# **Excel Reader Tutorial**

## Introduction

The “Excel Reader” application is designed to transform the content of an input Excel file and return a transformed output file, which matches the requirements specified by the user. The requirements in this case mean a configuration file which acts as an instruction for the application, so that it can carry out the designated tasks.

When the application is run, considering the content of the configuration file, the input data is going to be transformed and the relevant output, will be returned in the form of an output Excel file.

## Excel Reader Functionality

The user now has, the following functionality of the application at his disposal:

1. **FUNCTIONALITY NO.1** - *Filter and Replace the data from the input file and return the results to the output file.*

2. **FUNCTIONALITY NO.2 -** *Make use of the Excel Functions, to make changes to the data,* in the input file and return the results to the output file.

3. **ADDITIONAL FUNCTIONALITY -** *Specify the input and output file, so that the data can be easily transferred, to and from, the chosen data files.*

**FUNCTIONALITY NO.1**

*Filter and Replace the Data*

When the user reads in the configuration file, he can filter and replace the data according to the structure of the JSON object.

Considering the configuration file, the user can choose a column for which the data is going to be filtered(**columnName**).

When the user chooses a column, he can pick the cells to be replaced (**before**) and replace that cell with a value of his liking(**after**). This means that the content of the input file is going to be transformed and the relevant output, is going to be visible in the output file.

Below I present the functionality mentioned above, present within the configuration file:

***FUNCTIONALITY NO.1***

**"replacements":**

**[**

**{**

**"columnName":"Numbers",**

**"before":"123","after":"456"**

**},**

Considering the above snippet of code, when the user runs the application, the app is going to choose the Numbers column present within the Excel worksheet.

In this column the application is going to find all cells with values: “123” and replace those values with “456”.

**FUNCTIONALITY NO. 2**

*Making changes to the data considering Excel Functions*

The user can make use of the Excel Functions to make changes in the data within the Excel worksheet.

The Excel functions which the user has currently at his disposal:

* EQUALS – considering the filter value, the user can check if that filter value, is equal to a specific cell in a column, within the chosen Excel file. A separate column is going to be created which contains the Results for the EQUALS function, in the output file.
* NOT EQUALS - considering the filter value, the user can check if a filter value, is not equal to a specific cell in a column, within the chosen Excel file. A separate column is going to be created which contains the Results for the NOT EQUALS function, in the output file.
* CONTAINS – considering the filter value, the user can check if a specific cell in a column contains the filter value, within the chosen Excel worksheet. A separate column is going to be created which contains the Results for the CONTAINS function, in the output file.
* NOT CONTAINS - considering the filter value, the user can check if a specific cell in a column does not contain the filter value, within the Excel worksheet. A separate column is going to be created which contains the Results for the NOT CONTAINS function, in the output file.
* LESS – considering the filter value, the user can check, if a cell in a column is less than the filter value, within the Excel worksheet. A separate column is going to be created which contains the Results for the LESS function, in the output file.
* GREATER - considering the filter value, if a cell in a column is greater than the filter value, within the Excel worksheet. A separate column is going to be created which contains the Results for the GREATER function, in the output file.
* BEFORE – to be developed.
* AFTER – to be developed.
* BETWEEN – to be developed.

The below snippet of code from the configuration file, corresponds to **FUNCTIONALITY NO.2** in the application.

**FUNCTIONALITY NO.2**

"filteringList":

[

{

"columnName":"Country",

"filterValue":"Germany, India, Poland",

"filterOption":"EQUALS"

},

{

"columnName":"Country",

"filterValue":"Finland",

"filterOption":"NOT\_EQUALS"

},

Below I will try and explain what the individual pieces of code mean in a little bit more detail:

The **filteringList** contains filteringList arguments. Each argument corresponds to a block of code which is enclosed within the “curly” brackets. -> {}

Each argument contains the following elements:

- **column names,**

**- filter values**

**- filter options**

By making use of filteringList arguments, the user can choose a column of his liking and in that column, he is able to make use of the Excel functions.

Functionality of the Excel functions corresponds to the **filterOption** which are used with the conjunction of **filterValues.**

In the above example we are making use of EQUALS and NOT\_EQUALS Excel functions with the **filterValues** such as: Germany, India, Poland or Finland.

This basically means that each **filterValue** will be used with the conjunction of an Excel Function**(filterOption)**, to carry out the relevant operations on the data.

