

# Java Parallel Streams Internals: Order of Processing Overview

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://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

# 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



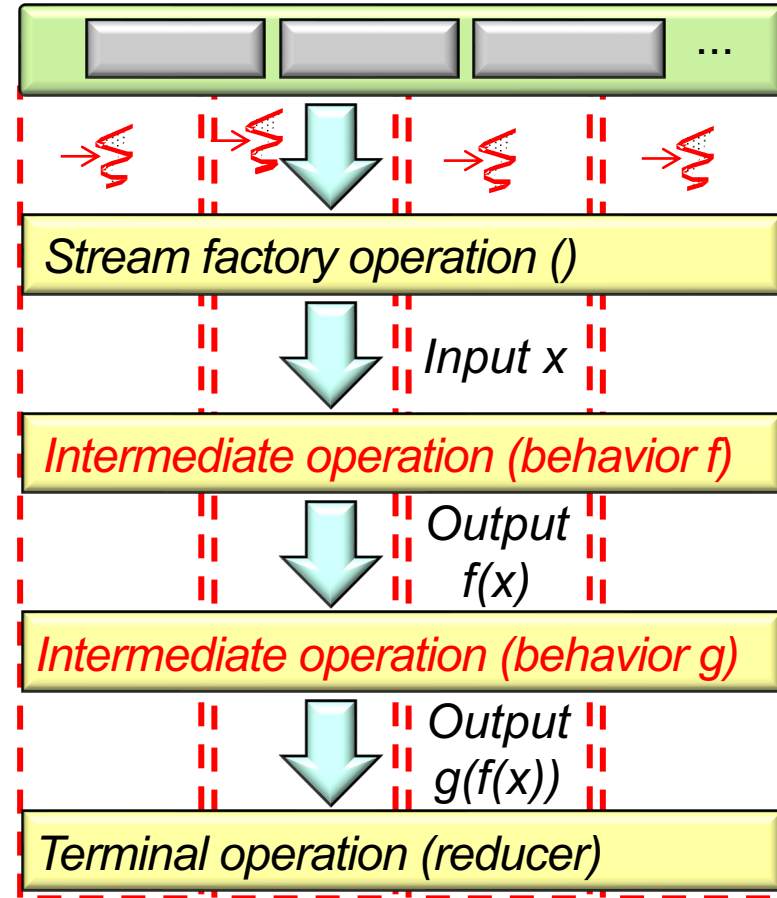
*Non-deterministic  
processing order*



*Deterministic  
processing results*

# Java Parallel Stream Processing Order

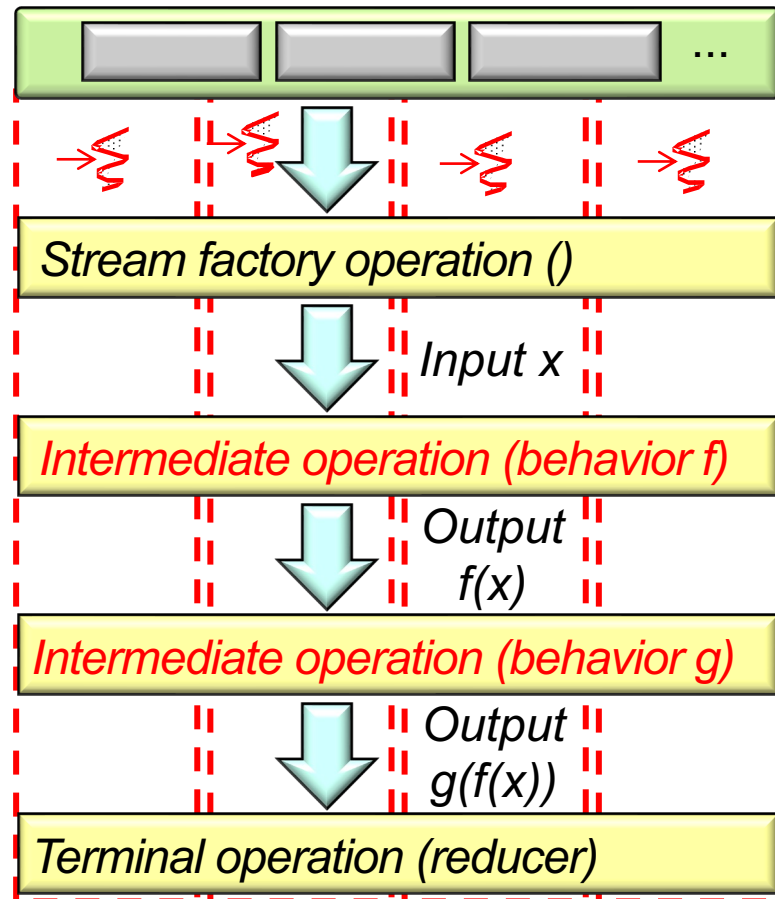
