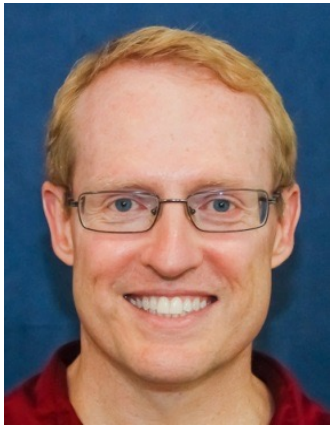
# Common Java Streams Factory Methods

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

- Recognize common factory methods used to create streams

# Common Factory Methods for Creating Streams

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream

See docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection

```
List<String> wordsToFind =
  List.of("do", "re", "me", ...);

List<SearchResults> results =
  wordsToFind.stream()
    ...
```

or

```
List<SearchResults> results =
  wordsToFind.parallelStream()
    ...
```

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection

```
List<String> wordsToFind =
  List.of("do", "re", "me", ...);

List<SearchResults> results =
  wordsToFind.stream()
    ...
```

or

```
List<SearchResults> results =
  wordsToFind.parallelStream()
    ...
```

See docs.oracle.com/javase/tutorial/collections/streams

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection

```
List<String> wordsToFind =
  List.of("do", "re", "me", ...);

List<SearchResults> results =
  wordsToFind.stream()
     ...
```

or

```
List<SearchResults> results =
  wordsToFind.parallelStream()
     ...
```

*We use this approach in the SimpleSearchStream program*

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection

```
List<String> wordsToFind =
  List.of("do", "re", "me", ...);

List<SearchResults> results =
  wordsToFind.stream()
    ...
```

or

```
List<SearchResults> results =
  wordsToFind.parallelStream()
    ...
```

See docs.oracle.com/javase/tutorial/collections/streams/parallelism.html

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection

```
List<String> wordsToFind =
  List.of("do", "re", "me", ...);

List<SearchResults> results =
  wordsToFind.stream()
    ...
```

or

```
List<SearchResults> results =
  wordsToFind.stream()
  ...
  .parallel()
```

*A call to parallel() can appear anywhere in a stream & will have same effect as parallelStream()*

See docs.oracle.com/javase/8/docs/api/java/util/stream/BaseStream.html#parallel

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
    - StreamSupport.stream() factory method

```
SearchResults searchForPhrase
   (String phrase, CharSequence input,
    String title, boolean parallel) {
   return new SearchResults
      (..., phrase, ..., StreamSupport
         .stream(new PhraseMatchSpliterator
                        (input, phrase),
                 parallel)
         .toList());
}
```

*Create a stream that contains all the phrases that match in an input string.*

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
    - StreamSupport.stream() factory method

*Create a new sequential or parallel stream from a Java spliterator*

```
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`

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
    - StreamSupport.stream() factory method

```
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 Collection interface defines two default methods using this capability*

See jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Collection.java

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
    - StreamSupport.stream() factory method

```
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*

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
    - StreamSupport.stream() factory method

```
SearchResults searchForPhrase
   (String phrase, CharSequence input,
    String title, boolean parallel) {
   return new SearchResults
      (..., phrase, ..., StreamSupport
        .stream(new PhraseMatchSpliterator
                          (input, phrase),
                parallel)
        .toList());
}
```

*Create a stream that contains all the phrases that match in an input string.*

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Arrays.stream(a);

stream.forEach(s ->
                  System.out.println(s));
```

or

```
stream.forEach(System.out::println);
```

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Arrays.stream(a);

stream.forEach(s ->
                    System.out.println(s));
```

*Create stream containing all elements in an array*

or

```
stream.forEach(System.out::println);
```

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Arrays.stream(a);

stream.forEach(s ->
                    System.out.println(s));
```

*Print all elements in the stream*

or

```
stream.forEach(System.out::println);
```

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array
  - From a static factory method

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Stream.of(a);

stream.forEach(s ->
                  System.out.println(s));
```

or

```
stream.forEach(System.out::println);
```

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array
  - From a static factory method

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Stream.of(a);

stream.forEach(s ->
                  System.out.println(s));
```

or

```
stream.forEach(System.out::println);
```

*Create stream containing all elements in an array*

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

# Common Factory Methods for Creating Streams

- There are several common ways to obtain a stream, e.g.
  - From a Java collection
  - From an array
  - From a static factory method

```
String[] a = {
  "a", "b", "c", "d", "e"
};

Stream<String> stream = Stream.of(a);

stream.forEach(s ->
                    System.out.println(s));
```

*Print all elements in the stream*

or

```
stream.forEach(System.out::println);
```

# End of Common Java Streams Factory Methods