Considering the whole argument, which is presented in the below piece of code:

{

"columnName":"Country",

"filterValue":"Germany, India, Poland",

"filterOption":"EQUALS"

},

Taking it into account, we can check if the cells in the column “Country” are equal to Germany, India or Poland. If this is the case, another column is going to be created at the end of the output file so that it will contain the relevant results for the Countries, which satisfy the match.

**FUNCTIONALITY NO.3**

*Additional Functionality*

* **Specifying the input and output file**

For the application to run the **input** and **output** files must be specified.

In the configuration file there are two main statements that correspond to the **input** and **output** **file**:

1. **inputFile** entry in the **configuration file**:

{

"C:\\FullyFunctionalApllicationToBeRunFromTheCommandLine\\ChargedHoursNew.xlsb",

1. **outputFile** entry in the **configuration file**:

"outputFile":

{

"newFileName":"C:\\FullyFunctionalApllicationToBeRunFromTheCommandLine\\

ChargedHoursNew.xlsx",

"columns": ["Firstname", "Lastname"]

}}

Considering the two examples mentioned above the user can specify the input and output file of his liking. The data can then be transferred between the two.

**!!! Important!!!**

The whole of the application was designed as such that the input file can be in xlsb format.

That was an additional challenge considering the application requirements.

If the user has such a desire, he can specify the input file to be of some other format e.g xls or xlsx.

The logic behind converting the input file is that the user must specify the input file to be in a valid Excel Format(xls,xlsx or xlsb). That input file will later be converted into something which is more convenient to use for a common user(e.g an Excel File in the xlsx format).

After the input file is read in, it is later converted to the output file so that the format of the data can be chosen according to the users liking. In this example the file format of the output file is .xlsx.

Considering the functionality mentioned above it was developed with the use of “**Aspose Cells**”. What exactly is “**Aspose Cells”**?

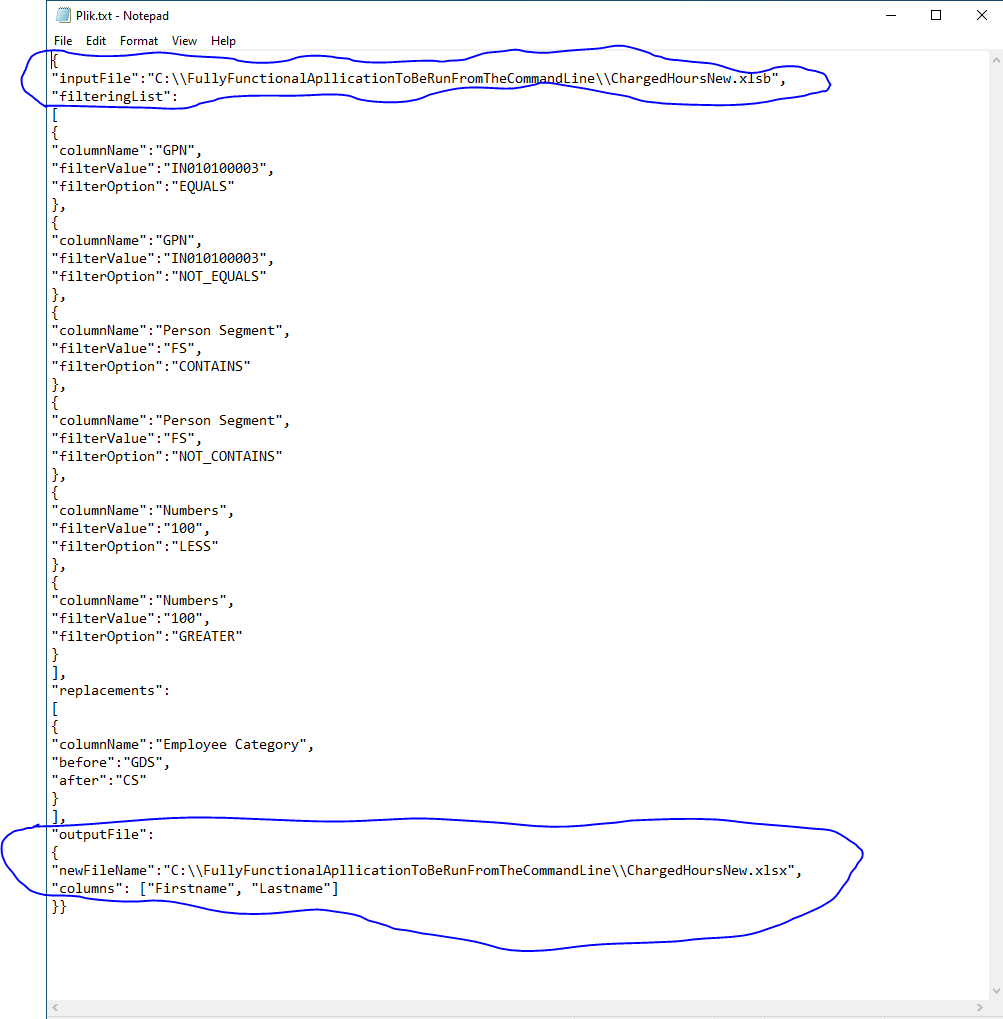
According to : Aspose (no date) *Aspose*, *Java Excel API*. Available at: https://products.aspose.com/cells/java/ (Accessed: 28 November 2023).

