Visualizing Java Streams in Action

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

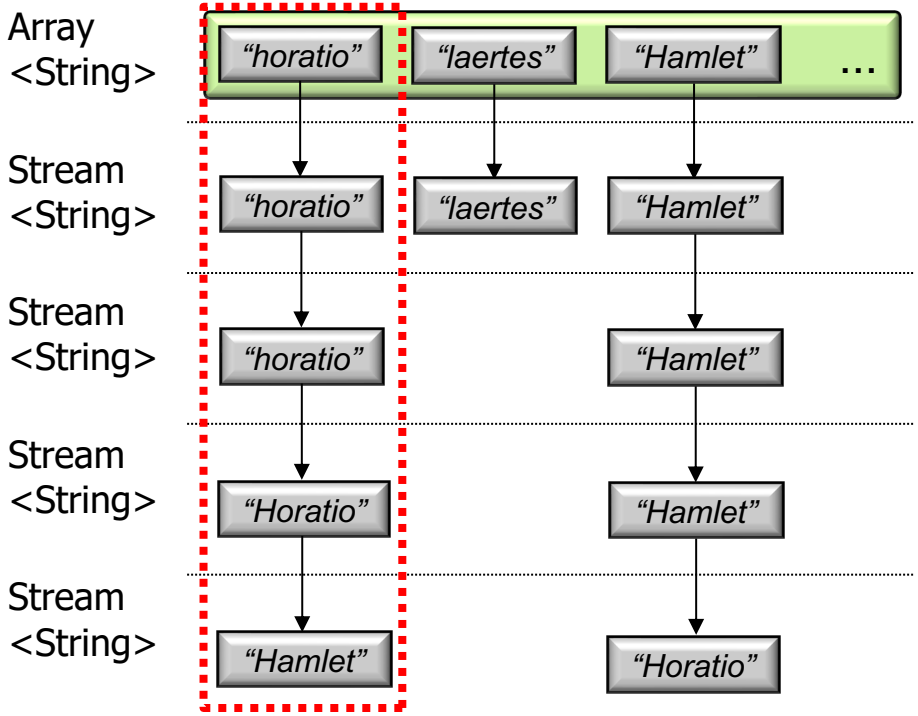
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand Java streams structure & functionality, e.g.
  - Fundamentals of streams
  - Three streams phases
  - Operations that create a stream
  - Aggregate operations in a stream
  - Visualizing streams in action

Learning Objectives in this Part of the Lesson

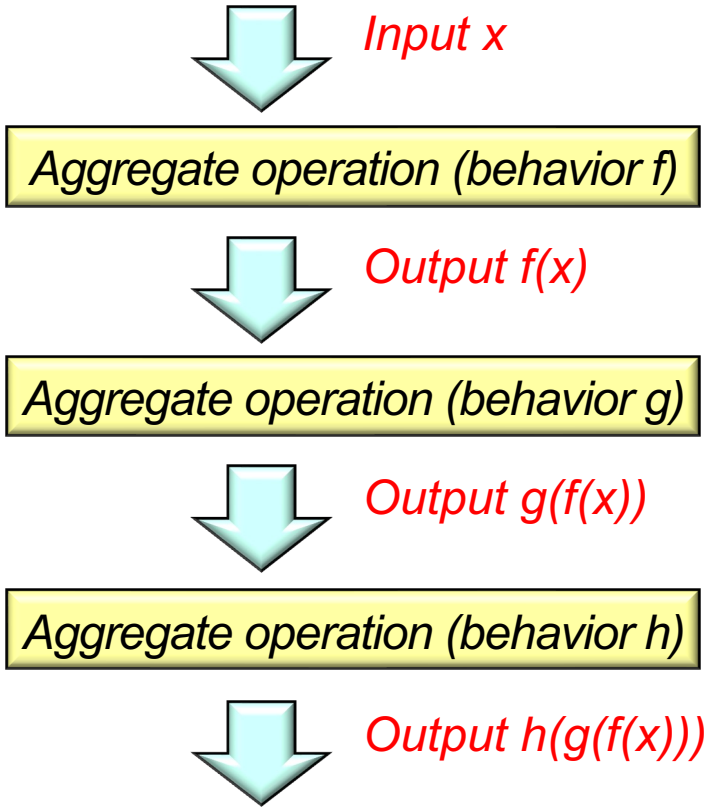
- Understand Java streams structure & functionality, e.g.
  - Fundamentals of streams
  - Three streams phases
  - Operations that create a stream
  - Aggregate operations in a stream
  - Visualizing streams in action
    - Be aware of how Java streams work in practice



Visualizing Streams in Action

Visualizing Streams in Action

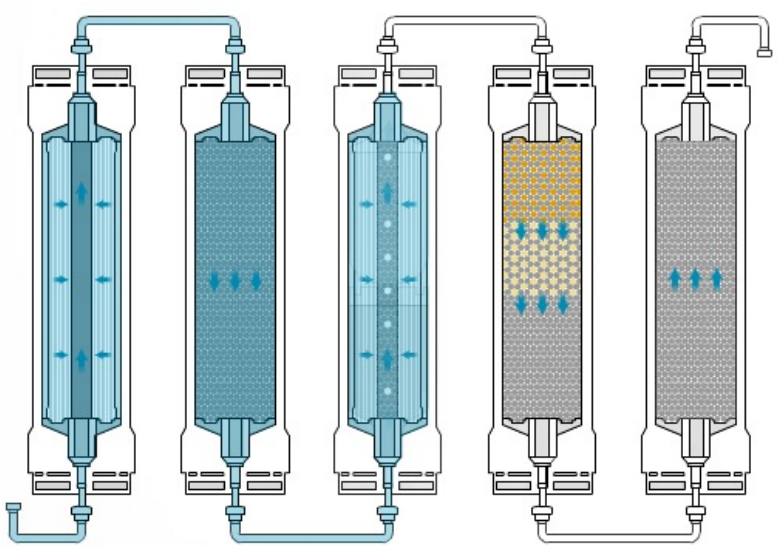
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together



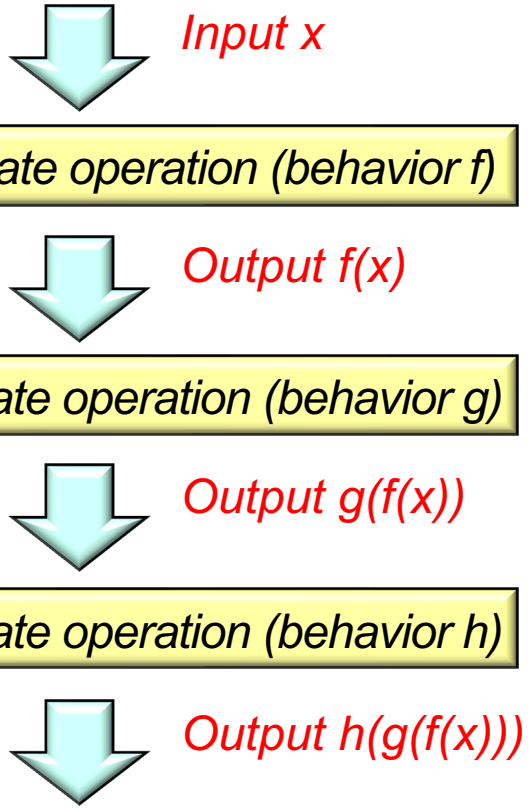
See en.wikipedia.org/wiki/Pipeline_(software)

