Functional Programming with Scala

ITI 43 Project

Problem Statement:

A huge retail store wants a rule engine that qualifies orders' transactions to discounts based on a set of ***qualifying rules***. And automatically calculates the proper discount based on some ***calculation rules*** as follows:

QUALIFYING RULES	CALCULATION RULE
less than 30 days remaining for the product to expire (from the day of transaction, i.e. timestamp)	If 29 days remaining -> 1% discount if 28 days remaining -> 2% discount if 27 days remaining -> 3% discount etc …
Cheese and wine products are on sale	cheese -> 10% discount wine -> 5% discount
Products that are sold on 23rd of March have a special discount! (Celebrating the end of java project?)	50% discount !!
bought more than 5 of the same product	6 – 9 units -> 5% discount 10-14 units -> 7% discount More than 15 -> 10% discount

- Transactions that didn't qualify to any discount will have 0% discount.
- Transactions that qualified to more than one discount will get the top 2 and get their average.

- We need the system to be up and running 24 hours. Raw transactions' files will be pushed to the following machine (IP: X.X.X.X) on the following path (……/raw_data). We need the calculations to be done automatically once the file is pushed.

- After ingesting the raw data and calculating the discount please also calculate the final price and load the result in a database table of your choice.

- The raw data needs to be removed from the raw_data directory after successfully writing the processed data in the database.

- It is required to log the engine's events in a log file "rules_engine.log". Please use the following logging format.

  TIMESTAMP LOGLEVEL MESSAGE

Technical considerations:

- We can have wrappers of impure functions to interact with the outside world. However, the core logic must be written in a pure functional manner.

In the core functional logic:

- Use only vals, no vars allowed.
- No mutable data structures allowed.
- No loops allowed.
- No Null values.
- Make sure all your functions are pure:
  - Output depends solely on input.
  - Input to the function doesn't get mutated.
  - Has a predictable behavior.

- The code should be well commented.

- The code should be clean, easy to read and self-explainable. (remember that one of the objectives of functional programming is having a readable code)

Delivery:

- Project should be delivered as a Github repo with a well written readme file.
- Deadline for sending Github repo link is March 29th at 12 Midnight.
- Project discussion is March 30th.

Teams:

Team 1: Mohamed Abdelsalam – Mariem – Sherouk

Team 2: Malak – Ibrahim – Hesham

Team 3: Sara – Islam – Mohamed Fathy

Team 4: Ahmed Ali – Kareem – Ahmed Ehsan

Team 5: Dina – Habiba – Yousra

Team 6: Mohamed Labib – Abdelaziz – Fatma - Aly