Stream API

Способы создания стримов:

1. Классический: Создание стрима из коллекции - collection.**stream**()

Collection<String> collection = Arrays.asList("a1", "a2", "a3");

Stream<String> streamFromCollection = collection.stream();

2. Создание стрима из значений - **Stream.of**(*значение1*,… *значениеN*)

Stream<String> streamFromValues = Stream.of("a1", "a2", "a3");

3. Создание стрима из массива - **Arrays.stream**(*массив*)

String[] array = {"a1","a2","a3"};

Stream<String> streamFromArrays = Arrays.stream(array);

4. Создание стрима из файла (каждая строка в файле будет отдельным элементом в стриме) - **Files.lines**(*путь\_к\_файлу*)

Stream<String> streamFromFiles = Files.lines(Paths.get("file.txt"))

5. Создание стрима из строки - «строка».**chars**()

IntStream streamFromString = "123".chars()

6. С помощью Stream.builder - Stream.**builder**().**add**(...)....**build**()

Stream.builder().add("a1").add("a2").add("a3").build()

7. Создание параллельного стрима - collection.**parallelStream**()

Stream<String> stream = collection.parallelStream();

8. Создание бесконечных стрима с помощью Stream.iterate - **Stream.iterate**(*начальное\_условие*, *выражение\_генерации*)

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

9. Создание бесконечных стрима с помощью Stream.generate - **Stream.generate**(*выражение\_генерации*)

Stream<String> streamFromGenerate = Stream.generate(() -> "a1")

Example:

**public class** StreamDemo {  
 **public static void** main(String[] args) {  
 List<String> myList = Arrays.*asList*(**"a1"**, **"a2"**, **"b1"**, **"c2"**, **"c1"**);  
 myList  
 .stream() // открывает поток для выполнения  
 .filter(s -> s.startsWith(**"c"**)) // фильтр  
 .map(String::toUpperCase)  
 .sorted() // сортировка  
 .forEach(System.***out***::println);  
 }  
}

Описание методов Stream:

.filter(s -> s.startsWith(**"c"**)) - Отфильтровывает записи, возвращает только записи, соответствующие условию.

условия:

(s -> s.startsWith(**"c"**)) – делает поиск значений начинающихся с символа «С». Поиск можно задавать из нескольких символов.

(s -> s.startsWith(**"c", 1**)) – делает такой же поиск, цифра после запятой указывает с какой позиции начинается поиск.

(s -> s.equals(**"a1"**)) – ищет точное совпадение.

(s -> s.equalsIgnoreCase(**"A1"**)) – ищет совпадения игнорируя регистр.

(s -> s.contains(**"re"**)) – ищет совпадения символов, совпадения могут быть в любом месте строки.

myList.stream().filter(**"a1"**::equals).count() – ищет совпадения и возвращает количество совпадений.

.skip(2) - Позволяет пропустить N первых элементов