

Artemis 6.4.0


☰

Course Overview



ge64baw ▾

Courses > [Praktikum: Grundlagen der Programmierung WS22/23](#) > Exercises > [W09H01 - Pinguin Ausflug](#)




W09H01 - Pinguin Ausflug

Hausaufgabe

Easy

Submission due:

7 months ago

Points: 13 of 13 Assessment: automatic 

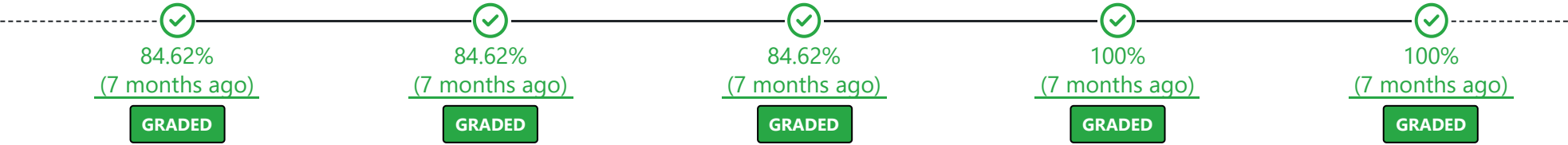
Complaint due:

7 months ago

 [100% \(7 months ago\)](#)

GRADED

Recent results:



Show all results ▾

Tasks:

W09H01 - Pinguin Ausflug

Es wird endlich wieder Sommer in der Antarktis und die Pinguine der Familie Fluss können ihren ersten Ausflug nach dem langen Winter kaum noch erwarten. Damit auch alles glattgeht und der Ausflug zu einem vollen Erfolg wird, brauchen sie aber Deine Hilfe! Sie würden ihren Ausflug nämlich gern analysieren, bevor sie aufbrechen. Da sie gelesen haben, dass sich Streams dafür sehr gut eignen, würden sie diese These gerne von Dir testen lassen.

Streams

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

Hinweise/Tipps:

- Wenn Ihr bei einer Teilaufgabe nicht weiter kommt, hilft oft ein Blick in die [Stream API](#). Alle Aufgaben lassen sich in wenigen Zeilen Code (bzw. mit wenigen Methodenaufrufen auf einem Stream) lösen. Oft reduziert sich die Schwierigkeit nur darauf, die richtige Stream-Methode zu finden. Beachtet auch, dass es neben generischen Streams auch spezielle Implementationen, u. a. [IntStream](#), [LongStream](#) und [DoubleStream](#) gibt, welche besondere Funktionen (wie z.B. `sum()` oder `average()`) bereitstellen.
- Als kleines Weihnachtsgeschenk (und damit die Gesamtpunktzahl am Ende eine schöne 250 ergibt) verteilen wir auf die Methoden dieser Aufgabe 5 Extrapunkte (**nicht Bonuspunkte!**). Die Aufgabe gibt also 13 statt der normalen 8 Punkte. Sie ist allerdings als 8 Punkte Aufgabe konzipiert worden, sollte also nicht umfangreicher sein, als gewöhnliche H01 Aufgaben.
- Auch diese Woche gilt für die H01, dass die Punktzahl, die euch durch die Public Tests vergeben wird, sicher ist, solange ihr keine der verbotenen Kontrollstrukturen / Methoden etc. verwendet.

 **Keine verbotenen Konstrukte verwendet** [1 of 1 tests passing](#)

Gegebene Klassen

Euch sind bereits die Klassen `WayPoint` und `OneWay` gegeben. Diese enthalten einige Methoden für geometrische Berechnungen, Umwandlung von Strings zu Objekten und Ausgabe. Schaut euch die Klassen gut an und verwendet die gegebenen Methoden nach Möglichkeit.

Aufgabe

Vervollständige nun die Methoden in der Klasse `PinguTrip` wie im Folgenden beschrieben!

1. Route einlesen

Die Pinguine haben sich schon den ganzen Winter lang Gedanken gemacht, wo sie denn bei ihrem ersten Ausflug überall hin möchten. Da sie aber erst im Oktober angefangen haben zu programmieren, stehen ihre ganzen Notizen noch in einer .txt Datei und sie hätten gerne eine Methode `readWayPoints`, welche ihre Wegpunkte einliest und in `WayPoint` Objekte umwandelt, damit sie die Notizen besser analysieren können.

