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 Java parallel streams framework allows for variability in the order of its processing, while still being deterministic in the processing results.
The order in which chunks in a parallel stream are processed is non-deterministic.
Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic.

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

- Stream factory operation()
- Input x
- Intermediate operation (behavior f)
- Output f(x)
- Intermediate operation (behavior g)
- Output g(f(x))
- Terminal operation (reducer)
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.
• 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!

*Java Parallel Stream Processing Order*

---

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 Understand Java Parallel Streams Internals: Order of Processing Overview