Mapping Java Completable Future Features Onto Reactive Programming Principles

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

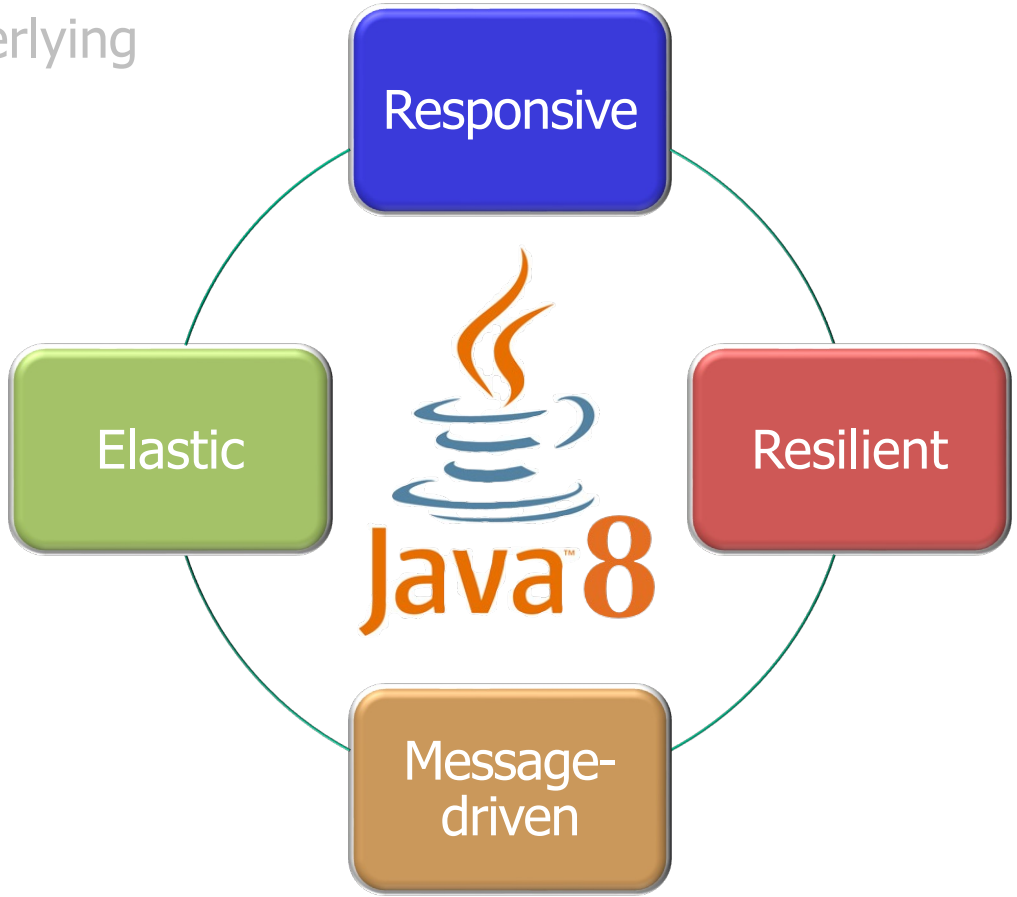
Institute for Software Integrated Systems

