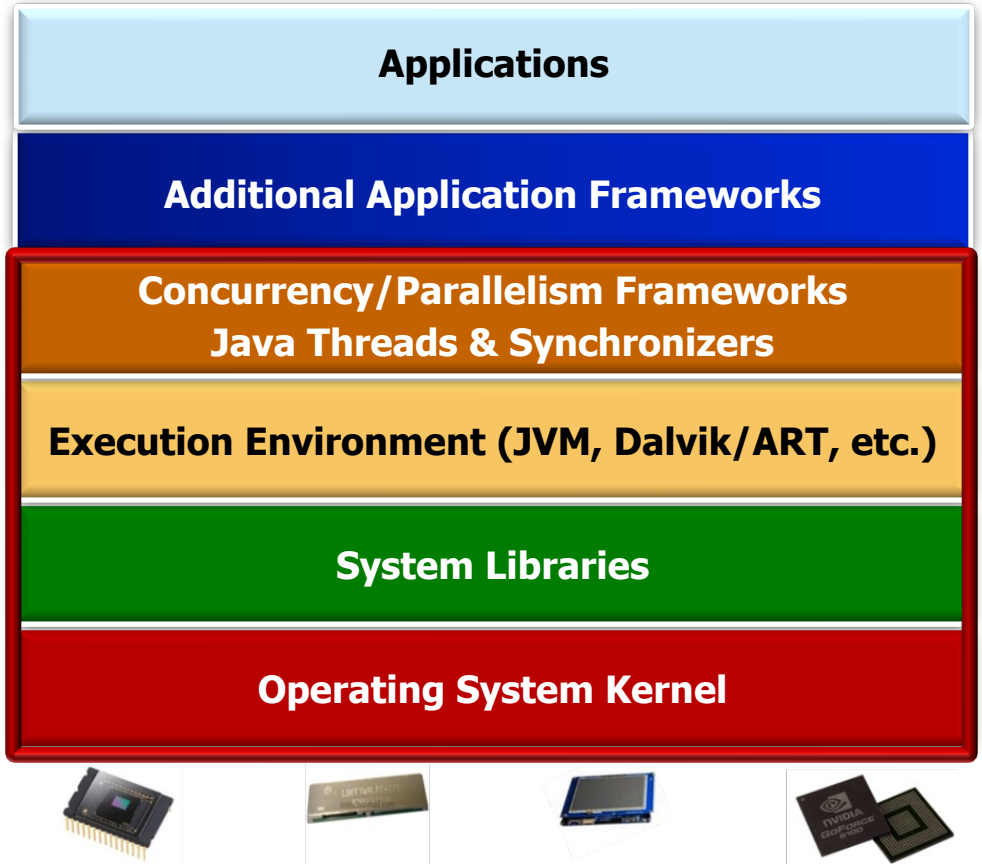
Strive to be a Full-Stack Developer!

- Full stack" developers should be fluent with every layer!!
  - Full-stack developers have expertise in front-end & back -end development, e.g.
    - Can build & maintain all aspects of web apps
      - Such as creating user interfaces, managing databases, developing server-side logic, & even handling deployment & cloud infrastructure



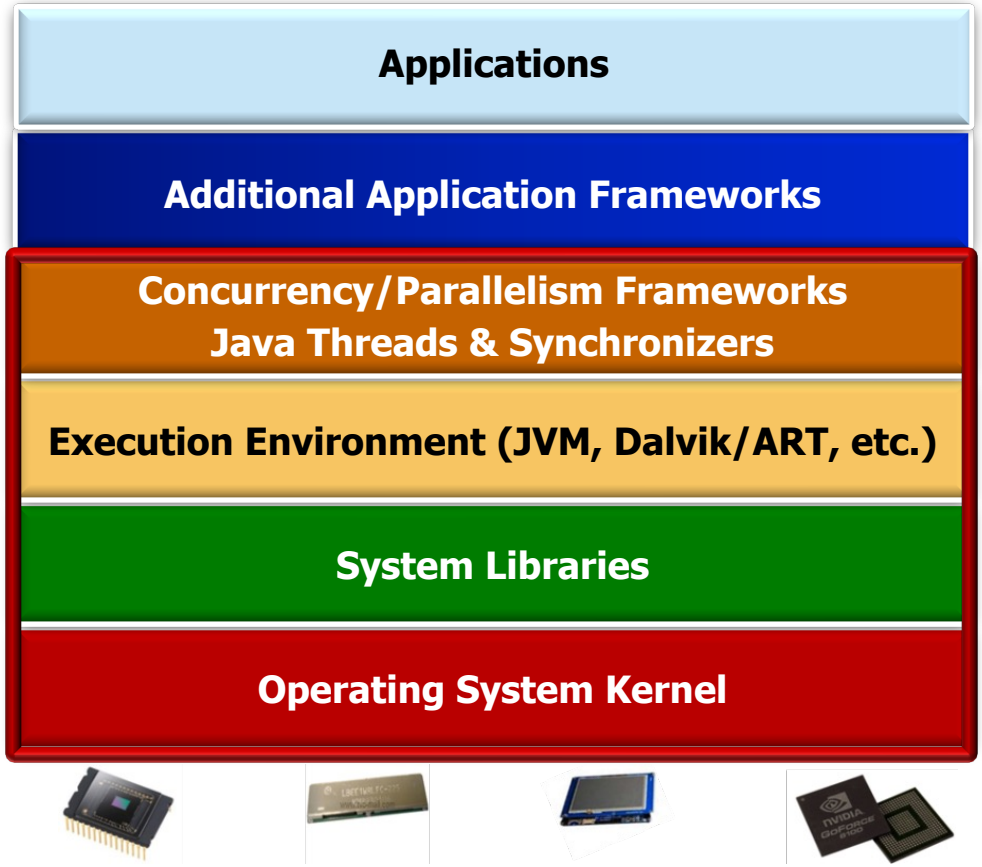
Strive to be a Full-Stack Developer!

