Product Age Calculator - Manual

1. Usage

The jar is executed by normal java command “java -jar” followed by its name (product-age-calculation.jar) and the parameters in quotes. In the section 2 - Parameters, you can learn more about the parameters.

The jar will:

1-Receive two dates;

2-Read an internal File with all orders registered (this file works as our database, it can be modified later).

3-Find all orders made between those dates;

4-Retrieve all products inside each of the orders found in 2.

5-Classify each product inside an age. The ages are based on intervals given by the user (or default ones).

6-Count how many of the products were classified in each age

7-Create and fill an OutputData file with all ages and quantity of orders

8-Read the content of the OutputData file and print it in the shell output

1. Parameters

The valid parameters are:

[mandatory]”<date 1>”

[mandatory]”<date 2>”

[optional] “n”

[optional] “to<timeout limit in seconds>”

[optional] “<interval 1>” [optional] “<interval 2>” [optional] “<interval 3>” …

The first two arguments are mandatory and must be put in that position. All the optional arguments can be put at any order.

Example -> java -jar product-age-calculation.jar "2022-01-01 00:00:00" "2023-01-01 00:00:00" ">1" "<123" ">123" "5-19" "to10" "n"

**2.1 Date 1 and Date 2**

These parameters are mandatory and will be used to search for all orders made in that period of time between the two dates. The program will automatically determine the later date, so the user can send the dates in any order.

Example ->

java -jar product-calculation.jar "2022-01-01 00:00:00" "2023-01-01 00:00:00"   
is the same as:

java -jar product-calculation.jar "2023-01-01 00:00:00" "2022-01-01 00:00:00"

**2.2 Option Now (“n”)**

The program will calculate the age of a product using the date of the order as its comparison date.  
 For instance: A product created in 2020-01-01 is 1month old in an order made in 2020-02-01. The same product will be 12months old in an order made in 2021-01-01.

The idea behind this implementation is that the Age is always a relevant information in the context of when the order was made.

However, if the user want to consider today’s date in the calculation instead, they can use the option “n”, and so the platform will calculate the product age based on today’s date.

Therefore: A product created in 2020-01-01 would be 25months old today (2022-03-11, when this document is being written), independently of when the order was made.

**2.3 Option Interval**

Intervals are the way to classify the age of a product. The user can use “>n” or “<n” to search for products younger or older then a certain amount of months (n being the number of months) or it can use n1-n2, to search for products in a specific range of age (between n1 months and n2 months). If no intervals are sent in the arguments, the system will use the default intervals, those being: “1-3”, “4-6”, “7-12” and “<12”.

**2.4 Option Timeout (“to”)**

The solution was designed considering that each given interval needs to be matched with all orders. With that in mind, a large amount of intervals can scalate very quickly if we use a numerous base of orders, and so multi-threading was applied to enhance the overall performance of the product.

One thread is created to check each of the given intervals, and they operate in parallel to reduce the waiting time after executing the jar.

However, if the user decides to keep a bare minimum of time to wait, they use the “toN” option to set a time limit (in seconds) they want to wait before the system gives its return (N being the number of seconds).

But don’t worry. Even if the output is not ready before the timeout, the threads will still generate an OutputData file with the content the user needs.