Vanderbilt University

Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

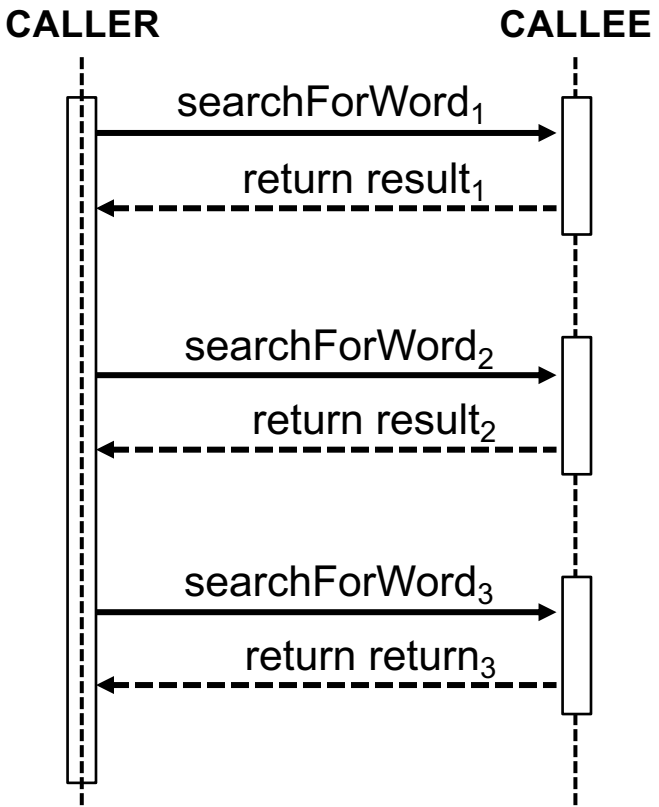
- Understand the key principles underlying reactive programming
- Recognize the Java completable futures framework's structure & functionality
- Learn how the Java completable futures framework maps to key reactive programming principles



Reactive Programming & Java Completable Futures

Reactive Programming & Java Completable Futures

- Java completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
    - Avoid blocking caller code
      - Blocking underutilizes cores, impedes inherent parallelism, & complicates program structure

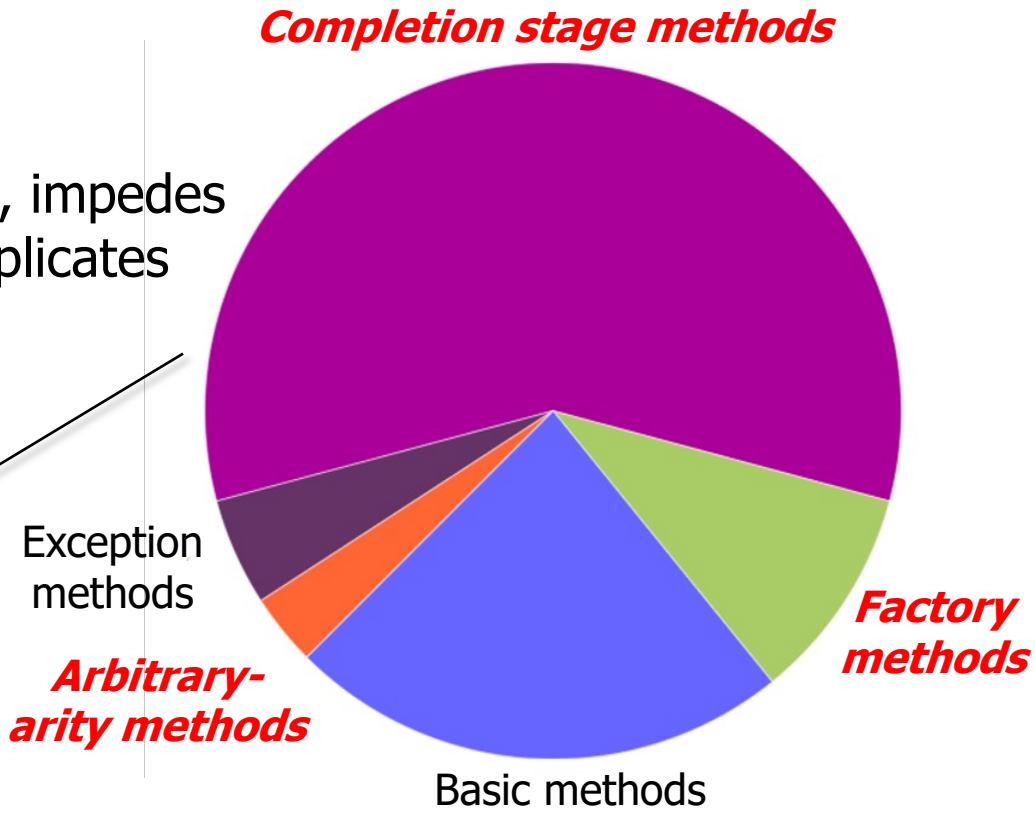


See www.nastel.com/10-reasons-your-java-apps-are-slow

Reactive Programming & Java Completable Futures

- Java completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
    - Avoid blocking caller code
      - Blocking underutilizes cores, impedes inherent parallelism, & complicates program structure

