Java Stream API and Functional Programming

Rahman Usta
rahmanusta@kodedu.com

Agenda

- Imperative programming
- Declarative programming
- Functional programming
- Functional Interfaces
- Java Stream API
  - filter, map, flatMap, collect, etc.

Imperative programming

- Developer decides every detail
- Developer asks question 'How to'?
- It is procedurel
  - Some flow, some loops, some temp. variables

Declarative programming

- Details are hidden from developers
  - Library cares about 'How to'
- More abstraction
  - Less code
  - Correctness
- Developer asks mostly 'What to'?

Functional programming

- A subset of declarative programming
  - FP is declarative
- A series of transformations
  - by using higher-order functions
- Composing functions
- Lazy evaluation

Higher-order functions

- Functions accept function(s) as argument
- Functions can return function

Functions in Java

- There is no direct component called function in Java
- Lambda expressions or Method references are kinda 'functions'

Functional Interfaces

- functions in Java expressed as Functional Interfaces (java.util.function.*)
- Lambda expressions or method references are translated to instance of FIs.

Java Stream API

Provides higher-order functions two write FP style coding in Java.

- Stream API
  - filter
  - map
  - flatMap
  - collect
  - sort, distinct, etc.

Getting Stream<T> instance

- Collection#stream
- Stream.* (Stream.of)
- IntStream.*, LongStream.*, ..
- Arrays.stream()
- etc.

Stream#filter()

- Stream<T> --(Filter)-→ Stream<T>
- Filter is a predicate
- Next Stream contains only matched elements.
- java.util.function.Predicate is used

Stream#map()

- Stream<T> --(Mapping)-→ Stream<R>
- Can transform value of data
  - data → data*2
- Can transform type of data, or both.
  - data → data*2.0
- java.util.function.Function is used

Stream#collect()

- Converts Stream to Non-Stream
  - List, Set, Map, int, double, etc.

Thanks