auto.ru

ФУНКЦИОНАЛЬНЫЕ СТРИМЫ

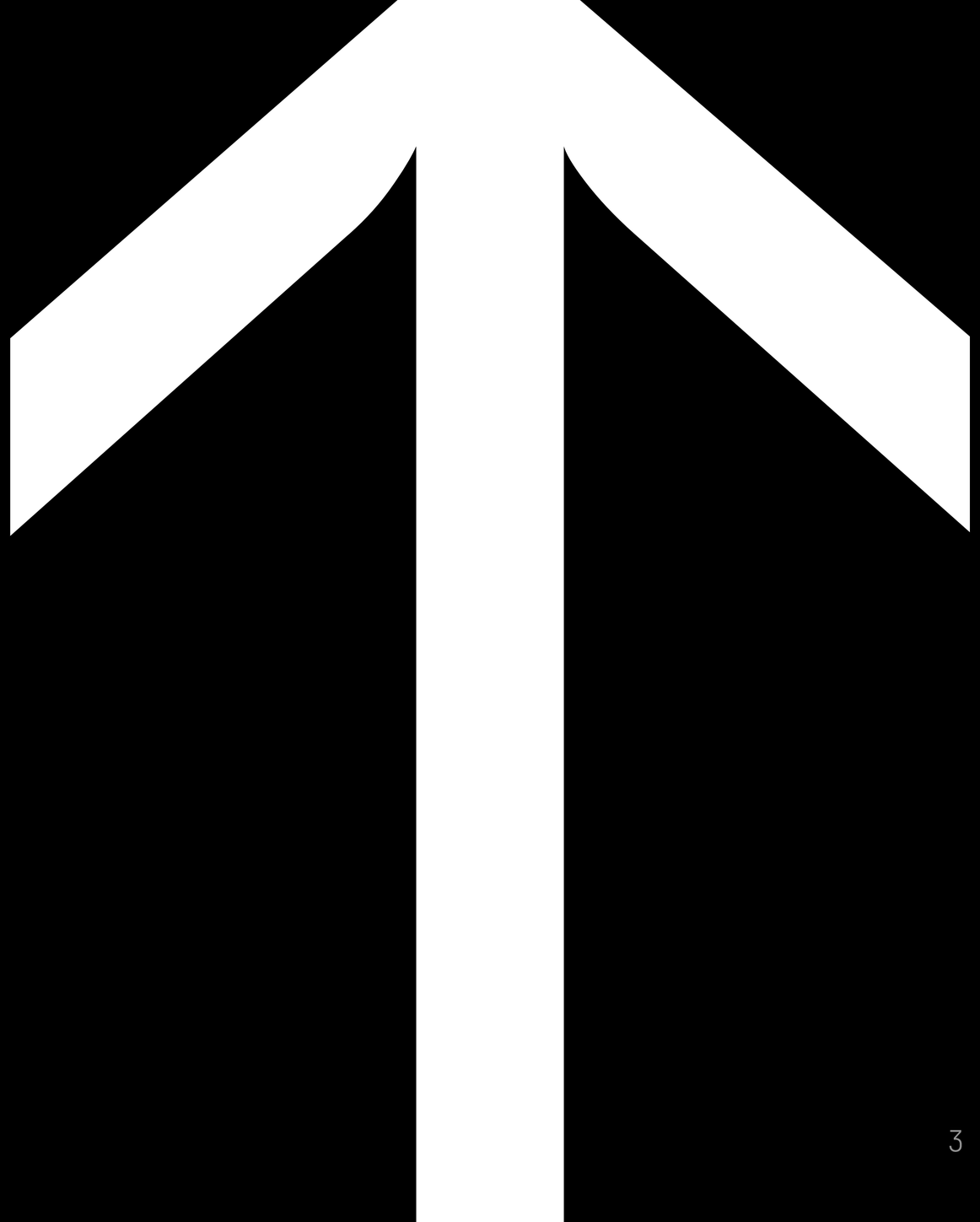
Ахтям Сакаев

О ЧЁМ

Моделирование процессов с примерами на Java

ЦЕЛЬ

Вдохновить на эксперименты



О СЕБЕ

Engineering Manager в Авто.ру

7 лет

Программировал на Scala

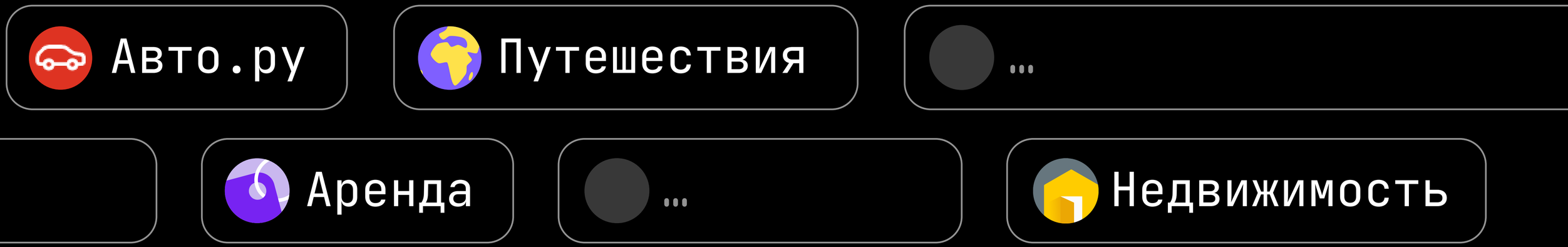
Разрабатывал:

- трейдинг
- мессенджер
- классифайд

Помогаем решать любые вопросы с автомобилями

Пишем серверную часть на Scala

JVM В ЯНДЕКСЕ



ПЛАН

01 Надёжные программы

02 Моделирование процессов

03 Бесконечность и время

КОРОТКО О ПОГОДЕ

Задача

Хранить историю температуры в Петербурге

01

Измерение

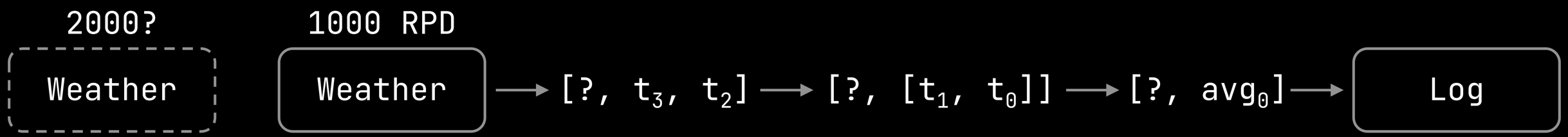
Средняя температура за час

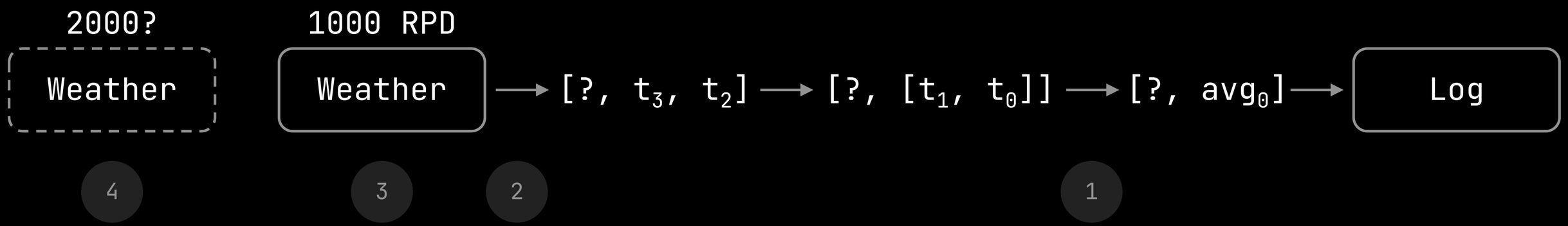
02

Ограничение

1000 вызовов в сутки

03


2000?
Weather
1000 RPD
Weather
[?, t3, t2]
[?, [t1, t0]]
[?, avg0]
Log


2000?
1000 RPD
Weather
Weather
[?, t3, t2]
[?, [t1, t0]]
[?, avg0]
Log
4
3
2
1

СРЕДНЕЕ АРИФМЕТИЧЕСКОЕ

```
Integer avg(List<Integer> xs) {
  if (xs.isEmpty()) return 0;
  else {
    var acc = 0;
    for (Integer x : xs) {
      acc = acc + x;
    }
    return acc / xs.size();
  }
}
```

Паттерн, у которого есть обобщение

СВЁРТКА

```
import java.util.stream.Stream;

Integer avg(List<Integer> xs) {
  return xs.isEmpty() ?
    0 :
    xs.stream().reduce(0, (acc, x) -> acc + x) / xs.size();
}
```

ИЗБАВЛЕНИЕ ОТ ХРУПКОСТИ

Повторяющийся код с циклами for/while

Хрупкое изменяемое состояние

REDUCE

Паттерн, рекурсия, fold

Управление состоянием

«Сокращает» данные, например списки

Зачем это использовать?

