Java Sequential SearchStreamGang Example: Implementing 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

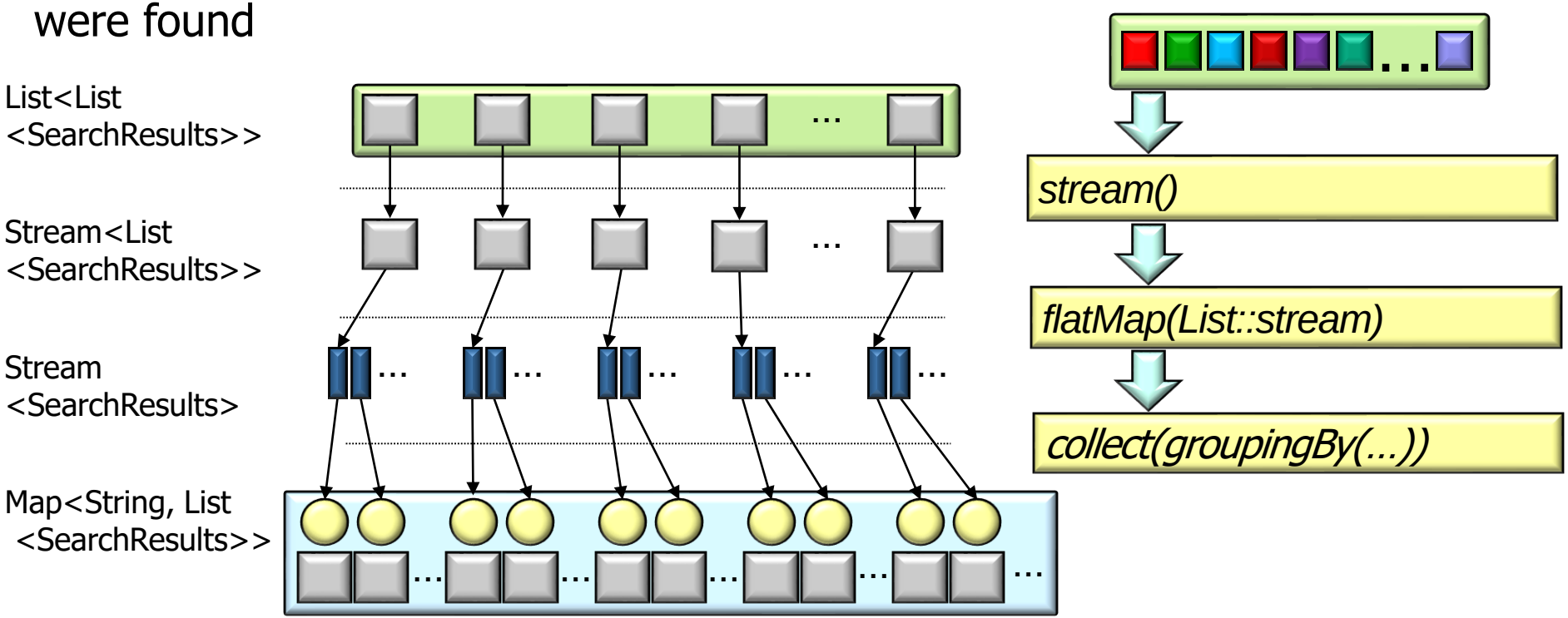
```
void printPhrases(List<List<SearchResults>>
                    listOfListOfSearchResults) {
  Map<String, List<SearchResults>> resultsMap =
    listOfListOfSearchResults
      .stream()
      .flatMap(List::stream)
      .collect(groupingBy(SearchResults::getTitle,
                          TreeMap::new,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

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

See github.com/douglascraigschmidt/LiveLessons/tree/master/SearchStreamGang

Implementing printPhrases() as a Sequential Stream

Implementing printPhrases() as a Sequential Stream

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



Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

See SearchStreamGang/src/main/java/livelessons/streamgangs/SearchStreamGang.java

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

The list of lists of search results param was obtained from the processStream() method

See "*Java Sequential SearchStreamGang Example: Implementing Hook Methods*"

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Converts the list of lists of search results into a stream of lists of search results

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Output stream flattens the stream of lists of search results into a stream of search results

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#flatMap

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Any/all empty lists are ignored!

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#flatMap

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

# of output stream elements may differ from the # of input stream elements

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

Converts stream of lists of search results into a stream of search results

flatMap() can transform its input stream type into a different output stream type

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Triggers all of the intermediate operations

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#collect

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Groups elements via a classification function & return results in a map

See docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html#groupingBy

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

Classification function creates keys in the map from play titles

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

TreeMap stores its contents in sorted order based on key

See docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

This "downstream collector" defines a list collector that is applied to the results of the classifier function

See www.baeldung.com/java-groupingby-collector

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

Associates phrases found in input with titles where they were found

See docs.oracle.com/javase/8/docs/api/java/util/Map.html

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
    .stream()

    .flatMap(List::stream)

    .collect(groupingBy(SearchResults::getTitle,
                        TreeMap::new, toList()));

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

Iterates thru all key/value pairs in map

See docs.oracle.com/javase/8/docs/api/java/util/Map.html#forEach

Implementing printPhrases() as a Sequential Stream

- printPhrases() uses a stream to display phrases associated with each play

```
void printPhrases(List<List<SearchResults>> listOfListOfResults){
  Map<String, List<SearchResults>> map = listOfListOfResults
     .stream()

     .flatMap(List::stream)

     .collect(groupingBy(SearchResults::getTitle,
                         TreeMap::new, toList()));

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

Displays titles (keys) & phrases (values)

See SimpleSearchStream/src/main/java/search/SearchResults.java

End of Java Sequential SearchStreamGang Example: Implementing printPhrases()