Time Series fetching and calculations - Java homework assignment task

Introduction

In TSRD we pull time series market data from various external providers and perform hundreds of calculations on the obtained data in order to synthesize additional factors required by the downstream systems to compute the market risk levels.

Your task

In this excercise, we would like to emulate a simplified version of the scenario described above, by asking you to develop a simple application that is going to read time series data from the specified input file and then applying several simple algorithms to the data received,

Input data

The input file format is very simple and it's basically a list of data records (one record - one input file line), where each record looks as follows:

<INSTRUMENT_NAME>,<DATE>,<VALUE>

and:

INSTRUMENT_NAME - name of the financial instrument associated with a given data point; a real-life example would be e.g. USD/EUR exchange rate or a company stock name etc.

DATE - a date on which a given instrument had a given value

VALUE - evaluation of a given instrument on a given date (e.g. price)

Example: *USDPLEXRATE,12-Mar-2015,4.02*

Calculations

Depending on the instrument, we would like to apply different algorithms to the received time series, namely:

- **INSTRUMENT1** – mean of all the values
- **INSTRUMENT2** – mean of all the values from ***November 2014***
- **INSTRUMENT3** – any other statistical calculation that we can compute "on-the-fly" as we read the file (it's up to you)
- any other instrument from the input file - sum of the newest 10 instrument values (in terms of the date).

The values above, however, are not just the values read from the input file as we would like to emulate the fact that in the real life there are often multiple factors influencing calculations performed on the instrument prices.

So as part of your task we would like you to set up a database with only one table, called INSTRUMENT_PRICE_MODIFIER with the following columns:

column	meaning
ID	primary key (just a db sequence number, no business meaning)
NAME	instrument name (corresponding to the names from the input file)
MULTIPLIER	double value that specifies the factor by which the original instrument value should be multiplied.

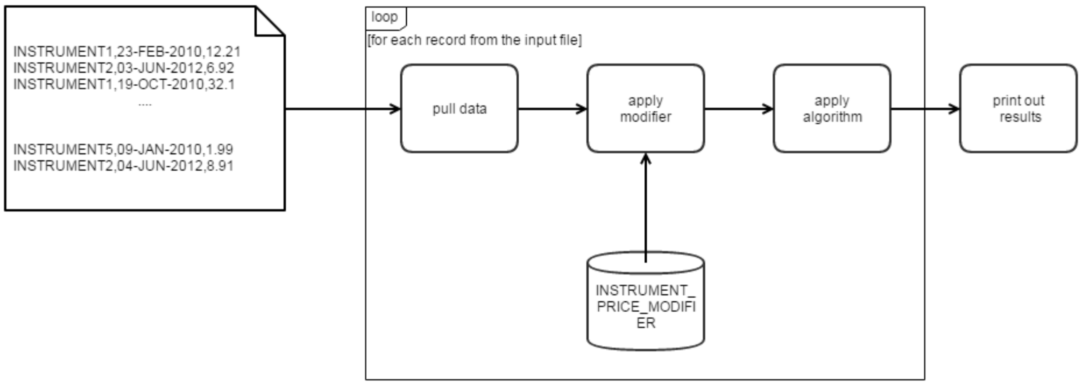
Moreover, we are only interested in business dates, i.e. Monday-Friday. Discard entries with non-business date.

To summarize then, in order to determine the final value of a given instrument price for a given date you should do the following:

1. read the line from the input file;
2. check if it's a record for a business date or not, if not - discard;
3. if it's a business date record - query the database to see if there is an entry for the <INSTRUMENT_NAME> you read in the 1st step;
4. if there is - multiply the original <VALUE> by the factor you found in the step 3;
5. if there is no entry - simply use the original <VALUE> from the file.

Please assume that the values in the INSTRUMENT_PRICE_MODIFIER table can change frequently independently of the process you're implementing, but not more often than once every 5 seconds.

The following picture summarizes the task:


INSTRUMENT1,23-FEB-2010,12.21
INSTRUMENT2,03-JUN-2012,6.92
INSTRUMENT1,19-OCT-2010,32.1
....
INSTRUMENT5,09-JAN-2010,1.99
INSTRUMENT2,04-JUN-2012,8.91
loop
[for each record from the input file]
pull data
apply modifier
apply algorithm
print out results
INSTRUMENT_PRICE_MODIFIER

The final results can be printed out simply on the console, we **don't want** you to come up with any fancy UI or anything. Focus on the core of the assignment.

Technical requirements

- create the **Maven (3+)** project for your solution; we must be able to build your program using Maven;
- use at least the **JVM 1.7** version;
- use a lightweight database (e.g. h2, Derby), memory or file-based - your choice, that doesn't require installation of any prerequisite software and can be launched and torn down easily as part of your program run;
- you’re allowed to use any Open Source libraries and frameworks you like, so e.g. you’re free to use Mockito, JUnit, Guava, Apache Commons, Spring etc.
- provide instructions on how to run your program (e.g. a java command with arguments allowing us to execute your program)
- provide a description of your solution: important decisions made etc.

Guidelines and gotchas

- model the problem using OO principles; think of what's likely to be a changing factor in this application and what's fairly constant; design your solution accordingly;
- test your work;
- bear in mind that your program should also work for a file that has many gigabytes of data; it should be capable of handling it without crashing along the way;
- assume it's vitally important to calculate the abovementioned metrics as quickly as possible; your solution should scale effectively;
- do not try to show off and don't over-engineere: your solution won't be evaluated on the basis of the number of frameworks used but rather on the beauty of your design and the efficiency of the algorithms used;
- send the source code back via email to us in a zip archive (**but please don't include any executables or jars !** otherwise it will be blocked by our corporate SPAM filter)

ABOVE SPECIFICATION IS AND ALWAYS WILL BE USED FOR CANDIDATE SKILLS EVALUATION ONLY.