Overview of Java 8 Streams (Part 3)

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

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations
  - “Splittable iterators” (Spliterators)

See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html
Learning Objectives in this Part of the Lesson

- Understand the structure & functionality of Java 8 streams, e.g.,
  - Fundamentals of streams
  - Common stream aggregate operations
  - “Splittable iterators” (Spliterators)
  - We’ll show how a Spliterator is used in the SimpleSearchStream

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

stream()
map (this::searchForWord)
filter (not(SearchResults::isEmpty))
collect (toList())
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream)
Overview of the Java Spliterator
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8

See docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- Iterator - It can be used to traverse elements of a source
  - e.g., a collection, array, etc.

```java
List<String> quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...");

for (Spliterator<String> s = quote.spliterator();
     s.tryAdvance(System.out::print)
     != false;
     )
{
    continue;
}
```

Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
  - e.g., a collection, array, etc.

```java
List<String> quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...);

for (Spliterator<String> s = quote.spliterator();
     s.tryAdvance(System.out::print) != false;
     continue;
) {
}
```

The source is an array/list of strings
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
  - e.g., a collection, array, etc.

```java
List<String> quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...);

for (Spliterator<String> s = quote.spliterator();
    s.tryAdvance(System.out::print)
    != false; )
    continue;
```

Create a spliterator for the entire array/list
Overview of the Java Spliterator

• A Spliterator is a new type of “splittable iterator” in Java 8

• Iterator – It can be used to traverse elements of a source
  • e.g., a collection, array, etc.

```
List<String> quote = Arrays.asList
  ("This ", "above ", "all- ",
  "to ", "thine ", "own ",
  "self ", "be " , "true", ",\n", ...");

for (Spliterator<String> s = quote.spliterator();
    s.tryAdvance(System.out::print)
    != false;
    )
    continue;
```

tryAdvance() combines the hasNext() & next() methods of Iterator
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- *Iterator*: It can be used to traverse elements of a source
  - e.g., a collection, array, etc.

```java
List<String> quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...);

for (Spliterator<String> s = quote.spliterator();
     s.tryAdvance(System.out::println)
     != false;
)
    continue;
```

*Print value of each string in the quote*
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
- **Split** – It can also partition all elements of a source

```java
class quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...");
Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf = secondHalf.trySplit();

firstHalf.forEachRemaining(System.out::print);
secondHalf.forEachRemaining(System.out::print);
```

Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
  - *Iterator* – It can be used to traverse elements of a source
  - *Split* – It can also partition all elements of a source

List<String> quote = Arrays.asList("This ", "above ", "all- ",
  "to ", "thine ", "own ",
  "self ", "be ", "true" , ",\n", ...");

Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf =
  secondHalf.trySplit();

firstHalf.forEachRemaining(System.out::print);
secondHalf.forEachRemaining(System.out::print);

Create a spliterator for the entire array/list
Overview of the Java Spliterator

- A Spliterator is a new type of “splittable iterator” in Java 8
- Iterator – It can be used to traverse elements of a source
- Split – It can also partition all elements of a source

```java
List<String> quote = Arrays.asList
    ("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...");
Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf = secondHalf.trySplit();
firstHalf.forEachRemaining(System.out::print);
secondHalf.forEachRemaining(System.out::print);
```

trySplit() returns a spliterator covering elements that will no longer be covered by the invoking spliterator
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
  - *Iterator* – It can be used to traverse elements of a source
  - *Split* – It can also partition all elements of a source

```
List<String> quote = Arrays.asList
    ("This ", "above ", "all- ",
    "to ", "thine ", "own ",
    "self ", "be ", "true", ",\n    ...");

Spliterator<String> secondHalf =
    quote.spliterator();
Spliterator<String> firstHalf =
    secondHalf.trySplit();

firstHalf.forEachRemaining
    (System.out::print);
secondHalf.forEachRemaining
    (System.out::print);
```

Ideally a spliterator splits the original input source in half!
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
- **Split** – It can also partition all elements of a source

```java
List<String> quote = Arrays.asList
        ("This ", "above ", "all- ",
         "to ", "thine ", "own ",
         "self ", "be ", "true", ",\n",
         ...);

Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf = secondHalf.trySplit();

firstHalf.forEachRemaining
        (System.out::print);
secondHalf.forEachRemaining
        (System.out::print);
```

*Performs the action for each element in the spliterator*
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
- **Split** – It can also partition all elements of a source

```
List<String> quote = Arrays.asList("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", ",\n", ...");

Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf = secondHalf.trySplit();

firstHalf.forEachRemaining(System.out::print);
secondHalf.forEachRemaining(System.out::print);
```

Print value of each string in the quote
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
- **Iterator** – It can be used to traverse elements of a source
- **Split** – It can also partition all elements of a source
- Mostly used with Java 8 parallel streams

See blog.logentries.com/2015/10/java-8-introduction-to-parallelism-and-spliterator
Overview of the Java Spliterator

- A Spliterator is a new type of "splittable iterator" in Java 8
  - *Iterator* – It can be used to traverse elements of a source
  - *Split* – It can also partition all elements of a source

We’ll focus on traversal now & on partitioning after covering parallel streams
Overview of the Java Spliterator

- The `StreamSupport.stream()` method creates a new sequential or parallel stream from a spliterator.
Overview of the Java Spliterator

- The StreamSupport.stream() method creates a new sequential or parallel stream from a spliterator
- e.g., the Collection interface defines two default methods using this capability

```java
public interface Collection<E> extends Iterable<E> {
    ...
    default Stream<E> stream() {
        return StreamSupport
            .stream(spliterator(),
                    false);
    }

    default Stream<E> parallelStream()
        return StreamSupport
            .stream(spliterator(),
                    true);
}
```