Visualizing Streams in Action

- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together



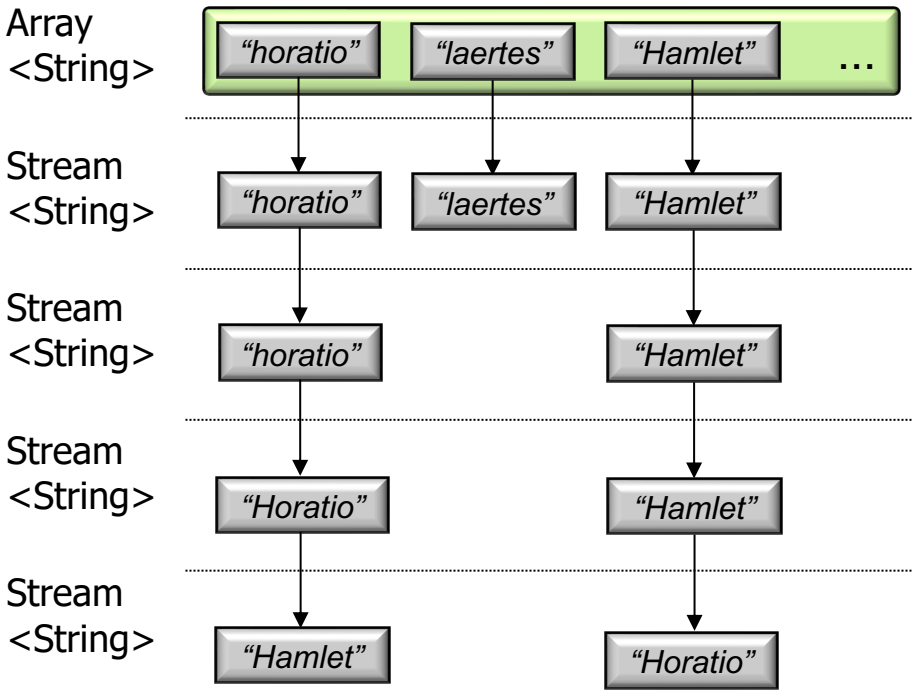
Each aggregate operation in the pipeline can filter and/or transform the stream.

