Java Sequential SearchStreamGang Example: Object-Oriented Implementation

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 Lesson

Know how to apply object-oriented features to the SearchStreamGang program

```
protected List<List<SearchResults>> processStream() {
   List<List<SearchResults>> listOfListOfResults =
      new ArrayList<>();
                                                             Starting SearchStreamGangTest
   for (CharSequence inputSeq
                                                             PARALLEL SPLITERATOR executed in 409 msecs
                                                             COMPLETABLE FUTURES INPUTS executed in 426 msecs
           : getInput())
                                                             COMPLETABLE_FUTURES_PHASES executed in 427 msecs
                                                             PARALLEL STREAMS executed in 437 msecs
      listOfListOfResults
                                                             PARALLEL STREAM PHASES executed in 440 msecs
                                                             RXJAVA_PHASES executed in 485 msecs
          .add(processInput(inputSeq);
                                                             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
   return listOfListOfResults;
                                                             Ending SearchStreamGangTest
```

Learning Objectives in this Lesson

Know how to apply object-oriented features to the SearchStreamGang program

```
protected List<List<SearchResults>> processStream() {
   List<List<SearchResults>> listOfListOfResults =
      new ArrayList<>();
                                                              Starting SearchStreamGangTest
   for (CharSequence inputSeq
                                                             PARALLEL_SPLITERATOR executed in 409 msecs
                                                              COMPLETABLE FUTURES INPUTS executed in 426 msecs
           : getInput())
                                                              COMPLETABLE_FUTURES_PHASES executed in 427 msecs
                                                              PARALLEL STREAMS executed in 437 msecs
      listOfListOfResults
                                                              PARALLEL STREAM PHASES executed in 440 msecs
                                                              RXJAVA_PHASES executed in 485 msecs
          .add(processInput(inputSeq);
                                                              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
   return listOfListOfResults;
                                                             Ending SearchStreamGangTest
```

Also understand the limitations with object-oriented programming...

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>>_processStream() {
      This method is not actually implemented with a stream...
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return listOfListOfResults;
```

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
                   Results are stored in a list of lists of search results
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return listOfListOfResults;
```

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream()
                      Stores # of times a phrase appeared in an input string
   List<List<SearchResults>> listOfListOfResults
                                                                             <<Java Class>>
      new ArrayList<>();
                                                                            SearchResults
                                                                      mThreadId: long
                                                                      mWord: String
   for (CharSequence inputSeq : getInput())

    mTitle: String

      listOfListOfResults
                                                                      mCycle: long
                                                                      SearchResults()
         .add(processInput(inputSeq));
                                                                      SearchResults(long,long,String,String)
                                                                      getTitle():String
                                                                      headerToString():String
                                             <<Java Class>>
   return listOfListOfResults;
                                                        #mList
                                                                      add(int):void
                                               Result
                                                                      isEmpty():boolean
                                             mIndex: int
                                                                      size():int
                                                                      toString():String
                                             print():SearchResults
```

See <u>livelessons/utils/SearchResults.java</u>

processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
      Must remember to initialize this list or chaos & insanity will result...
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
       .add(processInput(inputSeq));
  return listOfListOfResults;
```

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
                                          Explicitly loop thru all the
  return listOfListOfResults;
                                            works of Shakespeare
```

processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
                                                        <<Java Class>>
                                                       G StreamGang<E>
  for (CharSequence inputSeq : getInput())
                                                      getInput():List<E>
    listOfListOfResults
                                                     setInput(List<E>):List<E>
                                                     .add(processInput(inputSeq));

⋄ initiateStream():void

                                                     return listOfListOfResults;
                                      <<Java Class>>
                                   SearchStreamGang
```

The getInput() method is defined in the StreamGang framework

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
                                           CharSequence optimizes
                                        subSequence() to avoid memory
  return listOfListOfResults;
                                         copies (cf. String substring())
```

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return listOfListOfResults;
                   Applying processInput() to each work
```

processInput() returns a list of SearchResults—one list for each input string

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return\listOfListOfResults;
    Add the list of search results to the list of lists
```

• processStream() sequentially searches for phrases in lists of input "strings"

protected List<List<SearchResults>> processStream() {

```
List<List<SearchResults>> listOfListOfResults =
  new ArrayList<>();
for (CharSequence inputSeq : getInput())
  listOfListOfResults
    .add(processInput(inputSeq));
return listOfListOfResults;
```

If listOfListOfResults isn't initialized properly chaos & insanity will result.

processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return listOfListOfResults:
                                        Return the search results
```

• processStream() sequentially searches for phrases in lists of input "strings"

```
protected List<List<SearchResults>> processStream() {
      Indicates how many times a search phrase appeared in each input string
  List<List<SearchResults>> listOfListOfResults =
    new ArrayList<>();
  for (CharSequence inputSeq : getInput())
    listOfListOfResults
      .add(processInput(inputSeq));
  return listOfListOfResults;
```

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeg);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
                                               The input is a section of
  for (String phrase : mPhrasesToFind) {
                                               a text file managed by
                                               the test driver program
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq) ;
                                       The input is split into two parts
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

processInput() searches an input string for all occurrences of phrases to find

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
                                       subSequence() avoids overhead
  CharSequence input = inputSeq. -
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
```

```
if (res.size() > 0) listOfResults.add(res);
return listOfResults;
  See SearchStreamGang/src/main/java/livelessons/utils/SharedString.java
```

for (String phrase : mPhrasesToFind) {

searchForPhrase(phrase, input, title);

SearchResults res =

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
                                               Create empty list
                                                to store results
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

processInput() searches an input string for all occurrences of phrases to find

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults > listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```



Again, you must remember to initialize this list or chaos & insanity will result...

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
                              Explicitly loop thru all phrases to search input
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

processInput() searches an input string for all occurrences of phrases to find

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
                              Apply this function to all phrases in the input
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
```

return listOfResults;

processInput() searches an input string for all occurrences of phrases to find

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
```

SearchResults res = searchForPhrase(phrase, input, title); if (res.size() > 0) listOfResults.add(res);

return listOfResults;

Explicitly check if results are non-empty

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
                                         Add non-empty SearchResults
  return listOfResults;
                                           to the list of search results
```

processInput() searches an input string for all occurrences of phrases to find

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
  return listOfResults;
```

There are control constructs in this code, which makes it harder to read!

