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
Visualizing Streams in Action
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

Visualizing Streams in Action

Input $x$

Aggregate operation (behavior $f$)

Output $f(x)$

Aggregate operation (behavior $g$)

Output $g(f(x))$

Aggregate operation (behavior $h$)

Output $h(g(f(x)))$

See [en.wikipedia.org/wiki/Pipeline_(software)](en.wikipedia.org/wiki/Pipeline_(software))
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](en.wikipedia.org/wiki/Water_filter#Point-of-use_filters)
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

Visualizing Streams in Action

Array of names of (“horatio”, “laertes”, “Hamlet”, …)

Stream of names

Stream of names starting with ‘h’

Stream of capitalized names

Stream of sorted names

See [github.com/douglascairnschmidt/LiveLessons/tree/master/Java8/ex12](github.com/douglascairnschmidt/LiveLessons/tree/master/Java8/ex12)
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

```
Stream <String>  ----------> Stream of names starting with ‘h’
    `filter(s->toLowerCase(s.charAt(0))...)
```

```
Stream <String>  ----------> Stream of capitalized names
    `map(this::capitalize)
```

```
Stream <String>  ----------> Stream of sorted names
    `sorted()
```

```
Array of names
of(“horatio”, “laertes”, “Hamlet”, …)
```
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

<table>
<thead>
<tr>
<th>Array &lt;String&gt;</th>
<th>Stream &lt;String&gt;</th>
<th>Stream &lt;String&gt;</th>
<th>Stream &lt;String&gt;</th>
<th>Stream &lt;String&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>“horatio”</td>
<td>“laertes”</td>
<td>“Hamlet”</td>
<td>“Horatio”</td>
<td>“Hamlet”</td>
</tr>
<tr>
<td>“Horatio”</td>
<td>“Hamlet”</td>
<td>“Hamlet”</td>
<td>“Hamlet”</td>
<td>“Horatio”</td>
</tr>
</tbody>
</table>

Array of names

Stream of names

Stream of names starting with ‘h’

Stream of capitalized names

Stream of sorted names

Visualizing Streams in Action
Visualizing Streams in Action

- Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

Array

Stream

Stream

Stream

Stream

Array of names

Stream of names

Stream of names starting with ‘h’

Stream of capitalized names

Stream of sorted names
**Visualizing Streams in Action**

- Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

**Diagram:**
- Array `<String>`: “horatio” “laertes” “Hamlet” ...
- Stream `<String>`: “horatio” “laertes” “Hamlet”
- Stream `<String>`: “Horatio” “Hamlet”
- Stream `<String>`: “Hamlet” “Horatio”

**Operations:**
- Array of names: `of(“horatio”, “laertes”, “Hamlet”, …)`
- Stream of names: `filter(s->toLowerCase(s.charAt(0)...)`
- Stream of names starting with ‘h’: `map(this::capitalize)`
- Stream of capitalized names: `sorted()`
- Stream of sorted names
Visualizing Streams in Action

- Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together:

- **Array** `<String>`
  - “horatio”
  - “laertes”
  - “Hamlet”
  - ...

- **Stream** `<String>`
  - “horatio”
  - “laertes”
  - “Hamlet”

- **Stream** `<String>`
  - “horatio”
  - “Hamlet”

- **Stream** `<String>`
  - “Horatio”
  - “Hamlet”

- **Stream** `<String>`
  - “Hamlet”
  - “Horatio”

1. **Array of names** of(“horatio”, “laertes”, “Hamlet”, …)
2. **Stream of names**
3. **Array of names starting with ‘h’** filter(s->toLowerCase(s.charAt(0)…))
4. **Stream of capitalized names** map(this::capitalize)
5. **Stream of sorted names** sorted()
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

Array `<String>`

Stream `<String>`

Stream `<String>`

Stream `<String>`

Array of names of(“horatio”, “laertes”, “Hamlet”, …)

Stream of names

Stream of names starting with ‘h’

Stream of capitalized names

Stream of sorted names

Visualizing Streams in Action
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.
End of Visualizing Java Streams in Action