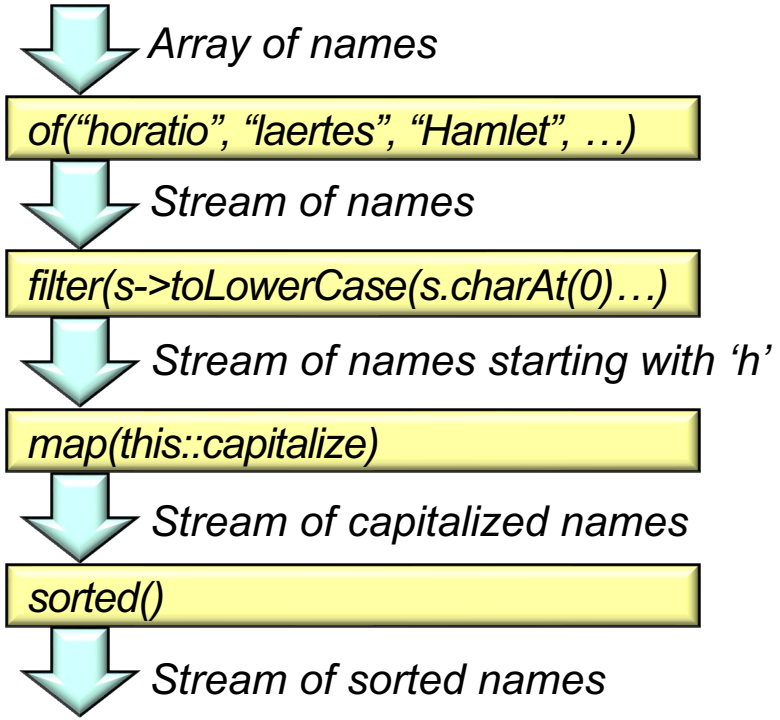


See en.wikipedia.org/wiki/Water_filter#Point-of-use_filters

Visualizing Streams in Action

- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

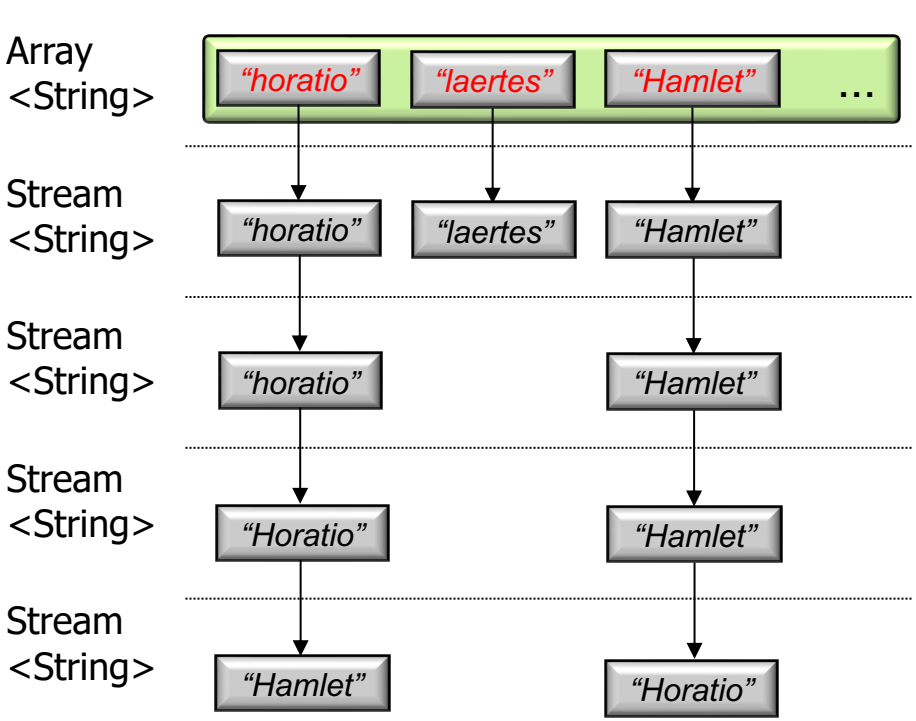


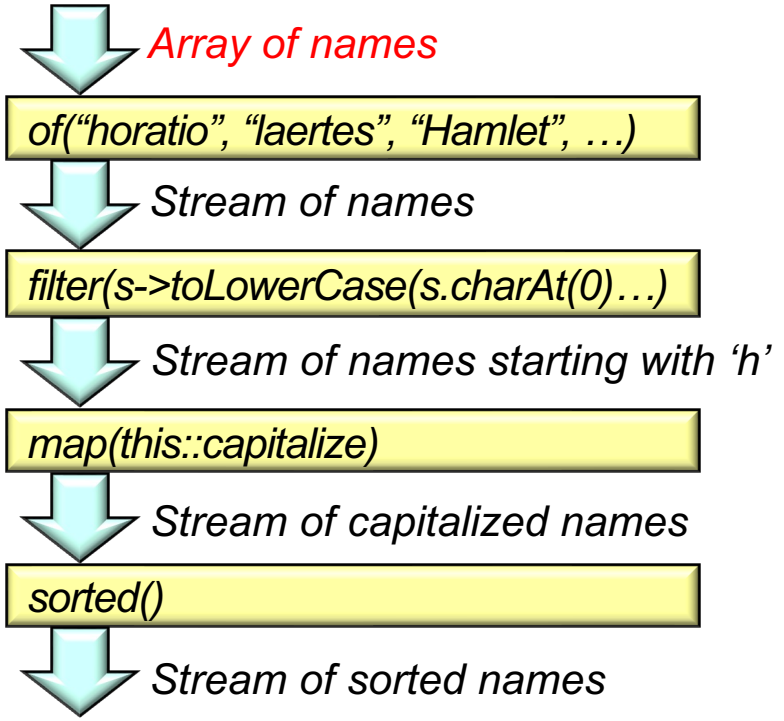


See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex12

Visualizing Streams in Action

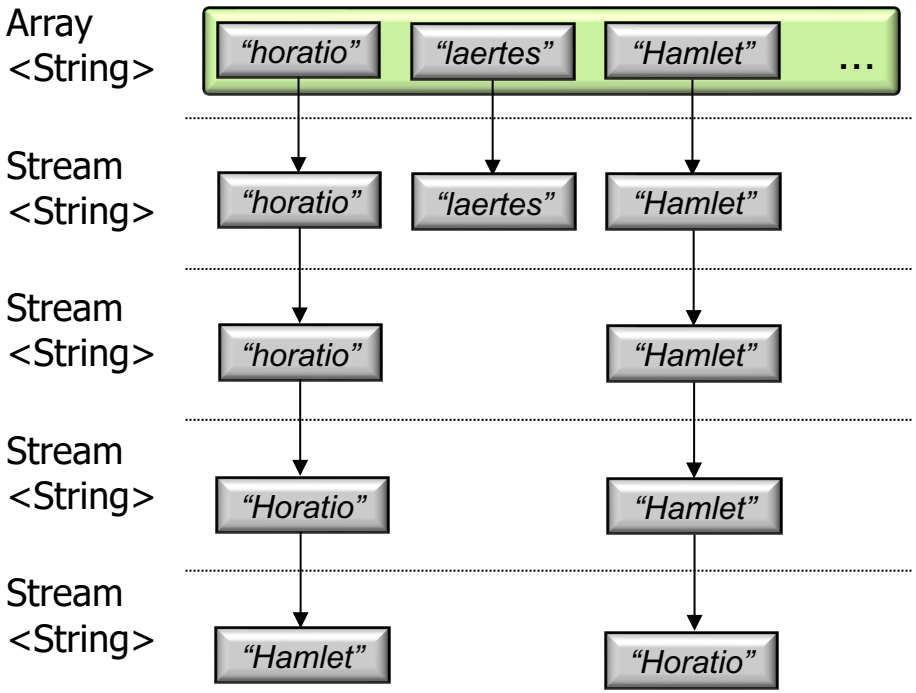
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

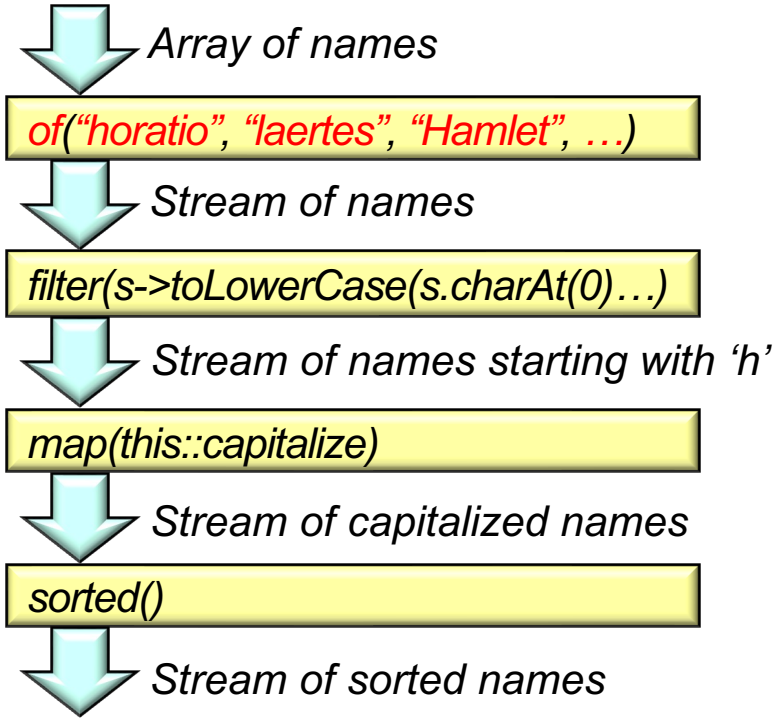




Visualizing Streams in Action

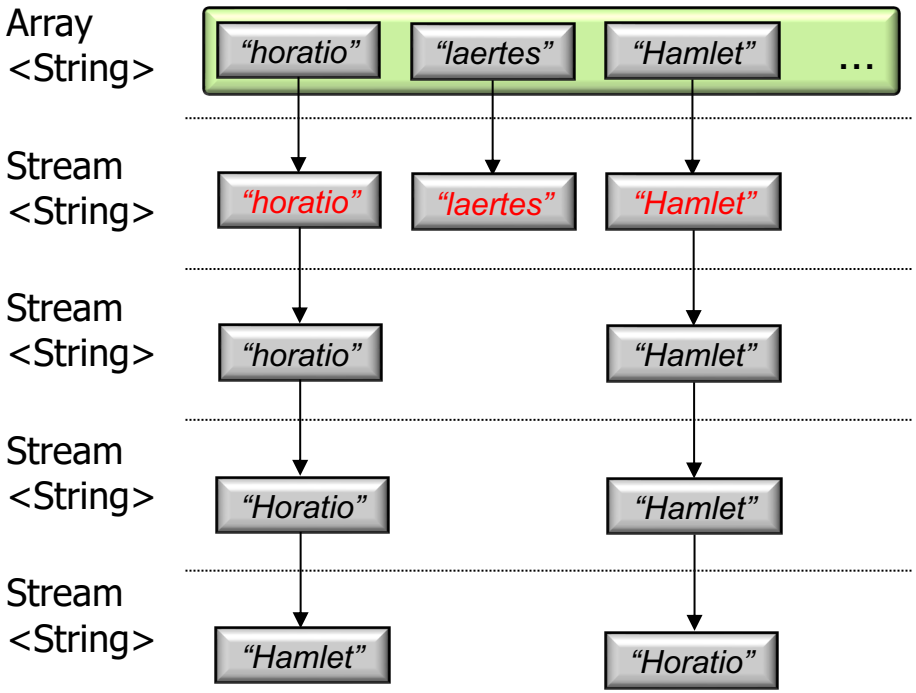
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

