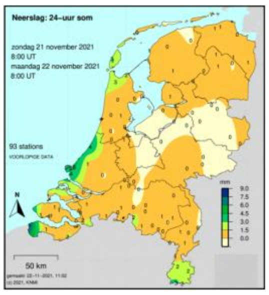
Algorithms and Data Structures

Sets, Maps, Streams: KNMI Climate Analysis

Assignment-4

Version: November 22th, 2021

Koninklijk Nederlands Meteorologisch Instituut
Ministerie van Infrastructuur en Waterstaat



Introduction

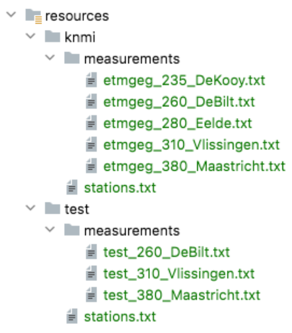
In July 2021, North-Western Europe were caught by the surprise of unprecedented heavy rainfalls. Several Dutch, Belgian and German citizens and companies were impacted by severe damage to their properties when the Meuse and Geul rivers flooded. You are requested to process some historical meteorological data that may give some leads of what trends are developing and what kind of extreme weather situations we still can expect from the future.

As of the year 1901 The "Koninklijk Nederlands Meteorologisch Instituut" has been tracking climate data from multiple weather stations across the Netherlands. At https://www.knmi.nl/nederland-nu/klimatologie/daggegevens you find the historical data of daily measurements of these stations in .csv format. The key questions are:

1. What is the trend of the average temperature in the Netherlands over the past century?
2. Is there a trend in events of extreme volumes of precipitation (neerslag) along this period?
3. What about the extremes in wind?

You will use Set and Map data structures to organize this data and Stream processing to perform the analysis.

The data



```
resources
  knmi
    measurements
      etmgeg_235_DeKooy.txt
      etmgeg_260_DeBilt.txt
      etmgeg_280_Eelde.txt
      etmgeg_310_Vlissingen.txt
      etmgeg_380_Maastricht.txt
    stations.txt
  test
    measurements
      test_260_DeBilt.txt
      test_310_Vlissingen.txt
      test_380_Maastricht.txt
    stations.txt
```

For the purpose of this assignment you will use some 200,000 consolidated daily meteorological records of five weather stations: "De Kooy", "De Bilt", "Eelde", "Vlissingen" and "Maastricht". The source data has been downloaded already for your convenience and configured in the resources/knmi folder of 'Assignment-4-KNMI_Climate_Analysis-starter.zip'. The files in resources/test contain a small extract of this data for testing purposes.

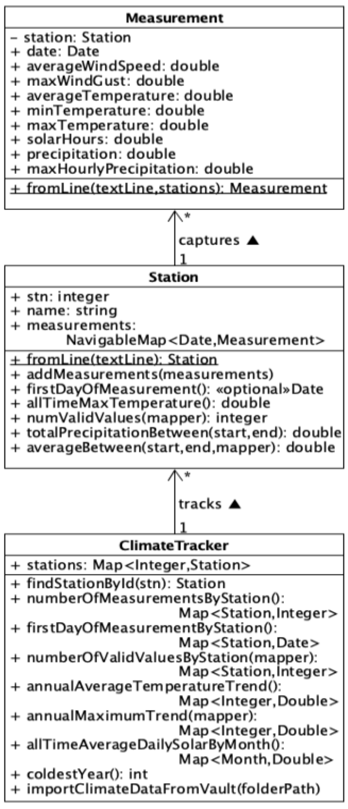
The format of the data is documented in the header of each file. We will only use the following columns of the comma-separated data:

Column Nr	Field code / Header	Description	Unit of measure
00	# STN	Station number: a unique integer id. Should be the same for all rows in a single file	-
01	YYYYMMDD	Measurement date: 8-digit integer value in YYYYMMDD format	-
04	FG	Daily average windspeed	0.1 m/s
09	FXX	Maximum wind gust	0.1 m/s
11	TG	Daily average temperature	0.1 °C
12	TN	Daily minimum temperature	0.1 °C
14	TX	Daily maximum temperature	0.1 °C
18	SQ	Number of hours of sunshine	0.1 h
22	RH	Daily precipitation (rain, snow, hail)	0.1 mm -1 indicates < 0.05 mm
23	RHX	Maximum hourly precipitation across the day. (heavy bursts)	0.1 mm -1 indicates < 0.05 mm

The units of measure in the files are a bit unconventional. All values are represented by integers. When importing the data, you need to multiply them by 0.1 to scale to the appropriate numerical value.

