МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

ЛАБОРАТОРНАЯ РАБОТА № 3

по курсу   
Большие данные

Выполнил студент  
 группы 6404-010302D

Дёмина Н.В.

Самара 2025

Задание

Выполнить следующие задания из набора заданий репозитория https://github.com/ververica/flink-training-exercises:

1. RideCleanisingExercise;
2. RidesAndFaresExercise;
3. HourlyTipsExerxise;
4. ExpiringStateExercise.

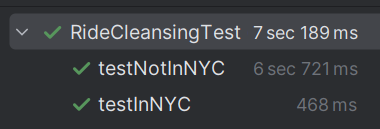
Ход выполнения лабораторной работы

1. RideCleanisingExercise

**Задание:** очистить поток событий Taxi Ride, удалив события, которые начинаются или заканчиваются за пределами Нью-Йорка.

**Решение:** используется метод GeoUtils.isInNYC(), который по координатам определяет находится ли локация в Нью-Йорке.

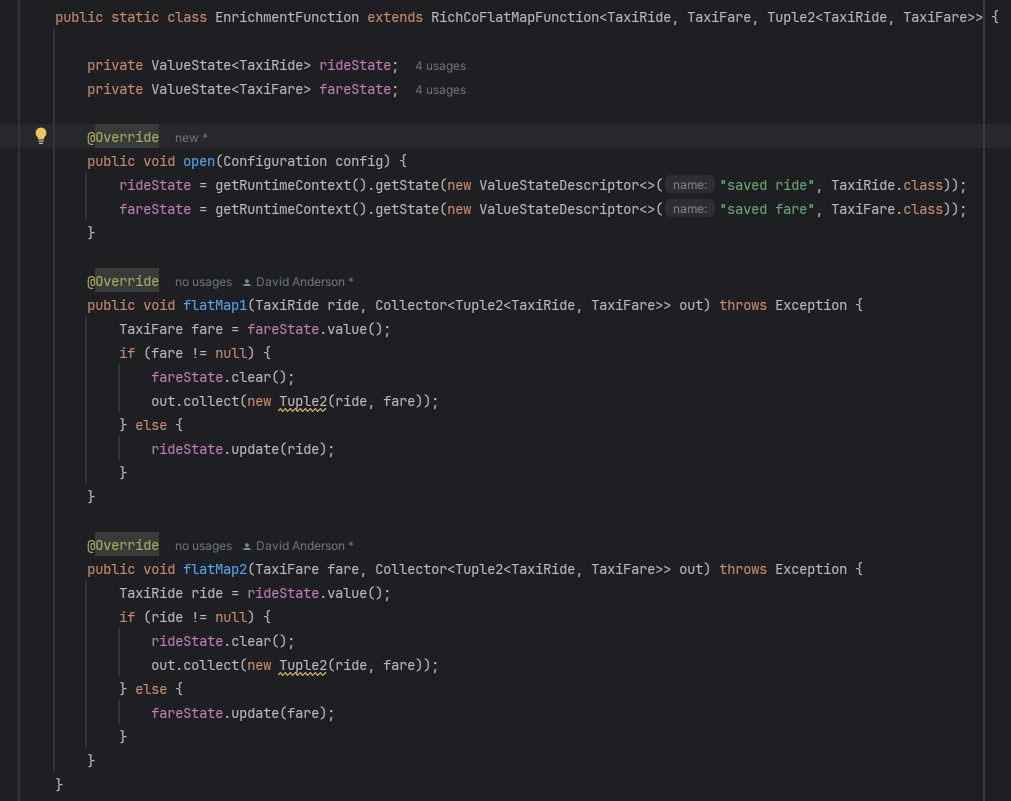


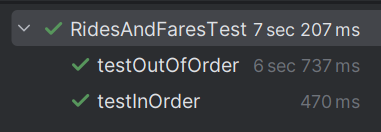


1. RidesAndFaresExercise

**Задание:** необходимо объединить TaxiRide и TaxiFare записи для каждой поездки.

**Решение:** когда поступает TaxiRide, метод flatMap1() проверяет, есть ли уже информация о стоимости. Если есть, он создает кортеж и отправляет его дальше. Если нет, он сохраняет информацию о поездке. Аналогично, когда поступает TaxiFare, метод flatMap2() проверяет, есть ли информация о поездке, и так же создает кортеж, если информация о поездке уже доступна.

