Visualize 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 `SimpleSearchStream`’s `WordSearcher.findWords()` method

```java
inputString

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

stream()

map(this::searchForWord)

filter(not(SearchResults::isEmpty))

collect(toList())

Let's start at the very beginning..

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://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

See SimpleSearchStream/src/main/java/search/WordSearcher.java
WordSearcher.findWords() searches for words in an input string

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

Convert collection to a (sequential) stream

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

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

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

**Output a stream of words to find**

```
List<String>
```

```
Stream<String>
```

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

```
wordsToFind
```

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

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

`Map(this::searchForWord)`

`List<String>`

`Stream<String>`

*Input a stream of words to find*

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

`stream()`

`map(this::searchForWord)`
WordSearcher.findWords() searches for words in an input string.

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

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

```
Visualizing the WordSearcher.findWords() Method
```

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

```
map(this::searchForWord)
```

```
stream()
```

```
Search for the word in the input string
```
Visualizing the WordSearcher.findWords() Method

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

```
Output a stream of search results

List <String>

Stream <String>

Stream <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

```
Input a stream of search results
```

List

```
< String >
```

Stream

```
< String >
```

Stream

```
< SearchResults >
```

```
stream()
```

```
map(this::searchForWord)
```

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

Note the transformation of types at this point in the stream!
Input a stream of search results

List
<String>

Stream
<String>

Stream
<SearchResults>

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

Visualizing the WordSearcher.findWords() Method

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

```
Map(this::searchForWord)
filter(not(SearchResults::isEmpty))
```
Visualizing the `WordSearcher.findWords()` Method

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

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

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

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

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

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))

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

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

wordsToFind

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

Trigger intermediate operation processing
WordSearcher.findWords() searches for words in an input string. The method takes a list of words to find, filters out any empty search results, and returns a list of non-empty search results.

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

Words to find include: "do", "re", "mi", "fa", "so", "la", "ti", "do"
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())
```

The “physical” processing of a stream differs from the “logical” model. However, each element is actually “pulled” from the source through each aggregate operation. This diagram visualizes the WordSearcher.findWords() method:

- **map** (this::searchForWord)
- **filter** (not(SearchResults::isEmpty))
- **collect** (toList())

The implementation is much more efficient and supports "short-circuit" operations.
End of Visualize the Word Searcher.findWords() Method