Java 8 Stream 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 Lesson

• Recognize common stream factory methods
Common Factory Methods for Creating Streams
Common Factory Methods for Creating Streams

- Streams can be obtained various ways

See [docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html](docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html)
Streams can be obtained various ways, e.g.,

- From a Java collection

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

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

or

```java
List<SearchResults> results = wordsToFind.parallelStream();
```
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
- From a Java collection

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

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

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

- Streams can be obtained various ways, 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/stream/parallelism.html](docs.oracle.com/javase/tutorial/collections/stream/parallelism.html)
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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
Streams can be obtained various ways, 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

- Streams can be obtained various ways, 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

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

Get & print all the elements in an array

See [docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#stream](docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#stream)
Streams can be obtained various ways, 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);
```

Get & print all the elements in an array
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
    - 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);
  - From an array
  - From a static factory method
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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));
  or
  stream.forEach(System.out::println);
  ```
  
  - From an array
  - From a static factory method
  
See [docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#of)
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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);
```

Get & print all the elements in an array
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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](docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#iterate)
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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);
```

Create the “seed,” which defines the initial element in the stream
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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

- Streams can be obtained various ways, 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

- Streams can be obtained various ways, 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);
```

Limit the stream to 100 elements

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)
Common Factory Methods for Creating Streams

• Streams can be obtained various ways, 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
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, 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);
```

Don’t use `iterate()` in a parallel stream! Use the *Stream.range()* method instead..

See docs.oracle.com/javase/8/docs/api/java/util/stream/IntStream.html#range
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file

```java
void printFileLines(String filename) {
    try (BufferedReader reader = Files.newBufferedReader(Paths.get(filename))) {
        reader.lines().forEach(System.out::println);
    } catch (IOException ex) {
    }
}
```

Create a stream containing all of the lines in a file

See docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html#lines
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file

```java
class Example {
    void printFileLines
        (String filename)
    {
        try (BufferedReader reader = Files.newBufferedReader
                     (Paths.get(filename))) {
            reader
                .lines()
                .forEach
                    (System.out::println);
        } catch (IOException ex) {...}
    }
}
```

Print each of the lines in the stream
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods

```java
void printFileLines(String filename){
    try(Stream<String> stream = Files.lines(Paths.get(fileName))) {
        stream.forEach(System.out::println);
    } catch (IOException ex) {...}
}
```

Create a stream containing all of the lines in a file

See docs.oracle.com/javase/8/docs/api/java/nio/file/Files.html#lines
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file
- Streams of file paths & lines can be obtained from Files methods

```java
void printFileLines(String filename){
    try(Stream<String> stream = Files.lines(Paths.get(fileName))) {
        stream.forEach(System.out::println);
    } catch (IOException ex) { ... }
}
```

*Print each of the lines in the stream*
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #'s can be obtained from Random.ints()

```
new Random()
    .ints(0,100)
    .limit(50)
    .forEach(System.out::println);
```
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #’s can be obtained from Random.ints()

Generate an “unbounded” stream of random #’s ranging between 0 & 100

```java
new Random()
   .ints(0,100)
   .limit(50)
   .forEach(System.out::println);
```

See [docs.oracle.com/javase/8/docs/api/java/util/Random.html#ints](http://docs.oracle.com/javase/8/docs/api/java/util/Random.html#ints)
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file
- Streams of file paths & lines can be obtained from Files methods
- A stream of random #'s can be obtained from Random.ints()

```java
new Random()
    .ints(0, 100)
    .limit(50)
    .forEach(System.out::println);
```

Limit the size of the stream to 50 elements
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - `BufferedReader.lines()` obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #’s can be obtained from `Random.ints()`

```java
new Random()
  .ints(0,100)
  .limit(50)
  .forEach(System.out::println);
```

Print each random # in the stream
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #'s can be obtained from Random.ints()
  - Other JDK stream-bearing methods

```java
Stream<String> getData(String filename, String splitter) {
    return Pattern.compile(splitter).splitAsStream(new String(Files.readAllBytes(Paths.get(filename).toURI())));
}
```
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #'s can be obtained from Random.ints()
  - Other JDK stream-bearing methods

```java
Stream<String> getInputData
    (String filename,
     String splitter)
{
    return Pattern
        .compile(splitter)
        .splitAsStream
            (new String
                (Files.readAllBytes
                    (Paths.get(filename)
                        .toURI())));
}
```

Splits a file into a stream of strings

See docs.oracle.com/javase/8/docs/api/java/util/regex/PATTERN.html#splitAsStream
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file
- Streams of file paths & lines can be obtained from Files methods
- A stream of random #'s can be obtained from Random.ints()
- Other JDK stream-bearing methods

```java
List<TreeMap<Long, String>>
listOfTreeMaps =
    Stream.generate
        (TreeMap<Long, String>::new)
        .limit(100)
        .collect(toList());
```

Generate an “infinite” stream of TreeMaps

See docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html#generate
Common Factory Methods for Creating Streams

- Streams can be obtained various ways, e.g.,
  - From a Java collection
  - From an array
  - From a static factory method
  - BufferedReader.lines() obtains lines of a file
  - Streams of file paths & lines can be obtained from Files methods
  - A stream of random #’s can be obtained from Random.ints()
  - Other JDK stream-bearing methods

```java
List<TreeMap<Long, String>>
listOfTreeMaps =
    Stream.generate
    (TreeMap<Long, String>::new)
    .limit(100)
    .collect(toList());
```

*Limit the stream to 100 elements*
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file
- Streams of file paths & lines can be obtained from Files methods
- A stream of random #'s can be obtained from Random.ints()
- Other JDK stream-bearing methods

```java
List<TreeMap<Long, String>>
listOfTreeMaps =
Stream.generate
   (TreeMap<Long, String>::new)
   .limit(100)
   .collect(toList());
```

Create a list of 100 TreeMaps
Streams can be obtained various ways, e.g.,

- From a Java collection
- From an array
- From a static factory method
- BufferedReader.lines() obtains lines of a file
- Streams of file paths & lines can be obtained from Files methods
- A stream of random #'s can be obtained from Random.ints()
- Other JDK stream-bearing methods

```java
List<TreeMap<Long, String>>
listOfTreeMaps =
Stream.generate
    (TreeMap<Long, String>::new)
    .limit(100)
    .collect(toList());
```

We’ll use this idiom in Java 8 programs covered later.
End of Java 8 Stream Factory Methods