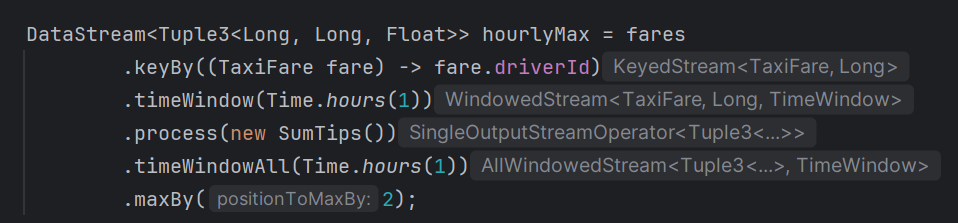


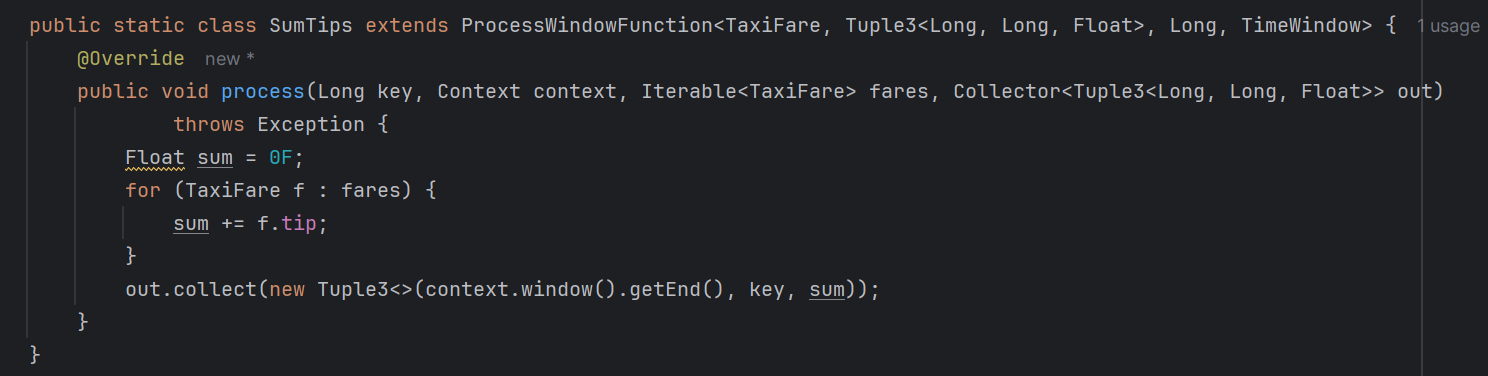


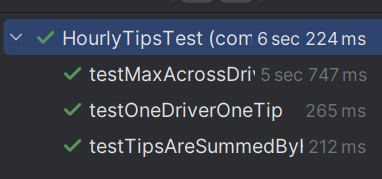
1. HourlyTipsExerxise

**Задание:** необходимо за каждый час определить водителя, получившего наибольшее количество чаевых.

**Решение:** выполняется группировка по водителям, устанавливается окно в 1 час, подсчитывается общее количество чаевых за час и выполняется поиск максимального значения чаевых.







1. ExpiringStateExercise

**Задание:** задача аналогична RidesAndFaresExercise, но в более сложной реализации с гибкой обработкой несоответствий и задержек в поступлении данных.

**Решение:** используется KeyedCoProcessFunction с поддержкой таймеров. В processElement1(), если информация о стоимости доступна, то удаляется связанный с ней таймер и результат отправляется в out, иначе регистрируется таймер на время события поездки для последующей обработки случая, когда соответствующая стоимость не приходит. В processElement2() реализуется аналогичное для поездок. Метод onTimer() вызывается при срабатывании таймера, здесь если есть значение в fareState/rideState, то оно отправляется в специальный выходной коллектор unmatchedFares/unmatchedRides.



