Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

ИТиП

Лабораторная работа №8

Выполнил:

студент группы БВТ2203

Петухов Г. В.

Москва 2023

**Задание для выполнения лабораторной работы:**

Вам необходимо разработать приложение, которое считывает данные из

исходного источника (например, файл, база данных или сетевой ресурс),

применяет к ним различные операции с использованием Stream API, и

сохраняет результаты в новый источник данных.

1. Создайте аннотацию @DataProcessor, которая будет использоваться

для пометки методов обработки данных.

2. Создайте класс DataManager, который будет отвечать за

многопоточную обработку данных. Этот класс должен иметь методы:

o registerDataProcessor(Object processor): Регистрирует объект-

обработчик данных с аннотацией @DataProcessor.

o loadData(String source): Загружает данные из исходного

источника.

o processData(): Запускает многопоточную обработку данных,

применяя методы с аннотацией @DataProcessor с

использованием Stream API.

o saveData(String destination): Сохраняет обработанные данные в

новый источник.

3. Создайте несколько классов, представляющих различные

обработчики данных, и пометьте их аннотацией @DataProcessor.

Например, можно создать классы для фильтрации, трансформации

и агрегации данных.

4. Используйте многопоточность из java.util.concurrent для

эффективной обработки данных параллельно.

5. Протестируйте ваше приложение, загрузив данные из исходного

источника, применив различные обработчики с помощью Stream

API, и сохраните результаты в новый источник.

**Реализация**

1) Для начала создадим классы обработчики данных. Всего три класса Filter (Рис. 1), который фильтрует список слов и убирает слова меньше заданной длины. Transformer (Рис. 2), который заменяет слова на другое слово. Aggregator (Рис. 3), который соединяет все слова в одно большое слово.



Рисунок 1 – Листинг Filter.

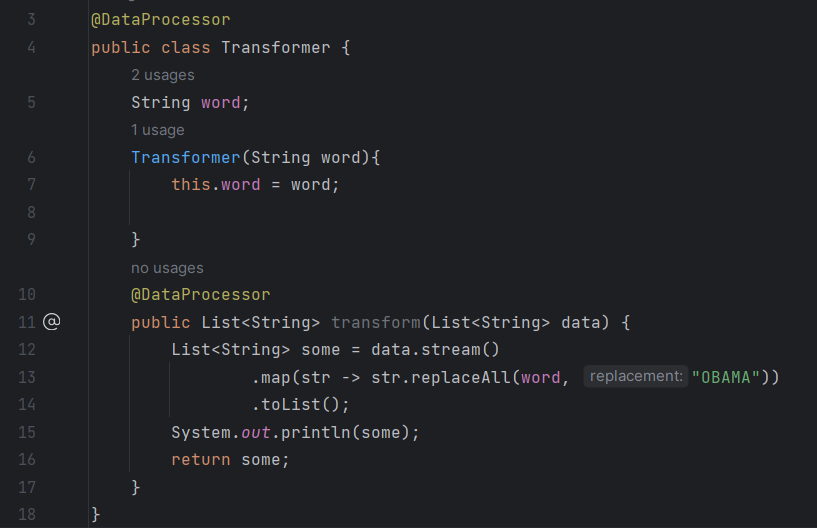


Рисунок 2 – Листинг Transformer.

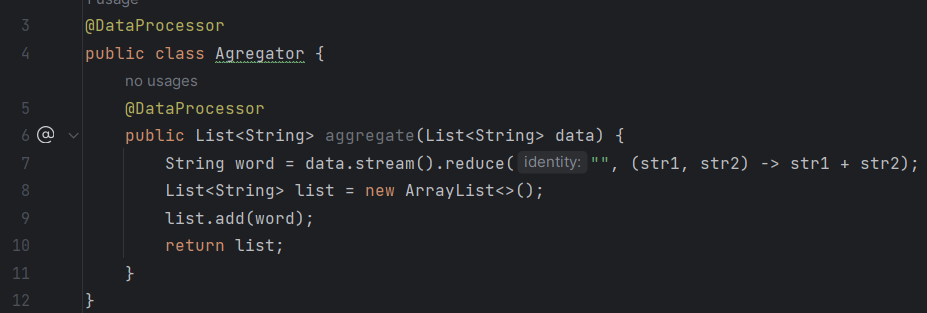


Рисунок 3 – Листинг Aggregator.

