ESCOLA POLITÉCNICA

Programa IT Academy – Processo Seletivo – Edição #18

Nome Completo: Mateus Campos Caçabuena E-mail: mateus.cacabuena@edu.pucrs.br

Etapa 1 – Questões de lógica

Esta seleção possui 15 questões de lógica de caráter eliminatório. As questões são apresentadas no formulário de Exercício Técnico e devem ser respondidas no próprio formulário online, que deverá ser acessado através do link a seguir: https://forms.gle/yZtVcv1b5fCgScLBA

Etapa 2

RESUMO DA SOLUÇÃO

Análise

Lendo o texto, percebi que seria uma dificuldade como conseguir armazenar todas as distâncias entre cidades, já que elas estão em uma matriz e nunca programei ler arquivos de *.csv* que estão em uma matriz. Porém, para chegar nesta etapa, precisaria primeiro desenvolver as outras classes, como caminhões e itens. Para conseguir me situar corretamente, grifei as partes do texto que seriam usadas para codificar, separando as cores de classes e atributos por cores diferentes do habitual.

Caminhão e Item

Comecei o código pela classe de caminhões, implementando como atributos:

• Carga - Fiquei indeciso por qual variável usar, pois os itens podem variar, exemplo do celular que possuía peso de ½ quilograma. Então, para não ter que fazer a conversão em um método mais para a frente, decidi colocar como *double* e atribuir 1000kg como 1 tonelada, por exemplo.

• Custo - Este foi mais simples, um *double* que seria baseado em R$ por KM.

Finalizei a construção da classe o construtor, *getters* e *setters*.

Em seguida, como dito, segui para a classe de itens e implementei como atributos nome para identificação e peso, sem impedimentos nesta classe.

Transporte

Nesta classe, inicialmente, implementei os atributos *string* origem e destino, sinalizando que seriam os nomes das cidades. Depois, pensei em introduzir uma nova classe só com as distâncias entre cidades e, com um método de leitura de arquivos na classe transporte, introduzi-las automaticamente.

Passei algumas horas tentando implementar este método corretamente, pois como é uma matriz, não basta ler e atribuir o valor no próximo atributo, teria que ter uma estrutura de matriz também, como o *.csv*.
Não queria fazer com uma matriz pois achei que não seria prático, além de não ter muita familiaridade, tentei implementar como um *ArrayList*. Portanto, fiz um *ArrayList* da classe distância e, como não consegui pensar em uma maneira de pensar em como implementar a cidade origem, destino e distância juntos no mesmo objeto, comecei a pensar em como implementar apenas a primeira linha do *.csv*, que era a cidade origem e cidade destino no mesmo objeto.
Assim, pensei em implementar um *for*(i) e dentro um *for*(j): o *for*(i) simbolizava a cidade origem e registrava todas as cidades destino que é o *for*(j), passando um por um, até o fim da linha. Desta maneira, registraria todos os possíveis trajetos entre as cidades, faltando apenas a distância entre elas.
Por algum motivo, estava dando erro. Nos testes apareciam repetidas vezes as mesmas cidades de origem, no destino e, no debug, percebi que o nome das cidades destino eram corretos, mas no print da tela estava dando errado.
Revisando o código, percebi o erro: o método *getDestino()* que estava na classe *Distancia* estava retornando a *cidadeNome* que é a origem, por isso estava dando erro no *toString()*.
Após arrumar isso, o código teste fluiu normalmente registrando todos os possíveis trechos entre cidades.
Agora, com os elementos de todos os possíveis trechos inseridos no *ArrayList*, comecei a pensar em maneiras de colocar a distância entre eles e, como os elementos estavam em ordem, bastava apenas um for, que acabava assim que chegasse ao final da linha, numerando a distância de cada elemento existente na lista.
Ao testar o código, percebi que a linha não pulava para a outra, então pesquisei e aprendi que o *br.readline*() é o que faz a linha ser "pulada". Porém, testei implementando-o entre um for e outro e não funcionou, pois, a variável *line* torna-se *null* ao fim do *for*(i) e *for*(j). Então, pensei em fazer outro *while*, após o fim do primeiro, com o novo for para determinar a distância. Testei e deu certo.
Para conseguir executar os testes, retirei temporariamente os parâmetros do método construtor da classe Transporte e coloquei um *forEach* com a chamada do método *toString*() de cada objeto de distância que havia no *ArrayList*, para checar se estavam corretos. Assim que os testes deram certo, implementei os parâmetros no construtor novamente e retirei o *forEach*.

Frota e App

A classe Frota é responsável por obter todo o registro de transportes, além ser quem é instanciada para executar os programas do sistema. Ao mesmo tempo que codifiquei esta classe, lia novamente as funcionalidades requisitadas para implementar a execução corretamente.
Logo na primeira funcionalidade, percebi que precisaria instanciar a classe Transporte, que instanciaria a classe Distância com atributos similares. Portanto, pensei na possibilidade de deletar a classe Distancia e fazer toda a tabela de distâncias entre cada cidade com um *ArrayList* chamado "trechos", que sinalizaria que é a lista de trechos e distância entre cada cidade. Assim, o método que lê os arquivos da tabela de distância passou a ser da classe Frota.
Porém, estaria muito confuso, o parâmetro do construtor da classe Transporte passou a ter as cidades, carga e distância. Para registrar apenas a distância entre as cidades seria

inútil ter o parâmetro carga, assim como para registar um transporte não seria necessário retratar a distância. Dito isso, pensei na possibilidade de recriar a classe Distancia e separar novamente os atributos, pensei também em fazer uma herança com a classe Distancia sendo filha da classe Transporte, mas não haveria um nome e nem uma função para a classe mãe. Outrora, veio a ideia de fazer apenas um método de calcular a distância, mas seria desnecessário pois não há cálculo, então, caiu a ficha que apenas um atributo distância seria o suficiente.

