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.

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
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”

- Stream `<String>`:
  - “Horatio”

- Stream `<String>`:
  - “Hamlet”

Visualizing Streams in Action

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

Visualizing Streams in Action

Array
<String>

Stream
<String>

Stream
<String>

Stream
<String>

Stream
<String>

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

Visualizing Streams in Action:

- Array `<String>`
  - “horatio” “laertes” “Hamlet” …

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

- Stream `<String>`
  - “horatio”

- Stream `<String>`
  - “Hamlet”

- Stream `<String>`
  - “Horatio”

- Stream `<String>`
  - “Hamlet”

- List of names of(“horatio”, “laertes”, “Hamlet”, …)

- Stream of names

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

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

- Stream of sorted names
  - `sorted()`
Streams enhance flexibility by forming a “processing pipeline” that composes multiple aggregate operations together.

Array

```
horatio  laertes  Hamlet
...
```

Stream

```
horatio  laertes  Hamlet
```

Stream

```
horatio
```

Stream

```
Horatio
```

Stream

```
Hamlet
```

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

![Diagram of Stream Operations]

- Array: 
  - "horatio" 
  - "laertes" 
  - "Hamlet" 
  - ...

- Stream: 
  - "horatio" 
  - "laertes" 
  - "Hamlet"

- Stream: 
  - "horatio"
  - "Hamlet"

- Stream: 
  - "Horatio"
  - "Hamlet"

- Stream: 
  - "Hamlet"
  - "Horatio"

- List 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:
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”

- **List 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

```
Array
<String>
horatio  laertes  Hamlet ...

Stream
<String>
horatio  laertes  Hamlet

Stream
<String>
horatio  Hamlet

Stream
<String>
Horatio  Hamlet

Stream
<String>
Hamlet  Horatio
```

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

- Array of names: "horatio", "laertes", "Hamlet", ...
- Stream of names: "horatio", "laertes", "Hamlet"
- Stream of names starting with ‘h’: "Horatio", "Hamlet"
- Stream of capitalized names: "Hamlet", "Horatio"
- Stream of sorted names: "Hamlet", "Horatio"

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