Understand Java Streams Spliterator

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 “Splittable iterators” (Spliterators)

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

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

```java
public interface Spliterator<T>

Type Parameters:
T - the type of elements returned by this Spliterator

All Known Subinterfaces:
Spliterator.OfDouble, Spliterator.OfInt, Spliterator.OfLong,
Spliterator.OfPrimitive<T,T_CONS,T_SPLITR>

All Known Implementing Classes:
Spliterators.AbstractDoubleSpliterator,
Spliterators.AbstractIntSpliterator,
Spliterators.AbstractLongSpliterator,
Spliterators.AbstractSpliterator
```

An object for traversing and partitioning elements of a source. The source of elements covered by a Spliterator could be, for example, an array, a Collection, an IO channel, or a generator function.

A Spliterator may traverse elements individually (tryAdvance()) or sequentially in bulk (forEachRemaining()).

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

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

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

This source is an array/list of strings

See github.com/douglascraigschmidt/LiveLessons/tree/master/Java8/ex13
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 = List.of
("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.

```java
List<String> quote = List.of
    ("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*

See [docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#tryAdvance](http://docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#tryAdvance)
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 = List.of
("This ", "above ", "all- ",
 "to ", "thine ", "own ",
 "self ", "be ", "true", 
\n",
...
);

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

```java
boolean tryAdvance(Consumer<? super T> action) {
    if (noMoreElementsRemain)
        return false;
    else {
        action.accept
            (nextElement);
        return true;
    }
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#tryAdvance](http://docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#tryAdvance)
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 = List.of
    ("This ", "above ", "all- ",
     "to ", "thine ", "own ",
     "self ", "be ", "true", "\n",
     ...);

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

```java
boolean tryAdvance(Consumer<? super T> action) {
    if (noMoreElementsRemain)
        return false;
    else {
        action.accept(nextElement);
        return true;
    }
}
```
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 = List.of
("This ", "above ", "all- ",
"to ", "thine ", "own ",
"self ", "be ", "true", "\n",
...);

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

```java
boolean tryAdvance(Consumer<? super T> action) {
    if (noMoreElementsRemain)
        return false;
    else { ...
        action.accept
            (nextElement);
        return true;
    }
}
```
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 = List.of
    ("This ", "above ", "all- ",
    "to ", "thine ", "own ",
    "self ", "be ", "true", "\n",
    ...);

for (Spliterator<String> s =
     quote.spliterator();
    s.tryAdvance(System.out::print)
    != 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
List<String> quote = List.of("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

```java
List<String> quote = List.of
("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 = List.of
("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

See [docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#trySplit](docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html#trySplit)
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 = List.of("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", "\n", ...);

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

```java
Spliterator<T> trySplit() {
    if (input <= minimum size)
        return null
    else {
        split input in 2 chunks
        update “right chunk”
        return spliterator for “left chunk”
    }
}
```
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 = List.of
("This ", "above ", "all- ",
"to ", "thine ", "own ",
"self ", "be ", "true", "\n", ...
);

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

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

Spliterator<T> trySplit()
{
  if (input <= minimum size)
    return null
  else {
    split input in 2 chunks
    update "right chunk"
    return spliterator
      for "left chunk"
  }
}

trySplit() calls itself recursively until all chunks are <= to the minimize size.
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 = List.of
("This ", "above ", "all-",
 "to ", "thine ", "own ",
 "self ", "be ", "true", "\n",
...
);
Spliterator<String> secondHalf = quote.spliterator();
Spliterator<String> firstHalf = secondHalf.trySplit();
```

Ideally, a spliterator efficiently 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 = List.of("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", "\n", ...);

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

```java
Spliterator<T> trySplit() {
    if (input <= minimum size)
        return null
    else {
        split input in 2 chunks
        update "right chunk"
        return spliterator for "left chunk"
    }
}
```

The "right chunk" is defined by updating the state of this spliterator object
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 = List.of("This ", "above ", "all- ", "to ", "thine ", "own ", "self ", "be ", "true", "\n", ...);

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

Spliterator\()<\text{T}> trySplit() \{ 
  if (input <= minimum size) 
    return null 
  else {
    split input in 2 chunks 
    update "right chunk" 
    return spliterator 
    for "left chunk"
  }
\}

The "left chunk" is defined by creating/returning a new spliterator object
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 = List.of
("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 = List.of
    ("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 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

### Interface Spliterator<T>

**Type Parameters:**

T - the type of elements returned by this Spliterator

**All Known Subinterfaces:**


**All Known Implementing Classes:**


```java
class Spliterator<T> {
    public interface Spliterator<T>
    {
        An object for traversing and partitioning elements of a source. The source of elements covered by a Spliterator could be, for example, an array, a Collection, an IO channel, or a generator function.

        A Spliterator may traverse elements individually (tryAdvance()) or sequentially in bulk (forEachRemaining()).
    }
}
```
Overview of the Java Spliterator

The StreamSupport.stream() factory method creates a new sequential or parallel stream from a spliterator.

```java
public static <T> Stream<T> stream(Spliterator<T> spliterator, boolean parallel)

Creates a new sequential or parallel Stream from a Spliterator.

The spliterator is only traversed, split, or queried for estimated size after the terminal operation of the stream pipeline commences.

It is strongly recommended the spliterator report a characteristic of IMMUTABLE or CONCURRENT, or be late-binding. Otherwise, stream(java.util.function.Supplier, int, boolean) should be used to reduce the scope of potential interference with the source. See Non-Interference for more details.

Type Parameters:
T - the type of stream elements

Parameters:
spliterator - a Spliterator describing the stream elements
parallel - if true then the returned stream is a parallel stream; if false the returned stream is a sequential stream.

Returns:
a new sequential or parallel Stream
```
Overview of the Java Spliterator

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

```java
class 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)
Overview of the Java Spliterator

- The StreamSupport.stream() factory 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);
      }
  }
  ```

The ‘false’ parameter creates a sequential stream, whereas ‘true’ creates a parallel stream.
End of Understand Java Streams Spliterators