

Оглавление



Предисловие	13
Благодарности	15
О книге	17
Форум liveBook	19
От издательства.....	19
Об авторах	20

ЧАСТЬ I. ЗНАКОМСТВО

Глава 1. Знакомство со стриминговыми системами	22
Что такое потоковая обработка?	23
Примеры событий	23
Примеры стриминговых систем	24
Стриминговые системы и реальное время	25
Как работает стриминговая система	26
Приложения	26
Серверные службы	27
Системы пакетной обработки	29
Внутри системы пакетной обработки данных	30
Системы потоковой обработки	31
Внутри системы потоковой обработки	32
Преимущества многофазной архитектуры	33
Многофазная архитектура в системах пакетной и потоковой обработки ...	34
Сравнение систем	35

6 Оглавление

Эталонная система обработки событий	36
Итоги	37
Упражнение	37
Глава 2. Привет, стриминговые системы!	38
Начальнику нужен современный пункт оплаты	39
Вначале были запросы HTTP... и ничего не вышло	40
ЭйДжей и Миранда берут паузу, чтобы подумать	41
ЭйДжей размышляет о стриминговых системах	42
Сравнение серверных служб и потоковой обработки	43
Очереди: фундаментальная концепция	45
Передача данных в очередях	46
Потоковый фреймворк (вернее, его начало)	47
Обзор фреймворка Streamwork	48
Подробнее о ядре Streamwork	49
Основные стриминговые концепции	50
Подробнее о концепциях	51
Последовательность выполнения стримингового задания	52
Первое стриминговое задание	53
Выполнение задания	59
Ход выполнения задания	60
Внутри ядра	61
Перемещение событий	65
Жизненный цикл элемента данных	66
Краткий обзор концепций стриминга	67
Итоги	68
Упражнения	68
Глава 3. Параллелизация и группировка данных	69
Датчик генерирует больше событий	70
Даже в потоковых системах непросто добиться обработки в реальном времени	71
Новые концепции: параллелизм важен	72
Новые концепции: параллелизм данных	73
Новые концепции: независимость выполнения данных	74
Новые концепции: параллелизм задач	75
Параллелизм данных и параллелизм задач	76
Параллелизм и многозадачность	77
Параллелизация задания	78

Параллелизация компонентов	79
Параллелизация источников	80
Результат выполнения	81
Параллелизация операторов	82
Результат выполнения	83
События и экземпляры	84
Упорядочение событий	85
Группировка событий	86
Случайная группировка	87
Случайная группировка: внутренний механизм	88
Группировка по значениям полей	89
Группировка по значениям полей: внутренний механизм	90
Выполнение группировки событий	91
Заглянуть в ядро: диспетчер событий	92
Применение группировки по значениям полей в задании	93
Упорядочение событий	94
Сравнение поведения группировок	95
Итоги	96
Упражнения	96
Глава 4. Граф потока	97
Система обнаружения мошеннических действий с кредитными картами ...	98
Подробнее о системе обнаружения мошеннических действий с кредитными картами	99
Процедура обнаружения мошеннических действий	100
Потоковая обработка не всегда прямолинейна	101
Механизм работы системы	102
Подробнее о задании обнаружения мошеннических действий	103
Новые концепции	104
Предшествующие и последующие компоненты	105
Разветвление и объединение потока	106
Графы, направленные графы и DAG	107
DAG в системах потоковой обработки	108
Все новые концепции на одной странице	109
Разветвление потока к анализаторам	110
Что происходит внутри ядра	111
Проблема эффективности	112
Разветвление с несколькими потоками	113

8 Оглавление

Что происходит внутри ядра (еще раз)	114
Коммуникации между компонентами по каналам	115
Несколько каналов	116
Объединение потока в агрегаторе оценок	117
Объединение потоков в ядре	118
Краткий обзор разновидности объединения потоков — соединения	119
Система в целом	120
Графы и стриминговые задания	121
Примеры систем	122
Итоги	123
Упражнения	124
Глава 5. Семантика доставки	125
Требования к задержке в системе обнаружения мошеннических действий	126
Возвращаемся к заданию обнаружения мошеннических действий	127
О точности	128
Частичный результат	129
Новое стриминговое задание для контроля за использованием системы	130
Новое задание контроля использования системы	131
Требования к заданию контроля	132
Новые концепции: количество доставок и обработок	133
Новая концепция: семантика доставки	134
Выбор семантики	135
«Не более одного»	136
Задание обнаружения мошеннических действий	137
«Не менее одного»	138
«Не менее одного» с подтверждением	139
Отслеживание событий	140
Управление сбоями при обработке событий	141
Раннее обнаружение потерянных событий	142
Код подтверждения в компонентах	143
Новая концепция: контрольные точки	143
Новая концепция: состояние	145
Контрольные точки в задании контроля за использованием системы для семантики «не менее одного»	145
Контрольные точки и функции управления состоянием	147
Код управления состоянием в компоненте источника транзакций	148