Factory, completion stage, & arbitrary-arity methods avoid blocking threads



Reactive Programming & Java Completable Futures

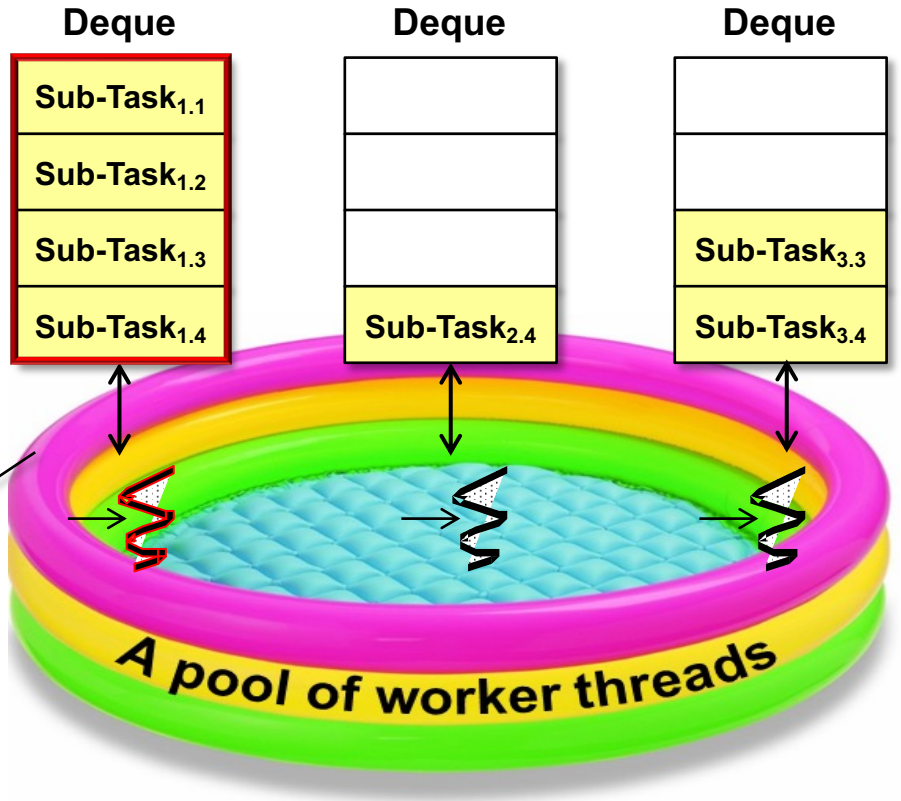
- Java completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
    - Avoid blocking caller code
    - Avoid changing threads
      - Incurs excessive overhead wrt synchronization, context switching, & memory/cache management

See www.dre.vanderbilt.edu/~schmidt/dissertations/schmidt-dissertation.pdf

Reactive Programming & Java Completable Futures

- Java completable futures map onto key reactive programming principles, e.g.
  - **Responsive**
    - Avoid blocking caller code
    - Avoid changing threads
      - Incurs excessive overhead wrt synchronization, context switching, & memory/cache management

