Stream API. Работа. Коллекциями.

Практика 36 - Айрат Хасьянов

Минутка Паттерна

- Кто желает?

На разогрев

- Как найти максимальный элемент коллекции, использую конкретный компаратор и Stream API?

```
Optional<Integer> maxValue =
testValues.stream().max(Integer::compareTo);
System.out.println("MaxValue="+maxValue);
```

На подогрев

- Как найти максимальный элемент коллекции, использую конкретный компаратор и Stream API, если среди элементов может быть null?

```
Optional<Integer> maxValue =
testValues.stream().max(Integer::compareTo);
System.out.println("MaxValue="+maxValue);
```

```
Optional<Integer> maxValueNotNull =  testValuesNull.stream().filter((p) ->
p !=  null).max(Integer::compareTo);
System.out.println("maxValueNotNull="+maxValueNotNull);
```

Разберём Домашний тьюториал

Первая книга с тэгом Java

```
public Article getFirstJavaArticle() {
    for (Article article : articles) {
        if (article.getTags().contains("Java")) {
            return article;
        }
    }
    return null;
}
```

```
public Optional<Article> getFirstJavaArticle() {
    return articles.stream()
        .filter(article -> article.getTags().contains("Java"))
        .findFirst();
    }
```

Все книги тэгом Java

```
public List<Article> getAllJavaArticles() {
    List<Article> result = new ArrayList<>();
    for (Article article : articles) {
        if (article.getTags().contains("Java")) {
            result.add(article);
        }
    }
    return result;
}
```

```
public List<Article> getAllJavaArticles() {
    return articles.stream()
        .filter(article -> article.getTags().contains("Java"))
        .collect(Collectors.toList());
    }
```

Сгруппируем статьи по автору

```
public Map<String, List<Article>> groupByAuthor() {
    return articles.stream()
        .collect(Collectors.groupingBy(Article::getAuthor));
}
```

Сгруппируем статьи по автору

```
public Set<String> getDistinctTags() {
    return articles.stream()
        .flatMap(article -> article.getTags().stream())
        .collect(Collectors.toSet());
}
```

Вопросы

- Конвейер в Stream API содержит источник, сколько-то промежуточных и одну терминальную операцию.
- Что может быть на месте терминальной операции?
    - Stream.forEach, Stream.reduce, Stream.collect, Stream.max, Stream.min, Stream.count
- Что может играть роль источника в StreamAPI?
    - Коллекция, массив, генерирующая функция или канал ввода/вывода.
- В чем различие между терминальными и конвейерными операциями?
- Что значит, что потоки в Java ленивые?

Преобразование одной коллекции в другую (map)

```
Stream<Book> map = articles.stream()
        .filter(p -> p.getPages() > 200)
        .map(article -> new Book(article));
```

```
Stream<Book> map = articles.stream()
        .filter(p -> p.getPages() > 200)
        .map(Book::new);
```

```
List<Book> books = articles.stream()
        .filter(p -> p.getPages() > 200)
        .map(Book::new)
        .collect(Collectors.toList());
```

```
List<Book> books = articles.stream()
        .filter(p -> p.getPages() > 200)
        .map(Book::new)
        .collect(new Collector<Book, List<Book>>() {});
```

Преобразование в конкретную коллекцию

```
List<Book> books = articles.stream()
        .filter(p -> p.getPages() > 200)
        .map(Book::new)
       .collect(Collectors.toCollection(ArrayList::new));
```

```
List<Book> books = articles.stream()
        .parallel()
        .filter(p -> p.getPages() > 200)
       .sequential()
        .map(Book::new)
       .collect(Collectors.toCollection(ArrayList::new));
```