See [jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Collection.java](jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Collection.java)
Using Java Spliterator in SimpleSearchStream
Using Java Spliterator in SimpleSearchStream

- The SimpleSearchStream program uses a sequential spliterator:

  ```java
  stream()
  .map(this::searchForWord)
  .filter(not(SearchResults::isEmpty))
  .collect(toList())
  ```

Search Words:
- "do", "re", "mi", "fa", "so", "la", "ti", "do"

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream)
Using Java Spliterator in SimpleSearchStream

- `searchForWord()` uses the spliterator to find all instances of a word in the input & return a list of all the `SearchResults`

```java
SearchResults searchForWord(String word) {
    return new SearchResults(..., word, ..., StreamSupport.stream(new WordMatchSpliterator(mInput, word), false).collect(toList()));
}
```

See `SimpleSearchStream/src/main/java/search/WordSearcher.java`
Using Java Spliterator in SimpleSearchStream

- searchForWord() uses the spliterator to find all instances of a word in the input & return a list of all the SearchResults

```java
SearchResults searchForWord
    (String word){
        return new SearchResults
            (... word, ..., StreamSupport
                .stream(new WordMatchSpliterator
                    (mInput, word),
                    false)
                .collect(toList()));
    }
```

StreamSupport.stream() creates a sequential stream via the WordMatchSpliterator class

See docs.oracle.com/javase/8/docs/api/java/util/stream/StreamSupport.html#stream
Using Java Spliterator in SimpleSearchStream

- `searchForWord()` uses the spliterator to find all instances of a word in the input & return a list of all the `SearchResults`

```
SearchResults searchForWord(String word) {
    return new SearchResults(..., word, ..., StreamSupport.stream(new WordMatchSpliterator(mInput, word), false).collect(toList()));
}
```

This stream is collected into a list of `SearchResults.Result` objects
Using Java Spliterator in SimpleSearchStream

- WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string.

```java
class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {
    private final Matcher mWordMatcher;

    public WordMatchSpliterator(String input, String word) {
        ...
        String regexWord = "\\b" + word.trim() + "\\b";

        mWordMatcher =
            Pattern.compile(regexWord,
                Pattern.CASE_INSENSITIVE)
                .matcher(input);
    }
```

See SimpleSearchStream/src/main/java/search/WordMatchSpliterator.java
Using Java Spliterator in SimpleSearchStream

- WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string.

```java
class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {
    private final Matcher mWordMatcher;

    public WordMatchSpliterator(String input, String word) {
        ...,
        String regexWord = "\\b" + word.trim() + "\\b";
        mWordMatcher =
            Pattern.compile(regexWord,
                Pattern.CASE_INSENSITIVE)
                .matcher(input);
    }
```

Create a regex that matches only a “word”

See [www.vogella.com/tutorials/JavaRegularExpressions/article.html](http://www.vogella.com/tutorials/JavaRegularExpressions/article.html)
Using Java Spliterator in SimpleSearchStream

- WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string

```java
class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {
    private final Matcher mWordMatcher;

    public WordMatchSpliterator(String input, String word) {
        ...
        String regexWord = "\\b" + word.trim() + "\\b";

        mWordMatcher =
            Pattern.compile(regexWord,
                Pattern.CASE_INSENSITIVE)
                .matcher(input);
    }
```

Compile the regex & create a matcher for the input string

See [docs.oracle.com/javase/8/docs/api/java/util/regex/PATTERN.html](http://docs.oracle.com/javase/8/docs/api/java/util/regex/PATTERN.html)
Using Java Spliterator in SimpleSearchStream

WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string.

class WordMatchSpliterator
  extends Spliterators.AbstractSpliterator<Result> {
    ...
    public boolean tryAdvance(Consumer<? super Result> action) {
      if (!mWordMatcher.find())
        return false;
      else {
        action.accept(new Result(mWordMatcher.start()));
        return true;
      }
    }
  }

Called by the Java 8 streams framework to attempt to advance the spliterator by one word match.
• WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string

```java
class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {
    ...
    public boolean tryAdvance(Consumer<? super Result> action) {
        if (!mWordMatcher.find())
            return false;
        else {
            action.accept(new Result(mWordMatcher.start()));
            return true;
        }
    }
}
```

Pass-by-reference param used to get next item back to framework
Using Java Spliterator in SimpleSearchStream

- WordMatchSpliterator uses Java regex to create a stream of SearchResults Result objects that match the # of times a word appears in an input string.

```java
class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {

    
    public boolean tryAdvance(Consumer<? super Result> action) {
        if (!mWordMatcher.find())
            return false;
        else {
            action.accept(new Result(mWordMatcher.start()));
            return true;
        }
    }

    }
WordMatchSpliterator uses Java regex to create a stream of SearchResult objects that match the # of times a word appears in an input string

class WordMatchSpliterator
    extends Spliterators.AbstractSpliterator<Result> {

    public boolean tryAdvance(Consumer<? super Result> action) {
        if (!mWordMatcher.find())
            return false;
        else {
            action.accept(new Result(mWordMatcher.start()));
            return true;
        }
    }

    }

If there's a match the consumer records the index where the match occurred
Using Java Spliterator in SimpleSearchStream

- Here’s the output that searchForWord() & WordMatchSpliterator produce

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

Search Words

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

List <String>

Stream <String>

Stream <SearchResults>

Stream <SearchResults>

List <SearchResults>
End of Overview of Java 8 Streams (Part 3)