Ровно один или фактически один?	149
Вспомогательная концепция: идемпотентные операции	150
Наконец, «ровно один»	151
Код управления состоянием в компоненте анализатора использования системы	152
Повторное сравнение семантик доставки	153
Итоги	154
Упражнения	154
Что дальше?	154
Глава 6. Краткий обзор стриминговых систем и взгляд в будущее	155
Компоненты стриминговых систем	156
Параллелизация и группировка событий	157
DAG и стриминговые задания	158
Семантика (гарантия) доставки	159
Семантика доставки в системе обнаружения мошеннических действий с кредитными картами	159
Что дальше?	161
Оконные вычисления	162
Соединение данных в реальном времени	163
Обратное давление	164
Вычисления с состоянием и без состояния	165
ЧАСТЬ II. ДВИЖЕМСЯ ДАЛЬШЕ	
Глава 7. Оконные вычисления	168
Сегментация данных в реальном времени	169
Анализ задачи	170
Анализ задачи (продолжение)	171
Два разных контекста	172
Окна в задании обнаружения мошеннических действий	173
Что именно называется окном?	174
Подробнее об окнах	175
Новая концепция: оконная стратегия	176
Фиксированные окна	177
Фиксированные окна в анализаторе оконного расстояния	178
Обнаружение попыток мошенничества в фиксированном временном окне	179
Фиксированные окна: время и количество	180

10 Оглавление

Скользящие окна	181
Скользящие окна: анализатор оконного расстояния	182
Обнаружение мошеннических действий благодаря использованию скользящих окон	183
Сеансовые окна	184
Сеансовые окна (продолжение)	185
Обнаружение мошеннических действий благодаря использованию сеансовых окон	186
Обзор оконных стратегий	187
Разбиение потока событий на наборы данных	188
Оконные системы: концепция или реализация	189
Другой взгляд	190
Хранилища пар «ключ — значение»	191
Реализация анализатора оконного расстояния	192
Время события и другие вехи	193
Водяной знак окна	194
Поздние события	195
Итоги	196
Упражнения	197
Глава 8. Операции соединения	198
Соединение данных выбросов в реальном времени	199
Задание контроля выбросов, версия 1	200
Преобразователь данных о выбросах	201
Точность становится проблемой	202
Обновленное задание контроля выбросов	203
Все внимание на соединение	204
Еще раз: что такое соединение?	205
Как работает стриминговое соединение	206
Соединение (join) потоков — разновидность объединения (fan-in)	207
События автомобилей и события температуры	208
Таблица как материализованное представление стриминга	209
Материализация событий автомобилей менее эффективна	210
Проблемы с целостностью данных	211
Проблемы с оператором соединения	212
Внутреннее соединение	213
Внешнее соединение	214
Внутренние и внешние соединения	215

Разные типы соединений	216
Внешние соединения в стриминговых системах	217
Новая проблема: ненадежное соединение	218
Оконные соединения	218
Соединение двух таблиц вместо соединения потока с таблицей	219
Снова о материализации представлений	221
Итоги	222
Глава 9. Обратное давление	223
Надежность критична	224
Обзор системы	225
Усовершенствование стриминговых заданий	226
Новые концепции: мощность, использование и резерв мощности	227
Подробнее об использовании и резерве мощности	228
Новая концепция: обратное давление	229
Измерение использования мощности	230
Обратное давление в ядре Streamwork	231
Обратное давление в ядре Streamwork: распространение	232
Стриминговое задание с обратным давлением	233
Обратное давление в распределенных системах	234
Новая концепция: водяные знаки обратного давления	239
Другой подход к управлению отстающими экземплярами: отбрасывание событий	240
Когда отбрасывание событий допустимо?	241
Обратное давление как возможный симптом	242
Останов и возобновление работы могут привести к пробуксовке	243
Решение проблемы пробуксовки	244
Итоги	245
Глава 10. Вычисления с состоянием	246
Миграция стриминговых заданий	247
Компоненты с состоянием в задании контроля за использованием системы	248
Снова о состоянии	249
Состояния в разных компонентах	250
Данные состояния и временные данные	251
Компоненты с состоянием и без состояния: код	252
Источник с состоянием и оператор в задании контроля за использованием системы	253

