

Overview of the Java SearchWith ParallelStreams Case Study

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

**Institute for Software
Integrated Systems**

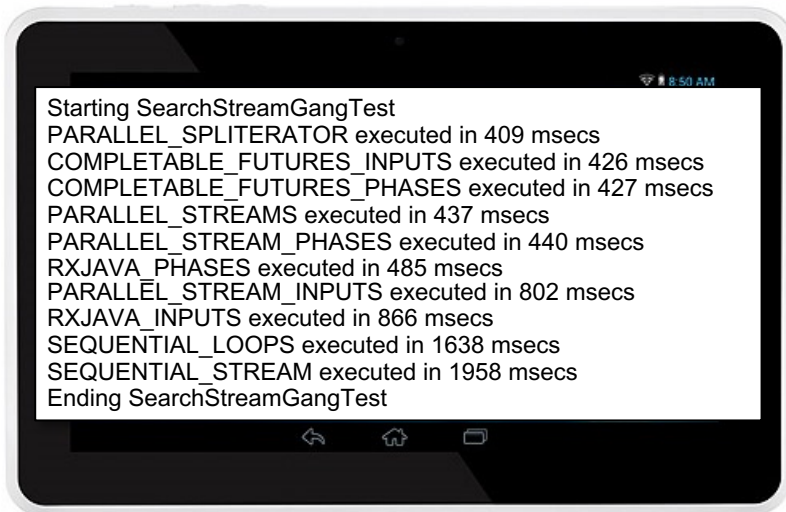
**Vanderbilt University
Nashville, Tennessee, USA**



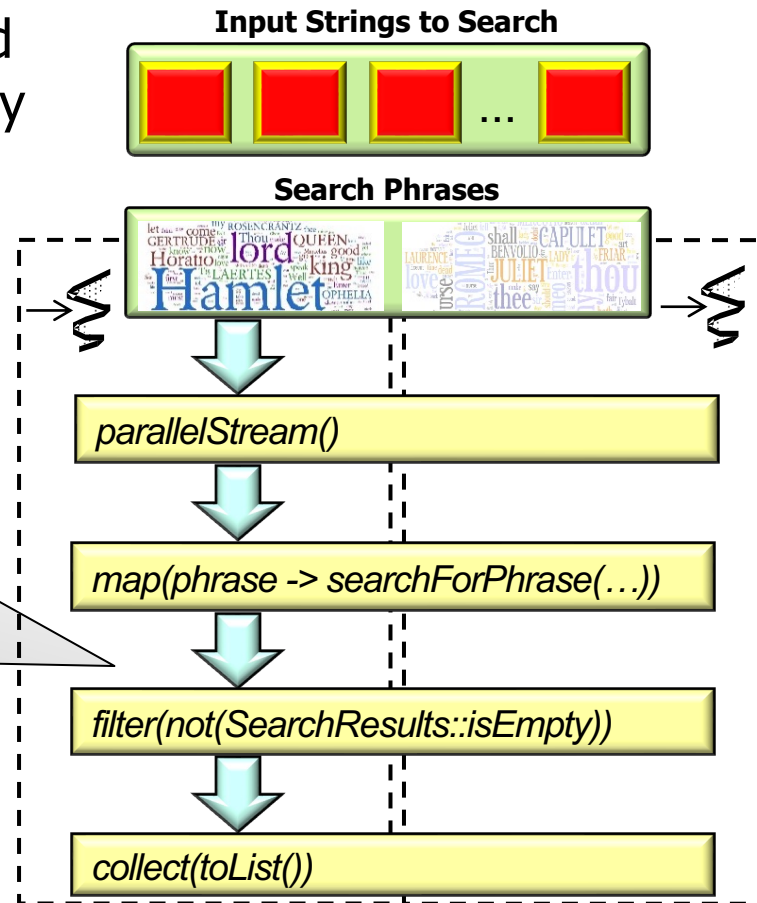
Learning Objectives in this Part of the Lesson