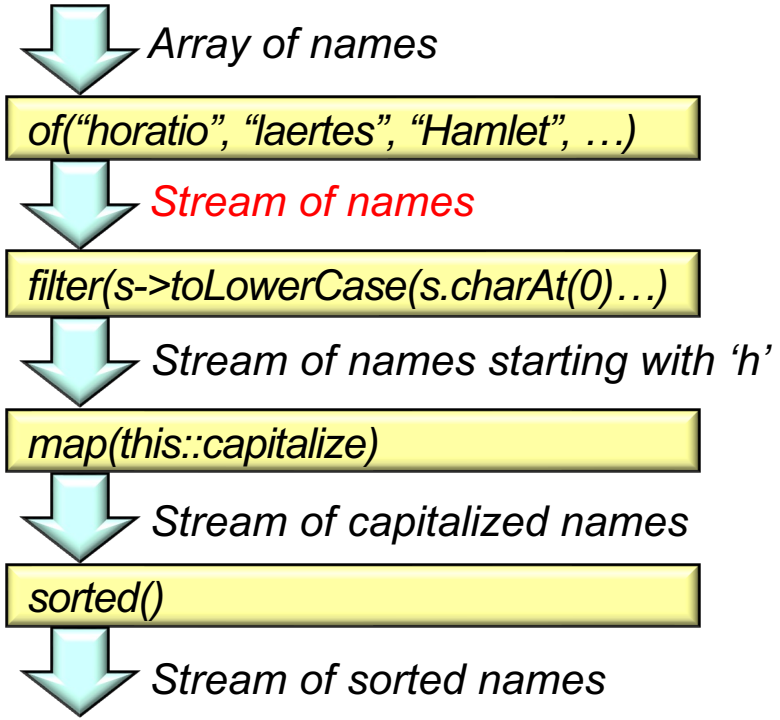




Visualizing Streams in Action

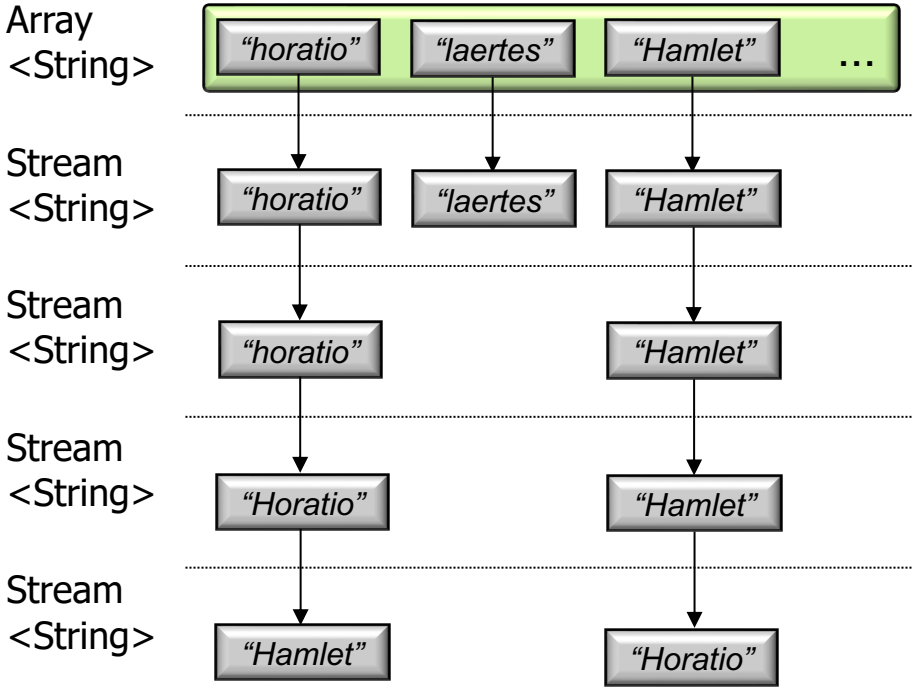
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