2) Далее создадим саму аннотацию DataProcessor (Рис. 4). Добавим метааннотации, которые указывают на то что наша аннотация может быть приписана над классами или методами, а так же то, что она доступна в Runtime.

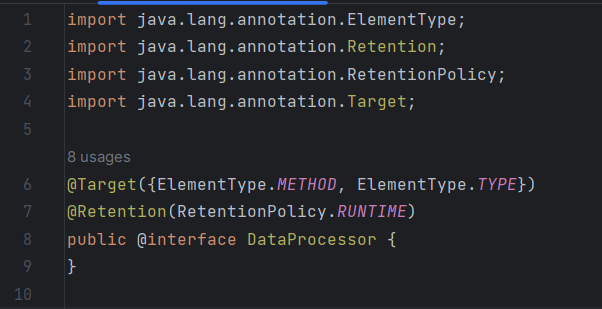


Рисунок 4 – Листинг аннотации DataProcessor.

3) Создадим класс DataManger (Рис. 5 -8), реализуем необходимые методы. Метод registerDataProcessor (Рис. 5) проверяет аннотирован ли класс и если да, то добавляет его в список объектов. Метод loadData (Рис. 6) загружает строку из файла, далее разделяет слова по пробелам и сохраняет в список.

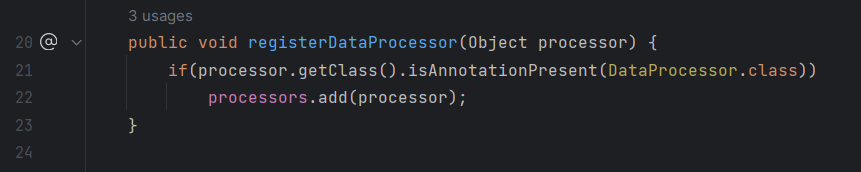


Рисунок 5 – Листинг метода registerDataProcessor.

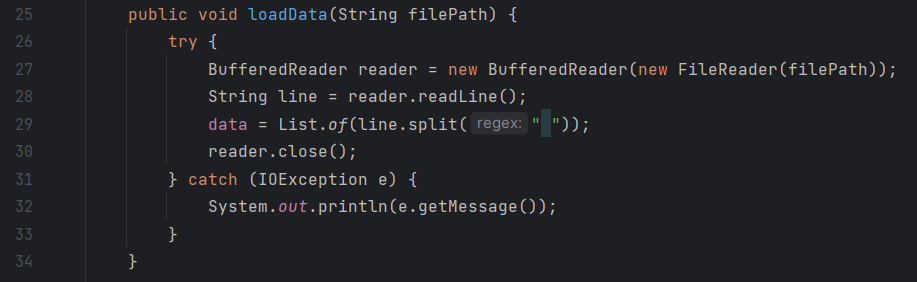


Рисунок 6 – Листинг loadData.