- Know how Java parallel streams are applied in the SearchWithParallelStreams case study

```
<<Java Class>>  
 SearchWithParallelStreams  
  
♦ processStream():List<List<SearchResults>>  
■ processInput(CharSequence):List<SearchResults>
```



```
Starting SearchStreamGangTest  
PARALLEL_SPLITTERATOR executed in 409 msecs  
COMPLETABLE_FUTURES_INPUTS executed in 426 msecs  
COMPLETABLE_FUTURES_PHASES executed in 427 msecs  
PARALLEL_STREAMS executed in 437 msecs  
PARALLEL_STREAM_PHASES executed in 440 msecs  
RXJAVA_PHASES executed in 485 msecs  
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  
Ending SearchStreamGangTest
```



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

Applying Parallel Streams to SearchStreamGang

Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()

<<Java Class>>

 **SearchWithParallelStreams**

◆ processStream():List<List<SearchResults>>

■ processInput(CharSequence):List<SearchResults>

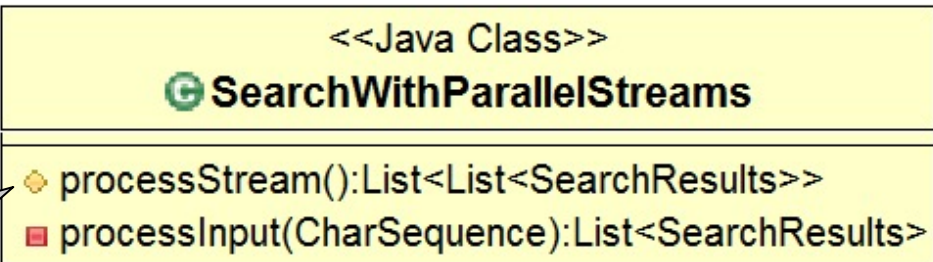
See github.com/douglasraigschmidt/LiveLessons/tree/master/SearchStreamGang

Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()

```
getInput()  
    .parallelStream()  
    .map(this::processInput)  
    .collect(toList());
```

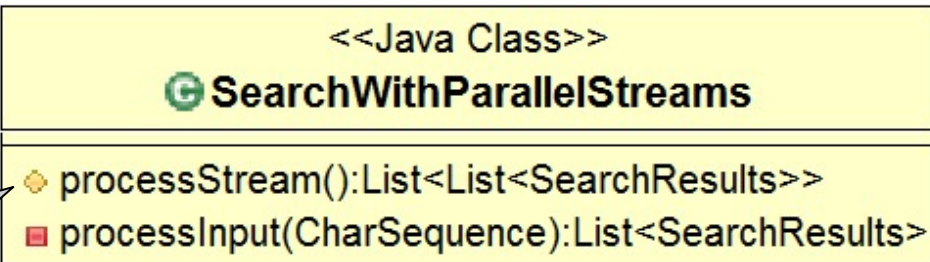
```
return mPhrasesToFind  
    .parallelStream()  
    .map(phrase -> searchForPhrase(phrase, input, title, false))  
    .filter(not(SearchResults::isEmpty))  
    .collect(toList());
```



Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()

```
getInput()  
    .parallelStream()  
    .map(this::processInput)  
    .collect(toList());
```



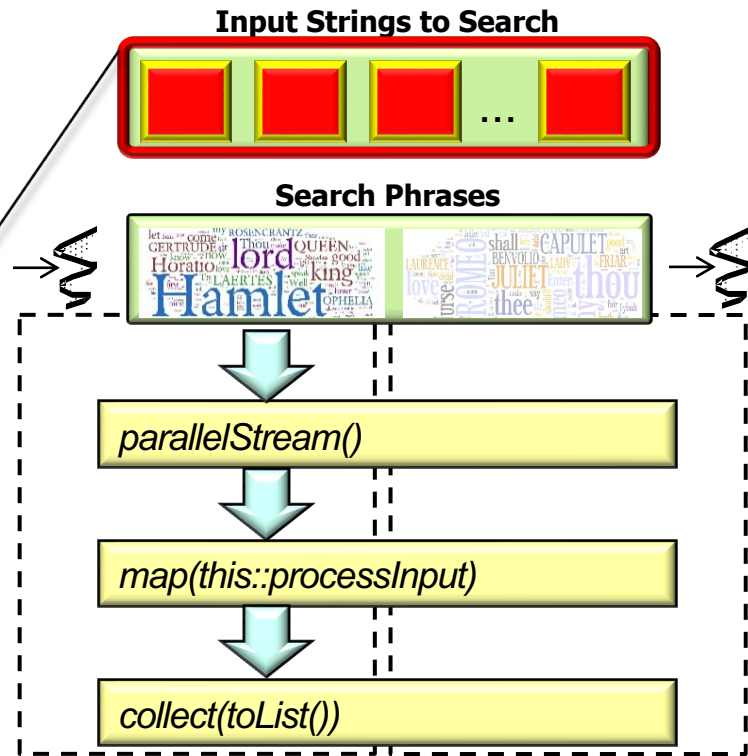
```
return mPhrasesToFind  
    .parallelStream()  
    .map(phrase -> searchForPhrase(phrase, input, title, false))  
    .filter(not(SearchResults::isEmpty))  
    .collect(toList());
```

