UserZoom Analytics Team - Technical Test

Summary

We have access to a system that allows to record the speed of a F1 car. The measurements are collected by two high-precision sensors that sample the speed every second. Each sensor stores their measurements in its own separate CSV file.

We want to write a program that reads and processes this data to output a summary of it.

The solution should be presented through a public repository that **must** include a README.md document where you explain how to execute the application.

This test is aimed at knowing how you code, the technical decisions you take along the process and the cleanliness of your code.

Guidelines

- **Max. duration: 4-6 hours.** Don't worry if your program isn't complete, we'd rather value that the items that you do finish meet your quality standards!
- The application must be written in **Java**.
- Use of third party libraries is allowed.

Description of the sensor system

1. The CSV files generated by the sensors contain, for each line, the following information, delimited by commas:

Field index	Field description
1	The timestamp of the sample, represented as the number of seconds that have elapsed since 00:00:00 Thursday, 1 January 1970
2	A fractional number representing the speed of the car in km/h

2. The program will run while the sensors are actively collecting data. Thus, in any moment, new measurements can appear at the end of each file. Keep in mind, however, that the CSV files are append-only, so the pre-existing content will never be modified.

3. Due to clock readjustments in the sensor controller, multiple samples can appear for the same timestamp in any of the two sensors. Also, take into account that both sensors can contain information for different sets of timestamps. For example:

Example content of the files at time t_0:

SpeedSensor1.csv	SpeedSensor2.csv
1548154670,281.0	**1548154670,282.0**
1548154671,282.0	**1548154671,283.0**
1548154671,285.0	**1548154672,288.0**
1548154672,287.0	**1548154673,289.0**
1548154674,287.0	**1548154674,287.0**
1548154676,290.2	
1548154677,291.5	

Example content of the files at time t_1:

SpeedSensor1.csv	SpeedSensor2.csv

1548154670,281.0	**1548154670,282.0**
1548154671,282.0	**1548154671,283.0**
1548154671,285.0	**1548154672,288.0**
1548154672,287.0	**1548154673,289.0**
1548154674,287.0	**1548154674,287.0**
1548154676,290.2	**1548154678,294.3**
1548154677,291.5	**1548154678,294.6**
1548154679,294.1	**1548154679,296.0**
	1548154681,299.5

Expected program behaviour

The program should consolidate the repeated data and return, for each timestamp, the average value of all the speeds recorded for that timestamp. For example, based on the data available at time t_0:

AverageSpeed_0.csv
1548154670,281.5
1548154671,283.3333333333333
1548154672,287.5
1548154673,289
1548154674,287
1548154676,290.2
1548154677,291.5

This process must be repeated periodically.

In particular, the program should:

- On the initial execution, it should read the current content of the 2 data files and write a new file called **AverageSpeed_0.csv** with the format described above.
- The program must be kept running indefinitely and, every 10 seconds, it should check for new data in any of the CSVs and write a new file called **AverageSpeed_N.csv** (with N=1, 2 etc), recalculating the average speed for each timestamp.

Other considerations

- Do use some software layer architecture pattern, so that each layer has limited and separated responsibilities.
- Implement some unit tests.
- The CSV files can be potentially big, avoid re-reading the same data and try to parallellise the reading steps.
- Assume you can acquire read permissions on the file.