Java Stream Api - продолжение

Бесконечные стримы и другие изменения Java 9

Stream.iterate(начальное_условие, выражение_генерации)

```
Stream<Integer> streamFromIterate = Stream.iterate(1, n -> n + 1)
```

https://vertex-academy.com/tutorials/ru/java-9-stream-uluchsheniya/

Много примеров

https://habr.com/ru/company/luxoft/blog/270383/

Группировка

https://habr.com/ru/post/348536/

Параллельные потоки

https://metanit.com/java/tutorial/10.9.php

Collectors

Метод	Описание
toList, toCollection, toSet	представляют стрим в виде списка, коллекции или множества
toConcurrentMap, toMap	позволяют преобразовать стрим в map
averagingInt, averagingDouble, averagingLong	возвращают среднее значение
summingInt, summingDouble, summingLong	возвращает сумму
summarizingInt, summarizingDouble, summarizingLong	возвращают SummaryStatistics с разными агрегатными значениями
partitioningBy	разделяет коллекцию на две части по соответствию условию и возвращает их как Map<Boolean, List>
groupingBy	разделяет коллекцию на несколько частей и возвращает Map<N, List<T>>
mapping	дополнительные преобразования значений для сложных Collector'ов

examples

examples

examples

examples

Collectors

Условие: Дана коллекция чисел Arrays.asList(1, 2, 3, 4), рассмотрим работу collect и toArray с ней

Задача	Код примера	Результат
Получить сумму нечетных чисел	numbers.stream().collect(Collectors.summingInt(((p) -> p % 2 == 1? p: 0)))	4
Вычесть от каждого элемента 1 и получить среднее	numbers.stream().collect(Collectors.averagingInt((p) -> p — 1))	1.5
Прибавить к числам 3 и получить статистику	numbers.stream().collect(Collectors.summarizingInt((p) -> p + 3))	IntSummaryStatistics{cou sum=22, min=4, average= max=7}
Разделить числа на четные и нечетные	numbers.stream().collect(Collectors.partitioningBy((p) -> p % 2 == 0))	{false=[1, 3], true=[2, 4]}