i.e., the map(), filter(), & collect() aggregate operations

Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()
- **processStream()**
 - Uses a parallel stream to search a list of input strings

Each input string contains a work of Shakespeare (e.g., Hamlet, MacBeth, etc.)

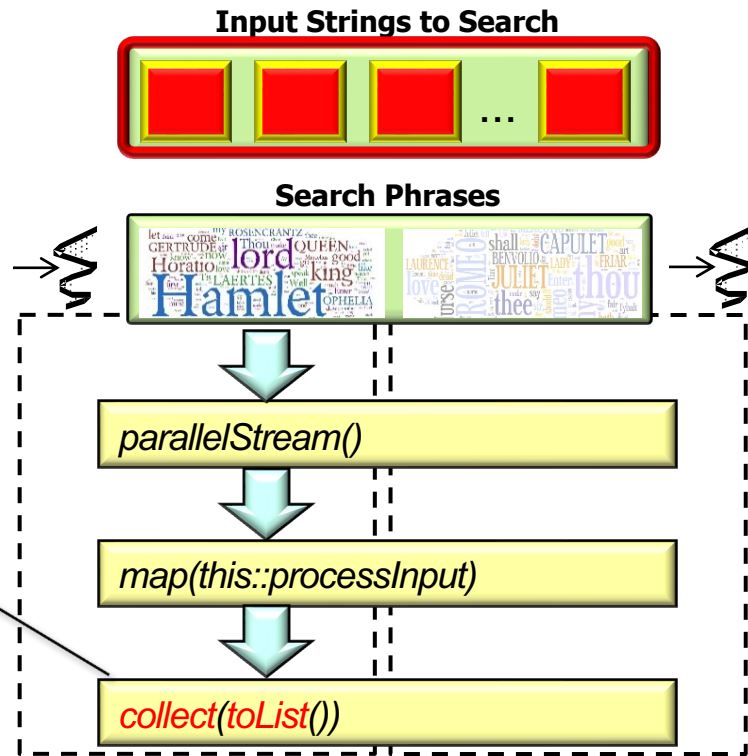


This parallel stream uses the common fork-join pool of worker threads

Applying Parallel Streams to SearchStreamGang

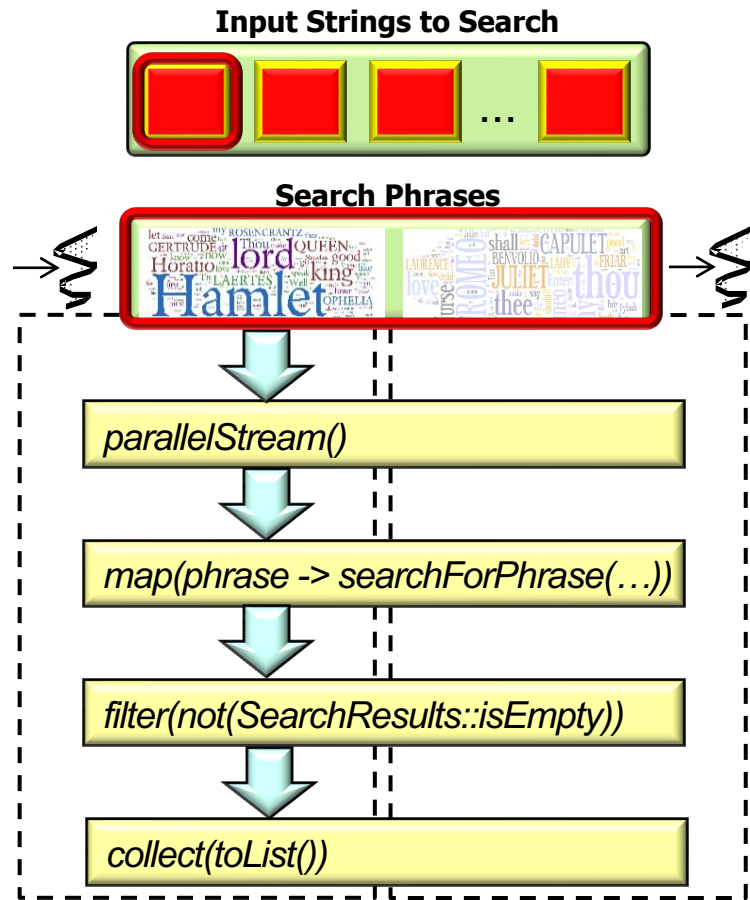
- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()
- **processStream()**
 - Uses a parallel stream to search a list of input strings

Returns a list of lists of SearchResults



Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()
- **processStream()**
- **processInput()**
 - Uses a parallel stream to search each input string & locate all occurrences of phrases



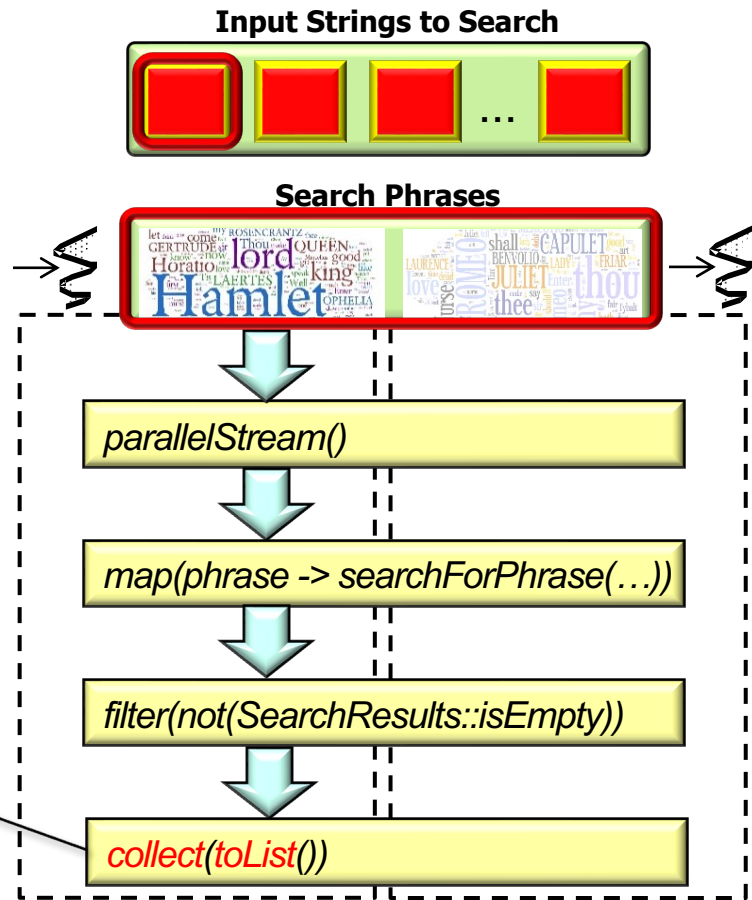
This parallel stream also uses the common fork-join pool of worker threads

Applying Parallel Streams to SearchStreamGang

- We focus on parallel streams in the SearchWithParallelStreams methods processStream() & processInput()
 - **processStream()**
 - **processInput()**
 - Uses a parallel stream to search each input string & locate all occurrences of phrases



*Returns a list of
SearchResults*



End of Overview of the Java SearchWithParallelStreams Case Study