Die Methode soll den Pfad zu den Notizen als String bekommen und einen Stream, welcher die `WayPoint` Objekte beinhaltet, zurückgeben. Jede Zeile in der Notiz ist dabei genau in einem von drei Formaten:

1. Eine Koordinate `x;y`. Diese soll in einen `WayPoint` umgewandelt werden. x und y stehen hierbei für beliebige doubles als String.

2. Ein Kommentar `//` Startend mit zwei Schrägstrichen und gefolgt von einer beliebigen Zeichenfolge. Kommentare sollen ignoriert werden.

3. Ein Schlusstrich `---`. Als die Pinguine gemerkt haben, dass ihre Route schon viel zu lang ist haben sie einen Schlusstrich gezogen. Alles, was danach kommt, soll nicht mehr behandelt werden.

Alle anderen Fälle, wie etwa eine Zeile die nicht in einem der drei Formate ist, werden nicht getestet und müssen nicht behandelt werden.

Sollte beim Einlesen ein Fehler auftreten, soll ein leerer Stream zurückgegeben werden.

▼ Beispiel

Eine Datei mit folgendem Inhalt:

```
4.0;11.5
// comment
19.1;3.2
---
9.11;1.1
```

soll umgewandelt werden in:

```
Stream.of(new WayPoint(4.0, 11.5), new WayPoint(19.1, 3.2));
```

✓ **Public Tests (= Punkte)** [2 of 2 tests passing](#)

✓ **Hidden Tests** [2 of 2 tests passing](#)

2. Wegpunkte zu Wegen wandeln

Da nur die Wegpunkte zu haben manchmal sehr umständlich sein kann, wollen die Pinguine ihre Wegpunkte nun in Wege - repräsentiert durch **OneWay** Objekte - umwandeln. Dafür sollen in der Methode **transformToWays** die Elemente einer Liste von Wegpunkten verbunden werden und zwar so, dass immer der i-te mit dem i+1-ten Wegpunkt zu einem Weg wird. Die Methode wird nur mit Streams, welche mindestens zwei Wegpunkte enthalten, getestet. Leere Streams oder Streams mit nur einem Element müssen nicht behandelt werden.

Tipp: Der Weg wird hier aus einem Grund als eine Liste und nicht wie in den anderen Methoden als ein Stream übergeben. Mit **waysPoints.stream()** zu beginnen, ist also vermutlich nicht der richtige Weg. Überlege dir stattdessen, mit was für einem Stream du deine Berechnungen beginnen willst. Die Methode **IntStream.range()** könnte hilfreich sein.

▼ Beispiel

```
List.of(new WayPoint(4.0, 11.5), new WayPoint(19.1, 3.2), new WayPoint(2.1, 7.4));
```

Soll umgewandelt werden in:

```
Stream.of(new OneWay(new WayPoint(4.0, 11.5), new WayPoint(19.1, 3.2)), new OneWay(new WayPoint(19.1, 3.2), new WayPoint(2.1, 7.4)))
```

✓ **Public Tests (= Punkte)** [1 of 1 tests passing](#)

✓ **Hidden Tests** [1 of 1 tests passing](#)

3. Länge des Weges

Die Pinguine wollen wissen, wie lang ihr Weg überhaupt ist. Dafür soll die Länge der gesamte Strecke in der Methode **pathLength** berechnet werden.

▼ Beispiel

```
Stream.of(new OneWay(new WayPoint(0.0, 0.0), new WayPoint(1.0, 0.0)));
```

Sollte dementsprechend **1.0** zurück geben.

✓ **Public Tests (= Punkte)** [1 of 1 tests passing](#)

✓ **Hidden Tests** [1 of 1 tests passing](#)

4. Kinderausflug

In der großen Pinguin-Familie sind auch Küken, welche nicht so weite Strecken laufen können. Daher sollen in der Methode **kidFriendlyTrip** ein Ausflug erstellt werden, der auch für die kleinen Pinguine geeignet ist. Dafür sollen vom Start aus die Wege genommen werden, welche nicht länger als die durchschnittliche Weglänge in der gesamten Route sind. Sobald der erste Weg länger ist, soll der Kinderausflug beendet werden und keine weiteren Wege mehr enthalten (exklusive dem Weg, welcher länger als der Durchschnitt war).

Hinweis: Dass hier wieder Listen statt Streams übergeben und zurückgegeben werden, hat ebenfalls seine Gründe. Hier musst du dir darüber allerdings nicht allzu viele Gedanken machen. **oneWays.stream()** zu verwenden ist hier eine gute Idee.

