**Teste Java**

Este teste tem o intuito de observar suas habilidades em **I/O, Stream, Optional, Collection, Collector, Lambda expressions, e lógica de programação, voltada a tratamentos de dados, usando Java Core 8**, imersa em uma atividade que resume a principal tarefa que o desenvolvedor enfrenta nessa equipe.

Como explicado anteriormente, nossos processos se baseiam em leitura de arquivos (**.csv, .xml, .xlms, .txt, ...),** filtros e/ou modificações nesses dados, fundir dados entre diferentes arquivos, e no final como saída um novo arquivo texto (ou uma atualização de um arquivo existente) em um formato específico. Temos de sempre entregar o retorno do trabalho o mais rápido possível, aplicando a melhor lógica possível, consumindo o mínimo de memória possível, como um jogo de balança, buscando o melhor dos dois mundos.

A estrutura inicial do projeto já foi gerada. Tente aplicar as regras de negócio das tarefas utilizando o que foi desenvolvido (isso poupará tempo). Mas você está aberto a modificar tudo e criar novas coisas, se achar necessário, mas que isso faça sentido e que você saiba explicar o porquê. Você pode adicionar novas dependências Maven, se achar necessário, ou se tiver algo para apresentar na sua solução, mas o que já foi disponibilizado já é suficiente para a tarefa.

**O código inicialmente foi desenvolvido por um desenvolvedor júnior. Ele ainda não conhece bem os conceitos, e podem ter cometido equívocos, portanto revalide e refatore, se necessário, o que já foi desenvolvido.** Você precisa terminar o desenvolvimento do código que ele já iniciou.

Lembre-se: Nós vamos avaliar **I/O, Stream, Optional, Collection, Collector, Lambda expressions, e lógica de programação, voltada a tratamentos de dados, usando Java Core 8**, então esperamos que use esses conceitos como um desenvolvedor pleno/sênior!

Ex:

*List<Object> list = .... ;*

**- Dev Júnior**

*for( int i = 0; i < list.size(); i++ ) {*

*System.out.println(list.get(i));*

*}*

**- Dev Pleno**

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

**- Dev Sênior**

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

**Iniciando as tarefas**

1. ***Ler o arquivo “people.csv”***

O arquivo “**people.csv**” encontra-se em “**src/main/resources/people.csv**”.

O arquivo possui as seguintes colunas:

**ID,NAME,GENDER,AGE,DATE,COUNTRY**

Onde, obrigatoriamente:

1. **ID,** e **AGE**: são números inteiros
2. **NAME**, **GENDER, DATE,** e **COUNTRY**: são textos

Siga a obrigatoriedade do tipo de dado (validação de informação) que deve conter cada coluna, e caso o **tipo de dado seja inválido, ignore a linha**. Lembre-se que **linhas com dados de texto vazio (colunas com informação vazia) ou com o valor “null” também devem ser ignoradas**. As linhas devem ter **somente 6 colunas, linhas com menos ou mais colunas devem ser ignoradas** também. Deve-se **remover linhas duplicadas**, mantendo somente uma informação válida.

1. ***Ler o arquivo “customers.csv”***

O arquivo “**customers.csv**” encontra-se em “**src/main/resources/customers.csv**”.

O arquivo possui as seguintes colunas:

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

Nesse caso, todos os tipos de dados de cada coluna desse arquivo já foram validados. Você precisará somente extrair as colunas **Company, Country,** e **Email**, sem preocupação. Todas eles estão em formato de texto e não são nulas, ou vazias.

1. ***Fusão dos arquivos***

Precisamos **fundir as duas tabelas gerada com a leitura dos arquivos**. Retorne todos os registros da tabela à esquerda (**people**), onde a idade é maior ou igual a 18, e os registros correspondentes da tabela à direita (**customers**), onde a chave é a coluna **people.COUNTRY = customers.Country**.



Ex:

- Table1 (KEY, NAME):

1, Silva

2, Miranda

3, Oliveira

- Table2 (KEY, JOB):

1, Dev

1, Ops

3, Msg

- JOIN por 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, Msq

1. **Escrita da fusão das tabelas**

Agora precisamos salvar a nova tabela em um arquivo de saída “**output.csv**”. Ela deve conter as colunas do arquivo **people** e as outras 3 mapeadas do arquivo **customers**. **Se o arquivo de saída conter dados, eles devem ser apagados**, e adicionado os novos dados. Não esqueça do cabeçalho (use o nome **COUNTRY** para a coluna da chave da fusão).

1. ***Obter informação do arquivo “people.csv”***

Leia (novamente, se necessário) o arquivo “**people.csv**”. Encontre a primeira linha que na coluna “**COUNTRY**” seja o valor “**France**”, substitua o valor por “**Portugal**”, e depois imprima na tela.

**System.out.println(PersonModel.toString())**

**>> PersonModel{id=1, name='abc', gender='abc', age=20, date='12-12-12', country='Portugal'}**

Caso não encontre nenhuma coluna com o valor esperado, imprima “**PersonModel{id=0, name='', gender='', age=0, date='', country=''}”**