Aufgabe Flugdaten **af_bne_lambda_flug**

generiert mit Bing

Name:	
Klasse:	
Datum:	

Schuljahr: 2024/25

Für den Flug einer Sportmaschine soll die Reihenfolge bestimmt werden, in der vorgegebene Geopositionen (diese werden über latitude, longitude und altitude angegeben) angeflogen werden.

Ausgehend von der aktuellen Position soll immer die Position als nächstes angeflogen werden, die der aktuellen Position am nächsten liegt. Die anzufliegenden Positionen liegen in einer CSV-Datei vor (Auszug).

```
Latitude,Longitude,Altitude
48.3538,11.7861,0.0
48.40150416666667,11.756599999999999,909.0909090909091
48.44920833333333,11.7271,1818.1818181818182
48.4969125,11.6976,2727.2727272727275
```

1. Erstelle eine Klasse `GeoPos`, die die einzelnen Daten aus der CSV-Datei aufnimmt und füge entsprechende Konstrukturen, getter, setter, ... hinzu.

```
public class GeoPos {

   double latitude;
   double longitude;
   double altitude;
   // ...
}
```

2. Erstelle eine Klasse `CsvReader` mit der nachfolgenden Methode, welche die CSV-Daten als Liste zurückliefert.

```
public static List<GeoPos> readCsv(final String filePath)
```

3. Erstelle eine Klasse `GeoCalculator` mit der nachfolgenden Methode, die die Distanz zwischen zwei Geopositionen berechnet. Mache Dich schlau, wie diese Berechnung funktioniert.

```
public static double getDistance(final GeoPos pos1, final GeoPos pos2)
```

4. Erweitere die Klasse `GeoCalculator` mit der nachfolgenden Methode, die folgende Berechnungen durchführen soll.

```
public static GeoPos[] calculateFlight(final List<GeoPos> geoPositions)
```

   - Das Liste `geoPositions` kann durch den Algorithmus verändert werden.
   - Die Geoposition mit dem Index 0 der Liste `geoPositions` wird zur ersten aktuellen Position. Diese Position wird im Array `flightPositions` gespeichert (Startposition!) und kann dann aus der Liste der noch anzufliegenden Positionen (`geoPositions`) gelöscht werden.
   - Solange noch Geopositionen im Array geoPositions vorhanden sind:
     - Ermittle die Position in der Liste `geoPositions` mit der kürzesten Entfernung zur aktuellen Position.

- Diese Position wird im Array `flightPositions` gespeichert und zur neuen aktuellen Position.
- Diese Position kann dann aus der Liste `geoPositions` gelöscht werden.

Hinweis: Der größte Doublewert kann mit `Double.MAX_VALUE` abgerufen werden.

5. Setze die Aufgabe so um, dass die Klasse `GeoCalculator` nur Schleifen etc. verwendet.
6. Erstelle nun die Klasse `GeoCalculator2`, die die Aufgabe mit Lambda Streams umsetzt.
7. Zum Testen erstelle die Klasse `GeoMain`, welche zuerst die Berechnung mit Schleifen durchführt und das Ergebnis ausgibt und dann mit Lambda-Streams. Die Ausgaben müssen dabei identisch sein, sonst ist was „schief“ gelaufen. Beachte, dass die Pfadangaben angepasst werden müssen!

```
System.out.println("=== mit Schleifen");

final List<GeoPos> geoliste = CsvReader.readCsv("daten/
flight_data_muc_to_nue.csv");

final GeoPos[] calc = GeoCalculator.calculateFlight(geoliste);
for (final GeoPos geoPos : calc) {
   System.out.println(geoPos);
}

System.out.println("\n=== mit Lambda Streams");
final List<GeoPos> geoliste2 = CsvReader.readCsv("daten/
flight_data_muc_to_nue.csv");

final GeoPos[] calc2 = GeoCalculator2.calculateFlight(geoliste2);
for (final GeoPos geoPos : calc2) {
   System.out.println(geoPos);
}
```