Состояния и контрольные точки	254
Создание контрольных точек: сложность выбора момента времени	255
Событийный отсчет времени	256
Создание контрольных точек с использованием событий контрольных точек	257
Событие контрольной точки обрабатывается исполнителями экземпляров	259
Путь события контрольной точки через задание	260
Создание контрольных точек с использованием событий контрольных точек на уровне экземпляров	261
Синхронизация событий контрольных точек	263
Загрузка контрольных точек и обратная совместимость	264
Хранилище контрольных точек	265
Компоненты с состоянием и компоненты без состояния	267
Ручное управление состоянием экземпляров	268
Лямбда-архитектура	269
Итоги	270
Упражнения	271
Глава 11. Продвинутые концепции в стриминговых системах	272
Это действительно все?	273
Оконные вычисления	274
Основные виды окон	275
Соединение данных в реальном времени	276
SQL и стриминговые соединения	277
Внутренние и внешние соединения	278
Неожиданности в стриминговых системах	279
Обратное давление: замедление источников или предшествующих компонентов	280
Другой подход к управлению отстающими экземплярами: отбрасывание событий	281
Обратное давление может быть симптомом постоянной проблемы	282
Компоненты с состоянием и контрольные точки	282
Событийный отсчет времени	284
Компоненты с состоянием и компоненты без состояния	285
У вас все получилось!	286
Ключевые концепции, рассмотренные в книге	287

Последовательность выполнения стримингового задания

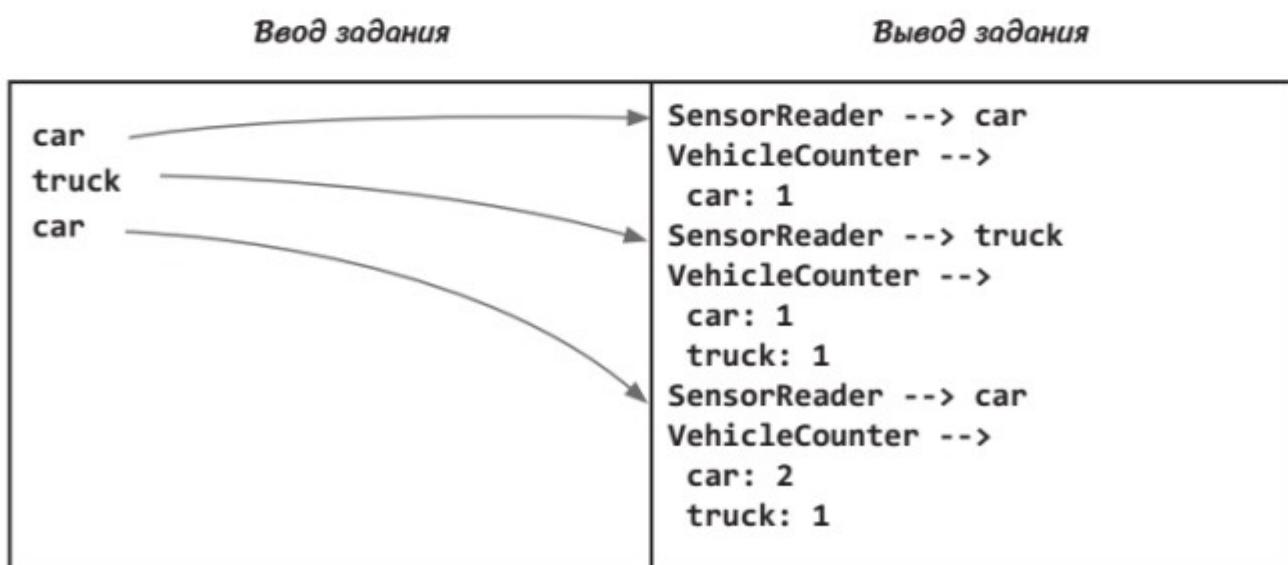
С концепциями, рассмотренными на двух предыдущих страницах, можно составить визуализацию стримингового задания для подсчета автомобилей, состоящего из двух компонентов и одного потока между ними, как показано справа.



