ORACLE

Lesson 2-6: Stream Interface: Terminal Operations

Terminal Operations

- Terminates the pipeline of operations on the stream
- Only at this point is any processing performed
  - This allows for optimisation of the pipeline
    - Lazy evaluation
    - Merged/fused operations
    - Elimination of redundant operations
    - Parallel execution
- Generates an explicit result or a side effect

Matching Elements

- `findFirst(Predicate p)`
  - The first element that matches using the given `Predicate`
- `findAny(Predicate p)`
  - Works the same way as `findFirst()`, but for a parallel stream
- `boolean allMatch(Predicate p)`
  - Whether all the elements of the stream match using the `Predicate`
- `boolean anyMatch(Predicate p)`
  - Whether any of the elements of the stream match using the `Predicate`
- `boolean noneMatch(Predicate p)`
  - Whether no elements match using the `Predicate`

Collecting Results

- `collect(Collector c)`
  - Performs a mutable reduction on the stream
- `toArray()`
  - Returns an array containing the elements of the stream

Numerical Results

Object Stream

- `count()`
  - Returns how many elements are in the stream
- `max(Comparator c)`
  - The maximum value element of the stream using the `Comparator`
  - Returns an `Optional`, since the stream may be empty
- `min(Comparator c)`
  - The minimum value element of the stream using the `Comparator`
  - Returns an `Optional`, since the stream may be empty

Numerical Results

Primitive Type Streams (IntStream, DoubleStream, LongStream)

- `average()`
  - Return the arithmetic mean of the stream
  - Returns an `Optional`, as the stream may be empty
- `sum()`
  - Returns the sum of the stream elements

Iteration

- `forEach(Consumer c)`
  - Performs an action for each element of this stream
- `forEachOrdered(Consumer c)`
  - Like `forEach`, but ensures that the order of the elements (if one exists) is respected when used for a parallel stream

- Use with caution!
  - Encourages non-functional (imperative) programming style
  - More detail in week 3

Folding A Stream

Creating A Single Result From Multiple Input Elements

- `reduce(BinaryOperator accumulator)`
  - Performs a reduction on the stream using the `BinaryOperator`
  - The `accumulator` takes a partial result and the next element, and returns a new partial result
  - Returns an `Optional`
  - Two other versions
    - One that takes an initial value (does not return an `Optional`)
    - One that takes an initial value and `BiFunction` (equivalent to a fused map and reduce)

Section 6

Summary

- Terminal operations provide results or side effects
- Many types of operation available
- Ones like `reduce` and `collect` need to be looked at in more detail
  - We'll do this in week 3


ORACLE®