


Artemis 6.4.0

Course Overview

  ge64baw

Courses > Praktikum: Grundlagen der Programmierung WS22/23 > Exercises > W09H02 - Bahn Analyse mit Streams

 W09H02 - Bahn Analyse mit Streams

Hausaufgabe

Medium

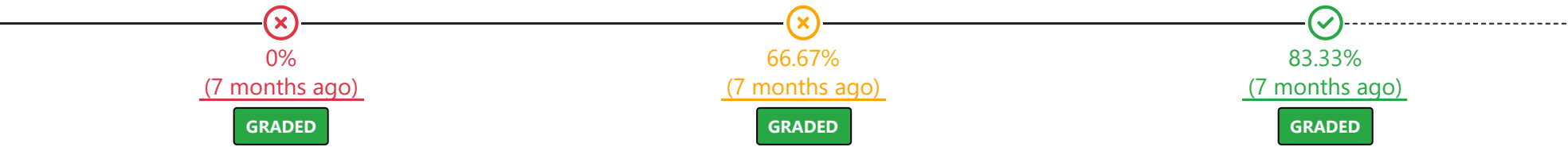
Submission due: 7 months ago

Complaint due: 7 months ago

Points: 5 of 6 Assessment: automatic ?

 83.33% (7 months ago) GRADED

Recent results:



Show all results

Tasks:

W09H02 Bahn-Analyse mit Streams

Im Folgenden wollen wir die Deutsche Bahn und den ÖPNV mit der Hilfe von Streams analysieren.

Streams

Diese Aufgabe dient dazu Javas Streams kennenzulernen, daher sind alle konventionellen Kontrollstrukturen verboten, dazu gehören: `for (;;)`-, `for (:)`- und `(do-)while` Schleifen, die Methode `.forEach()` auf `Stream` und `List` sowie Rekursion. `if`-Statements / `if-else`-Statements (und der Ternary Operator) sind im Gegensatz zur ersten Hausaufgabe explizit erlaubt. Diese Einschränkungen werden in einem public Test automatisch überprüft. Sollte es Probleme damit geben, würden wir uns auf Zulip über Feedback freuen.

Wenn Ihr bei einer Teilaufgabe nicht weiter kommt, hilft oft ein Blick in die [Stream API](#). Beachtet auch, dass es neben generischen Streams auch spezielle Implementationen, u. a. `IntStream`, `LongStream` und `DoubleStream` gibt, welche besondere Funktionen bereitstellen.

Hinweise/Tipps:

- Nochmals: `if`-Statements / `if-else`-Statements sind in dieser Aufgabe explizit erlaubt und sind bei mindestens einer Teilaufgabe sehr nützlich.
- Für Teilaufgaben 2, 4 und 6:** Die Verspätung eines Zuges an einer Station in Minuten kann mithilfe der bereits von uns implementierten Methode `TrainStop.getDelay()` ermittelt werden. Diese gibt die Differenz (in Minuten, wie gesagt) zwischen tatsächlicher und geplanter Ankunftszeit zurück wenn diese positiv ist, 0, wenn nicht. Die unerwünschten negativen Verspätungen werden durch `getDelay()` also bereits automatisch auf 0 gesetzt.
- Für Teilaufgaben 3, 4 und 6:** Die Methode `Stream.flatMap()` (genauso wie die von ihr abgeleiteten Methoden `flatMapToInt()`, ...) ist eine Variante von `Stream.map()`. Sie nimmt eine Funktion entgegen, die aus einem Stream-Element einen Stream macht. Würde man `map()` mit so einer Funktion aufrufen, würde man einen Stream von Streams erhalten. `flatMap()` nimmt nun noch diesen Stream von Streams und kombiniert die inneren Streams zu einem großen Gesamt-Stream. Diese Methode mag in einigen Situationen nützlich sein.
[Stream.flatMap\(\)-JavaDoc](#)
[Stream.flatMap\(\) Erklärung](#)
- Für Teilaufgaben 5 und 6:** Die Methode `Stream.collect()` in Kombination mit den Kollektoren `Collectors.groupingBy()` und `Collectors.toMap()` sind hier hilfreich. Ersterem wird eine Funktion übergeben, die ein Stream-Element auf einen Key abbildet, zurückgegeben wird dann eine Map von jeweiligen Key auf eine Liste aller Stream-Elemente, die auf diesen Key abbilden. Letzterem werden zwei Funktionen übergeben, eine, die aus dem gegebenen Stream-Element einen Key baut und eine zweite, die aus einem gegebenen Stream-Element einen Value baut. Die Map mit einem Key-Value-Paar pro Stream-Element, wobei der Key durch Anwenden ersterer, der Value durch Anwenden zweiterer Funktion aus dem Stream-Element erzeugt wird, wird dann zurückgegeben. Nutze nun erst `groupingBy()` um die `TrainConnections` (Aufg. 5) bzw. die `TrainStops` (Aufg. 6) geeignet in eine Map von Typ der Verbindung bzw. Stunde auf die Liste aller Verbindungen mit diesem Typ bzw. Halte in dieser Stunde zu überführen. Danach mache aus der erhaltenen Map mit `Map.entrySet().stream()` wieder einen Stream und Nutze nun `Stream.collect(Collectors.toMap(...))` geeignet um das Endergebnis zu erzeugen.
[Collectors.groupingBy\(\)-JavaDoc](#)
[Collectors.groupingBy\(\) Erklärung](#)
[Collectors.toMap\(\)-JavaDoc](#)
[Collectors.toMap\(\) Erklärung](#)