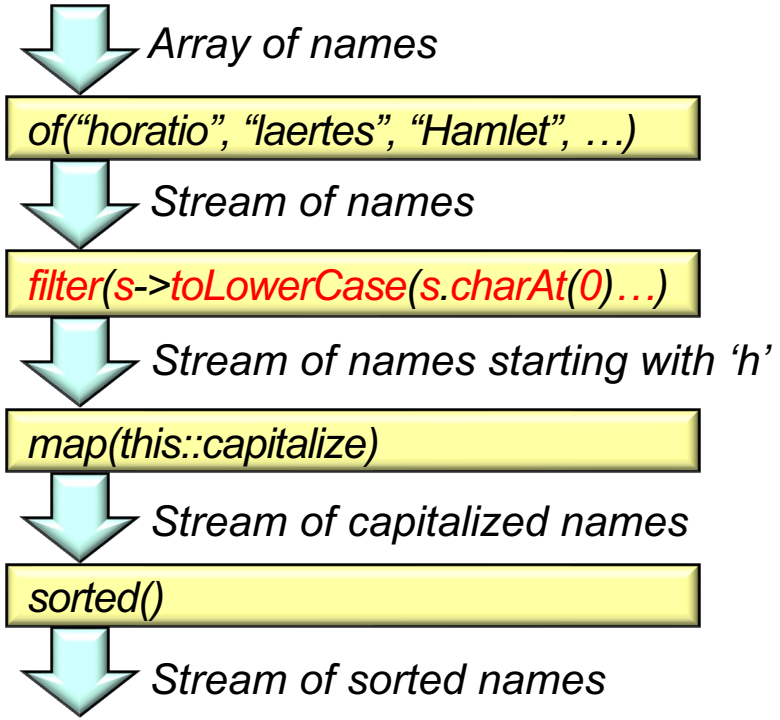




Visualizing Streams in Action

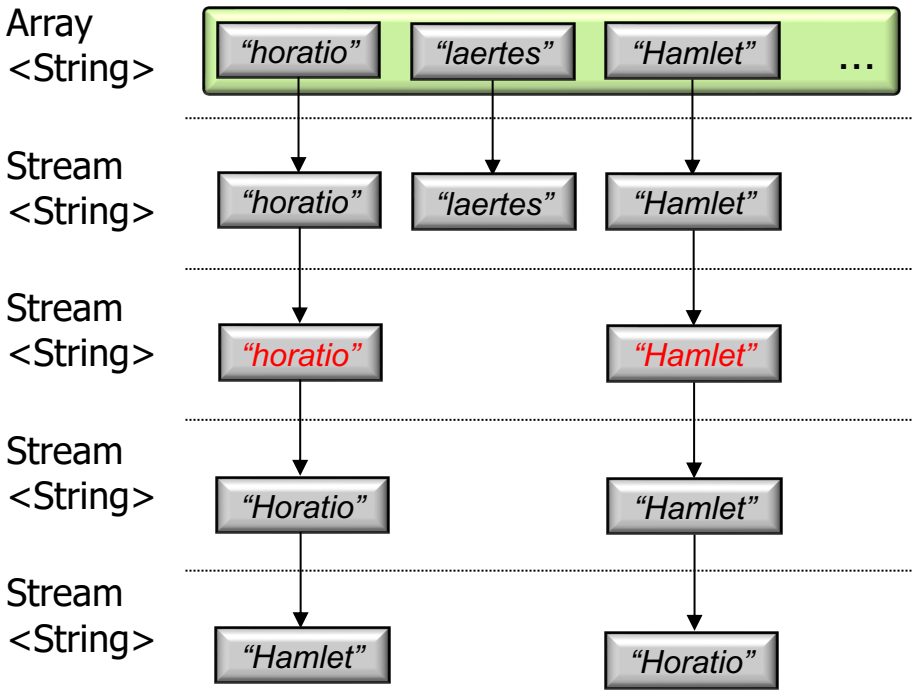
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

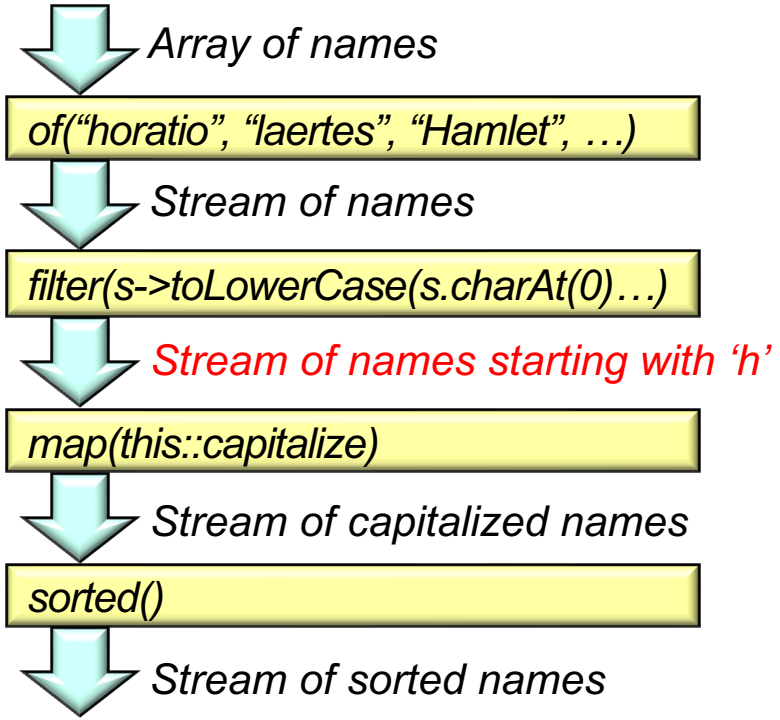




Visualizing Streams in Action

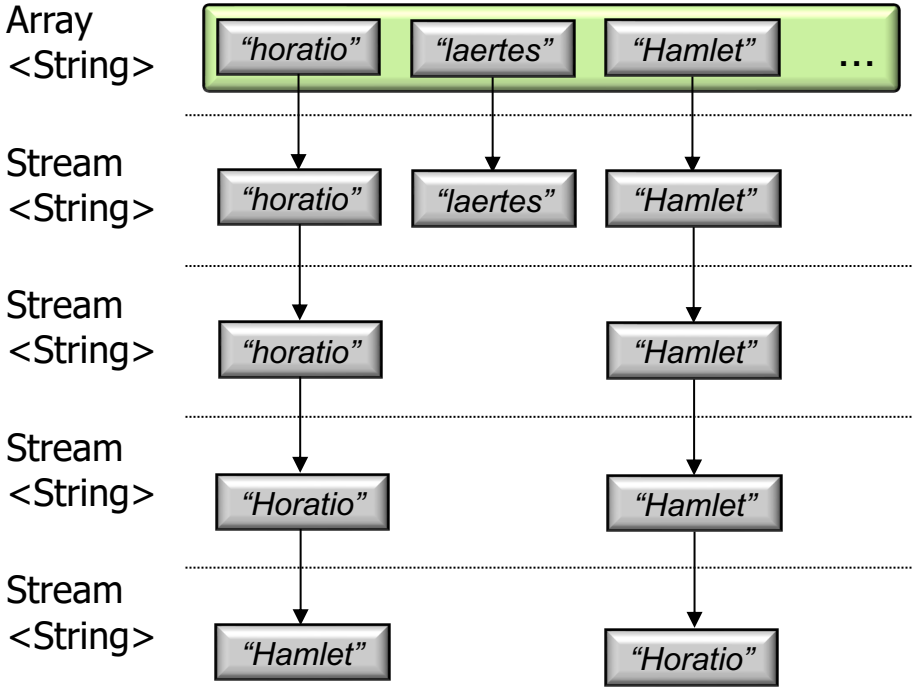
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