БЕСКОНЕЧНЫЕ ДАННЫЕ

```
Stream<Integer> s = Stream.iterate(0, x -> x + 1);
// [0, 1, 2, ...]
```

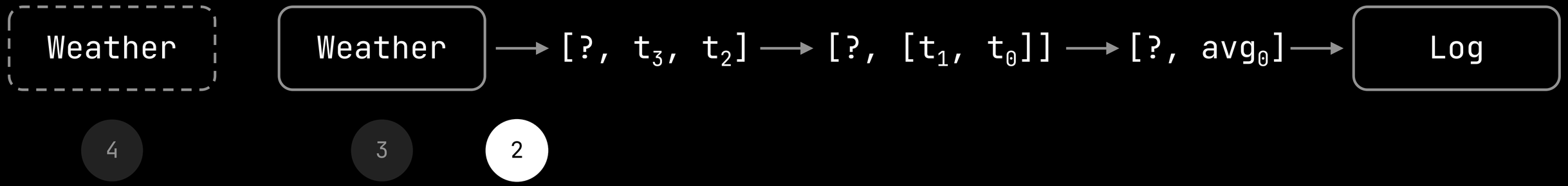
ЛЕНИВЫЕ ВЫЧИСЛЕНИЯ

```
Integer avgN(Stream<Integer> xs, Integer n) {
  return n == 0 ?
    0 :
    xs.limit(n).reduce(0, (acc, x) -> acc + x) / n;
}
```

Стрим — описание ещё не вычисленных данных

ВЫВОД

Свёртка возможна без размещения данных в памяти


Weather
Weather
[?, t_3, t_2]
[?, [t_1, t_0]]
[?, avg_0]
Log
4
3
2

Что значит бесконечность?

ПРИМЕРЫ

Последовательности и прогрессии из математики

Фракталы в природе — волны, лес, горы

Генеративное искусство, анимация

ЕЩЁ ПРИМЕРЫ

Биржевой тикер

Действия пользователя в браузере

Байтовый поток TCP соединения

Как создать бесконечные данные?

ТЕМПЕРАТУРА

```
interface Weather {
  Integer temp(String city);
}

Stream<Integer> temp(Weather w, String city) {
  return Stream.generate(() -> w.temp(city));
}
```

Пагинация и поллинг — «бесконечные» данные

auto.ru

01

Где ещё может пригодиться бесконечность?

ПОДБРАСЫВАНИЕ МОНЕТЫ

```
Stream<Boolean> coin() {
  var rnd = new Random();
  return Stream.generate(() -> rnd.nextBoolean());
}
```