- Блок чтения данных получает данные с датчика и сохраняет события в очереди. Это источник.
- Счетчик автомобилей отвечает за подсчет автомобилей в потоке. Это оператор.
- Непрерывное перемещение данных от источника к оператору — поток событий автомобилей.

Блок чтения данных с датчика становится началом задания, а счетчик автомобилей — его концом. Линия, соединяющая блок чтения (источник) со счетчиком автомобилей (оператором), представляет поток типов автомобилей (событий), проходящий от блока чтения данных к счетчику автомобилей.

В этой главе такая система будет описана более подробно. Она будет выполняться на ваших локальных компьютерах с двумя терминалами: один получает ввод пользователя (левый столбец), а другой — вывод задания (правый столбец).



Первое стриминговое задание

Стриминговое задание создается средствами Streamwork API и включает следующие шаги.

1. Создание класса задания.
2. Построение источника.
3. Построение оператора.
4. Соединение компонентов.

Первое стриминговое задание: создание класса события

Событие — один фрагмент данных в потоке, который должен обрабатываться заданием. Во фреймворке Streamwork класс API `Event` отвечает за хранение или инкапсуляцию пользовательских данных. Аналогичные концепции существуют и в других стриминговых системах.

В задании каждое событие представляет один тип автомобилей. Для простоты будем считать, что каждый тип представляет собой строку (например, `car` или `truck`). В нашем примере будет использоваться класс события `VehicleEvent`, производный от класса `Event` из API. Каждый объект `VehicleEvent` содержит информацию об автомобиле, которую можно получить вызовом функции `getData()`.

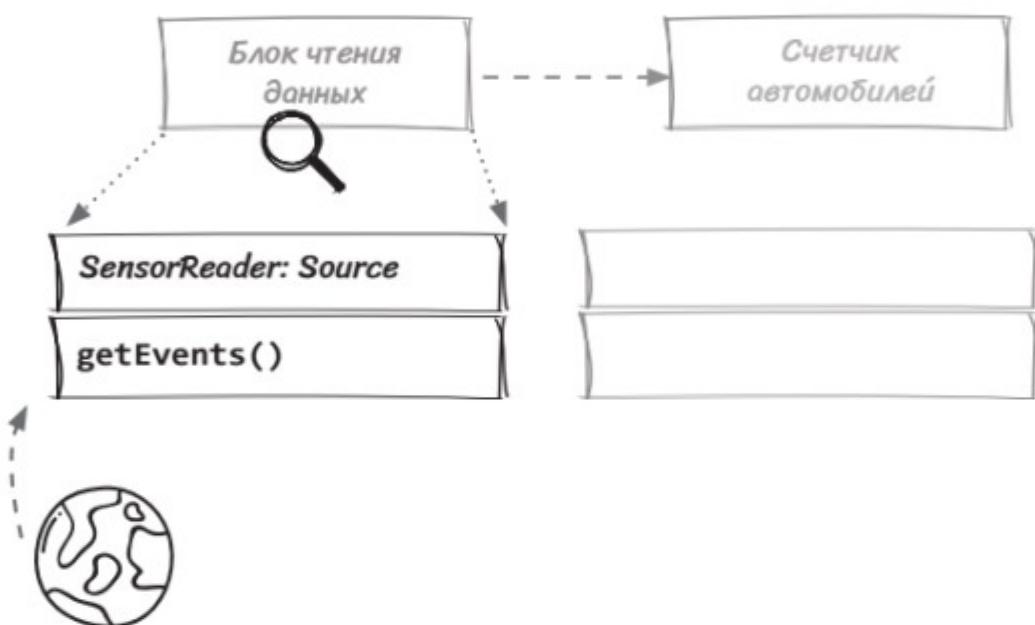
```
public class VehicleEvent extends Event {
    private final String vehicle; ← Внутренняя строка для
                                обозначения типа автомобиля.

    public VehicleEvent(String vehicle) {
        this.vehicle = vehicle; ← Конструктор получает vehicle в виде
                                строки и сохраняет значение.
    }

    @Override
    public String getData() {
        return vehicle; ← Получает данные, хранящиеся в событии.
    }
}
```

Первое стриминговое задание: источник данных

Источником (source) называется компонент, который вводит внешние данные в стриминговую систему. Земной шар на следующей диаграмме обозначает данные, внешние по отношению к заданию. В стриминговом задании блок чтения данных от датчика вводит данные, полученные от локального порта, в систему.