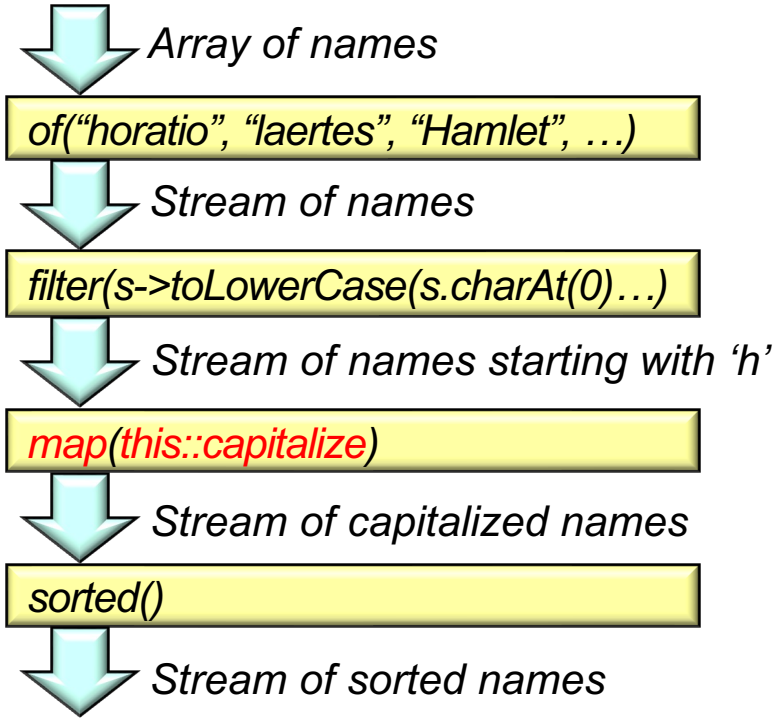




Visualizing Streams in Action

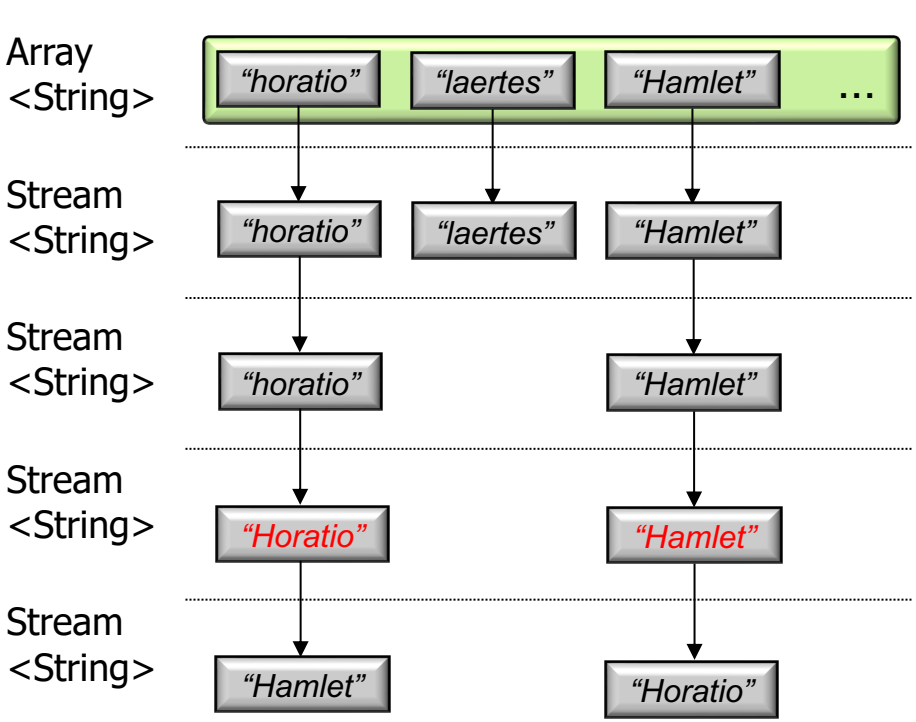
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

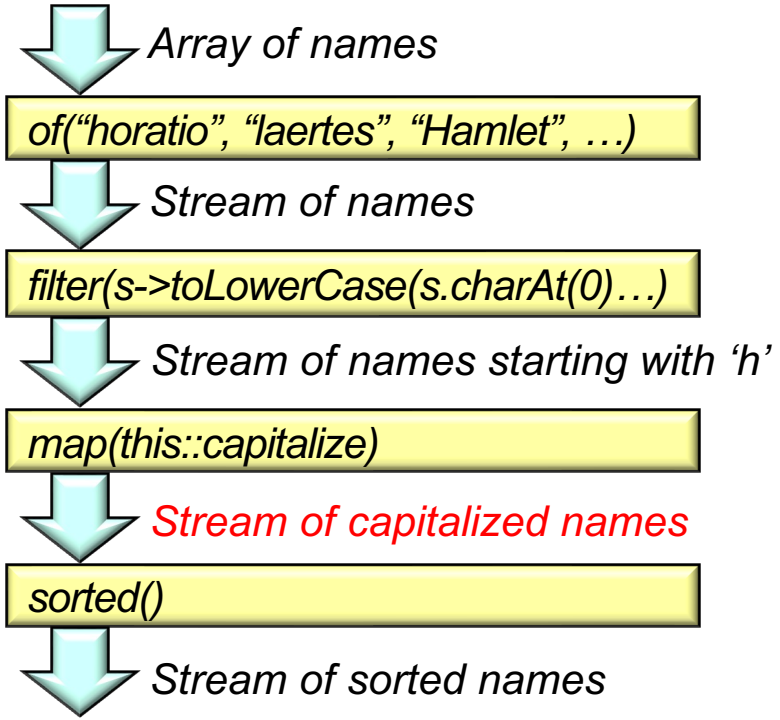




Visualizing Streams in Action

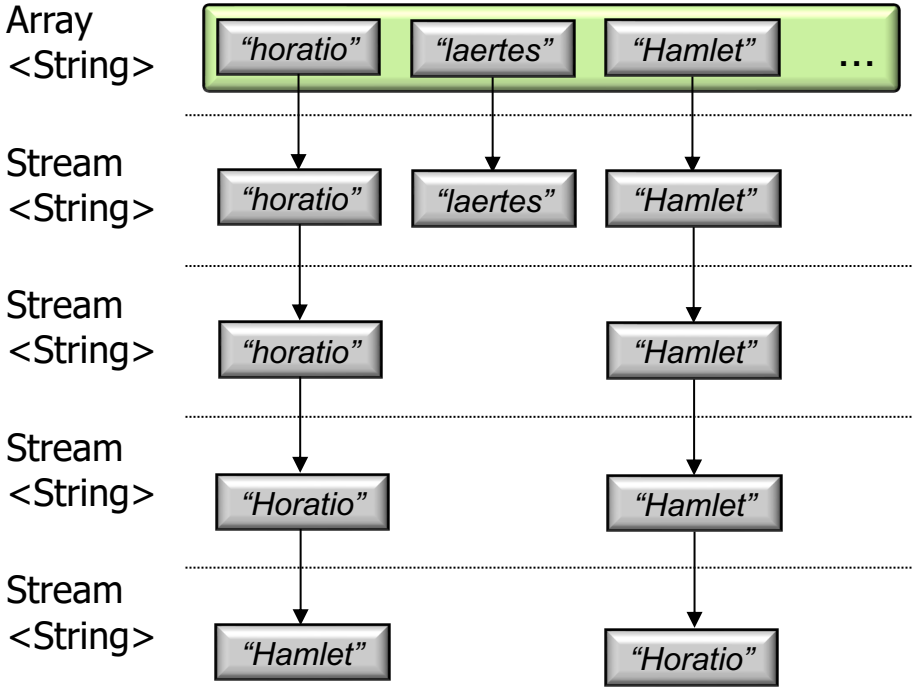
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

