Java SearchWithParallelStreams
Example: Visualizing 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 Java parallel streams are applied in SearchWithParallelStreams

```
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
```

See [github.com/douglasclarkschmidt/LiveLessons/tree/master/SearchStreamGang](https://github.com/douglasclarkschmidt/LiveLessons/tree/master/SearchStreamGang)
Visualizing the processStream() Method
Visualizing the `processStream()` Method

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

List `<String>`

Each input string contains a work of Shakespeare (e.g., Hamlet, MacBeth, etc.)
Visualizing the `processStream()` Method

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

Convert collection to a parallel stream, i.e., substreams with chunks of input strings
Output a stream of input strings

Chunks of input strings are processed in parallel on separate threads/cores
Visualizing the `processStream()` Method

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

List `<String>`

Stream `<String>`

**Input a stream of input strings**

- `parallelStream()`
  - `map(this::processInput)`

A pool of worker threads

Input Strings to Search
• processStream() searches a list of input strings in parallel

Call processInput() to search for phrases in a given input string in parallel
ProcessStream() searches a list of input strings in parallel

Output a stream of lists of search results

Input Strings to Search

parallelStream()

map(this::processInput)

List<String>

Stream<String>

Stream<List<SearchResults>>

A pool of worker threads

Some lists of search results may be empty if no phrases matched an input string
Visualizing the `processStream()` Method

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

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

```
List<String>...
parallelStream()
map(this::processInput)
Stream<List<SearchResults>>...
collect(toList())
```

**Input Strings to Search**

![A pool of worker threads]
processStream() searches a list of input strings in parallel.

Trigger intermediate operation processing to run on multiple worker threads & cores.
Visualizing the `processStream()` Method

- `processStream()` searches a list of input strings in parallel.

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

```
Stream<String>
```

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

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

Returns a list of lists of search results based on “encounter order”.

Input Strings to Search

```
parallelStream()
```

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

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

A pool of worker threads
Visualizing the processInput() Method
• `processInput()` finds phrases in an input string in parallel

`List <String>`

*Input a list of phrases to find*

`parallelStream()`
**Visualizing the `processInput()` Method**

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

Convert collection to a parallel stream, i.e., substreams with chunks of phrases.
Visualizing the processInput() Method

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

Output a stream of phrases to find

List <String> →

Stream <String> →

A pool of worker threads

parallelStream()

Search Phrases

Different chunks of phrases are processed in parallel on multiple worker threads & cores
Visualizing the `processInput()` Method

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

```
List <String>

Stream <String>
```

Input a stream of phrases to find

```
parallelStream()

map(phrase -> searchForPhrase(…))
```

A pool of worker threads
Visualizing the `processInput()` Method

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

List `<String>`

Stream `<String>`

Perform parallel search for phrases in a given input string
Visualizing the `processInput()` Method

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

Output a stream of search results

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

Search Phrases

`parallelStream()`

`map(phrase -> searchForPhrase(...))`
Visualizing the `processInput()` Method

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

**Input a stream of search results**

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

---

`parallelStream()`

`map(phrase -> searchForPhrase(…))`

`filter(not(SearchResults::isEmpty))`
Visualizing the `processInput()` Method

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

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

Remove empty search results from substreams in parallel
processInput() finds phrases in an input string in parallel. Output a stream of non-empty search results.

List <String>

Stream <String>

Stream <SearchResults>

Visualizing the processInput() Method

Search Phrases

parallelStream()

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

filter(not(SearchResults::isEmpty))

The stream of search results may be empty if no phrases matched an input string.
processInput() finds phrases in an input string in parallel

Input a stream of non-empty search results

List
\(<\text{String}\)\

Stream
\(<\text{String}\)\

Stream
\(<\text{SearchResults}\)\

Stream
\(<\text{SearchResults}\)\

A pool of worker threads

parallelStream()\

map(phrase -> searchForPhrase(…))\

filter(not(SearchResults::isEmpty))\

collect(toList())

Search Phrases

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

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

Trigger intermediate operation processing to run on multiple threads/cores

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

parallelStream()

map(phrase -> searchForPhrase(…))

filter(not(SearchResults::isEmpty))

collect(toList())
```
Visualizing the `processInput()` Method

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

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

Stream `<SearchResults>`

List `<SearchResults>`

Search Phrases

```
parallelStream()
map(phrase -> searchForPhrase(...))
filter(not(SearchResults::isEmpty))
collect(toList())
```

Return a list of search results in the originating thread based on “encounter order”
Visualizing the `processInput()` Method

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

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

This list will be empty if none of the phrases matched the input string
Visualizing the `processInput()` Method

- Note that the actual processing of parallel streams differs from this visualization.

List `<String>`

Stream `<String>`

Stream `<SearchResults>`

Stream `<SearchResults>`

List `<SearchResults>`

End of Java Search
With ParallelStreams Example:
Visualizing Hook Methods