```
List<SearchResults> processInput(CharSequence inputSeq) {
  String title = getTitle(inputSeq);
  CharSequence input = inputSeq.
    subSequence(title.length(), inputSeq.length());
  List<SearchResults> listOfResults = new ArrayList<>();
  for (String phrase : mPhrasesToFind) {
    SearchResults res =
      searchForPhrase(phrase, input, title);
    if (res.size() > 0) listOfResults.add(res);
                                   Return list result to processStream()
  return listOfResults; -
```

searchForPhrase() uses a regex to find phrases in an input string

List<SearchResults.Result> results = new ArrayList<>(); String regex = phr.trim()

SearchResults searchForPhrase(String phr, String in, String ti){

Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL);

while (match.find()) results.add(new Result(match.start())); return new SearchResults (Thread.currentThread().getId(),

Matcher match = pat.matcher(in);

.replaceAll("\\s+", "\\\s+")

.replace("?", "\\?");

0, phr, ti, results); ...

searchForPhrase() uses a regex to find phrases in an input string

Pattern pat = compile(regex, CASE_INSENSITIVE | DOTALL);
Matcher match = pat.matcher(in);

while (match.find()) results.add(new Result(match.start()));
return new SearchResults(Thread.currentThread().getId(),

0, phr, ti, results); ...

searchForPhrase() uses a regex to find phrases in an input string

Matcher match = pat.matcher(in);

SearchResults searchForPhrase(String phr, String in, String ti){
 List<SearchResults.Result> results = new ArrayList<>();
 String regex = phr.trim()
 .replaceAll("\\s+", "\\\\s+")
Create regex to search for phrase
by collapsing extrapeous whitespace

.replaceAll("\\s+", "\\\s+")
.replace("?", "\\?");

Pattern pat = compile(regex, CASE_INSENSITIVE | DOTALL);

while (match.find()) results.add(new Result(match.start()));

return new SearchResults(Thread.currentThread().getId(),
0, phr, ti, results); ...

searchForPhrase() uses a regex to find phrases in an input string

```
SearchResults searchForPhrase(String phr, String in, String ti){
  List<SearchResults.Result> results = new ArrayList<>();
```

```
String regex = phr.trim()
  .replaceAll("\\s+", "\\\s+")
  .replace("?", "\\?");
```

Compile the regex

```
Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL);
Matcher match = pat.matcher(in);
```

```
while (match.find()) results.add(new Result(match.start()));
return new SearchResults (Thread.currentThread().getId(),
```

0, phr, ti, results); ... See docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

searchForPhrase() uses a regex to find phrases in an input string

SearchResults searchForPhrase(String phr, String in, String ti){

List<SearchResults.Result> results = new ArrayList<>();

String regex = phr.trim() .replaceAll("\\s+", "\\\s+")

.replace("?", "\\?");

Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL); Matcher match = pat.matcher(in);

while (match.find()) results.add(new Result(match.start()));

return new SearchResults (Thread.currentThread().getId(), 0, phr, ti, results); ...

Create a matcher

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

 searchForPhrase() uses a regex to find phrases in an input string SearchResults searchForPhrase(String phr, String in, String ti){ List<SearchResults.Result> results = new ArrayList<>(); String regex = phr.trim()

.replaceAll("\\s+", "\\\s+")

.replace("?", "\\?");

Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL); Matcher match = pat.matcher(in);

```
Explicitly loop thru all regex matches
while (match.find()) results.add(new Result(match.start()));
```

return new SearchResults (Thread.currentThread().getId(),

36

0, phr, ti, results); ...

```
    searchForPhrase() uses a regex to find phrases in an input string

  SearchResults searchForPhrase(String phr, String in, String ti){
    List<SearchResults.Result> results = new ArrayList<>();
    String regex = phr.trim()
```

.replaceAll("\\s+", "\\\s+")

.replace("?", "\\?");

Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL);

Matcher match = pat.matcher(in);

```
Explicitly add starting index of all regex matches
while (match.find()) results.add(new Result(match.start()));
```

return new SearchResults (Thread.currentThread().getId(),

0, phr, ti, results); ...

searchForPhrase() uses a regex to find phrases in an input string

SearchResults searchForPhrase(String phr, String in, String ti){ List<SearchResults.Result> results = new ArrayList<>(); String regex = phr.trim() .replaceAll("\\s+", "\\\s+") .replace("?", "\\?"); Pattern pat = compile(regex, CASE INSENSITIVE | DOTALL); Matcher match = pat.matcher(in);

while (match.find()) results.add(new Result(match.start()));
return new SearchResults(Thread.currentThread().getId(),

0, phr, ti, results); ...

Return search results to processInput()

End of Java Sequential SearchStreamGang Example: Object-Oriented Implemention