 

Programa IT Academy – Processo Seletivo – Edição #16

## Etapa 2 – Enunciado

Nesta etapa, você vai escrever um programa de computador. Para isso deve ser feita a leitura do arquivo .csv enviado junto com este enunciado. Neste arquivo você encontra dados sobre medicamentos disponíveis no Brasil. Você deve implementar as seguintes funcionalidades:

1. [Consultar medicamentos pelo nome] Permitir que o usuário informe o nome do medicamento (ou parte do nome do medicamento) que desejar e como resultado o programa deverá exibir:
   1. Uma lista com os medicamentos encontrados e suas informações (Nome, Produto, Apresentação e valor PF Sem Impostos);

Atenção: somente devem aparecer no resultado os registros de produtos que foram comercializados em 2020 (observar a coluna de dados “COMERCIALIZAÇÃO 2020”).

1. [Buscar pelo código de barras] O programa deverá solicitar ao usuário o número correspondente ao código de barras de um produto (coluna de dados “EAN 1”, por exemplo ‘525516020019503’) e então:
   1. Localizar todos os registros referentes a este produto, independentemente de terem sido comercializados ou não em 2020;
   2. Dentre todos os registos encontrados, identificar o Preço Máximo ao Consumidor (alíquota de 0%, coluna de dados “PMC 0%”) mais alto e o mais baixo. Exibir na tela o mais alto, o mais baixo e a diferença entre eles.
2. [Comparativo da LISTA DE CONCESSÃO DE CRÉDITO TRIBUTÁRIO (PIS/COFINS)] Com base somente nos produtos que foram comercializados em 2020, o programa deverá:
   1. Consultar a coluna de dados “LISTA DE CONCESSÃO DE CRÉDITO TRIBUTÁRIO (PIS/COFINS)” para determinar o percentual de produtos classificados como “Negativa”, “Neutra” ou “Positiva” para esta coluna.
   2. Mostrar os respectivos valores percentuais da seguinte maneira (dados fictícios):

*[\* repare que a quantidade de asteriscos é proporcional ao respectivo percentual, por exemplo, neste caso são 21 asteriscos para a classificação Negativa.]*

|  |  |  |
| --- | --- | --- |
| CLASSIFICACAO | PERCENTUAL | GRAFICO |
| Negativa | 21,33% | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| Neutra | 45,18% | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| Positiva | 33,49% | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

TOTAL

100,00%

## Observações:

1. Sugere-se o desenvolvimento de um programa **na linguagem de sua preferência**, com uma **interface também de sua preferência podendo ser gráfica ou textual/console**, com um **menu com as opções enumeradas nos requisitos**;
2. Juntamente a este enunciado foi fornecido um arquivo no formato CSV contendo nomes, valores em decimais, bem como o respectivo dicionário de dados;
3. Você deve escrever o código que lê o arquivo e armazena os dados lidos em memória (do jeito que você quiser).
4. Não é necessário gravar dados em nenhum formato, nem usar sistemas de banco de dados.
5. O programa deverá lidar com dados de entrada inválidos e informar uma mensagem adequada caso ocorram.
6. Para facilitar, não é necessário lidar com a acentuação de palavras.
7. Na escrita do relatório apresente comentários sobre como você realizou os testes. Não esqueça de incluir uma autoavaliação.

-x-

**Explicação da Solução:**

Para solucionar o problema, foram criadas quatro Classes principais, cada uma visando abordar um ponto específico do problema. Além destas, existem três classes auxiliares, uma das quais não faz parte do programa final. A pasta com a solução foi dividida em duas sub-pastas; a primeira (testes) foi criada com o intuito de fazer a leitura e análise do arquivo – como estavam distribuidas as informações na tabela, quais colunas continham quais informações e etc; a segunda, que contém o programa final, é dividido em um Main (classe principal, que chama o método execute da Classe App, que por sua vez contem três chamadas para as outras classes – cada uma de acordo com um dos pontos do exercício). Para ler o arquivo, utilizei a Classe Scanner do java.util e a Classe File do java.io, depois iterei sobre o arquivo, quebrando cada linha lida no Scanner em arranjos unidimencionais. A localização do arquivo e o fato de ele ser separado nos pontos e vírugula (;) estão definidos em um arquivo externo as pastas denominado “CsvConfig.java”, que exporta o caminho e o separador para todas as outras classes. As três classes das etapas dos exercícios foram resolvidas de modos muito similares, possuindo um método principal chamado pelo App, que percorre a tabela e procura pelos valores de acordo com as condições do exercício e classes internas que executam funções – melhores explicadas no código – específicas para apresentar cada informação na tela.

**Explaining of the Solution:**

In order to solve the problem, four main classes were created, each one addressing a specific bullet point of the problem. In addition to these, there are three auxiliary classes, one of which is not a part of the final program. The folder with the solution was divided into two sub-folders; the first (“testes”) was created with the aim of reading and analyzing the file – how the data was distributed in the table and which columns contained which information; the second, contains the final program, is divided into a Main (the most important class, which calls the execute method from the App Class, which in turn includes three calls to the other classes – each one according to one of the bullet points of the exercise) . To read the file, I used the Scanner Class from java.util and the File Class from java.io, then I iterate over the file, breaking each line read by the Scanner into one-dimensional arrays. The location of the file and the fact that it is separated by semicolons (;) rather than the usual comma (,) are located in an external file to the two main folders named “CsvConfig.java”, which exports the path and separator for all of the other classes. All of the three questions were solved in a similar way, each Class having a main method called by the App according to an user input, which traversed the table and the values ​​according to the conditions of the and inner classes that perform functions – better explained in the code – specific to present each information on the screen.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

Autoavaliação

Sobre o desempenho; pontos de destaque e de dificuldade.

Apesar de estar muito feliz com o resultado final da minha aplicação, foi certamente um dos desafios mais complicados que fiz nos últimos tempos; ao mesmo tempo, foi um dos mais gratificantes e que envolveu mais áreas do conhecimento e tempo de pesquisa. Me vi forçado a estudar melhor como funcionava um arquivo .csv, como atualizar uma planilha em Excel com uma quantidade muito além do padrão (para mim) de colunas e linhas e como identificar pontos de problema externos – como o idioma do Windows, que impactou em muito no começo do meu desafio. Isso