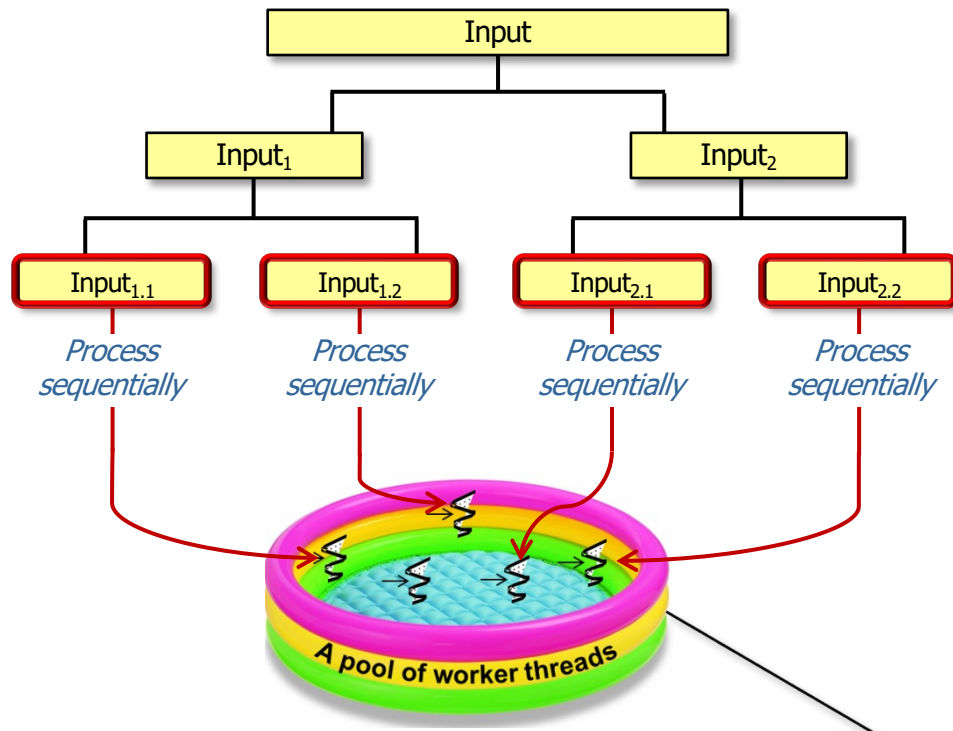
- The *order* in which chunks in a parallel stream are processed is non-deterministic



See [en.wikipedia.org/wiki/Nondeterministic\\_algorithm](https://en.wikipedia.org/wiki/Nondeterministic_algorithm)

# Java Parallel Stream Processing Order

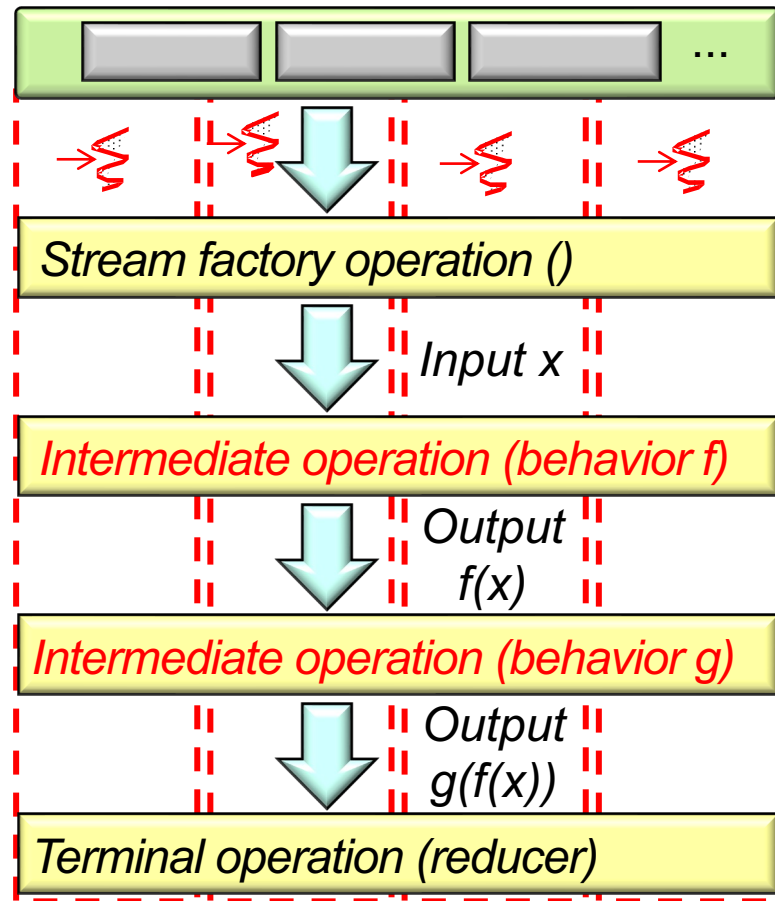
- The *order* in which chunks in a parallel stream are processed is non-deterministic