Application design.

Here you find a UML diagram of the proposed application design. The Station class represents a weather station which has organised all its daily measurements since its first day of operation in a NavigableMap. This map holds the data in a Binary Search Tree organized by date. The Station class also implements some basic processing on its collection of Measurements.


Measurement
- station: Station
+ date: Date
+ averageWindSpeed: double
+ maxWindGust: double
+ averageTemperature: double
+ minTemperature: double
+ maxTemperature: double
+ solarHours: double
+ precipitation: double
+ maxHourlyPrecipitation: double
+ fromLine(textLine,stations): Measurement
*
captures ▲
1
Station
+ stn: integer
+ name: string
+ measurements: NavigableMap<Date,Measurement>
+ fromLine(textLine): Station
+ addMeasurements(measurements)
+ firstDayOfMeasurement(): «optional»Date
+ allTimeMaxTemperature(): double
+ numValidValues(mapper): integer
+ totalPrecipitationBetween(start,end): double
+ averageBetween(start,end,mapper): double
*
tracks ▲
1
ClimateTracker
+ stations: Map<Integer,Station>
+ findStationById(stn): Station
+ numberOfMeasurementsByStation(): Map<Station,Integer>
+ firstDayOfMeasurementByStation(): Map<Station,Date>
+ numberOfValidValuesByStation(mapper): Map<Station,Integer>
+ annualAverageTemperatureTrend(): Map<Integer,Double>
+ annualMaximumTrend(mapper): Map<Integer,Double>
+ allTimeAverageDailySolarByMonth(): Map<Month,Double>
+ coldestYear(): int
+ importClimateDataFromVault(folderPath)

De Measurement class stores all data values from the various types of measurement instruments that are available at the Station. Some values may be missing for some dates, in which case you store a Double.NaN and do not include that value in any processing.

The ClimateTracker class organizes all Stations and provides aggregate processing of the data across multiple stations. The method 'importClimateDataFromVault' imports all data files into your data structures. The methods 'numberOfMeasurementsByStation', 'firstDayOfMeasurementByStation' and 'numberOfValidValuesByStation' support quality assurance that your data was loaded correctly and that you prove successful exclude missing values from your stream calculations.

Thereafter you can proceed with the other methods and calculate some results of your climate analysis.

Details about the functionality and constraints of each of the methods are provided in the JavaDoc of the code in the starter project.

Assignment.

Unpack the starter project into your development environment and proceed with implementing the missing parts:

1. Complete the implementations of the Station and Measurement classes:
   a) you need to complete the toString method to obtain a proper text representation in output. Stations shall be displayed with format: stationNumber/name
   b) you need to complete the static fromLine methods which you will need for converting text lines of the source files into object instances. In Measurement.fromLine you should apply the scaling of the unit of measure and replace missing values with Double.NaN
   c) you need to implement additional methods to support Hash and Tree data structures.
2. Implement the missing parts of the ClimateTracker.
3. Leverage the capabilities of Sets, Maps and Streams in your implementations. Specifically, you shall (investigate and) use each of the following Java methods at least once in a purposeful way:

Set.of or Set.copyOf	.subMap	.map() or .mapToInt() or .mapToDouble()
.filter()	.flatMap()	.average() or Collectors.averagingDouble()
.max() or .min() or .sorted()	.orElse()	.count() or .sum()
.collect()	Collectors.toMap() or Collectors.groupingBy()	

4. You may add more private and public methods to any of the classes, but not change any signature of the specified public methods.
5. Run all provided unit tests and add unit tests for relevant code or situations that are not covered by the provided unit tests.
6. Run the main program and compare the results with below output.

7. Prepare your report with explanations of seven code snippets. Stream operations can be explained by explaining what is represented by the data at each stage of the stream. Also include your full console output at the end of your report.

