Prof. Dr. Christoph Bockisch
MSc Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Companion exercises
Objektorientierte Programmierung: Wintersemester 2021/2022

No. 5, due until 29.11.2021

Task 5.1: Of Max and Min — 5 Points

This task is about implementing different operations on Arrays. You are only allowed to use every type of loop (for, while foreach) and recursion once in this whole task (one per subtask).

Implement the following methods:

a) `void testAll()` that tests all of the methods below. Implement your tests before implementing the other methods.
b) `float getMinimum(float[] numbers)` that returns the minimum of the numbers given.
c) `float getMaximum(float[] numbers)` that returns the maximum of the numbers given.
d) `float calculateAverage(float[] numbers)` that returns the calculated average of the numbers given.
e) `boolean isSorted(float[] numbers, boolean ascending)` that checks if numbers is sorted in ascending (ascending == true= or descending (ascending == false) order.

Task 5.2: Run, Forest, run! 7 Points

The resistance band of a given runner saves its GPS coordinates once every second. A GPS coordinate consists of three values, x, y and z. You can assume that the unit of each value is metres.

The band saves these values on a little SD-card from which they can be extracted as an array. This array looks like this:

```
double[] gps = {x1,y1,z1,x2,y2,z2,...};
```

Write tests for the following methods to analyse the GPS coordinates and implement them afterwards:

a) `double distance(double[] gps)` that approximately calculates the distance run.
b) `double velocity(double[] gps)` that calculates the average velocity of the runner (m/s).
c) `double maxVelocity(double[] gps)` that calculates the maximum velocity the runner achieved (calculated between two coordinates).
d) `double[] partialGPS(double[] gps, double[] start, double[] end)` that returns an array with a part of the given GPS coordinates starting at `start` and ending on `end` (`start` and `end` have a length of 3).

   If either value is `null`, can not be found or if `end` lies before `start`, the original array should be returned.
e) Test your methods properly.

Attention: To calculate a distance in three dimensions, you may use the following equation:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Attention: You are allowed to use `Math.pow`, `Math.sqrt` and `Arrays.copyOfRange(oldArray, startIndex, endIndex)` (endIndex being exclusive) in your solution.

You may use the array below for testing purposes.

```
{-20.0, 0.0,200.0, -18.5, -0.647,200.577, -16.85, -1.237,
 ↪ 201.16, -15.035, -1.763, 201.739, -13.038, -2.219,
 ↪ 202.299, -10.842, -2.599, 202.824,  -8.426,  -2.894,
 ↪ 203.289, -5.769, -3.096, 203.667, -2.846, -3.195, 203.918,
 ↪ 0.369, -3.182, 203.998, 3.861, -3.205, 203.85, 7.284,
 ↪ -3.176, 203.469, 10.638, -3.209, 202.868, 13.926, -3.175,
 ↪ 202.06, 17.147, -3.209,201.059, 20, -3.174, 199.877}
```