 Keine verbotenen Konstrukte verwendet 1 of 1 tests passing

Die Daten

Die Datengrundlage für unsere Analyse kommt von [expert.bahn](#), einer öffentlichen [API](#) für verschiedene Daten rund um Bus und Bahn. Die API nutzt `JSON` als Datenformat, die Umwandlung in die entsprechenden Objekte `TrainConnection`, `TrainStop` und `Station` (diese Klassen bitte nicht modifizieren) ist bereits in der Klasse `DataAccess` implementiert. Hier gibt es einmal die Möglichkeit Verbindungen aus einer Datei zu laden oder von der API

abzurufen. Wir würden Euch bitte beim Testen mit gespeicherten Daten zu arbeiten und die API so wenig wie möglich zu nutzen, eine Anfrage findet ihr in der Datei `connections_test/sample.json`.

Die Analyse

Da wir die Daten nun laden können, wollen wir im Folgenden schauen, welche Schlüsse sich ziehen lassen. Schaut Euch die Klassen `TrainConnection`, `TrainStop` und `Station` an und stellt sicher, dass ihr alle Attribute und Funktionen, welche nicht fürs Parsing zuständig sind, versteht. Beachtet beim bearbeiten bitte Folgendes:

- *Negative Verspätung*: Es gibt keine negativen Verspätungen! Sollte ein Zug zu früh sein, darf das in der Statistik nicht andere Verspätungen ausgleichen.
- *CANCELLED Stops*: Außer in der ersten und dritten Aufgabe muss keine Rücksicht darauf genommen werden ob ein Stop gestrichen wurde. `CANCELLED` Stops sollen daher wie alle anderen behandelt werden.
- *Leere Stops*: Die Liste mit Stops einer `TrainConnection` wird immer mindestens ein Element enthalten.

1. Clean Dataset

In der Methode `cleanDataset` wollen wir sicherstellen, dass wir eine saubere Datengrundlage haben. Dafür wollen wir den übergebenen Stream:

1. Von Duplikaten befreien, da die API teilweise leider doppelte Verbindungen übergibt. (Da `TrainConnection`, `TrainStop` und `Station` als `Records` implementiert sind haben sie eine "richtige" `.equals()` Methode, welche nicht auf Referenz-Gleichheit, sondern auf Inhalt prüft, dementsprechend gilt zwei Verbindungen `v1` und `v2` als Duplikate voneinander, wenn `v1.equals(v2)` gilt - in dem Fall soll nur eine von beiden behalten werden).
2. Sortieren nach geplanter Abfahrtszeit am Abfahrtsbahnhof (der nullte Index in der Stops-Liste).
3. Aus den Stops der Verbindungen sollen diejenigen aussortiert werden, welche nicht angefahren werden, und daher das `Kind` Attribut `CANCELLED` haben. (Die Methode `TrainConnection.withUpdatedStops()` könnte hilfreich sein).

- ✓ **Public Tests** [1 of 1 tests passing](#)
- ✓ **Hidden Tests** [5 of 5 tests passing](#)
- ✓ **Punkte** [1 of 1 tests passing](#)

2. Schlimmste Verspätung

In der Methode `worstDelayedTrain` soll aus allen Verbindungen diejenige gefunden werden, welche die schlimmste Verspätung an einem Halt im Laufe der Fahrt hatte. Wenn keine Verbindungen übergeben wurden, soll `null` zurückgegeben werden. Sollte es keine Verspätung geben, kann eine beliebige Verbindung zurückgegeben werden. Sollten mehrere Verbindungen die gleiche maximale Verspätung an einer ihrer Stops gehabt haben, kann eine beliebige aus diesen zurückgegeben werden.

