Streams

Primitive Streams

Primitive Streams

- when working primitive values, it's more convenient to use primitive streams
- there are three types of primitive streams:
  - `IntStream` - used for `int`, `short`, `byte` and `char`
  - `LongStream` - used for `long`
  - `DoubleStream` - used for `double` and `float`
- everything you know about streams apply to primitive streams as well
- only difference is that primitive streams have some unique methods

Unique Primitive Streams Methods (1/3)

Method	Primitive stream	What it does
`OptionalDouble average()`	`IntStream, LongStream, DoubleStream`	Arithmetic mean of elements
`Stream<T> boxed()`	`IntStream, LongStream, DoubleStream`	`Stream<T>` where `T` is wrapper class associated with primitive value
`OptionalInt max()`	`IntStream`	Maximum element of the stream
`OptionalLong max()`	`LongStream`	Maximum element of the stream
`OptionalDouble max()`	`DoubleStream`	Maximum element of the stream
`OptionalInt min()`	`IntStream`	Minimum element of the stream
`OptionalLong min()`	`LongStream`	Minimum element of the stream
`OptionalDouble min()`	`DoubleStream`	Minimum element of the stream

Unique Primitive Streams Methods (2/3)

Method	Primitive stream	What it does
`IntStream range(int a, int b)`	`IntStream`	Returns `IntStream` from a (inclusive) to b (exclusive)
`LongStream range(long a, long b)`	`LongStream`	Returns `LongStream` from a (inclusive) to b (exclusive)
`IntStream rangeClosed(int a, int b)`	`IntStream`	Returns `IntStream` from a (inclusive) to b (inclusive)
`LongStream rangeClosed(long a, long b)`	`LongStream`	Returns `LongStream` from a (inclusive) to b (inclusive)
`int sum()`	`IntStream`	Returns sum of elements in stream
`long sum()`	`LongStream`	Returns sum of elements in stream
`double sum()`	`DoubleStream`	Returns sum of elements in stream

Unique Primitive Streams Methods (3/3)

Method	Primitive stream	What it does
`IntSummaryStatistics summaryStatistics()`	`IntStream`	Returns object containing numerous stream statistics (avg, min, max, etc.)
`LongSummaryStatistics summaryStatistics()`	`LongStream`	Returns object containing numerous stream statistics (avg, min, max, etc.)
`DoubleSummaryStatistics summaryStatistics()`	`DoubleStream`	Returns object containing numerous stream statistics (avg, min, max, etc.)

```
// example #1
IntStream intStream = IntStream.of(7, 11, 21);
OptionalDouble avg = intStream.average();
System.out.println(avg.getAsDouble());
  => 13.0

// example #2
DoubleStream doubleStream = DoubleStream.of(3.14, 2.72, 1.618);
doubleStream.forEach(System.out::println);
  => 3.14
     2.72
     1.618
```

```
// example #3
IntStream intStream = IntStream.range(2, 5);
intStream.forEach(System.out::print);
  => 234

// example #4
IntStream intStream = IntStream.rangeClosed(2, 5);
intStream.forEach(System.out::print);
  => 2345
```

Mapping Streams

Source stream class	**To create:** `Stream`	**To create:** `DoubleStream`	**To create:** `IntStream`	**To create:** `LongStream`
`Stream<T>`	`map()`	`mapToDouble()`	`mapToInt()`	`mapToLong()`
`DoubleStream`	`mapToObj()`	`map()`	`mapToInt()`	`mapToLong()`
`IntStream`	`mapToObj()`	`mapToDouble()`	`map()`	`mapToLong()`
`LongStream`	`mapToObj()`	`mapToDouble()`	`mapToInt()`	`map()`

```
// mapping example

Stream<String> objStream = Stream.of("John", "Paul", "George", "Ringo");

IntStream intStream = objStream.mapToInt(s -> s.length());

intStream.forEach(System.out::print);

  => 4465
```

argument is `ToIntFunction`

```
// you can use flatMap() in the same way as before:
List<List<Integer>> listOfLists = new ArrayList<>();
listOfLists.add(Arrays.asList(1, 2, 3));
listOfLists.add(Arrays.asList(4, 5));
listOfLists.add(Arrays.asList(6, 7, 8, 9));
System.out.println(listOfLists);

IntStream intStream = listOfLists.stream()
  .flatMapToInt(list -> list.stream().mapToInt(n -> n));

intStream.forEach(System.out::print);
```

```
[[1, 2, 3], [4, 5], [6, 7, 8, 9]]
123456789
```

```
// primitive streams use optionals

var myIntStream = IntStream.rangeClosed(2, 7);

OptionalDouble myAvg = myIntStream.average();

myAvg.ifPresent(System.out::println);
  => 4.5

System.out.println(myAvg.getAsDouble());
  => 4.5

System.out.println(myAvg.orElseGet(() -> Double.NaN));
  => 4.5
```

as with any other `Optional`

`getAsDouble()` instead of `get()`

takes `DoubleSupplier` instead of `Supplier`

Summarizing Statistics

- `summaryStatistics()` method performs many calculations on the stream:
  - `getCount()` : gives number of values (`long`)
  - `getAverage()` : returns an average value (`double`) or `0` if the stream is empty
  - `getSum()` : returns a sum (`double` or `long`)
  - `getMin()` : returns the smallest number (`double`, `int` or `long`)
    - if the stream is empty returns the largest numeric value based on the type
  - `getMax()` : returns the largest number (`double`, `int` or `long`)
    - if the stream is empty returns the smallest numeric value based on the type

```
// example using summarizing statistics
var intStream = IntStream.of(7, 2, -4, 11, 27);
IntSummaryStatistics stats = intStream.summaryStatistics();
System.out.println(stats.getCount());
  => 5
System.out.println(stats.getAverage());
  => 8.6
System.out.println(stats.getSum());
  => 43
System.out.println(stats.getMin());
  => -4
System.out.println(stats.getMax());
  => 27
```