

## Code Challenge: Senior Java Developer

Please find the Coding Challenge for Senior Java Developer below. The solution should not take more than 3 hours for a suitably qualified developer. If there are any issues in understanding anything, please contact us.

### Tips for this Coding Challenge

- You only get one opportunity to submit a solution to us!
- Read the document completely and consider the requirements.
- Spare some time to develop your response.
- Don't try to attempt to do the Coding Challenge in multiple stages – your head will not be in the right place and you will come up with an inferior response.

### Overview

This Coding Challenge is a good opportunity for you to show how you approach a specific problem. We are looking for programmatic, maintainable code, **as well as tests** to prove your code works.

Feel free to create additional classes or use any 3rd party libraries you need to support the design of your solution.

However, **do not use** Spring, Hibernate, or in-memory databases as we want you to *find a simple way of solving the problem!* You will be **penalised for using tools** such as Spring, Hibernate or **any** in-memory database.

When creating your solution, please **build this in a manner** as if this was being deployed into a production environment.

## The Code Challenge - The Transaction Analyser

Consider the following simplified financial transaction analysis system. The goal of the system is to display statistical information about processed financial transactions.

*A transaction record* will contain the following fields:

**ID** - A string representing the transaction id.

**Date** - The date and time when the transaction took place (format "DD/MM/YYYY hh:mm:ss").

**Amount** - The value of the transaction (dollars and cents).

**Merchant** - The name of the merchant this transaction belongs to.

**Type** - The type of the transaction, which could be either PAYMENT or REVERSAL.

**Related Transaction** - (Optional) - In the case of a REVERSAL transaction, this field will contain the ID of the transaction it is reversing.

### The Problem

The system will be Initialised with an input file in CSV format containing a list of transaction records.

Once initialised, the system **should report** the total number of transactions and the average transaction value for a specific merchant in a specific date range.

An additional requirement is that, if a transaction record has a REVERSAL transaction, then it should not be included in the computed statistics, even if the reversing transaction **is outside** of the requested date range.

Input CSV Example:

ID, Date, Amount, Merchant, Type, Related Transaction

WLMFRDGD, 20/08/2020 12:45:33, 59.99, Kwik-E-Mart, PAYMENT,

YGXKOEIA, 20/08/2020 12:46:17, 10.95, Kwik-E-Mart, PAYMENT,

LFVCTEYM, 20/08/2020 12:50:02, 5.00, MacLaren, PAYMENT,

SUOVOISP, 20/08/2020 13:12:22, 5.00, Kwik-E-Mart, PAYMENT,

AKNBVHMN, 20/08/2020 13:14:11, 10.95, Kwik-E-Mart, REVERSAL, YGXKOEIA

JYAPKZfZ, 20/08/2020 14:07:10, 99.50, MacLaren, PAYMENT,

Given the above CSV file and the following input arguments:

fromDate: 20/08/2020 12:00:00

toDate: 20/08/2020 13:00:00

merchant: Kwik-E-Mart

### The output will be:

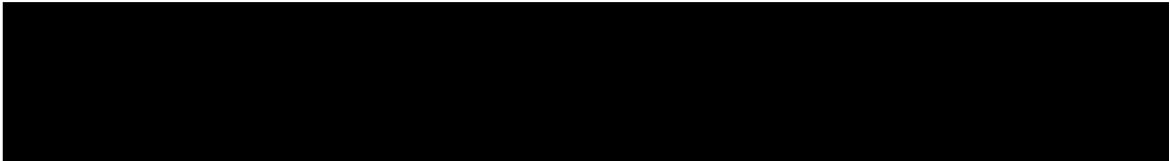
Number of transactions = 1

Average Transaction Value = 59.99

### Assumptions

For the sake of simplicity, you can assume that Transaction records are listed in correct time order.  
The input file is well formed and is not missing data.

### Deliverables

1. Please send us the source code and *make sure there are no compilation errors*.
2. Use as much of Java 8 (and beyond) components or Kotlin components as you can.
3. 
4. Please include a README file at the root of the project describing how to build and run your solution.
5. Whether it's via a main class or a unit test method that we can modify, we should have an easy way of providing the solution *with our own csv file* and input params to validate your solution.

This is a test of your capability to:

- properly read requirements from end to end.
- utilise your understanding of fundamentals of code design and logic.
- use of Java 8 and Kotlin frameworks.
- make a simple and elegant solution – **not a complex one!**

Again, your solution should be “production-ready”. Good luck and we look forward to seeing your response.