

17 Lambdas and Streams

Objectives

In this chapter you'll:

- Learn various functional-programming techniques and how they complement object-oriented programming.
- Use lambdas and streams to simplify tasks that process sequences of elements.
- Learn what streams are and how stream pipelines are formed from stream sources, intermediate operations and terminal operations.
- Create streams representing ranges of `int` values and random `int` values.
- Implement functional interfaces with lambdas.
- Perform on `IntStreams` intermediate operations `filter`, `map`, `mapToObj` and `sorted`, and terminal operations `forEach`, `count`, `min`, `max`, `sum`, `average` and `reduce`.
- Perform on `Streams` intermediate operations `distinct`, `filter`, `map`, `mapToDouble` and `sorted`, and terminal operations `collect`, `forEach`, `findFirst` and `reduce`.
- Process infinite streams.
- Implement event handlers with lambdas.

Outline

1. 17.1 Introduction
2. 17.2 Streams and Reduction
 1. 17.2.1 Summing the Integers from 1 through 10 with a `for` Loop
 2. 17.2.2 External Iteration with `for` Is Error Prone
 3. 17.2.3 Summing with a Stream and Reduction
 4. 17.2.4 Internal Iteration
3. 17.3 Mapping and Lambdas
 1. 17.3.1 Lambda Expressions
 2. 17.3.2 Lambda Syntax
 3. 17.3.3 Intermediate and Terminal Operations
4. 17.4 Filtering
5. 17.5 How Elements Move Through Stream Pipelines
6. 17.6 Method References
 1. 17.6.1 Creating an `IntStream` of Random Values
 2. 17.6.2 Performing a Task on Each Stream Element with `forEach` and a Method Reference
 3. 17.6.3 Mapping Integers to String Objects with `mapToObj`
 4. 17.6.4 Concatenating Strings with `collect`
7. 17.7 `IntStream` Operations
 1. 17.7.1 Creating an `IntStream` and Displaying Its Values
 2. 17.7.2 Terminal Operations `count`, `min`, `max`, `sum` and `average`
 3. 17.7.3 Terminal Operation `reduce`
 4. 17.7.4 Sorting `IntStream` Values

8. [17.8 Functional Interfaces](#)
9. [17.9 Lambdas: A Deeper Look](#)
10. [17.10 Stream<Integer> Manipulations](#)
 1. [17.10.1 Creating a Stream<Integer>](#)
 2. [17.10.2 Sorting a Stream and Collecting the Results](#)
 3. [17.10.3 Filtering a Stream and Storing the Results for Later Use](#)
 4. [17.10.4 Filtering and Sorting a Stream and Collecting the Results](#)
 5. [17.10.5 Sorting Previously Collected Results](#)
11. [17.11 Stream<String> Manipulations](#)
 1. [17.11.1 Mapping Strings to Uppercase](#)
 2. [17.11.2 Filtering Strings Then Sorting Them in Case-Insensitive Ascending Order](#)
 3. [17.11.3 Filtering Strings Then Sorting Them in Case-Insensitive Descending Order](#)
12. [17.12 Stream<Employee> Manipulations](#)
 1. [17.12.1 Creating and Displaying a List<Employee>](#)
 2. [17.12.2 Filtering Employees with Salaries in a Specified Range](#)
 3. [17.12.3 Sorting Employees By Multiple Fields](#)
 4. [17.12.4 Mapping Employees to Unique-Last-Name Strings](#)
 5. [17.12.5 Grouping Employees By Department](#)
 6. [17.12.6 Counting the Number of Employees in Each Department](#)
 7. [17.12.7 Summing and Averaging Employee Salaries](#)

13. 17.13 Creating a Stream<String> from a File
14. 17.14 Streams of Random Values
15. 17.15 Infinite Streams
16. 17.16 Lambda Event Handlers
17. 17.17 Additional Notes on Java SE 8 Interfaces
18. 17.18 Wrap-Up
 1. Summary
 2. Self-Review Exercises
 3. Answers to Self-Review Exercises
 4. Exercises