“  
Aspose.Cells for Java is an award-winning Excel Spreadsheet Processing API that allows the developers to embed the functionality to read, write, manipulate, convert & print spreadsheets in their own applications, without needing Microsoft Excel application.

Aspose.Cells for Java is a mature and feature-rich library, offering many functions that are way beyond the simple data export capabilities of other components available on the market. API provides the capabilities to export data, format spreadsheets to the most granular level, import & export images, create charts & Pivot Tables, apply & calculate complex formulas using formula calculation engine, stream Excel data & save the result in various formats.“

The Aspose Cells library was imported into the app, as jar file and now it is an integral part of the whole application structure.

Coming back to the matter of input and output files considering, the whole of the configuration file, the input and output file entries are visible on the screenshot which I present below:



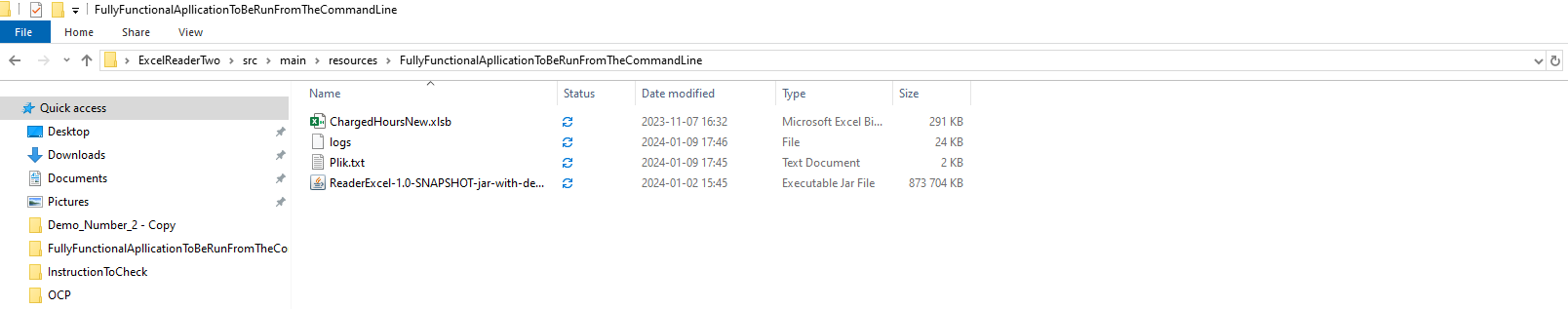
The above screenshot also presents a full configuration file. If the settings shown above were to be loaded the application would execute and return the relevant data which are of interest to the user.

# Structure of the folder with the data:

1. Plik.txt file – This is the configuration file. It contains a JSON object, which is used to process the data according to the requirements specified by the user. The name of the configuration file can be different and adjusted to the user’s personal needs.
2. ReaderExcel-1.0-SNAPSHOT-jar-with-dependencies.jar - this is the JAR file which is required to run the whole of the application from the Command Line.
3. Logs file – this file contains the logs from the whole of the application. The logs are the messages or the errors, which would normally be outputted in the command line, after the application was run.
4. Input data file e.g ChargedHoursNew.xlsb – this file is used to generate an output data file, which contains the results, that are of the interest to the user.
5. An output data file e.g ChargedHours.xlsx – this file is visible in the data folder after the application finished its run successfully. This file contains all the results, which are wanted by the user.