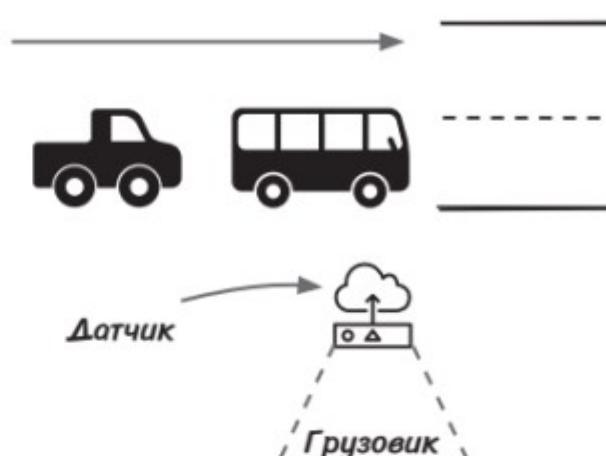
Во всех стриминговых фреймворках присутствует API, дающий возможность задавать для источников данных логику, которая представляет интерес только для вас. Во всех API источников данных присутствует некая разновидность *перехватчика жизненного цикла* (lifecycle hook), который будет вызываться для получения внешних данных. В этой точке код выполняется фреймворком.

Что такое перехватчик жизненного цикла?

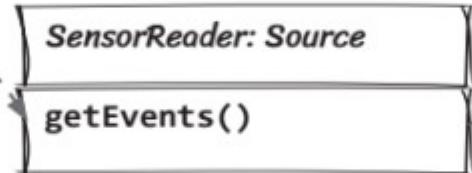
Перехватчиками жизненного цикла называются методы, которые вызываются по определенному повторяющему шаблону фреймворком, которому они принадлежат. Как правило, эти методы позволяют разработчикам настроить поведение приложения в фазах жизненного цикла фреймворка, на основе которого строится приложение. В случае фреймворка Streamwork имеется перехватчик жизненного цикла (или метод), называемый `getEvents()`. Он многократно вызывается фреймворком, чтобы вы могли получить внешние данные. Перехватчики жизненного цикла позволяют разработчикам писать логику, которая для них важна, и поручить рутинную работу фреймворку.

Первое стриминговое задание: источник данных (продолжение)

В этом задании блок чтения данных датчика будет читать события. В нашем упражнении вы будете моделировать датчик на мосту, самостоятельно создавая события и передавая их на открытый порт вашего компьютера, прослушиваемый стриминговым заданием. Блок чтения получает данные о типах автомобилей, отправленные в порт, и передает их потоковому заданию — так выглядит бесконечный (или неограниченный) поток событий.



1. `getEvents()` содержит логику чтения данных от датчика, определяемую пользователем.



2. События направляются в исходящий поток (очередь).