Results.

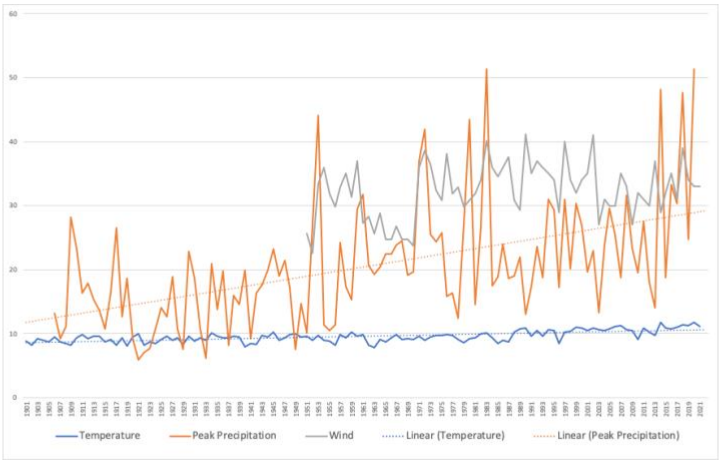
If all methods have been implemented correctly, then the main program will provide the following 10 results in its console output:

```
Welcome to the HvA Climate Analyser


1. Total Number of measurements by station:
{260/De Bilt=44152, 310/Vlissingen=42326, 280/Eelde=42326, 235/De Kooy=42326, 380/Maastricht=42326}
2. First day of measurement by station:
{260/De Bilt=1901-01-01, 310/Vlissingen=1906-01-01, 280/Eelde=1906-01-01, 235/De Kooy=1906-01-01, 380/Maastricht=1906-01-01}
3. All-time maximum temperature in de Bilt = 37,5 degC
4. Number of valid daily precipitation measurements by station:
{260/De Bilt=42296, 310/Vlissingen=23715, 280/Eelde=23698, 235/De Kooy=23698, 380/Maastricht=23698}
5. Total precipication in de Bilt in 1963 = 777 mm
6. Annual trend of average temperatures (in degC):
{1901=8.783561643835617, 1902=8.245205479452055, 1903=9.166575342465753, 1904=8.93360655737705, 1905=8.713698630136987,
1906=9.463703284258212, 1907=8.674849315068494, 1908=8.40584699453552, 1909=8.191123287671234, 1910=9.324383561643836, … ,
2021=11.151987577639751}
7. Annual trend of maximum hourly precipitation (in mm):
{1906=13.200000000000001, 1907=9.200000000000001, 1908=11.100000000000001, 1909=28.200000000000003, 1910=23.400000000000002,
1911=16.400000000000002, 1912=17.900000000000002, 1913=15.3, 1914=13.700000000000001, 1915=10.700000000000001, …
,2020=51.300000000000004, 2021=67.0}
8. Annual trend of maximum wind gust (in m/s):
{1951=25.7, 1952=22.6, 1953=33.4, 1954=36.0, … , 2016=35.0, 2017=31.0, 2018=39.0, 2019=34.0, 2020=33.0, 2021=33.0}
9. All-time monthly profile of daily solar hours:
{JANUARY=1.7026519583263693, FEBRUARY=2.6604576967912146, MARCH=3.9071492354057376, APRIL=5.574926338898839,
MAY=6.822385310860587, JUNE=6.862594277160459, JULY=6.451513802315227, AUGUST=6.111598467603131,
SEPTEMBER=4.852380952380952, OCTOBER=3.3691713014460514, NOVEMBER=1.9094002306805076, DECEMBER=1.3689705469845723}
10. Coldest year = 1963
```

If you export the data of the trend maps into a graph as shown here, you can draw your own conclusions about future expectations of the Dutch climate…

(You may share your conclusions in your report, but unfortunately, that cannot be awarded with extra ADS points…)



Grades.

At DLO you find the standard rubrics for this assignment with the three grading categories: Solution (50%), Report (30%) and Code Quality (20%). For a sufficient score on the solution category, you should have used all set, map and stream features as listed before in a useful way and have calculated correctly 7 out of the 10 items of the console output.