Где нужно параллельно, где нужно - последовательно

Использование reduce

```
String reduced2 = articles.stream()
        .reduce((acc, item) -> acc + " " + item)
        .get();
```

public T Optional<T>.get()

java.util.function.BinaryOperator.apply<T>{
  …
   T apply(T,T);
  …
}

Использование reduce

```
String reduced = articles.stream()
        .reduce("", (acc, article) -> acc + " " + article);
```

```
String reduced = articles.stream()
        .filter(article -> !article.startsWith(" "))
        .reduce("", (acc, article) -> acc + " " + article);
```

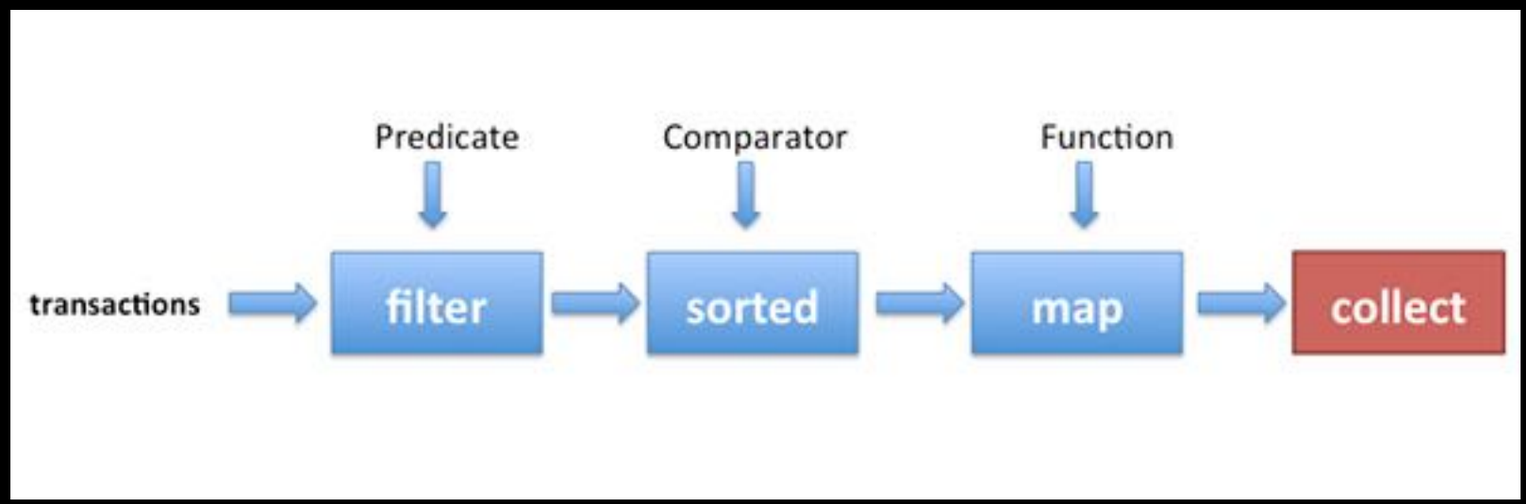
Задачи

- *Решить, хотим ли мы отменить крепостное право!*
- Всем жителям деревни дать одну фамилию;
- Сгруппировать/отфильтровать своих крепостных по возрасту, полу и имени;
- Преобразуйте всех ваших крепостных в свободных фермеров (используя map и лямбда);
- Создайте текст с именами всех жен свободных фермеров (каждую с новой строки);

Домашнее задание

- Создать коллекцию текстов из файлов;
- Преобразовать каждый текст в последовательность слов;
- Преобразовать коллекцию текстов в коллекцию слов с частотой употребления каждого слова;
- Отфильтровать ненужные слова не на кириллице;
- Найти наиболее и наименее употребительное слово;
- Сформировать лексический почерк исходных текстов (выбрать набор частот/слов из всего множества и выдать их в виде короткого текста, который потом можно использовать для сверки авторства)

Иллюстрация




```
List<Integer> transactionsIds =
    transactions.stream()
                .filter(t -> t.getType() == Transaction.GROCERY)
                .sorted(comparing(Transaction::getValue).reversed())
                .map(Transaction::getId)
                .collect(toList());
```