Java

Stream API

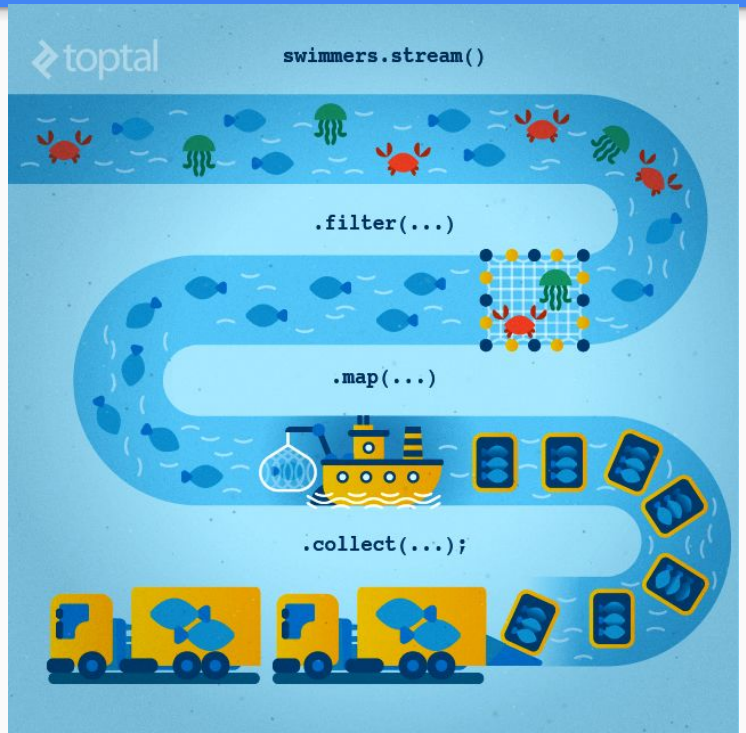
Функциональное программирование

- Программирование с математическими функциями
- Функции:
  - Тот же результат для тех же аргументов
  - Отсутствие побочных эффектов, работа с неизменяемыми данными

forEach

- Выполняется по порядку для каждого объекта
- Принимает Consumer
- Для Map принимает BiConsumer

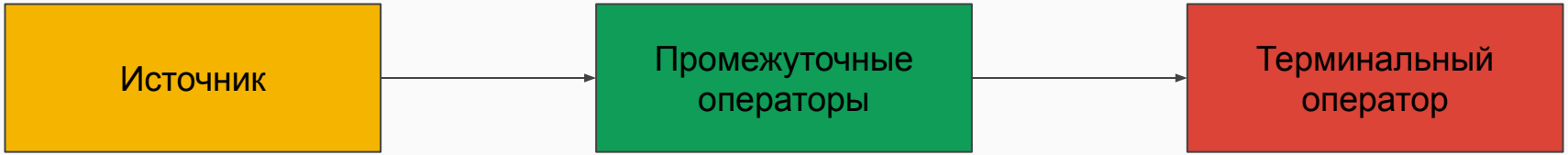
Stream API



Stream API

- Работа с данными в функциональном стиле
- Укорачивает код
- Упрощает работу с данными
- Упрощает распараллеливание процесса

Структура Stream



Объект Stream

- Описывает поток данных
- Содержит в себе промежуточные и терминальные операторы
- Обработка запускается только тогда когда вызван терминальный оператор
- Переиспользовать поток нельзя

Источники

- Пустой Stream.empty()
- Из указанных элементов Stream.of(obj1, obj2…)
- Из коллекций - list.stream()
- Из Map - map.entrySet().stream()
- Из массива Arrays.stream(array)
- Из итератора StreamSupport.stream(iterable.splitarator(), false)

Spliterator

- Splitting Iterator
- Позволяет проходить и разбивать последовательности
- tryAdvance - позволяет обрабатывать последовательность по частям. Если данных больше нет - возвращает false
- trySplit - Разбивает последовательность на две.
- estimatedSize - примерный размер. Integer.MAX_VALUE если бесконечный или слишком долго считать

NumberStream

- IntStream
- DoubleStream
- LongStream
- Методы:
  - sum
  - average
  - summaryStatistics

Промежуточные операции

- Обрабатывают данные
- Возвращают обработанный поток
- Может быть несколько

filter

- Принимает Predicate
- Если возвращает true то элемент будет оставлен в потоке, иначе будет убран
- Стоит использовать класс Objects

map()

- Позволяет заменить входящий элемент на другой
- Принимает Function

flatMap()

- Так же преобразует один тип в другой, но другой тип - поток
- Позволяет объединять потоки

sorted()

- Позволяет упорядочить поток
- Для работы нужно одно из двух:
  - Элементы должны реализовывать интерфейс Comparable
  - Нужно передать реализацию Comparator

distinct()

- Получить уникальные элементы
- Для работы используется equals

Другие

- peek() - совершить действие не меняя тип
- unsorted() - отменить сортировку
- limit() - ограничить количество
- dropWhile() / skipWhile() - выкидывать пока выполняется условие
- mapToInt/Double/Long - перевод в IntStream и тд
- skip() - пропустить N элементов
- parallel() - выполнять параллельно

Терминальные операции

- При вызове запускается весь поток
- В конце получается либо результат либо просто завершение операции

collect

- Принимает реализацию Collector
- Стандартные реализации можно взять из класса Collectors:
  - toList - в список
  - toMap - принимает две функции перевода: для ключа и для значения
  - toCollection - в абстрактную коллекцию
  - counting - подсчёт
  - joining - в строку, принимает разделитель

reduce

- Приводит поток к одному объекту
- Принимает функцию с двумя аргументами одного типа и возвращает этот же тип
- Пример - так можно реализовать сумму

find

- findFirst() - находит первый элемент потока
- findAny() - находит любой элемент потока

match

- allMatch - true если все элементы удовлетворяют предикату
- anyMatch - true если хотя бы один
- noneMatch - ни один не удовлетворяет

Другие

- count - счётчик количества элементов
- toArray - перевод в массив
- spliterator - получение сплитератора

Optional

- Контейнер для одного значения
- Optional.empty() - пустой
- Optional.of() - создать с элементом. Не может содержать null.
- Optional.ofNullable() - создать с элементом. Если передан null то будет empty.

Основные методы Optional

- get(): T - получение элемента. Кидает исключение если empty
- isPresent() / isEmpty() - проверка есть ли значение
- ifPresent(Consumer) - Если есть элемент то выполнить действие
- orElse(T): T / orElseGet(Supplier<T>): T - если нет элемента то вернуть другой
- or(Supplier<Optional<T>>): T - если пустой optional то создать другой
- orElseThrow() - если пустой то бросить исключение

Stream-методы в Optional

- filter - так же как в потоках, но над одним элементом
- map / flatMap - преобразовать в Optional другого типа (даже если пустой)
- stream - создать поток из одного элемента

Недостатки Stream API

- Могут ухудшить читаемость кода
- Сложнее отладка