OCP Java SE 8

Streams

Collections

- Externe Iteration
  - wird vom Entwickler in Applikationscode programmiert
- Interne Iteration
  - wird nicht vom Entwickler selbst programmiert, sondern wird im Framework realisiert . Man übergibt nur die auszuführende Aktion

Collections

- Externe Iteration
  - läuft sequenziell ab
  - Funktionalität und die Iteration werden gemischt
- Interne Iteration
  - Parallelverarbeitung möglich, solange die Aktionen für die einzelnen Elemente voneinander unabhängig sind.
  - Reihenfolge der Verarbeitung kann von der sequenziellen abweichen
  - Ist flexibel

Funktionelle Programmierung

- Man kann Funktionalität in Form von Sourcecode als Parameter übergeben (»Code as Data«) und an beliebiger Stelle ausführen.

Streams

- Pipeline für Datenströme
- Für den Einsatz von Lambdas konzipiert
- Kein wahlfreier Zugriff möglich (nur auf das erste Element)
- Lazy
- Können unendlich sein
- Können parallel verarbeitet werden

Intermediate Operations

- z.B. Filtern, Transformieren und Sortieren
- Verarbeitungsschritte, die sich einfach hintereinander schalten lassen
- es erfolgen noch keine Berechnungen, sondern lediglich die Abläufe werden beschrieben
- zustandslos und zustandsbehaftet

Intermediate Operations

- zustandslos (stateless)
  - Aktion wird auf jedes Element des Streams unabhängig von den anderen angewandt
    z.B. filter
- zustandsbehaftet (stateful)
  - Aktion erfordert die Kenntnis der anderen Elemente im Stream (oder zumindest eines Teils davon)

Intermediate Operations (zustandslos)

- filter()
  - Filtert alle Elemente aus dem Stream heraus, die nicht dem übergebenen Predicate<T> genügen
- map()
  - Transformiert Elemente mithilfe einer Function<T,R> vom Typ T auf solche mit dem Typ R
- flatMap()
  - Bildet verschachtelte Streams als einen flachen Stream ab.
- peek()
  - Führt eine Aktion für jedes Element des Streams aus.

Intermediate Operations (zustandsbehaftet)

- distinct()
  - Entfernt alle gemäß der Methode equals(Object) als Duplikate erkannte Elemente aus einem Stream.
- sorted()
  - Sortiert die Elemente eines Streams gemäß einem Sortierkriterium basierend auf einem Comparator<T>.
- limit()
  - Begrenzt die maximale Anzahl der Elemente eines Streams auf einen bestimmten Wert. Dies ist eine Short-circuiting Operation.
- skip()
  - Überspringt die ersten n Elemente eines Streams.

Terminal Operations

- lösen die Ausführung der durch die Intermediate Operations beschriebenen Verarbeitungsschritte aus

Optional

- Ein Container für ein optionales Ergebnis
- Kann einen Wert enthalten oder leer sein

Terminal Operations

- forEach()
  - Führt eine Aktion für jedes Element des Streams aus.
- toArray()
  - Überträgt die Elemente aus dem Stream in ein Array.
- collect()
  - Überträgt die Elemente aus dem Stream in eine Collection.
- reduce()
  - Verbindet die Elemente eines Streams.

Terminal Operations

- min()
  - Ermittelt das Minimum der Elemente eines Streams gemäß einem Sortierkriterium basierend auf einem Comparator<T>.
- max()
  - Ermittelt das Maximum der Elemente eines Streams gemäß einem Sortierkriterium basierend auf einem Comparator<T>.
- count()
  - Zählt die Anzahl an Elementen in einem Stream.
- anyMatch()
  - Prüft, ob es mindestens ein Element gibt, das die Bedingung eines Predicate<T> erfüllt. Dies ist eine Short-circuiting Operation.

Terminal Operations

- allMatch()
    - Prüft, ob alle Elemente die Bedingung eines Predicate<T> erfüllen. Dies ist eine Short-circuiting Operation, die allerdings abbricht, wenn sie das erste Gegenbeispiel gefunden hat.
- noneMatch()
    - Prüft, ob keines der Elemente die Bedingung eines Predicate<T> erfüllt. Dies ist eine Short-circuiting Operation.
- findFirst()
    - Liefert das erste Element des Streams, falls es ein solches gibt. Dies ist eine Short-circuiting Operation.
- findAny()
    - Liefert ein beliebiges Element, falls es ein solches gibt. Dies ist eine Short-circuiting Operation und kann manchmal günstiger sein als findFirst(), wenn es wirklich nur darum geht, einen beliebigen Treffer zu erhalten.

Terminal Operations

- sum()
- average()

Collectors

- joining()
  - Fasst Einträge vom Typ String zusammen
- groupingBy()
  - Fasst Einträge vom Typ String zusammen

Collectors

- partitioningBy()
  - Unterteilt die Eingabedaten basierend auf einer Realisierung eines Predicate < T > in zwei Partitionen