Die Ausgabe kann dann wie folgt aussehen:

```
=== mit Schleifen
GeoPos [latitude=48.3538, longitude=11.7861, altitude=0.0]
GeoPos [latitude=48.40150416666667, longitude=11.756599999999999, altitude=909.0909090909091]
GeoPos [latitude=48.44920833333333, longitude=11.7271, altitude=1818.1818181818182]
GeoPos [latitude=48.4969125, longitude=11.6976, altitude=2727.2727272727275]
GeoPos [latitude=48.54461666666667, longitude=11.668099999999999, altitude=3636.3636363636365]
GeoPos [latitude=48.59232083333333, longitude=11.638599999999999, altitude=4545.454545454546]
GeoPos [latitude=48.640025, longitude=11.6091, altitude=5454.545454545455]
GeoPos [latitude=48.687729166666664, longitude=11.5796, altitude=6363.636363636364]
GeoPos [latitude=48.73543333333333, longitude=11.550099999999999, altitude=7272.727272727273]
GeoPos [latitude=48.7831375, longitude=11.5206, altitude=8181.818181818182]
GeoPos [latitude=48.830841666666664, longitude=11.4911, altitude=9090.909090909092]
GeoPos [latitude=48.878545833333334, longitude=11.461599999999999, altitude=10000.0]
GeoPos [latitude=48.926249999999996, longitude=11.432099999999998, altitude=10000.0]
GeoPos [latitude=48.973954166666665, longitude=11.4026, altitude=9090.90909090909]
GeoPos [latitude=49.021658333333335, longitude=11.373099999999999, altitude=8181.818181818182]
GeoPos [latitude=49.0693625, longitude=11.343599999999999, altitude=7272.727272727272]
GeoPos [latitude=49.117066666666666, longitude=11.3141, altitude=6363.636363636364]
GeoPos [latitude=49.164770833333336, longitude=11.2846, altitude=5454.545454545454]
GeoPos [latitude=49.212475, longitude=11.255099999999999, altitude=4545.454545454545]
GeoPos [latitude=49.26017916666667, longitude=11.2256, altitude=3636.363636363636]
GeoPos [latitude=49.307883333333336, longitude=11.1961, altitude=2727.272727272727]
GeoPos [latitude=49.3555875, longitude=11.166599999999999, altitude=1818.181818181818]
GeoPos [latitude=49.40329166666667, longitude=11.137099999999998, altitude=909.0909090909081]
GeoPos [latitude=49.45099583333333, longitude=11.1076, altitude=0.0]

=== mit Lambda Streams
GeoPos [latitude=48.3538, longitude=11.7861, altitude=0.0]
GeoPos [latitude=48.40150416666667, longitude=11.756599999999999, altitude=909.0909090909091]
GeoPos [latitude=48.44920833333333, longitude=11.7271, altitude=1818.1818181818182]
GeoPos [latitude=48.4969125, longitude=11.6976, altitude=2727.2727272727275]
GeoPos [latitude=48.54461666666667, longitude=11.668099999999999, altitude=3636.3636363636365]
GeoPos [latitude=48.59232083333333, longitude=11.638599999999999, altitude=4545.454545454546]
GeoPos [latitude=48.640025, longitude=11.6091, altitude=5454.545454545455]
GeoPos [latitude=48.687729166666664, longitude=11.5796, altitude=6363.636363636364]
GeoPos [latitude=48.73543333333333, longitude=11.550099999999999, altitude=7272.727272727273]
GeoPos [latitude=48.7831375, longitude=11.5206, altitude=8181.818181818182]
GeoPos [latitude=48.830841666666664, longitude=11.4911, altitude=9090.909090909092]
GeoPos [latitude=48.878545833333334, longitude=11.461599999999999, altitude=10000.0]
GeoPos [latitude=48.926249999999996, longitude=11.432099999999998, altitude=10000.0]
GeoPos [latitude=48.973954166666665, longitude=11.4026, altitude=9090.90909090909]
GeoPos [latitude=49.021658333333335, longitude=11.373099999999999, altitude=8181.818181818182]
GeoPos [latitude=49.0693625, longitude=11.343599999999999, altitude=7272.727272727272]
GeoPos [latitude=49.117066666666666, longitude=11.3141, altitude=6363.636363636364]
GeoPos [latitude=49.164770833333336, longitude=11.2846, altitude=5454.545454545454]
GeoPos [latitude=49.212475, longitude=11.255099999999999, altitude=4545.454545454545]
GeoPos [latitude=49.26017916666667, longitude=11.2256, altitude=3636.363636363636]
GeoPos [latitude=49.307883333333336, longitude=11.1961, altitude=2727.272727272727]
GeoPos [latitude=49.3555875, longitude=11.166599999999999, altitude=1818.181818181818]
GeoPos [latitude=49.40329166666667, longitude=11.137099999999998, altitude=909.0909090909081]
GeoPos [latitude=49.45099583333333, longitude=11.1076, altitude=0.0]
```