Генерация — бесконечные данные

02

auto.ru

ГЕНЕРАТОР

Паттерн, корекурсия, unfold

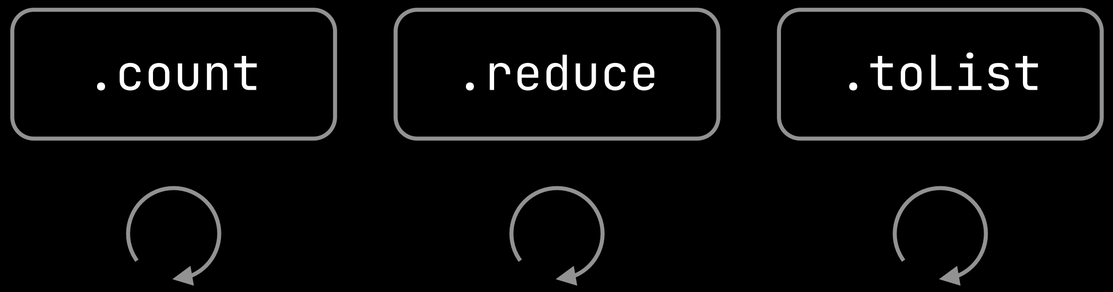
Коданные, описание вычислений

Представление бесконечности в коде

АССОЦИАЦИЯ

```
interface Iterator<E> {
  boolean hasNext();
  E next();
}
```

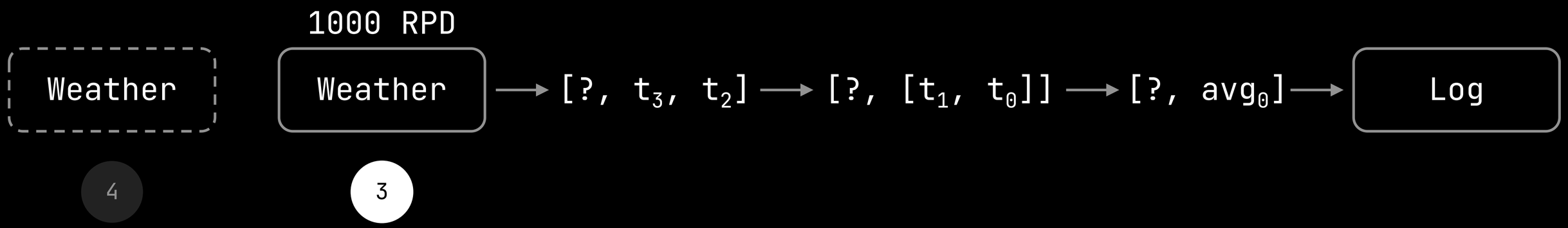
PULL-BASED



ДУАЛИЗМ

Рекурсия и корекурсия

fold и unfold


1000 RPD
Weather
Weather
[?, t_3, t_2]
[?, [t_1, t_0]]
[?, avg_0]
Log
4
3

Время

Кратчайшая история времени

ОГРАНИЧЕНИЕ ЧАСТОТЫ ВЫЗОВОВ

```
Stream<Integer> temp(Weather w, String city) {
  return Stream.generate(() -> w.temp(city));
}
```

java.util.stream.Stream

Понятия времени не существует

Моментальные ленивые вычисления

Долгие процессы предполагают время

Как моделировать время?

```
import reactor.core.publisher.Flux;
```

FLUX — ОПИСАНИЕ ПРОГРАММЫ

Производит ноль или несколько значений

Может завершаться успешно или с ошибкой

FLUX

Имплементация ленивых вычислений в Java

Работает со временем

ЧАСЫ

```
Flux<Long> clock = Flux.interval(Duration.ofSeconds(1));
```

БЕСКОНЕЧНАЯ ТЕМПЕРАТУРА

```
Flux<Integer> temp(Weather w) {
  return Flux.unfold("Saint Petersburg", city ->
    Optional.of(Tuples.of(w.temp(city), city))
  );
}
```

Add Flux.unfold #3897

https://github.com/reactor/reactor-core/pull/3897

ОГРАНИЧЕНИЕ ЧАСТОТЫ

```
Flux<Integer> tempSpaced(Weather w, Duration d) {
  return Flux
    .unfold("Saint Petersburg", city ->
      Optional.of(Tuples.of(w.temp(city), city))
    )
    .delayElements(d);
}
```

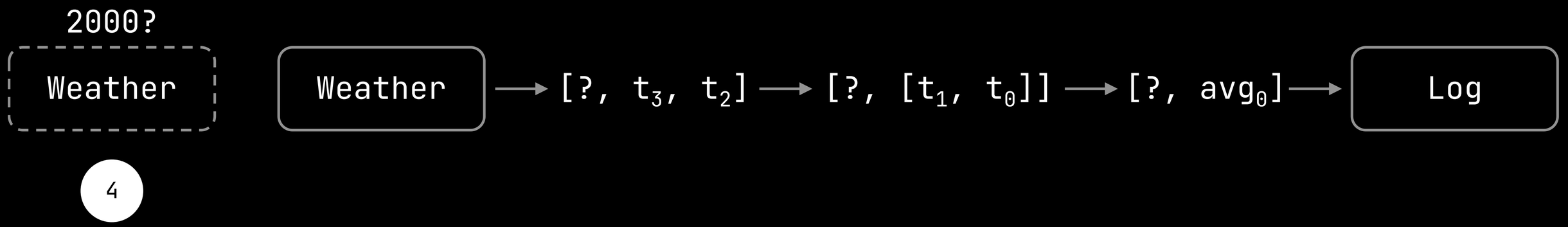
Время — триггер и состояние в этот момент

03

auto.ru

ВЫВОД

Стрим — данные, описание вычислений и время


2000?
Weather
4
Weather
[?, t3, t2]
[?, [t1, t0]]
[?, avg0]
Log

ОГРАНИЧЕНИЕ КОЛИЧЕСТВА

```
Flux<Integer> tempN(Weather w, Conf c) {
  return Flux
    .unfold("Saint Petersburg", city ->
      Optional.of(Tuples.of(w.temp(city), city))
    )
    .delayElements(c.interval())
    .take(c.limit());
}
```

Как улучшить качество данных?

