Java test (30 min – 40 min)

This test is intended to observe your skills in **I/O, Stream, Optional, Collection, Collector, Lambda expressions, and programming logic, focused on data analysis/processing, using Java Core 8**, immersed in an activity that summarizes the main task that the developer faces in this team.

As previously explained, our processes are based on reading files **(.csv, .xml, .xlms, .txt, ...)**, filters and/or modifications to these data, merging data between different files, and at the end as output a new text file (or an update of an existing file) in a specific format. We always have to deliver the return of the work as quickly as possible, applying the best possible logic, consuming as little memory as possible, like a balance game, seeking the best of both worlds.

The initial structure of the project has already been generated. Try to apply the business rules using what was developed (this will save time). But you are open to modifying everything and creating new things if you feel it is necessary.

The code was initially developed by a junior developer. He doesn't know the good practices, so revalidate and refactor, if necessary, what has already been developed. You need finish this code.

Remember: We are going to evaluate I/O, Stream, Optional, Collection, Collector, Lambda expressions, and programming logic, focused on data analysis/processing, using Java Core 8, so we expect you to use these concepts as a mid/senior developer!

Example:

List<Object> list = ... ;

- Dev Junior

```
for( int i = 0; i < list.size(); i++ ) {
        System.out.println(list.get(i));
}
```

- Dev Mid

```
list.forEach(d -> System.out.println(d));
```

- Dev Senior

```
list.forEach(System.out::println);
```

Starting tasks

1. ***Read the file "people.csv" (10 min) [PeopleRepository]***

The **"people.csv"** file can be found at "**src/main/resources/files/people.csv**".

The file contains the columns:

ID,NAME,GENDER,AGE,DATE,COUNTRY

Were, mandatory:

- Duplicate lines must be removed, keeping only one valid information.
- Rows should only have 6 columns, rows with different number of columns should be ignored too.
- Remember that lines with empty text data (columns with empty information) or with the value "null" must also need be ignored.
- Apply a filter to ignore the line if each column not contains the mandatory data type (information validation).
    - **ID,** and **AGE**: are integers
    - **NAME**, **GENDER, DATE,** and **COUNTRY**: are strings

2. ***Read the file "customers.csv" (3 min) [CustomersRepository]***

The **"customers.csv"** file is located in **"src/main/resources/files/customers.csv"**.

The file contains the columns:

Index,Customer Id,First Name,Last Name,Company,City,Country,Phone 1,Phone 2,Email,Subscription Date,Website

In this case, all data types for every column in that file have already been validated. You'll only need to extract the **Index (id), Company, Country**, and **Email** columns. Don't need any data validation.

3. Getting data from file "people.csv" (5 min) [GettingDataJob]

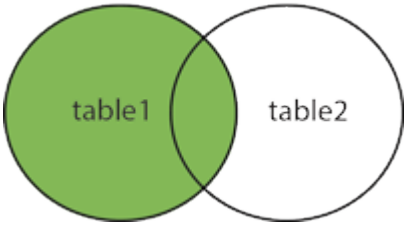
- Read the **"people.csv" file**.
- Set, to all person that is older 30 years, the **COUNTRY** value to null (java null).
- Then find the first line that in the column **"COUNTRY"** is the value **"France"**, and then **print it on screen**.

"**Person.toString()**"

If you don't find any row with the expected value, print a default **Person object**.

4. Merging the files (15 min) [MergingFilesJob]

We need to merge the two tables generated by reading the files. Return all records from the left table (**people.csv**) and the corresponding records from the right table (**customers.csv**), where the key is the column **people.COUNTRY = customers.Country**. The final list need contains: **NAME, GENDER, AGE, DATE,** and **COUNTRY** columns from ***people.csv***, and **Company,** and **Email** columns from ***customers.csv***.



Example:

- Table1 (KEY, NAME):

1, Silva
2, Miranda
3, Oliveira

- Table2 (KEY, JOB):

1, Dev
1, Ops
3, Msg

- JOIN per KEY (KEY, NAME, JOB):

1, Silva, Dev
1, Silva, Ops
2, Miranda, (*or 2, Miranda, "null" / or 2, Miranda, null / or 2, Miranda, "")*
3, Oliveira, Msg

5. Writing the merging of tables (3 min) [JoinsCPRepository]

Now we need to save the new table in an output file **"output.csv"**. It must contain the columns from the **people file** and the other 3 mapped from the **customers file**.

Were, mandatory:

- If the output file contains data, it must be deleted, and new data added.
- Don't forget the header (use the name **COUNTRY** for the merge key column).