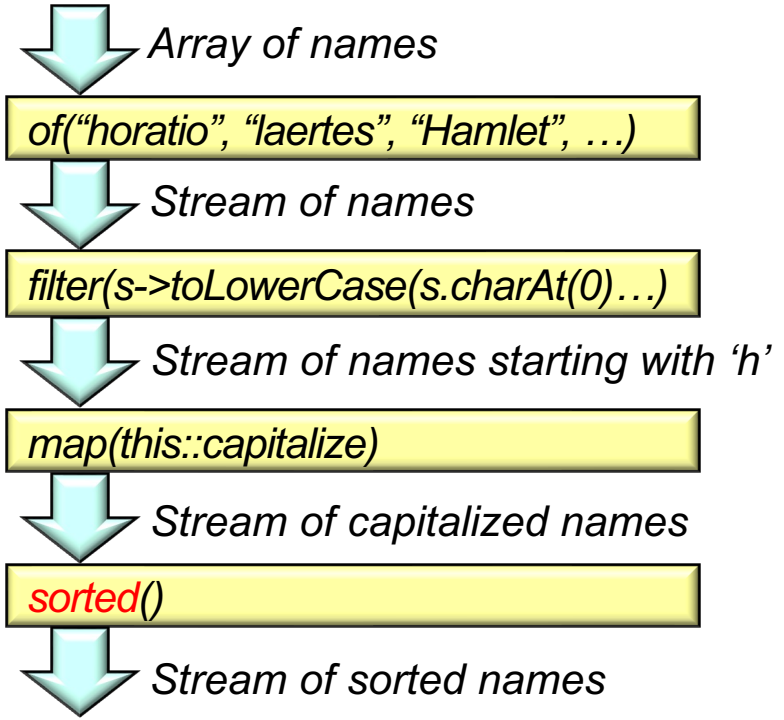




Visualizing Streams in Action

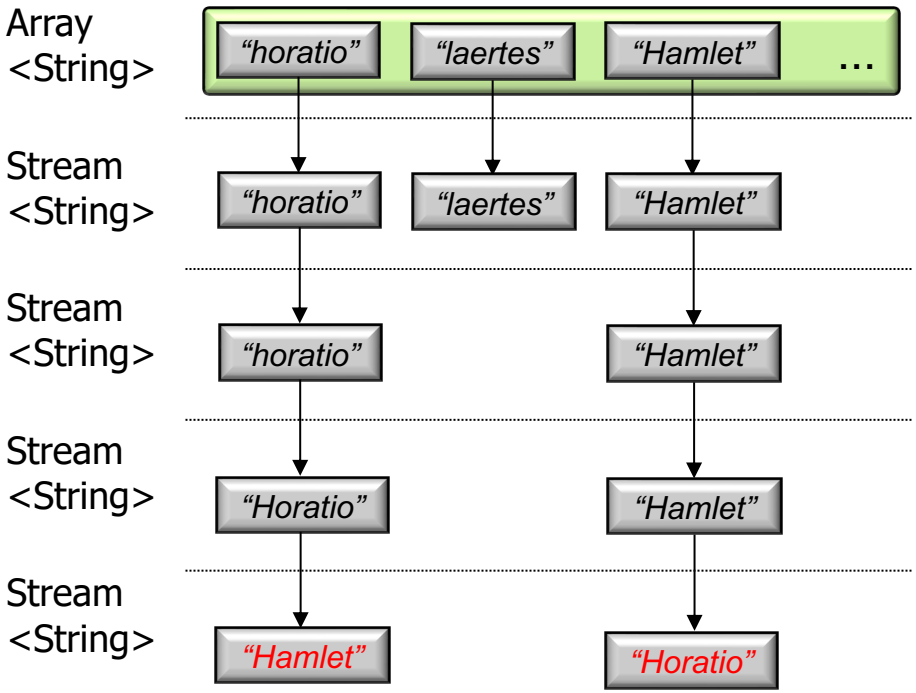
- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

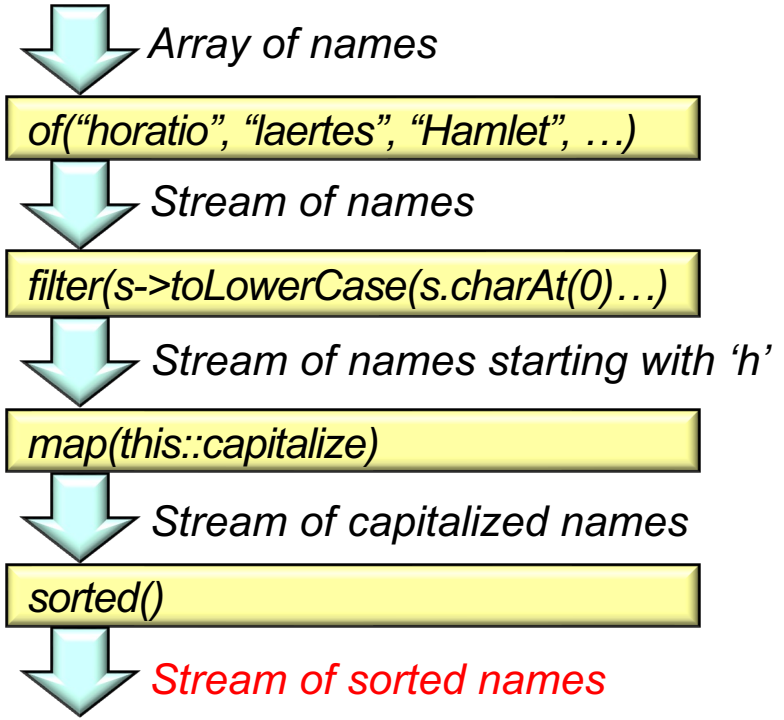




Visualizing Streams in Action

- Streams enhance flexibility by forming a "processing pipeline" that composes multiple aggregate operations together