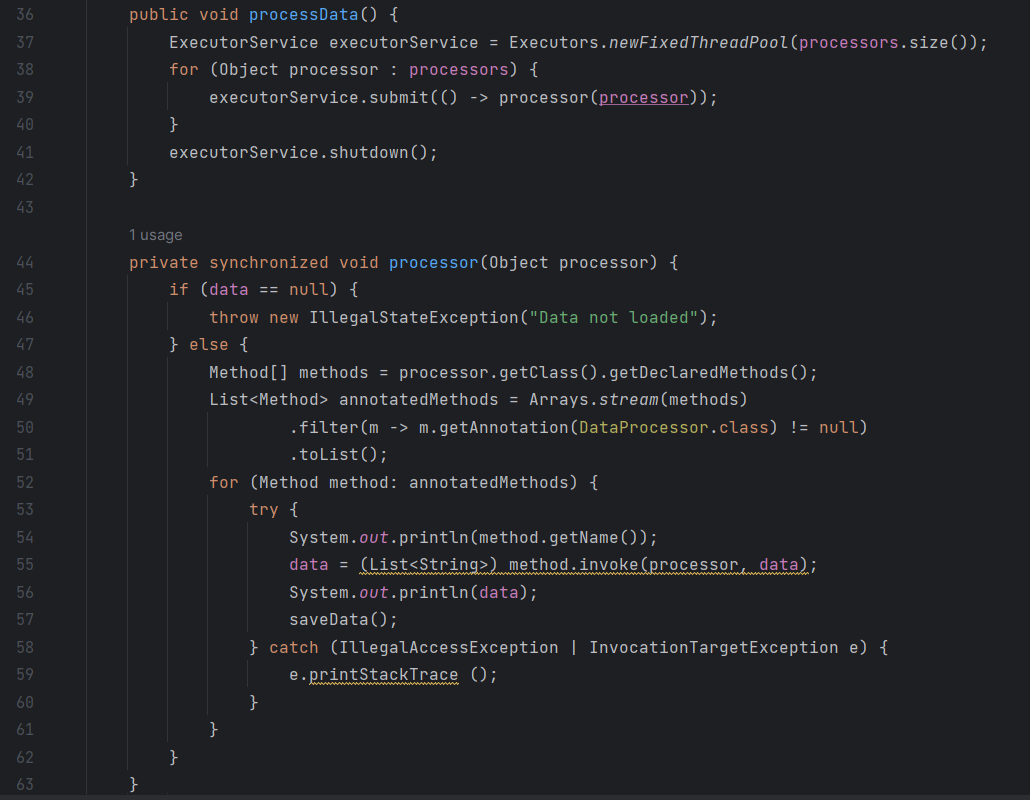


Рисунок 7 – Листинг ProcessData

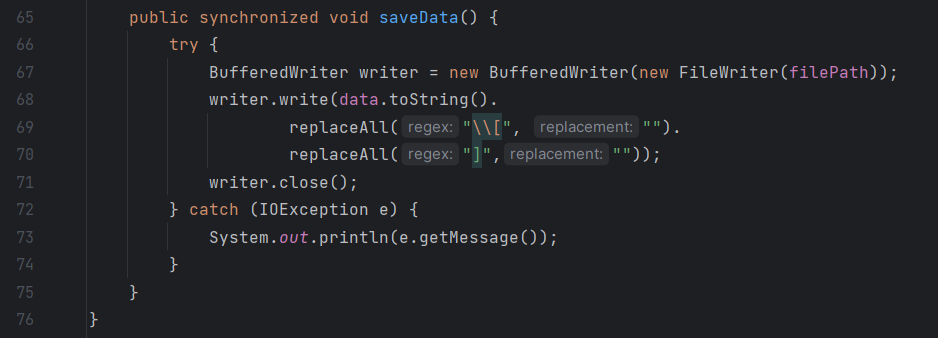


Рисунок 8 – Листинг saveData.

4) Для проверки создадим класс Main (Рис. 9) , где в main создадим экземпляр класса DataManager и вызовем метод processData, который запустит потоки. Содержимое файла 1.txt (Рис. 10), после чего над его данным производятся манипуляции и сохраняются в файл 2.txt (Рис. 11).

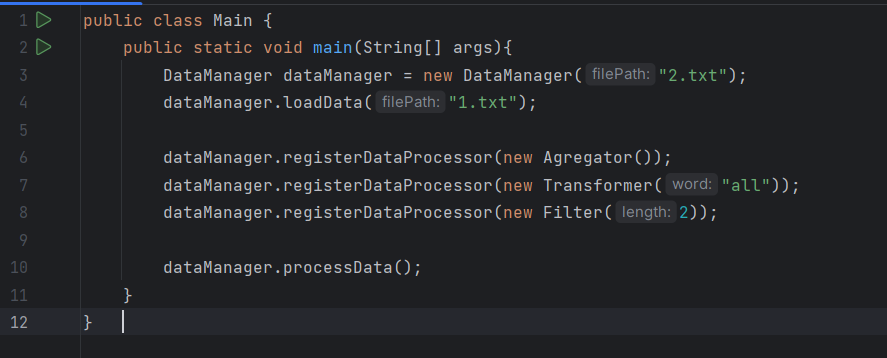


Рисунок 9 – Листинг Main

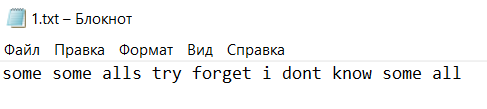


Рисунок 10 – 1.txt

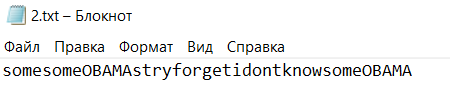


Рисунок 11 – 2.txt