Java Parallel Streams Internals:
Order of Processing

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 parallel stream internals, e.g.
• Know what can change & what can’t
  • Splitting, combining, & pooling mechanisms
• Order of processing
Java Parallel Stream Processing Order
The order in which chunks in a parallel stream are processed is non-deterministic.

See en.wikipedia.org/wiki/Nondeterministic_algorithm
The order in which chunks in a parallel stream are processed is non-deterministic.

Stream factory operation ()
Input x
Intermediate operation (behavior f)
Output f(x)
Intermediate operation (behavior g)
Output g(f(x))
Terminal operation (reducer)

The ordering can exhibit different behaviors on different runs, even for the same input.
The order in which chunks in a parallel stream are processed is non-deterministic.

Programmers have little/no control over how chunks are processed.
Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic
  - Programmers have little/no control over how chunks are processed
  - Non-determinism enables optimizations at multiple layers!

- e.g., scheduling & execution of tasks via fork-join pool, JVM, hardware cores, etc.
Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic
  - Programmers have little/no control over how chunks are processed
  - Non-determinism enables optimizations at multiple layers!

- e.g., fork-join framework’s support for work-stealing is a non-deterministic optimization

See upcoming lessons on “The Java Fork-Join Framework”
End of Java Parallel Stream Internals: Order of Processing