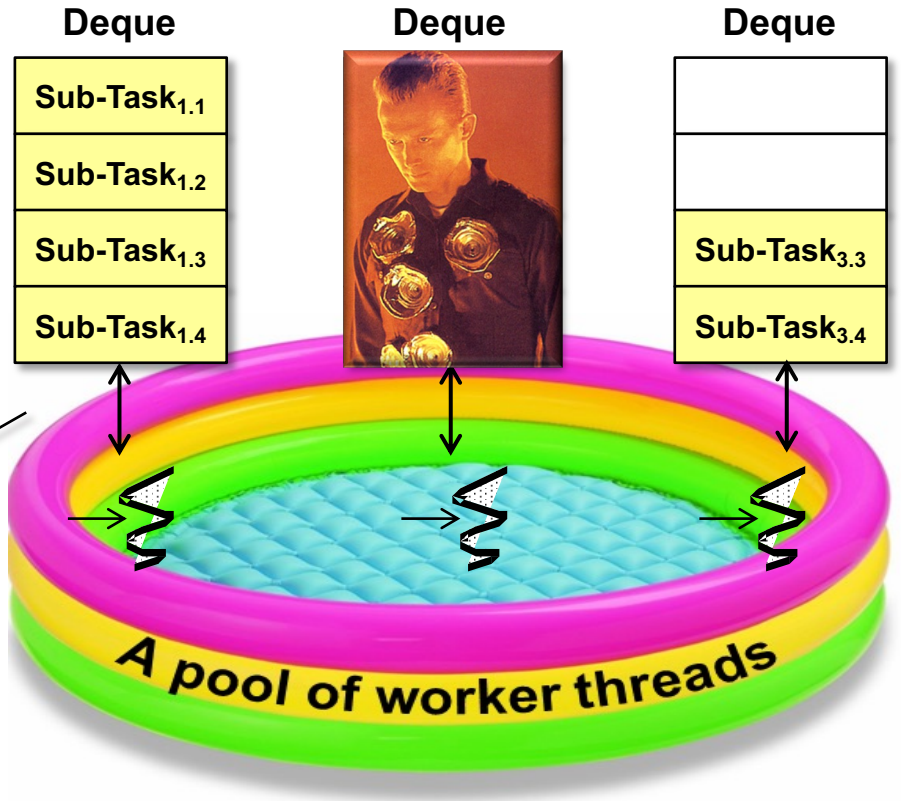
*The fork-join pool & non-*Async() methods avoid changing threads*



See gee.cs.oswego.edu/dl/papers/fj.pdf

Reactive Programming & Java Completable Futures

- Java completable futures map onto key reactive programming principles, e.g.
  - Responsive
  - **Resilient**
    - Exception methods make more programs resilient to failures

