Java Sequential SearchStreamGang

Example: Visualizing printPhrases()

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 printPhrases() method

```java
void printPhrases(List<List<SearchResults>> listOfListOfSearchResults) {
    Map<String, List<SearchResults>> resultsMap =
        listOfListOfSearchResults.stream()
            .flatMap(List::stream)
            .collect(groupingBy(SearchResults::getTitle,
                                TreeMap::new, toList()));

    resultsMap.forEach((key, value) -> {
        System.out.println("Title \"" + key + "\" contained");
        value.forEach(SearchResults::print);
    }); ...
}
```

Visualizing printPhrases()
Visualizing printPhrases()

- `SearchStreamGang.printPhrases()` displays phrases associated with each play...

**Title "The Tragedy of Hamlet, Prince of Denmark" contained**
- "It shall be so. Madness in great ones must not unwatch'd go." at [89594]
- "Give every man thine ear, but few thy voice" at [26207]
- "There is nothing either good or bad but thinking makes it so" at [62609]
- "To be, or not to be- that is the question" at [83061]
- "Neither a borrower nor a lender be" at [26556]
- "The lady doth protest too much, methinks" at [102267]
- "Get thee to a nunnery" at [86071|86953]
- "Brevity is the soul of wit" at [54747]

**Title "The Tragedy of Julius Caesar" contained**
- "The fault, dear Brutus, is not in our stars, but in ourselves that we are underlings" at [12243]
- "Friends, Romans, countrymen, lend me your ears" at [76850]
- "But for mine own part, it was Greek to me" at [19356]
- "Beware the Ides of March" at [6412|6495|6703]
- "The evil that men do lives after them, The good is oft interred with their bones" at [76950]

Plays are printed out in sorted ascending order by their title

This method shows the flatMap() & collect(groupingBy()) aggregate operations
printPhrases() uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

- **Visualizing printPhrases()**
  - **Input a list of lists of search results**
  - `List<List<SearchResults>>`
  - `stream()`

Some lists of search results may be empty if no phrases matched an input string.
Visualizing `printPhrases()`

- `printPhrases()` uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

Convert list to a (sequential) stream of lists of search results
printPhrases() uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

Output a stream of lists of search results

List<List<SearchResults>>

Stream<List<SearchResults>>

stream()
printPhrases() uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

Input a stream of lists of search results

List<List<SearchResults>>

Stream<List<SearchResults>>

stream()

flatMap(List::stream)
Visualizing `printPhrases()`

- `printPhrases()` uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

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

- Flatten the stream of lists of search results to a stream of search results:

```
stream()
flatMap(List::stream)
```

Flatten the stream of lists of search results to a stream of search results.
Visualizing `printPhrases()`

- `printPhrases()` uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

Empty lists will occur if no phrases match in an input string (Shakespeare work).
Visualizing `printPhrases()`

- `printPhrases()` uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

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

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

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

```
flatMap(List::stream)
```

```
stream()
```

11
printPhrases() uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

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

- **List**<List<SearchResults>>
- **Stream**<List<SearchResults>>
- **Stream**<SearchResults>
printPhrases() uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

- Visualizing printPhrases()

- List<List<SearchResults>>
- Stream<List<SearchResults>>
- Stream<SearchResults>

- stream()
- flatMap(List::stream)
- collect(groupingBy(...))

Trigger intermediate operation processing
Visualizing `printPhrases()`

- `printPhrases()` uses a stream that converts a list of lists of search results into a map that associates phrases with the works where they were found.

Create a map that groups phrases according to the works where they appear.
End of Java Sequential SearchStreamGang Example: Visualizing printPhrases()