ORACLE

Lesson 2: Introduction To The Streams API

Simon Ritter
Java Technology Evangelist

Lesson Agenda

- Introduction to functional programming concepts
- Elements of a Stream
- Streams of objects and primitive types
- Stream sources in JDK 8
- The Stream interface: Intermediate operations
- The Stream interface: Terminal operations
- The Optional class

ORACLE

Lesson 2-1: Introduction to Functional Programming Concepts

Imperative Programming

Names And Values

- Use variables as an association between names and values
- Use sequences of commands
  - Each command consists of an assignment
  - Can change a variable's value
  - Form is <variable_name> = <expression>
  - Expressions may refer to other variables
    - Whose value may have been changed by preceding commands
  - Values can therefore be passed from command to command
  - Commands may be repeated through loops

Functional Programming

Names And Values

- Based on structured function calls
- Function call which calls other functions in turn (composition)

  `<function1>(<function2>(<function3> … ) … )`
- Each function receives values from, and passes values back the calling function
- Names are only used as formal parameters
  - Once value is assigned it can not be changed
- No concept of a command, as used in imperative code
  - Therefore no concept of repetition

Names And Values

- Imperative
  - The same name may be associated with different values
- Functional
  - A name is only ever associated with one value

Execution Order

- Imperative
  - Values associated with names can be changed
  - The order of execution of commands forms a contract
    - If it is changed, the behaviour of the application may change
- Functional
  - Values associated with names cannot be changed
  - The order of execution does not impact the results
  - There is no fixed execution order

Repetition

- Imperative
  - Values associated with names may be changed by commands
  - Commands may be repeated leading to repeated changes
  - New values may be associated with the same name through repetition (loops)
- Functional
  - Values associated with names may not be changed
  - Repeated changes are achieved by nested function calls
  - New values may be associated with the same name through recursion

Functions As Values

- Functional programming allows functions to be treated as values
  - This is why Lambda expressions were required in JDK 8
  - To make this much simpler than anonymous inner classes

Section 1

Summary

- Imperative and functional approaches are very different
- Imperative
  - Values associated with names can be changed
  - Order of execution is defined as a contract
  - Repetition is explicit and external
- Functional
  - Values associated with names are set once and cannot be changed
  - Order of execution is not defined
  - Repetition is through the use of recursion


ORACLE®