Промежуточные операторы

Промежуточные операторы

Чтобы выполнить последовательность операций над элементами источника данных и агрегировать их результаты, понадобятся **три части**:

1. Сам источник данных
2. Промежуточные операции
3. Терминальные операции

Типы операций над стримом

Операции над стримом делятся на два типа:

1. Промежуточные операции
2. Терминальные операции

Промежуточные операции

Промежуточные операции работают с элементами стрима — изменяют или фильтруют их каким-либо образом.

Промежуточные операции могут быть объединены в цепочку, где каждая промежуточная операция возвращает стрим для дальнейшей обработки.

Терминальные операции

В отличие от промежуточных, терминальные операции выполняют окончательную обработку стрима.
Терминальная операция в стриме бывает только одна: она замыкает стрим и возвращает окончательный результат для дальнейшей обработки.

Важно:

1. Если стрим не закрыт терминальной операцией, то он не выполнится. Это свойство называют «ленивостью» стримов
2. Можно написать сколько угодно промежуточных операций, но без терминальной операции вы не получите результат

Промежуточные операторы

- **Метод map()** применяет переданную функцию к каждому элементу стрима и возвращает стрим изменённых элементов
- **Метод filter()**. Предположите, что вам нужно проверить каждый элемент на соответствие какому-то условию и оставить только те элементы, которые соответствуют этому условию. Для этого можно использовать именно данный метод
- **Метод flatMap()** поможет, если у вас есть сложная структура. Идея состоит в том, что вы «выравниваете» каждый элемент из сложной структуры, состоящей из нескольких внутренних элементов, в «плоский» поток, состоящий только из этих внутренних элементов, — отсюда и такое название
- **Метод distinct()** возвращает стрим из уникальных элементов данного стрима. Например, если в стриме будут повторяющиеся элементы

Промежуточные операторы

- С помощью **методов equals()** и **hashCode()** определяется уникальность элементов (аналогично вставкам в HashMap)
- **Метод sorted()** позволяет с помощью промежуточных операций не только преобразовать элементы стрима, но и отсортировать их в нужном порядке. Этот метод работает аналогично методу **Collections.sort()**. Это означает, что объекты, которые находятся в стриме, должны имплементировать интерфейс Comparable
- **Метод peek()** позволяет применить какой-либо метод к элементу стрима, тем самым изменяя его. Этот метод очень хорошо подходит для логирования. Крайне не рекомендуется использовать его для чего-то ещё, так как сами объекты внутри стрима ожидаемо не изменяются
- С помощью операции **limit()** можно ограничить количество элементов стрима. Размер исходного источника это не меняет

Запись стримов

Стримы принято записывать в виде цепочек вызовов:

```
Stream.of(1, 2, 3)
    .map(i -> i * 2)
    .forEach(System.out::println);
```

Изображение: Даниил Пилипенко

Вывод

В этом видео вы познакомились с промежуточными операторами в Java и разобрали основные примеры их применения.