- Full stack" developers should be fluent with every layer!!
  - Full-stack developers have expertise in front-end & back -end development, e.g.
    - Can build & maintain all aspects of web apps
    - May also have expertise with Java execution environments & operating systems layers



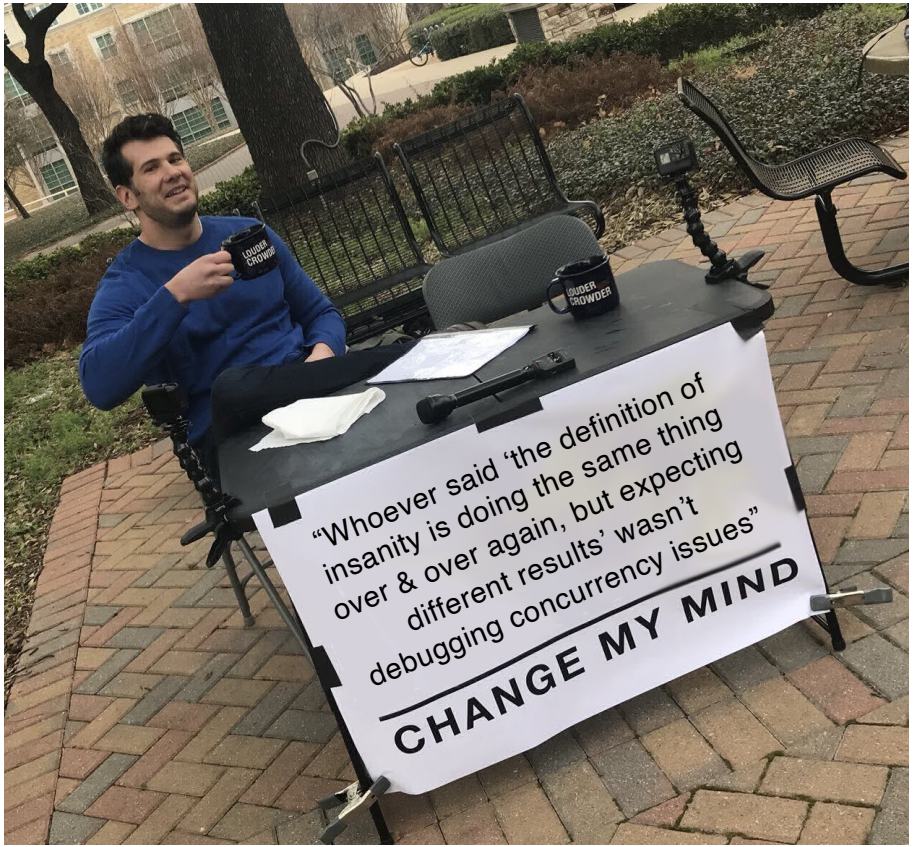
Strive to be a Full-Stack Developer!

- Full stack" developers should be fluent with every layer!!
  - Full-stack developers have expertise in front-end & back -end development, e.g.
    - Can build & maintain all aspects of web apps
    - May also have expertise with Java execution environments & operating systems layers
      - Useful for performance optimization, debugging, & building scalable applications



Strive to be a Full-Stack Developer!

- Regardless of which Java mechanisms you select, be prepared to master the accidental & inherent complexities of concurrent & parallel programming!



End of Evaluating the Concurrency & Parallelism Mechanisms in Java