Java Streams: Common 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

- Understand 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

```java
List<String> wordsToFind = Arrays.asList("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

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

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

or

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

See docs.oracle.com/javase/tutorial/collectionsstreams
Common Factory Methods for Creating Streams

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

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

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

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

We use this approach in the SimpleSearchStream program.

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleSearchStream](https://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

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

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

    or

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

See docs.oracle.com/javase/tutorial/collections/strips/parallelism.html
There are several common ways to obtain a stream, e.g.

- From a Java collection

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

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
    ```java
    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);
    ```
  - From an array
Common Factory Methods for Creating Streams

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

```java
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);
```

Create stream containing all elements in an array

See docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#stream
There are several common ways to obtain a stream, e.g.

- From a Java collection

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

Stream<String> stream = Arrays.stream(a);
stream.forEach(s ->
    System.out.println(s));
```

- From an array

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

Print all elements in the stream
There are several common ways to obtain a stream, e.g.

- From a Java collection

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

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

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

- From an array

- From a static factory method

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

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

Common Factory Methods for Creating Streams
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

```java
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

```java
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);
```

*Print all elements in the stream*
There are several common ways to obtain a stream, e.g.
- From a Java collection
- From an array
- From a static factory method

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
    f -> new BigInteger[]{f[1],
    f[0].add(f[1])})
    .map(f -> f[0])
    .limit(100)
    .forEach(System.out::println);
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#iterate](http://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#iterate)
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

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, 
  BigInteger.ONE}, 
  f -> new BigInteger[]{f[1], 
    f[0].add(f[1])})
  .map(f -> f[0])
  .limit(100)
  .forEach(System.out::println);
```

Generate & print the first 100 Fibonacci #'s
There are several common ways to obtain a stream, e.g.

- From a Java collection
- From an array
- From a static factory method

```
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
    f -> new BigInteger[]{f[1],
        f[0].add(f[1])})
    .map(f -> f[0])
    .limit(100)
    .forEach(System.out::println);
```

Create the "seed," which defines the initial element in the 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
  - From a static factory method

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
               f -> new BigInteger[]{f[1], f[0].add(f[1])})
               .map(f -> f[0])
               .limit(100)
               .forEach(System.out::println);
```

A lambda function applied to the previous element to produce a new element
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

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
    f -> new BigInteger[]{f[1],
                        f[0].add(f[1])})
    .map(f -> f[0])
    .limit(100)
    .forEach(System.out::println);
```

Convert the array to its first element
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

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
               f -> new BigInteger[]{f[1],
                                     f[0].add(f[1])})
  .map(f -> f[0])
  .limit(100)
  .forEach(System.out::println);
```

See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#limit](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#limit)
There are several common ways to obtain a stream, e.g.

- From a Java collection
- From an array
- From a static factory method

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
    f -> new BigInteger[]{f[1],
    f[0].add(f[1])})
    .map(f -> f[0])
    .limit(100)
    .forEach(System.out::println);
```

Print the Fibonacci #'s
There are several common ways to obtain a stream, e.g.

- From a Java collection
- From an array
- From a static factory method

```java
Stream.iterate(new BigInteger[]{BigInteger.ONE, BigInteger.ONE},
               f -> new BigInteger[]{f[1], f[0].add(f[1])}).
    .parallel()
    .map(f -> f[0])
    .limit(100)
    .forEach(System.out::println);
```

*Avoid using iterate() in a parallel stream!*
End of Java Streams: Common Factory Methods