Несколько токенов увеличат частоту измерений!

WEATHER С ТОКЕНОМ

```
<T> Flux<T> use(Integer token, Function<Weather, Flux<T>> f) {
  return Flux.using(
    () -> {
      System.out.println("Entering the Weather scope");
      return make();
    },
    f,
    w -> System.out.println("Leaving the Weather scope")
  );
}
```

Как try-with-resources в Java, только композируется

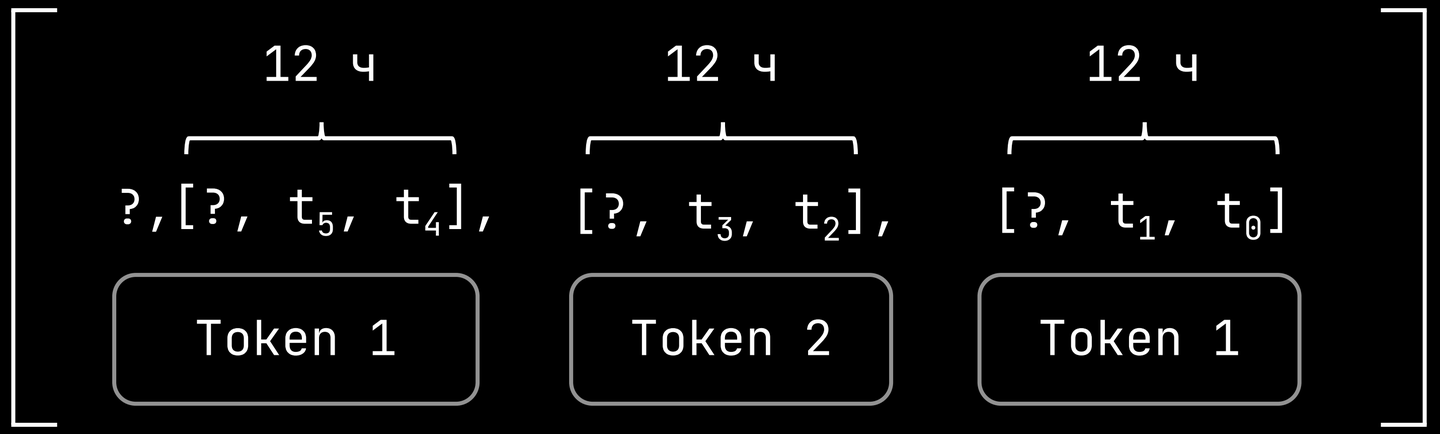
НЕСКОЛЬКО WEATHER

```
Flux<Integer> temp(Conf c) {
  return Flux
    .fromIterable(c.tokens())
    .concatMap(t ->
      WeatherSimulator.use(t, w -> tempN(w, c))
    )
    .repeat();
}
```

РЕСУРС

Время жизни

Область видимости


12 ч
12 ч
12 ч
?,[?, t5, t4],
[?, t3, t2],
[?, t1, t0]
Token 1
Token 2
Token 1

ВЫВОД

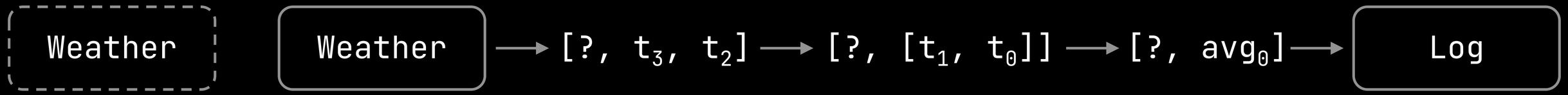
Стримы — управление жизненным циклом ресурсов