**PLEASE NOTE!!!**

The folder which contains the initial data, which is required for the application to run successfully should look like the following:

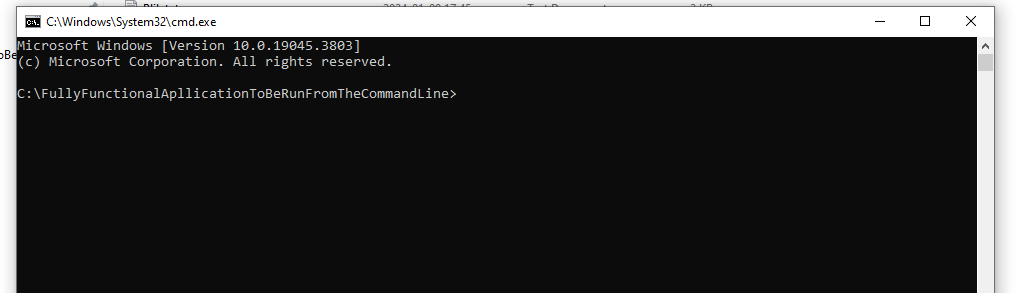


## RUNNING THE APPLICATION

In order to run the whole of the application the following steps need to be executed:

**1.**

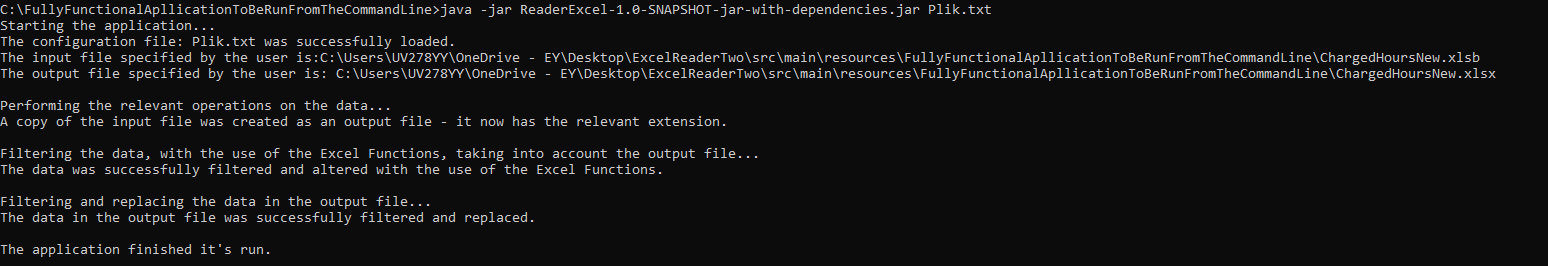
From the command line, the user needs to change the directory, to the directory where the files required to run the application, are contained, in this case it’s:



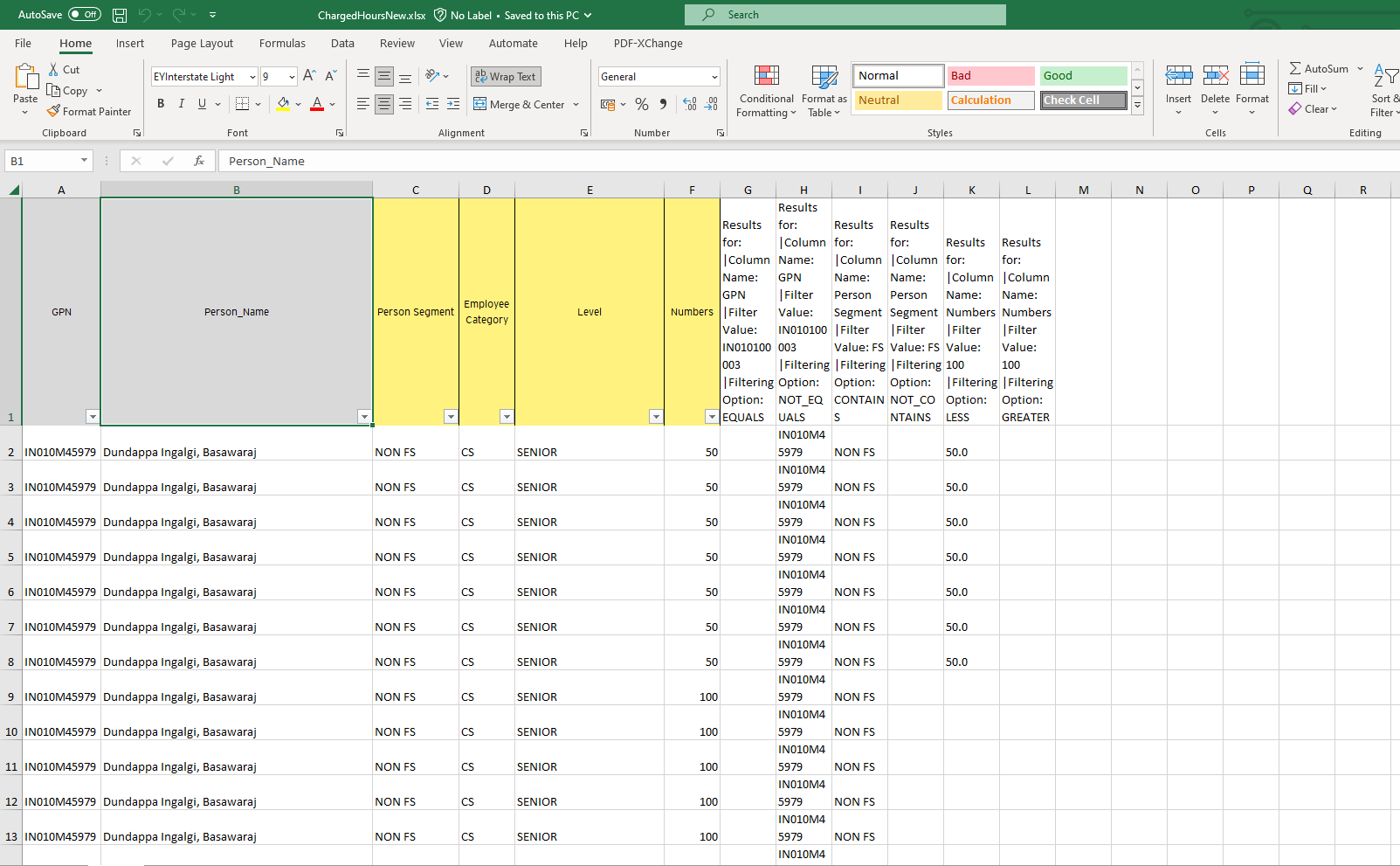
2.When the user is present in the directory with the application functionality, from the command line, he needs to execute the following command in order to run the whole of the app:

**java -jar ReaderExcel-1.0-SNAPSHOT-jar-with-dependencies.jar Plik.txt**

After the command within the command line was executed…



The changes to the chosen output file inside the data folder should now be visible.



## 

## MAKING CHANGES TO THE CONFIGURATION FILE

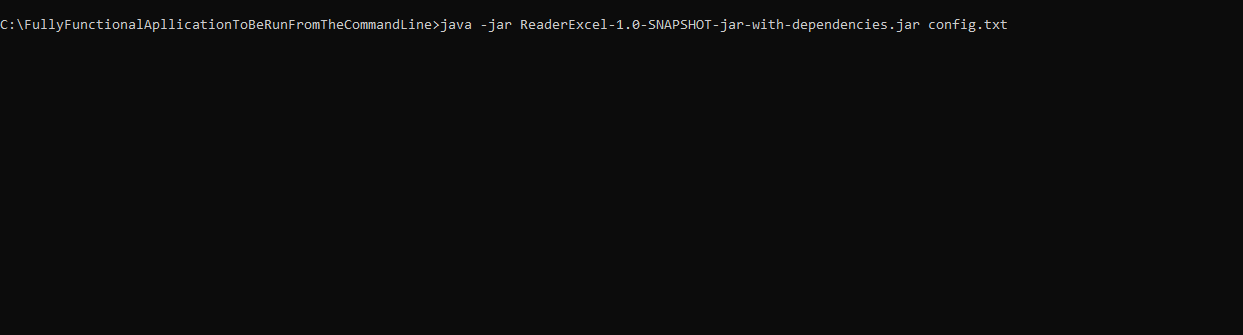
* **Altering the structure of the configuration file**

If you want to make changes to the data being filtered, you must alter the structure of the configuration file.

In this tutorial the configuration file is named Plik.txt. Please note, that the name of the configuration file can be different and adjusted to your personal needs.

E.g ConfigurationFile.txt, config.txt etc.

If you rename the configuration file, the input which is inserted into the console will be different like so:



If you want to make use of the **FUNCTIONALITY NO.1** and **FUNCTIONALITY NO.2** of the application, you must also make relevant changes to the configuration file.

Below I present two examples of what a config file looked like before and after it was changed:

(Please go to the next page to see the screenshot).

**BEFORE**



**AFTER**

****

The “columns” functionality of the outputFile entry is yet to be developed and it’s best if it’s left empty.

