Visualize Java Sequential SearchStreamGang Hook Methods

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

- Know how to apply sequential streams to the SearchStreamGang program
- Understand the SearchStreamGang process

Stream() & processInput() hook methods

Starting SearchStreamGangTest
PARALLEL_SPLITTERATOR executed in 409 msecs
COMPLETABLE_FUTURES_INPUTS executed in 426 msecs
COMPLETABLE_FUTURES_PHASES executed in 427 msecs
PARALLEL_STREAMS executed in 437 msecs
PARALLEL_STREAM_PHASES executed in 440 msecs
RXJAVA_PHASES executed in 485 msecs
PARALLEL_STREAM_INPUTS executed in 802 msecs
RXJAVA_INPUTS executed in 866 msecs
SEQUENTIAL_LOOPS executed in 1638 msecs
**SEQUENTIAL_STREAM** executed in 1958 msecs
Ending SearchStreamGangTest

**Search Phrases**

- **stream()**
- **map(phrase -> searchForPhrase(...))**
- **filter(not(SearchResults::isEmpty))**
- **collect(toList())**
Visualizing the `processStream()` Hook Method
Visualizing the `processStream()` Hook Method

- `processStream()` searches a list of input strings

```
List<String>
```

```
Input a list of input strings
```

```
stream()
```

```
Input Strings to Search
```

---

4
Visualizing the `processStream()` Hook Method

- `processStream()` searches a list of input strings

```
List<String> ...

stream()
```

Convert collection to a (sequential) stream
Visualizing the processStream() Hook Method

- processStream() searches a list of input strings

```
Output a stream of input strings
```

List `<String>`

Stream `<String>`
Visualizing the `processStream()` Hook Method

- `processStream()` searches a list of input strings

```
List <String> ...
```

```
Input a stream of input strings
```

```
Stream <String> ...
```

```
Input Strings to Search
```

```
stream()
```

```
map(this::processInput)
```

Visualizing the `processStream()` Hook Method

- `processStream()` searches a list of input strings

Call `processInput()` to search for phrases in each input string
Visualizing the processStream() Hook Method

- processStream() searches a list of input strings

Output a stream of lists of search results

List <String> → Stream <String> → Stream <List <SearchResults>>

Input Strings to Search

stream() → map(this::processInput)
processStream() searches a list of input strings

Output a stream of lists of search results

map() also transforms its input stream type into a different output stream type
• `processStream()` searches a list of input strings

Visualizing the `processStream()` Hook Method

**Input a stream of lists of search results**

- **List `<String>`**
- **Stream `<String>`**
- **Stream `<List<SearchResults>>`**

```
Input Strings to Search
```

```
stream()
```

```
map(this::processInput)
```

```
collect(toList())
```
processStream() searches a list of input strings

```
List<String>

Stream<String>

Stream<List<SearchResults>>
```

**Visualizing the processStream() Hook Method**

Input Strings to Search

```
<table>
<thead>
<tr>
<th>Input Strings to Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

```
stream()

map(this::processInput)

collect(toList())
```

**Trigger intermediate operation processing**
Visualizing the `processStream()` Hook Method

- `processStream()` searches a list of input strings

```
List<String>... 
```

```
Stream<String>... 
```

```
Stream<List<SearchResults>>... 
```

```
List<List<SearchResults>>... 
```

Return a list of lists of search results based on “encounter order”
Visualizing the processInput() Hook Method
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

`List <String>`. `stream()`

**Input a list of phrases to find**

Search Phrases

Hamlet
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

List

`<String>`

Convert collection to a (sequential) stream
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

**Output a stream of phrases to find**

List `<String>`

Stream `<String>`

Search Phrases

17
• `processInput()` finds phrases in an input string

```java
List<String> ...