ПОТОКОВЫЙ АЛГОРИТМ

```
Flux<Integer> avg(Flux<Integer> xs) {
  return xs
    .reduce(Tuples.of(0, 0), (acc, x) ->
      Tuples.of(acc.getT1() + x, acc.getT2() + 1)
    )
    .map(t -> t.getT2() == 0 ? 0 : t.getT1() / t.getT2())
    .flux();
}
```

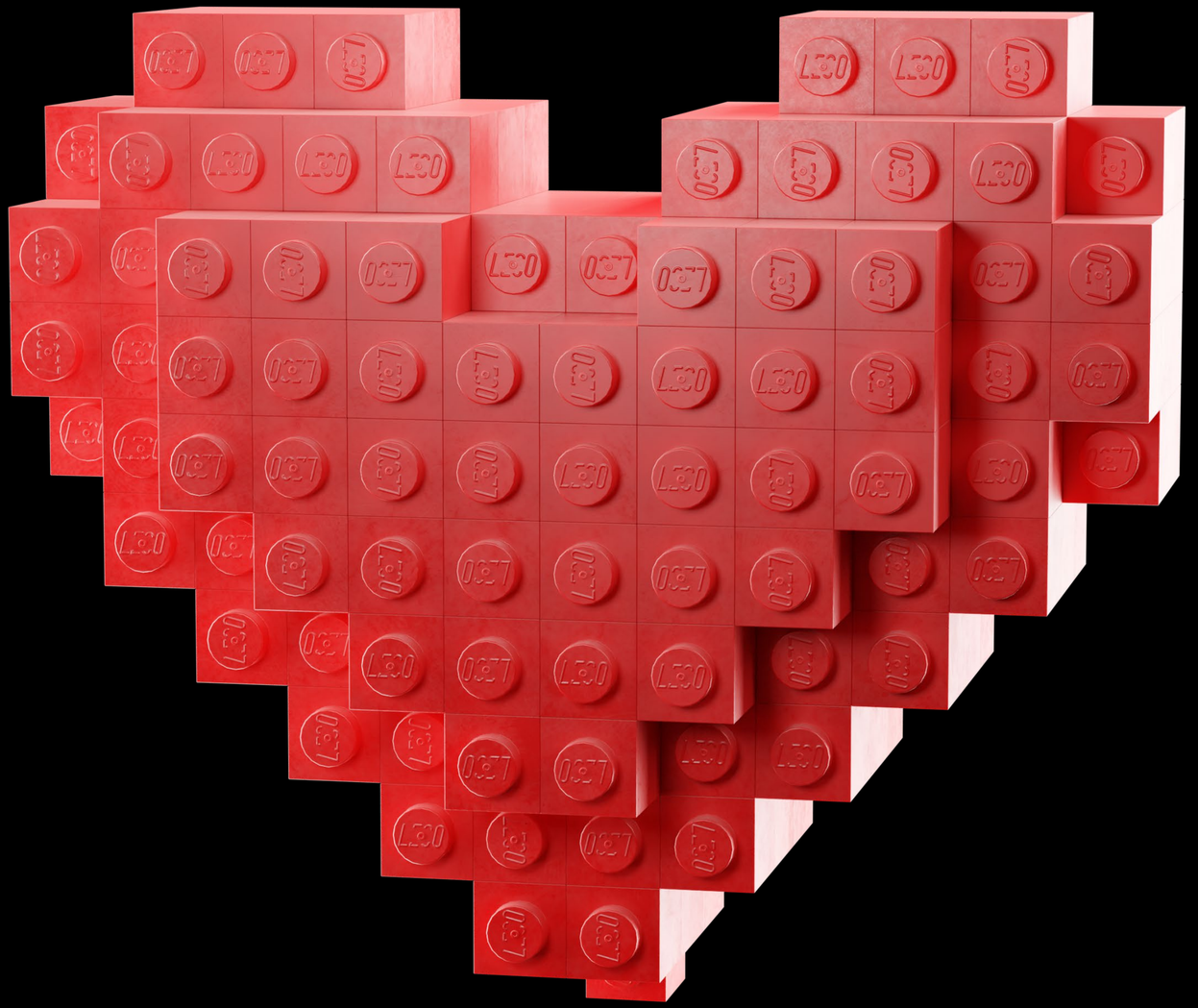
НАРЕЗАЕМ БЕСКОНЕЧНОСТЬ

```
Flux<Integer> avgs(Conf c) {
  return temp(c).window(c.window()).concatMap(xs -> avg(xs));
}
```


Weather
Weather
[?, t3, t2]
[?, [t1, t0]]
[?, avg0]
Log

Как собрать все части решения?

