**Comparator в Stream API**

В Stream Api, которые появились в Java 8, компаратор позволяет упрощать работу над элементами стримов. Например, нам нужна последовательность случайных чисел от 0 до 999 включительно:

Supplier<Integer> randomizer = () -> **new** Random().nextInt(1000);

Stream.generate(randomizer)

.limit(10)

.sorted(Comparator.naturalOrder())

.forEach(e -> System.out.println(e));

Мы могли бы и остановиться, но есть задачки поинтереснее. Например, нужно подготовить Map, где ключ — id сообщения. При этом мы хотим отсортировать эти ключи, чтобы ключи шли по порядку, от меньшего к большему. Начнём с такого кода:

Map<Integer, Message> collected = Arrays.stream(messages)

.sorted(Comparator.comparing(o -> o.id))

.collect(Collectors.toMap(o -> o.id, o -> o));

Нам вернут тут на самом деле HashMap. А как мы знаем, она не гарантирует какой либо порядок. Поэтому наши отсортированные по ID записи просто потеряли порядок. Не хорошо. Придётся изменить немного наш коллектор:

Map<Integer, Message> collected = Arrays.stream(messages)

.sorted(Comparator.comparing(o -> o.id))

.collect(Collectors.toMap(o -> o.id, o -> o,

(oldValue, newValue) -> oldValue, TreeMap::**new**));

Код стал выглядеть несколько страшнее, но задача теперь решена правильно благодаря явному указанию реализации карты TreeMap.