Input a stream of phrases to find
```

```java
Stream<String>
```

```java
map(phrase -> searchForPhrase(...))
```

**Visualizing the `processInput()` Hook Method**
• processInput() finds phrases in an input string

Search Phrases

Stream
<String>

map(phrase -> searchForPhrase(...))

List
<String>

Search for the phrase in each input string
Visualizing the processInput() Hook Method

- processInput() finds phrases in an input string

Output a stream of search results

```
List <String>
```

```
Stream <String>
```

```
Stream <SearchResults>
```

```
Search Phrases
```

```
stream()
```

```
map(phrase -> searchForPhrase(…))
```

20
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

**Input a stream of search results**

List
\[\text{<String>}\]

Stream
\[\text{<String>}\]

Stream
\[\text{<SearchResults>}\]

Search Phrases

- `stream()`
- `map(phrase -> searchForPhrase(…))`
- `filter(not(SearchResults::isEmpty))`
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

Remove empty search results from the stream
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

Output a stream of non-empty search results

List <String>

Stream <String>

Stream <SearchResults>

Stream <SearchResults>

List <String>

Stream <String>

Stream <SearchResults>

Stream <SearchResults>
• processInput() finds phrases in an input string

**Input a stream of non-empty search results**

List
(List<String>)

Stream
(Stream<String>)

Stream
(Stream<SearchResults>)

Stream
(Stream<SearchResults>)

Visualizing the processInput() Hook Method

Search Phrases

```
stream()
map(phrase -> searchForPhrase(…))
filter(not(SearchResults::isNotEmpty))
collect(toList())
```
processInput() finds phrases in an input string.

List <String>

Stream <String>

Stream <SearchResults>

Stream <SearchResults>

Search Phrases

stream()

map(phrase -> searchForPhrase(…))

filter(not(SearchResults::isEmpty))

collect(toList())

Trigger intermediate operation processing
Visualizing the `processInput()` Hook Method

- `processInput()` finds phrases in an input string

```
List <String>
```

```
Stream <String>
```

```
Stream <SearchResults>
```

```
Stream <SearchResults>
```

```
List <SearchResults>
```

**Search Phrases**

- Stream `<String>`
  - `stream()`
  - `map(phrase -> searchForPhrase(...))`
  - `filter(not(SearchResults::isEmpty))`
  - `collect(toList())`

**Return a (possibly empty) list of search results in encounter order**
Visualizing the `processInput()` Hook Method

- `processStream()` searches a list of input strings

**Output a stream of lists of search results**

List

\[
\text{List} \quad \langle \text{String} \rangle
\]

Stream

\[
\text{Stream} \quad \langle \text{String} \rangle
\]

Stream\langle List <\text{SearchResults}> >

- Some lists of search results may be empty if no phrases match an input string
Visualizing the `processInput()` Hook Method

- `processStream()` searches a list of input strings

```
List<String>
Stream<String>
Stream<List<SearchResults>>
List<List<SearchResults>>
```

Input Strings to Search

```
stream()
map(this::processInput)
collect(toList())
```

The `printPhrases()` method handles empty "list of search results" later

See "Java Sequential SearchStreamGang Example: Implementing `printPhrases()`"
We focus on sequential streams with one thread

```
List <String>
Stream <String>
Stream <SearchResults>
Stream <SearchResults>
List <SearchResults>
```

Visualizing the `processInput()` Hook Method

```
map(phrase -> searchForPhrase(...))
filter(not(SearchResults::isEmpty))
collect(toList())
```
Visualizing the processInput() Hook Method

- We focus on sequential streams with one thread
- We’ll cover parallel streams later

```
Visualizing the processInput() Hook Method
```

```
List <String>
```

```
Stream <String>
```

```
Stream <SearchResults>
```

```
List <SearchResults>
```

```
See upcoming lessons on "Overview of Java Parallel Streams"
```

```
Stream <SearchResults>
```

```
Stream <SearchResults>
```

```
List <SearchResults>
```

```
Search Phrases
```

```
parallelStream()
```

```
map(phrase -> searchForPhrase(…))
```

```
filter(not(SearchResults::isEmpty))
```

```
collect(toList())
```

```
45,000+ phrases
```

```
Search Phrases
```

```
Visualizing the processInput() Hook Method
```

```
List <String>
```

```
Stream <String>
```

```
Stream <SearchResults>
```

```
List <SearchResults>
```

```
See upcoming lessons on "Overview of Java Parallel Streams"
```
Minuscule changes are needed to transition from sequential to parallel streams!

**Visualizing the processInput() Hook Method**

- We focus on sequential streams with one thread
- We’ll cover parallel streams later
End of Visualize Java
Sequential SearchStreamGang
Hook Methods