Java-код класса `SensorReader` выглядит так:

```
public class SensorReader extends Source {
    private final BufferedReader reader;
    public SensorReader(String name, int port) {
        super(name);
        reader = setupSocketReader(port);
    }

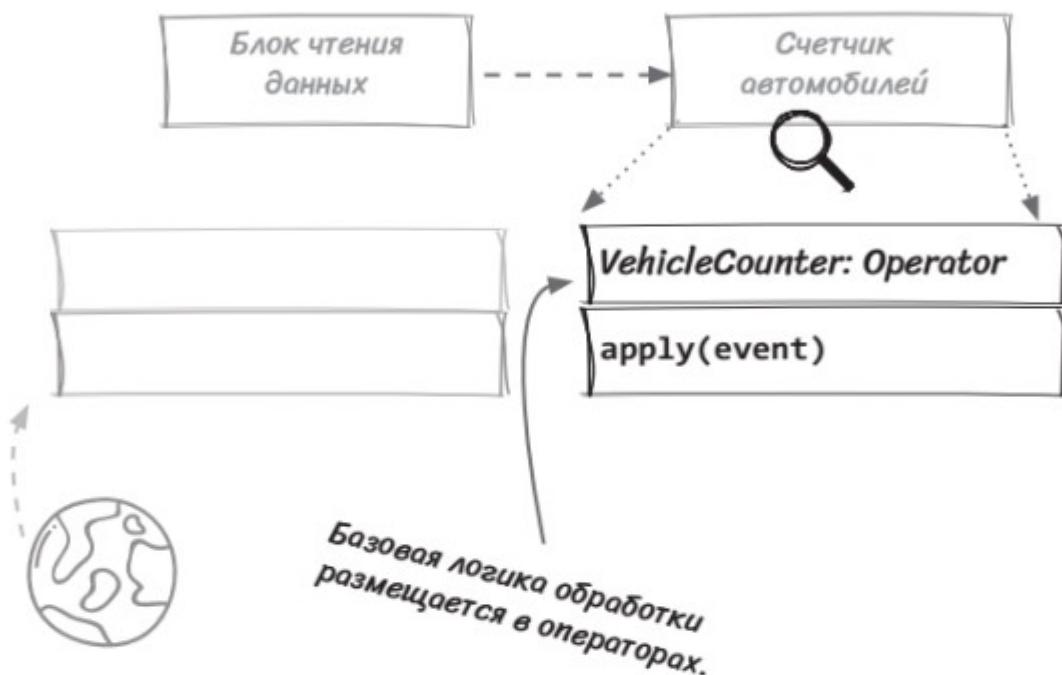
    @Override
    public void getEvents(List<Event> eventCollector) {
        String vehicle = reader.readLine();
        eventCollector.add(new VehicleEvent(vehicle));
        System.out.println("SensorReader --> " + vehicle);
    }
}
```

Перехватчик жизненного цикла стриминговой системы выполняет логику, определяемую пользователем.



Первое стриминговое задание: оператор

Вся пользовательская логика обработки содергится в операторах. Они отвечают за получение входящих событий для обработки и генерирование исходящих событий; следовательно, у них есть как ввод, так и вывод. Вся логика обработки данных в стриминговых системах обычно размещается в компонентах операторов.



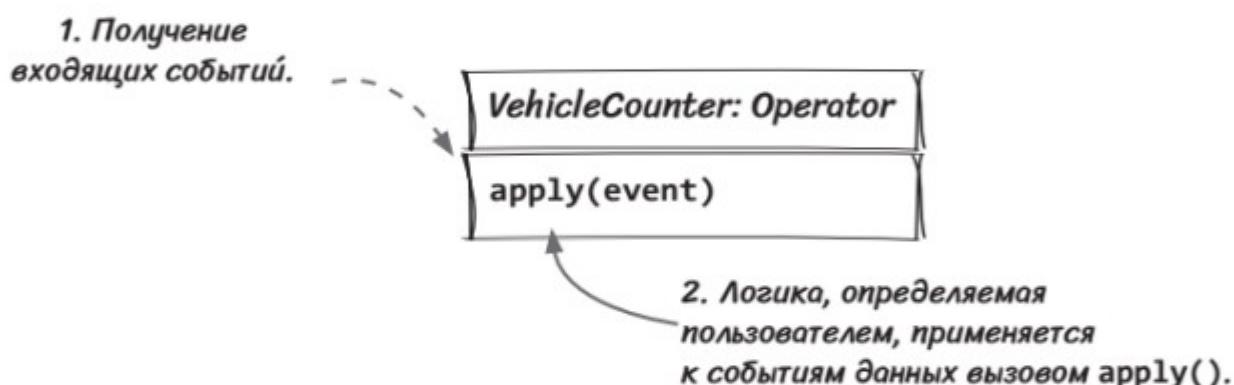
Для простоты мы ограничились одним источником и одним оператором. Рассматриваемая реализация счетчика автомобилей только подсчитывает автомобили, а затем регистрирует текущее значение счетчика в системе. Другой (возможно, лучший) способ реализации системы основан на направлении данных в новый поток. Тогда регистрация результатов может выполняться в дополнительном компоненте, который следует за счетчиком автомобилей. Как правило, в задании используются компоненты, которые имеют только одну функцию.