▼ Beispiel

```
List.of(new OneWay(new WayPoint(0.0, 0.0), new WayPoint(1.0, 0.0)),
        new OneWay(new WayPoint(1.0, 0.0), new WayPoint(3.0, 0.0)),
        new OneWay(new WayPoint(3.0, 0.0), new WayPoint(4.0, 0.0)));
```

soll zu folgender Liste umgewandelt werden, da die durchschnittliche Länge $(1 + 2 + 1)/3 = 1.33..$ ist, und der zweite Weg mit $2 > 1.33$ länger als der Durchschnitt ist und somit ab diesem Punkt keine weiteren Wege mehr mit aufgenommen werden.

```
List.of(new OneWay(new WayPoint(0.0, 0.0), new WayPoint(1.0, 0.0))
```

✔ **Public Tests (= Punkte)** [1 of 1 tests passing](#)

✔ **Hidden Tests** [1 of 1 tests passing](#)

5. Maximale Entfernung nach Hause

Damit sie ihren Ausflug auch gut spontan abbrechen können, wollen die Pinguine in der Methode `furthestAwayFromHome` gerne berechnet haben bei welchem Wegpunkt sie sich am weitesten von Zuhause entfernen. Wenn es keinen Wegpunkt gibt, soll der Wegpunkt ihres Zuhauses zurückgegeben werden. Sollten mehrere Wegpunkte gleich weit entfernt sein, kann von diesen ein Beliebiger zurückgegeben werden.

▼ Beispiel

```
Stream<WayPoint> path = Stream.of(new WayPoint(1.0, 0.0), new WayPoint(2.0, 0.0));
WayPoint home = new WayPoint(0.0, 0.0)
```

soll folgenden Wegpunkt wieder geben:

```
new WayPoint(2.0, 0.0)
```

✔ **Public Tests (= Punkte)** [1 of 1 tests passing](#)

✔ **Hidden Tests** [1 of 1 tests passing](#)

6. Freunde besuchen

Die Pinguine haben im Winter viele ihrer Freunde nicht gesehen und würden diese gerne besuchen. Dies ist aber nur sinnvoll, wenn ihre Freunde auf dem Weg ihrer Route leben. Gegeben ist eine Route und der Wegpunkt eines Freundes, die Methode `onTheWay` soll berechnen, ob ihr Freund auf dem geplanten Weg liegt.

▼ Beispiel

```
Stream.of(new OneWay(new WayPoint(0.0, 0.0), new WayPoint(1.0, 0.0));
WayPoint visit = new WayPoint(0.5, 0.0);
```

Soll `true` wieder geben.

✔ **Public Tests (= Punkte)** [1 of 1 tests passing](#)

✔ **Hidden Tests** [1 of 1 tests passing](#)

7. Schöne Richtungsangaben

Nachdem ihr Ausflug nun hinreichend analysiert wurde, wollen sich die Pinguine auf den Weg machen. Da der Mobilfunk-Ausbau in der Antarktis aber leider genau so schleppend voran geht wie in Deutschland, wollen sie sich ihre Wegbeschreibung ausdrucken. Dafür muss sie aber erstmal in der Methode `prettyDirections` in ein hübsches und lesbares Format umgewandelt werden. Jeder Weg soll in eine Zeile in folgendem Format gewandelt werden: `<gerundet (Entfernung / 0.7)> Schritte_Richtung <Grad mit Norden 0, Osten 90 etc.> Grad.`, (die Methode `prettyPrint` übernimmt diese Umwandlung bereits für euch).

▼ Beispiel

```
Stream.of(new OneWay(new WayPoint(0.0, 0.0), new WayPoint(1.0, 0.0)),
        new OneWay(new WayPoint(1.0, 0.0), new WayPoint(3.0, 0.0)),
        new OneWay(new WayPoint(3.0, 0.0), new WayPoint(4.0, 1.0)));
```

Soll folgenden String zurück geben.

- 1 Schritte Richtung 0 Grad.
- 3 Schritte Richtung 0 Grad.
- 2 Schritte Richtung 45 Grad.

✔ **Public Tests (= Punkte)** 1 of 1 tests passing

✔ **Hidden Tests** 1 of 1 tests passing

FAQ

Null Elemente: Ihr könnt davon ausgehen, dass alle Attribute von WayPoint und OneWay sowie die Parameter der Methoden immer ungleich **nu11** sind

[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. 