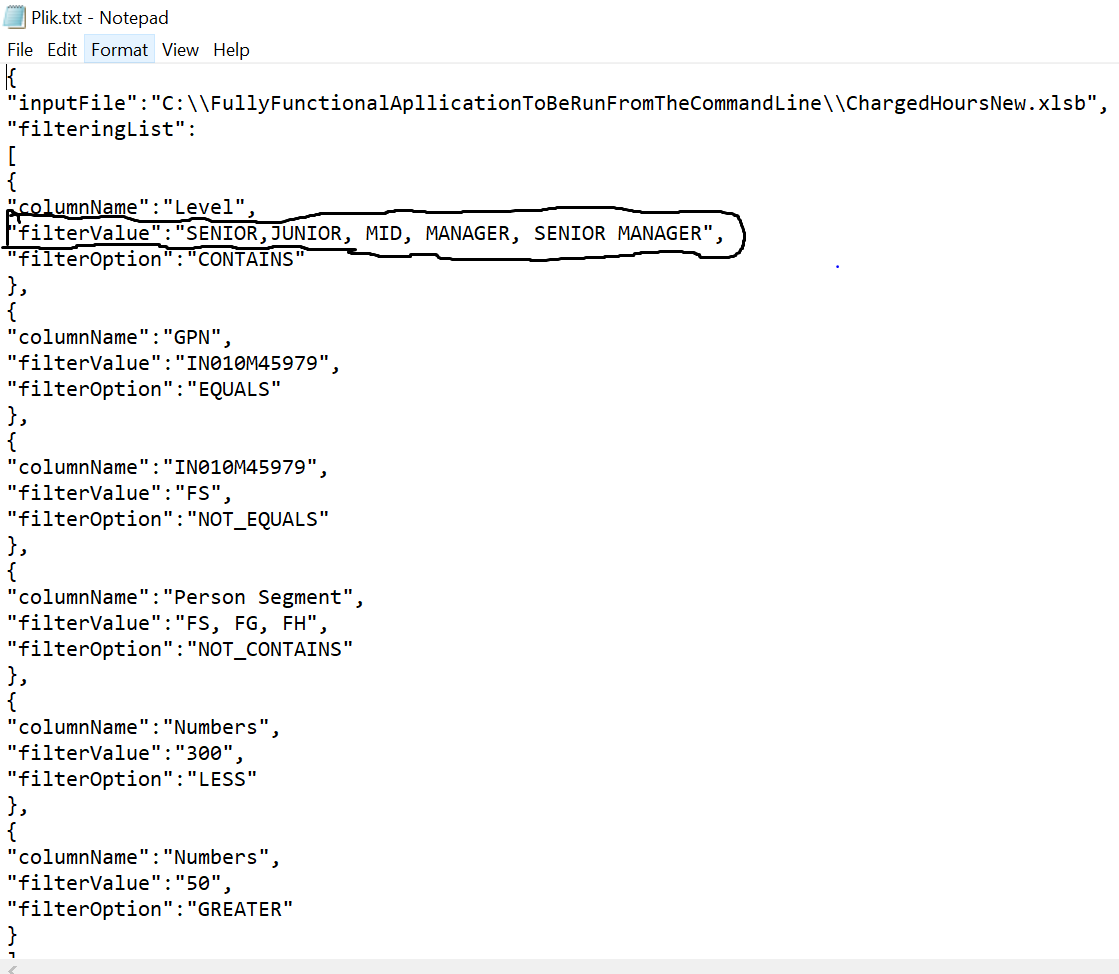
* **Providing multiple filter values**

The whole of the application was developed so that you can specify up to 5 filter values.

!!! If you insert a number that is above 5, the whole of the application will not be able to run!!!

Below I present an Example of the configuration file which contains 5 filter values.

Please observe that for those 5 filters values the application will make use of the CONTAINS function.



# What software the user needs to have to run the application:

Java JDK 19 needs to be installed and configured on your PC.

In case of any doubts please refer the tutorials mentioned below:

[Download and Install OpenJDK 19 on Windows (codejava.net)](https://codejava.net/java-se/install-openjdk-19-on-windows)

[Installing the JDK Software and Setting JAVA\_HOME (Using the GlassFish ESB Installation GUI) (oracle.com)](https://docs.oracle.com/cd/E19182-01/821-0917/inst_jdk_javahome_t/index.html)

## Additional Remarks:

The Excel functions such as BEFORE, AFTER and BETWEEN are yet to be developed.