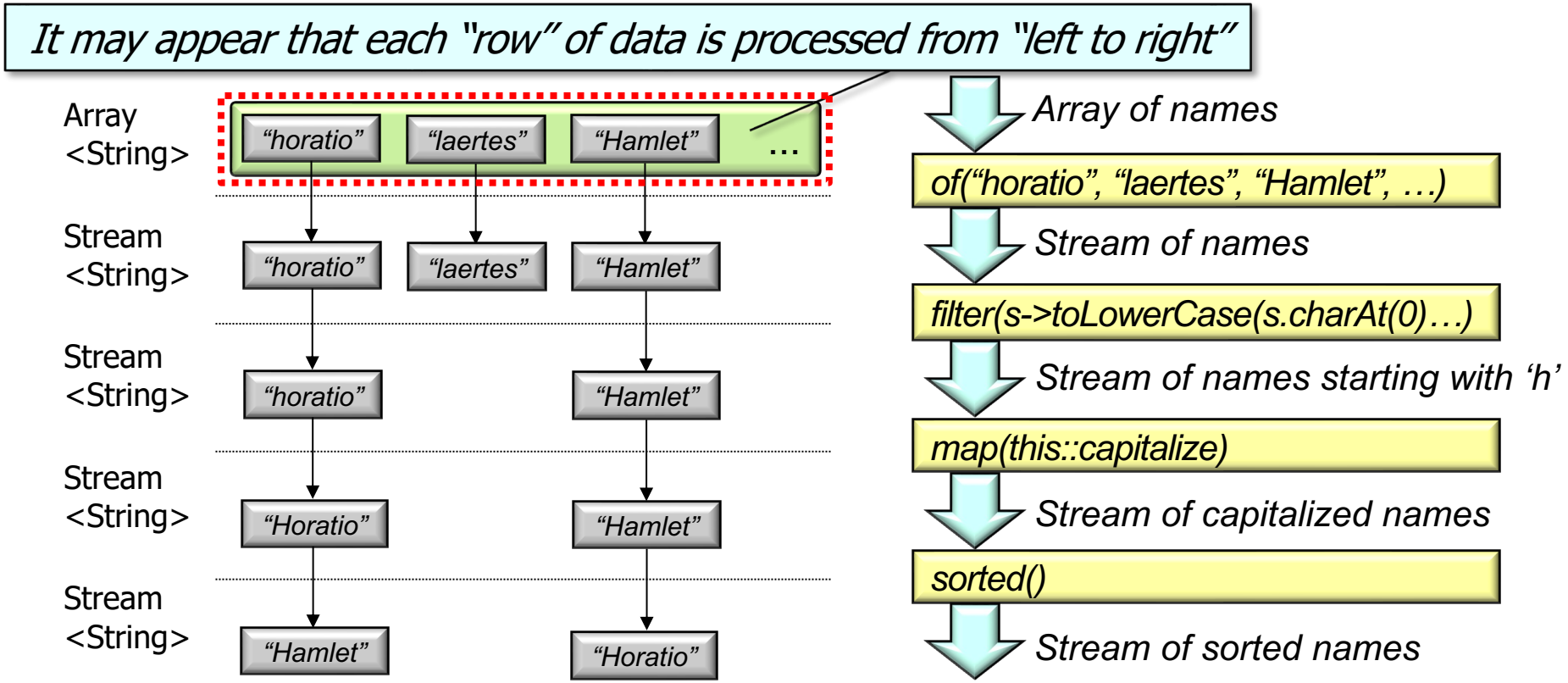




How Java Streams Processing Works in Practice

How Java Streams Processing Works in Practice

- The "physical" processing of a stream differs from the "logical" model

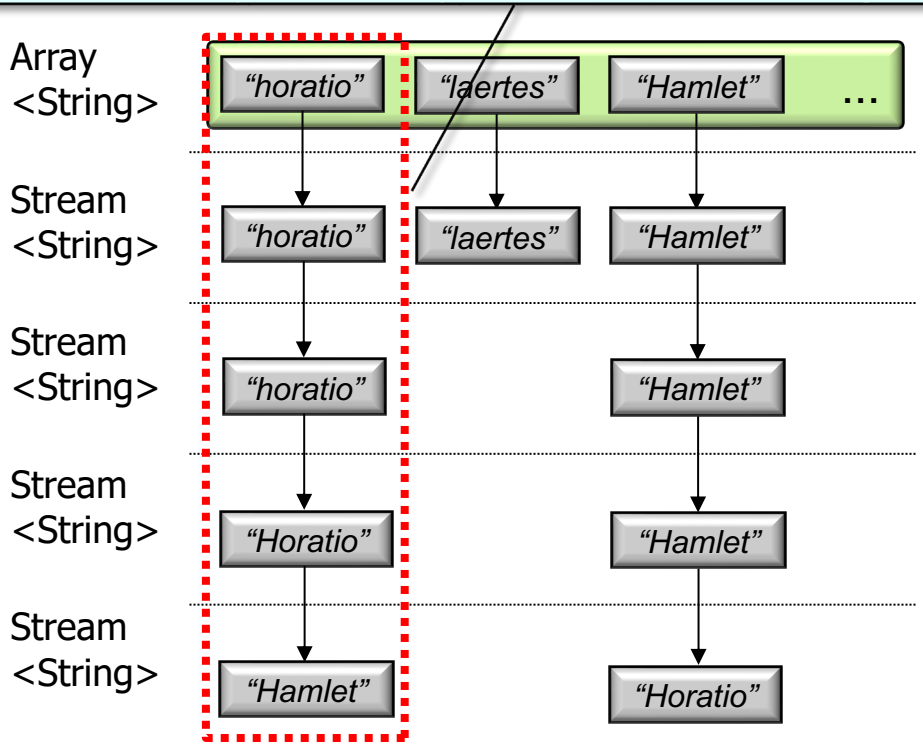


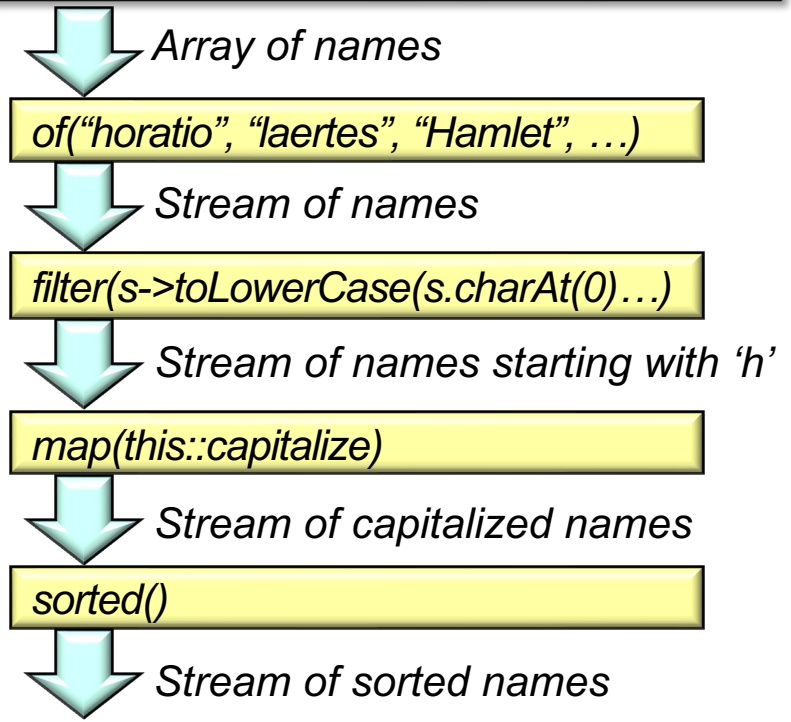
See www.ibm.com/developerworks/library/j-java-streams-3-brian-goetz

How Java Streams Processing Works in Practice

- The "physical" processing of a stream differs from the "logical" model

However, each element is actually "pulled" from the source thru each aggregate operation





This implementation is much more efficient & supports "short-circuit" operations

End of Visualizing Java Streams in Action