Visualizing the WordSearcher

.findWords() Method

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

• Recognize the structure & functionality of the SimpleSearchStream example

• Visualize aggregate operations in SimpleSearch Stream’s WordSearcher.findWords() method

Let's start at the very beginning..

wordsToFind
"do","re","mi","fa","so","la","ti","do"

stream()

map(this::searchForWord)

filter(not(SearchResults::isEmpty))

collect(toList())

See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream
Visualizing the Word Searcher.findWords() Method
WordSearcher.findWords() searches for words in an input string

Input a list of words to find

wordsToFind

"do", "re", "mi", "fa", "so", "la", "ti", "do"

List
<String>

See SimpleSearchStream/src/main/java/search/WordSearcher.java
Visualizing the WordSearcher.findWords() Method

- WordSearcher.findWords() searches for words in an input string

List `<String>`

```java
wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"
```

Convert collection to a (sequential) stream
Visualizing the WordSearcher.findWords() Method

- WordSearcher.findWords() searches for words in an input string

Output a stream of words to find

wordsToFind

"do", "re", "mi", "fa", "so", "la", "ti", "do"

List
<String>

Stream
<String>

stream()
**Visualizing the WordSearcher.findWords() Method**

- `WordSearcher.findWords()` searches for words in an input string

<table>
<thead>
<tr>
<th>List</th>
<th>Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;String&gt;</code></td>
<td><code>&lt;String&gt;</code></td>
</tr>
</tbody>
</table>

Input a stream of words to find

wordsToFind

"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream() -> map(this::searchForWord)
Visualizing the `WordSearcher.findWords()` Method

- `WordSearcher.findWords()` searches for words in an input string

List
`<String>`

Stream
`<String>`

```
8
```

Search for the word in the input string
WordSearcher.findWords() searches for words in an input string

Output a stream of search results

List
CTest<String>

Stream
CTest<String>

Stream
CTest<SearchResults>

wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream()

map(this::searchForWord)

SearchResults stores # of times a word appeared in the input string
Visualizing the WordSearcher.findWords() Method

- WordSearcher.findWords() searches for words in an input string

Output a stream of search results

wordsToFind

"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream()

map(this::searchForWord)

List
<String>

Stream
<String>

Stream
<SearchResults>

Note the transformation of types at this point in the stream!
WordSearcher.findWords() searches for words in an input string.

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

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

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

\[
\text{Input a stream of search results}
\]

\[
\text{wordsToFind: } "do", "re", "mi", "fa", "so", "la", "ti", "do"
\]

\[
\text{stream()}
\]

\[
\text{map(this::searchForWord)}
\]

\[
\text{filter(not(SearchResults::isEmpty))}
\]
Visualizing the WordSearcher.findWords() Method

- WordSearcher.findWords() searches for words in an input string

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

```
map(this::searchForWord)
filter(not(SearchResults::isEmpty))
stream()
```

List of words to find:
"do", "re", "mi", "fa", "so", "la", "ti", "do"

Remove empty search results from the stream
WordSearcher.findWords() searches for words in an input string

Output a stream of non-empty search results

List
<String>

Stream
<String>

Stream
<SearchResults>

Stream
<SearchResults>

wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream() → map(this::searchForWord) → filter(not(SearchResults::isEmpty))
WordSearcher.findWords() searches for words in an input string

Input a stream of non-empty search results

wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream()
map(this::searchForWord)
filter(not(SearchResults::isEmpty))
collect(toList())
WordSearcher.findWords() searches for words in an input string

```
Stream<SearchResults> stream()
map(this::searchForWord)
filter(not(SearchResults::isEmpty))
collect(toList())

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

**Trigger intermediate operation processing**

wordsToFind

"do", "re", "mi", "fa", "so", "la", "ti", "do"

45,000+ phrases

"do", "re", "mi", "fa", "so", "la", "ti", "do"
Visualizing the `WordSearcher.findWords()` Method

- `WordSearcher.findWords()` searches for words in an input string

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

```
wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"
```

```
return a list of non-empty search results
```

Diagram:
- Stream of input strings
- Map to `searchForWord`
- Filter non-empty search results
- Collect to list
- Return list of non-empty search results
Visualizing the WordSearcher.findWords() Method

- The “physical” processing of a stream differs from the “logical” model.

It may appear that each “row” of data is processed from “left to right.”

List
<String>

Stream
<String>

Stream
<SearchResults>

Stream
<SearchResults>

List
<SearchResults>

wordsToFind
"do", "re", "mi", "fa", "so", "la", "ti", "do"

stream()

map(this::searchForWord)

filter(not(SearchResults::isEmpty))

collect(toList())

See www.ibm.com/developerworks/library/j-java-streams-3-brian-goetz
The “physical” processing of a stream differs from the “logical” model.

However, each element is actually “pulled” from the source through each aggregate operation.

List
<String>

Stream
<String>

Stream
<SearchResults>

Stream
<SearchResults>

List
<SearchResults>

This implementation is much more efficient & supports “short-circuit” operations.
End of Visualizing the Word Searcher.findWords() Method