Композиция



ХРАНИЛИЩЕ

```
interface Log<T> {
  Boolean produce(T value);
}
```

ПРОЦЕСС

```
abstract Flux<Integer> tempN(Weather w, Integer n);
abstract Flux<Integer> temp(Conf c);
abstract Flux<Integer> avg(Flux<Integer> xs);
abstract Flux<Integer> avgs(Conf c);

Flux<Boolean> produce(Log<Tuple2<Integer, Instant>> log, Conf c) {
  return avgs(c).map(avg ->
    log.produce(Tuples.of(avg, Instant.now()))
  );
}

Flux<Boolean> program(Conf c) {
  return LogSimulator.use(log -> produce(log, c));
}
```

ПРИЛОЖЕНИЕ

```
public static void main(String[] args) {
  var c = new Conf(List.of(1, 2), 1000, Duration.ofSeconds(44), 81);
  Process.program(c).then().block();
}
```

```
limit=3, window=2, interval=1s, take=4

Entering the Log scope
  Entering the Weather [1] scope
    temp 13
    temp 11
  produce [12,2024-10-14T13:21:11.644038Z]
    temp 9
  Leaving the Weather [1] scope
  Entering the Weather [2] scope
    temp 14
  produce [11,2024-10-14T13:21:13.664125Z]
    temp 14
    temp 10
  produce [12,2024-10-14T13:21:15.671659Z]
  Leaving the Weather [2] scope
  Entering the Weather [1] scope
    temp 10
    temp 12
  produce [11,2024-10-14T13:21:17.679763Z]
  Leaving the Weather [1] scope
Leaving the Log scope
```

Стримы — сложные программы из простых блоков

05

auto.ru

СТРИМ — ЭТО ИДЕЯ

```
io.reactivex.rxjava3.core.Flowable
akka.stream.javadsl.Source
io.vavr.collection.Stream
```

СТРИМ

Способ описания программы

Высокоуровневый язык в языке

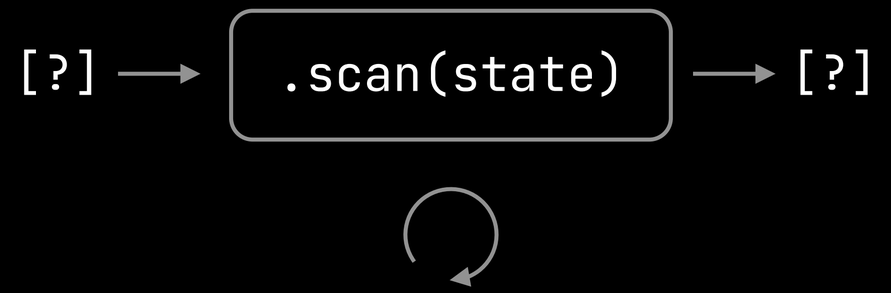
Парадигма data-flow programming

И последний пример...

ПРЕОБРАЗОВАНИЕ С СОСТОЯНИЕМ

```
Flux<String> pipe(Flux<Integer> xs) {
  return xs
    .scan(0, (acc, x) -> acc + x)
    .skip(1)
    .concatMap(n ->
      Flux.just(n.toString(), String.valueOf(n % 2 == 0))
    );
}

// [1, 1, 1] => [1, false, 2, true, 3, false]
```


[?]
.scan(state)
[?]

ВЫРАЗИТЕЛЬНОСТЬ

unfold

pipe

fold

СТРИМ

Управление состоянием и временем

Время жизни и область видимости

Процесс

ПОСЕТИТЕЛЬ КОНФЕРЕНЦИИ

```
record Slide() {}
record Brain() {}

Brain learn(Brain b, Slide s) {
  return b;
}

Flux<String> insight(Tuple2<Brain, Boolean> state) {
  return state.getT2() ? Flux.just("💡") : Flux.empty();
}

Flux<String> attendee(Flux<Slide> slides, Flux<Boolean> rnd) {
  return slides
    .scan(new Brain(), (brain, slide) -> learn(brain, slide))
    .zipWith(rnd, 1)
    .concatMap(state -> insight(state));
}
```

Озарение?

Доклад DDDamn good! на Joker 2021

Испытайте стримы в следующей задаче!