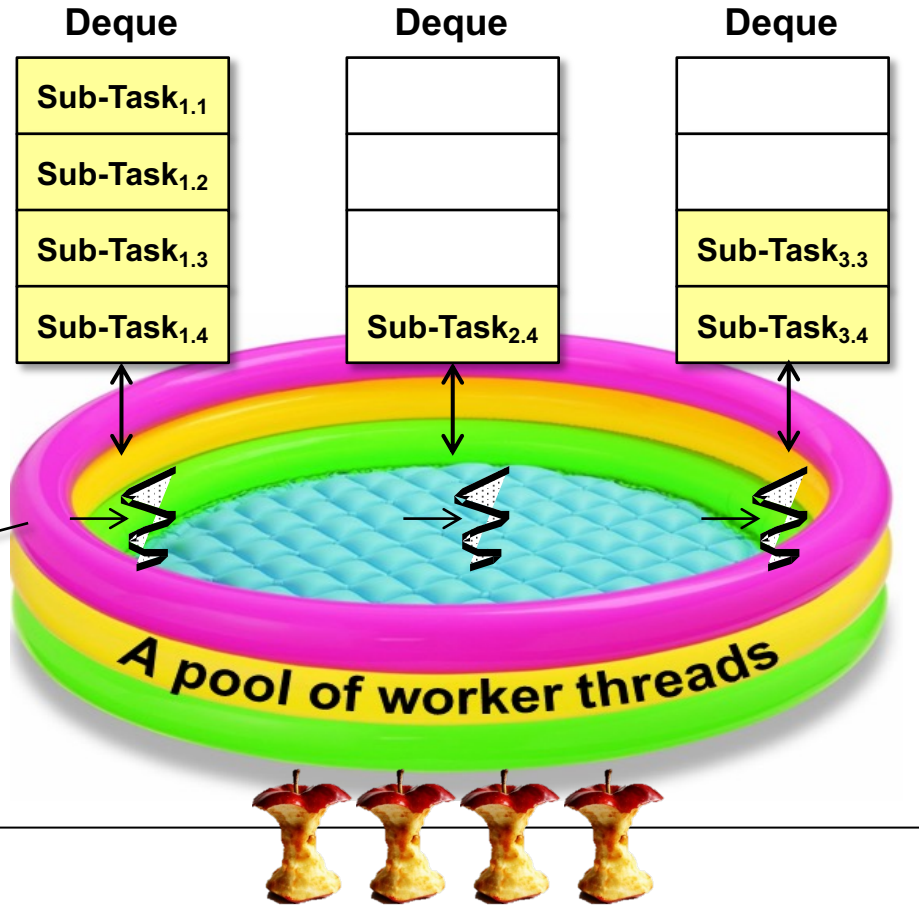


Exceptions decouple error processing from normal operations

Completable futures are localized to a single process, *not* a cluster!

Reactive Programming & Java Completable Futures

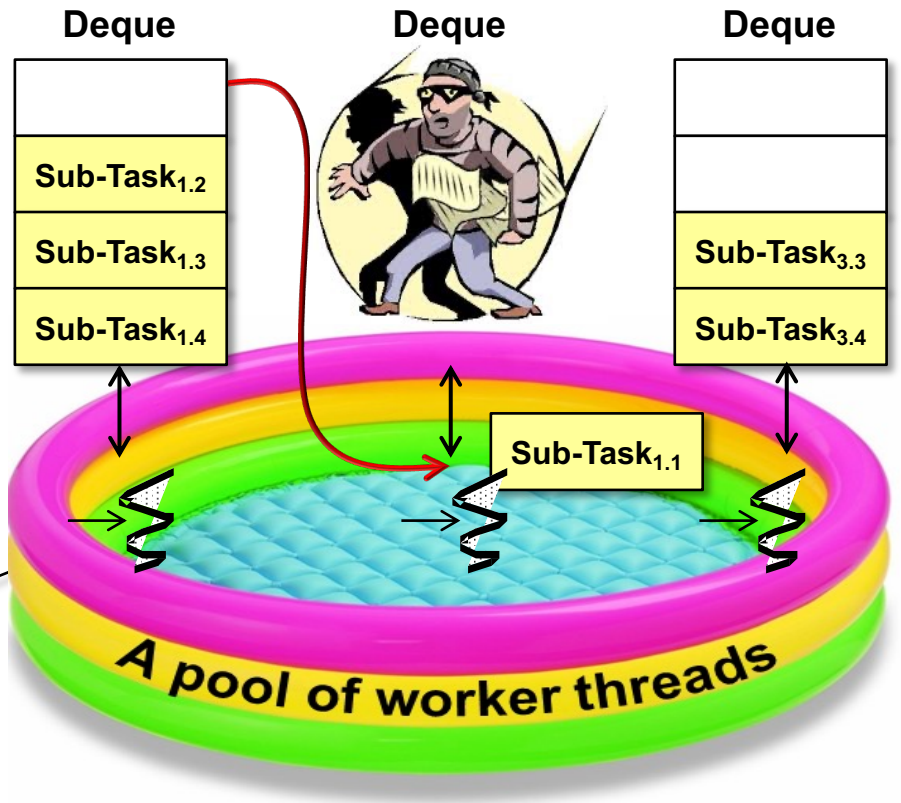
- Java completable futures map onto key reactive programming principles, e.g.
  - Responsive
  - Resilient
  - **Elastic**
    - Async computations can run scalably in a pool of threads atop a set of cores



Can be a (common) fork-join pool or a pre- or user-defined thread pool

Reactive Programming & Java Completable Futures

- Java completable futures map onto key reactive programming principles, e.g.
  - Responsive
  - Resilient
  - Elastic
  - **Message-driven**
    - Java's thread pools pass messages between threads in the pool internally



e.g., the Java completable futures & fork-join frameworks both use async message passing

See en.wikipedia.org/wiki/Work_stealing

End of Mapping Java Completable Future Features Onto Reactive Programming Principles