Кстати говоря, Сид занимает должность технического директора. Иногда он бывает старомодным, но он очень умен и интересуется новыми технологиями.

Первое стриминговое задание: оператор (продолжение)

В компоненте `VehicleCounter` карта `<vehicle, count>` используется для хранения счетчиков типов автомобилей в памяти. Она обновляется соответствующим образом при получении нового события. В стриминговом задании счетчик автомобилей представляет собой оператор, который подсчитывает события. Этот оператор завершает задание и не генерирует вывод для последующих операторов.

<i>Ключ (Vehicle)</i>	<i>Значение (Count)</i>
автомобиль	2
грузовик	1
микроавтобус	1



```
public class VehicleCounter extends Operator {
    private final Map<String, Integer> countMap =
        new HashMap<String, Integer>();
    public VehicleCounter(String name) {
        super(name);
    }
    @Override
    public void apply(Event event, List<Event> collector) {
        String vehicle = ((VehicleEvent)event).getData();
        Integer count = countMap.getOrDefault(vehicle, 0);
        count += 1;           ← Увеличение счетчика
        countMap.put(vehicle, count); ← Сохранение счетчика
        System.out.println("VehicleCounter --> ");
        printCountMap();       ← Вывод текущего
    }                         значения счетчика
}
```

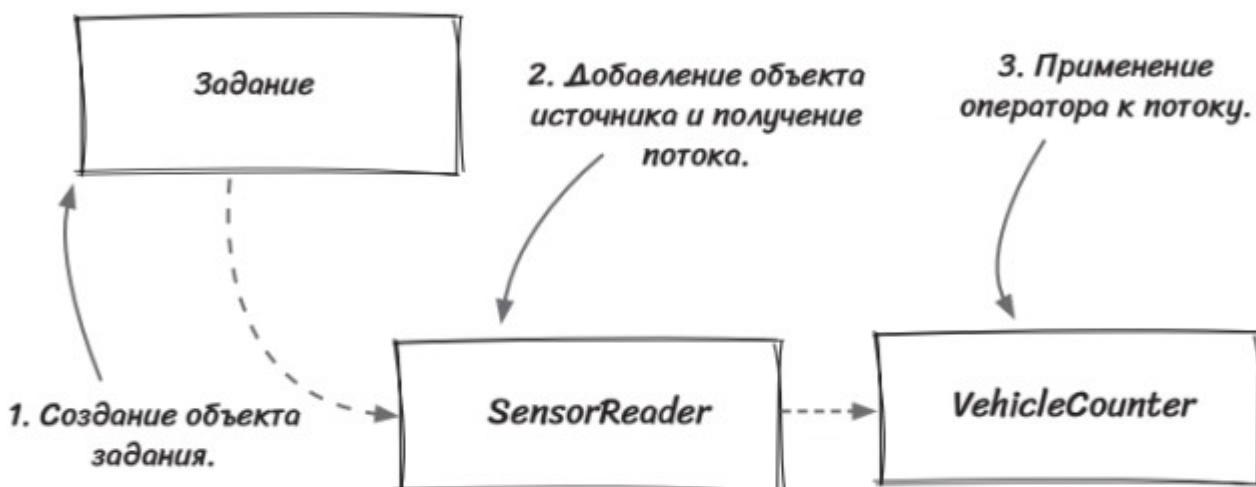
Получение счетчиков из карты

Сохранение счетчика обратно в карте

Первое стриминговое задание: сборка задания

Чтобы собрать стриминговое задание, необходимо добавить источник `SensorReader` и оператор `VehicleCounter` и соединить их. В классах `Job` и `Stream`, которые мы построили для вас, присутствуют перехватчики:

- `Job.addSource()` добавляет источник данных в задание.
- `Stream.applyOperator()` добавляет оператор в поток.



Следующий код выполняет описанные выше шаги:

```

public static void main(String[] args) {
    Job job = new Job();           ← Создание объекта задания
    Stream bridgeOut=job.addSource(new SensorReader()); ← Добавление объекта потока и получение потока

    bridgeOut.applyOperator(newVehicleCounter()); ← Применение оператора
    JobStarter starter = new JobStarter(job);          к потоку
    starter.start();                                ← Запуск задания
}

```