- ✓ **Public Tests** [1 of 1 tests passing](#)
- ✓ **Hidden Tests** [5 of 5 tests passing](#)
- ✓ **Punkte** [1 of 1 tests passing](#)

3. Wie viel Prozent sind ..

In der Methode `percentOfKindStops` soll berechnet werden, wie viel Prozent `[0, 100]` der Stops im Betrachtungszeitraum `CANCELLED`, `ADDITIONAL` bzw. `REGULAR` sind. Welches Attribut wir betrachten, wird der Methode als Argument `kind` übergeben.

- ✓ **Public Tests** [1 of 1 tests passing](#)
- ✗ **Hidden Tests** [4 of 5 tests passing](#)
- ✗ **Punkte** [0 of 1 tests passing](#)

4. Durchschnittliche Verspätung an ...

Die Methode `averageDelayAt` soll die durchschnittliche Verspätung an einer speziellen, übergebenen Station in Minuten berechnen. Beachtet wieder, dass `Station` als `Record` bereits eine `equals` Methode hat, welche benutzt werden kann.

- ✗ **Public Tests** [0 of 1 tests passing](#)
- ✗ **Hidden Tests** [3 of 5 tests passing](#)
- ✗ **Punkte** [0 of 1 tests passing](#)

5. Anteilige Verspätung pro Verkehrsmittel

Die Methode `delayComparedToTotalTravelTimeByTransport` soll eine Map zurückgeben, in welcher jedes im Datensatz vorkommende Verkehrsmittel (`type` Attribut in `TrainConnection`) auf seine anteilige Verspätung abbildet. Anteilig meint hierbei, wie viele Prozent der gesamten gefahrenen Zeit (für eine einzelne Verbindung ist das immer die Zeit zwischen dem ersten und letzten Stop) des Verkehrsmittels auf Verspätungen zurückzuführen sind. Wenn alle ICEs im Betrachtungszeitraum zusammen 24 Stunden fahren, aber nur 16 geplant waren, dann sprechen wir von einer anteiligen Verspätung von 33,3333...%, da ein Drittel der gefahrenen Zeit einer Verspätung zuzuordnen ist (vergleiche $(24 - 16)/24 = 0.3333 \dots$). (Beachtet hierbei auch die Methoden `totalTimeTraveledScheduled` und `totalTimeTraveledActual`).

- ✓ **Public Tests** [1 of 1 tests passing](#)
- ✓ **Hidden Tests** [5 of 5 tests passing](#)
- ✓ **Punkte** [1 of 1 tests passing](#)

6. Durchschnittliche Verspätung pro Stunde

Die Methode `averageDelayByHour` soll eine Map berechnen, wobei jede Stunde in der im Betrachtungszeitraum eine Station angefahren wurde, auf die durchschnittliche Verspätung in Minuten in dieser Stunde verweist. In einer Stunde sollen dabei jeweils die `TrainStops` betrachtet werden, die tatsächlich in dieser Stunde die jeweilige Station angefahren haben (`TrainStop.actual()`, nicht `TrainStop.scheduled()`). (`LocalDateTime.getHour()` findet die Stunde einer `LocalDateTime` heraus).

- ✔ **Public Tests** [1 of 1 tests passing](#)
- ✔ **Hidden Tests** [5 of 5 tests passing](#)
- ✔ **Punkte** [1 of 1 tests passing](#)


FAQ

Was sind Records: Das ist eine neue Art Klassen zu erstellen, welche besondere Eigenschaften haben: Alle Attribute sind `private final` und haben einen Getter `attributeName().equals(), toString()` und `hashCode()` sind bereits implementiert. Mehr könnt Ihr hier nachlesen: [Records](#)

[Lösungsvorschlag](#)
[Tests](#)

Exercise details

Release date:	Dec 15, 2022 18:30
Submission due:	Jan 15, 2023 18:00
Complaint due:	Jan 22, 2023 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left. 

You have already submitted a complaint 7 months ago **Complaint was accepted**

Response to your complaint 7 months ago:

Sehr geehrte Bewerberin / Sehr geehrter Bewerber, nachdem ich mir die Testergebnisse angesehen hatte, stellte ich fest, dass ich einen kleinen Fehler in meinem Code hatte, der leider zu vielen Punktabzügen führte :(

Da es zweimal der selbe Fehler war, bekommst du einen der zwei dadurch verlorenen Punkte zurück.

How useful is this feedback to you?



[About](#)

[Request change](#) [Release notes](#) [Privacy Statement](#) [Imprint](#)