*The ordering can exhibit different behaviors on different runs, even for the same input*

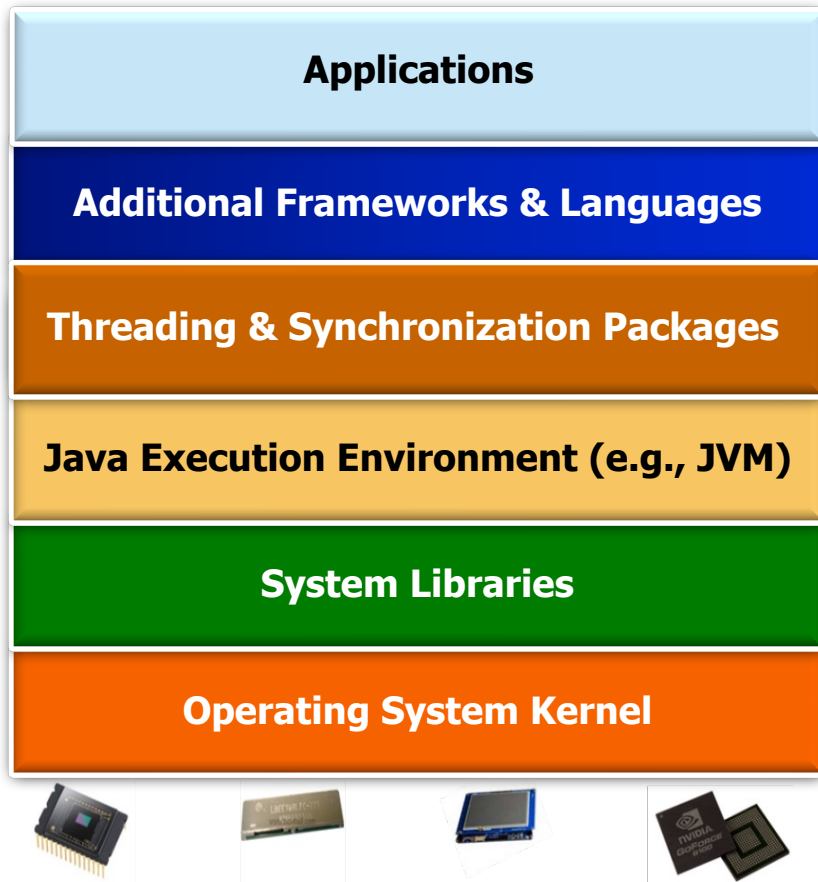
# Java Parallel Stream Processing Order

- The *order* in which chunks in a parallel stream are processed is non-deterministic
- Programmers have little/no control over how chunks are processed



# 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!



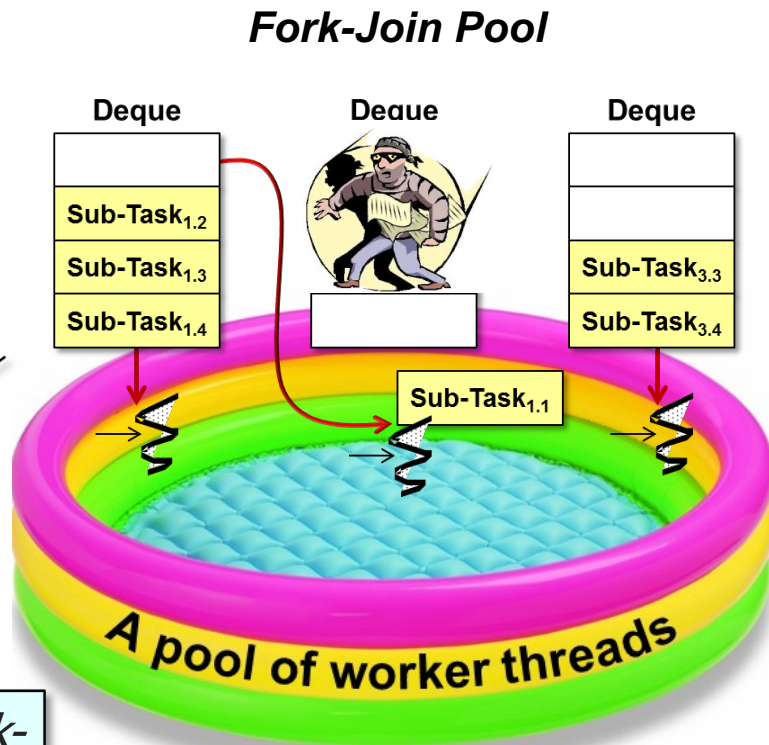
e.g., scheduling & execution of tasks via fork-join pool, JVM, hardware cores, etc.



# 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 Streams

## Internals: Order of Processing Overview