1ª Funcionalidade

Indo por partes, o primeiro desafio desta funcionalidade é mostrar para o usuário quais trajetos há disponíveis. Como eu passei um bom tempo pensando sem conseguir achar uma maneira de implementar, resolvi ao mesmo tempo iniciar a classe App que será a responsável por fazer o sistema funcionar fornecendo informações ao usuário.
Como base, utilizei um código similar ao que eu utilizei em 4 trabalhos feitos no semestre passado na cadeira de POO (Programação Orientada a Objetos), no qual o professor Yamaguti nos forneceu um início de código com *exceptions* e modelos de apresentação ao usuário.
Implementando e testando a primeira função do código que é ler os arquivos e dizer ao usuário quais trechos estão disponíveis, aconteceu um erro:
Procurei o erro com o debug e descobri que o arquivo não estava sendo lido, testei criando um arquivo.txt aleatório e não estava sendo lido. No fim, reiniciei o *vscode* e deu certo, era um erro da IDE. Utilizei uma expressão lambda linha 74 que também aprendi em POO, para simplificar o código.
Seguindo para o próximo problema, seria necessário juntar o transporte com o caminhão. Fiquei um longo tempo pensando em como implementar isso. As ideias não encaixavam pois, não queria usar a má prática de instanciar um caminhão, sendo que havia a frota e o transporte como classes que estariam sendo utilizadas para referenciá-lo.
Tive muita dificuldade neste método, pedi ajuda para uma colega da faculdade que está cursando no mesmo semestre para ver se ela sabia de algo que poderia agregar no meu código, recebi a dica de transformar os atributos da classe Caminhao em uma classe Porte que seria *enum*. Implementei o código e agora os únicos atributos da classe Caminhao é a classe Porte e o número da carga.
Após tirar uma pausa, voltei a tentar raciocinar para resolver este problema e cheguei à conclusão que deveria me basear em colocar as informações como parâmetro para classe Frota e o que ela não poderia fornecer, o transporte forneceria, apenas fiquei preocupado se conseguiria extrair informação da classe Caminhao que precisava. Cheguei a este código:
Ao testá-lo, deu quase tudo certo, o único erro foi o valor que deu errado por algum motivo que iria tentar descobrir executando e depurando o programa.
Depurando o programa, me encontrei no seguinte cenário: por algum motivo, mesmo executando o if, a conta encerra-se com tudo zerado.
Depois de 5 depurações diferentes para rever o código, descobri o erro: a distância não está sendo contabilizada. Para fazer a distância ser contabilizada, colocar um parâmetro distancia no método *calculaCusto*(), assim, imaginei que não precisaria ser a distância que há no atributo da classe Transporte.
Felizmente, o teste deu certo e assim pude encerrar o desenvolvimento da 1ª funcionalidade, quero implementar as *exceptions* para o código não encerrar caso o

usuário insira algo incorreto, mas deixarei isso para depois que acabar todas as funcionalidades, estou priorizando entregar todas as funcionalidades corretamente.

2ª Funcionalidade

Para implementar esta funcionalidade, quis ir em partes assim como na 1ª. Me afastei do teclado e comecei a fazer um brainstorm de ideias e, assim que pensei em uma maneira em que eu fosse capaz de desenvolver, comentei no código para não esquece-lo.

Detalhando o que pensei no comentário, permitia ao usuário dizer por quantas cidades seriam transportadas sua carga, assim, fazia um for até que i seja do mesmo tamanho que o *int* cidades. Neste for, haveria outro for, detalhando quantos itens diferentes o usuário iria transportar e, respectivamente, a quantia destes itens. Assim, consigo controlar a quantia do peso para calcular a carga e o custo através da quantia e porte de caminhões necessários. Além disso, caso o usuário deseje deixar certa quantia de carga em uma cidade para depois partir para outra, também conseguiria ser controlado.

Para o desenvolvimento deste método, pensei em duas coisas diferentes relacionados a *enum*:

• Transformar a classe Caminhao em *enum*, com os mesmos atributos da classe Porte e excluí-la. Pois, pensei que o caminhão não precisava adicionar nenhum parâmetro que pode variar, estes possíveis objetos já possuem parâmetros que são constantes(custo, carga limite) , sempre variará em 3 tipos, então, não vale a pena criar a classe sempre com os mesmos parâmetros e para isso transformei-o em um *enum*.

• Transformar a classe Item em *enum*, basicamente pelos mesmos motivos da classe Caminhao, já possuem parâmetros constantes (peso) que variam apenas de acordo com o nome do item.

O código ficou desta mesma maneira por algumas horas, sem nem mesmo testar. Pois, fiquei pensando em maneiras para conseguir contar a quantia de cada item, visto que, o exemplo de resposta fornecida no enunciado do exercício técnico mostrava e me preocupei com isso. Pensei em muitas possibilidades, como: adicionar um parâmetro que conta a quantidade de itens no transporte, fazer métodos novos em cada classe (frota e transporte), entre outros. Mas, o pensamento que mais fez sentido em minha cabeça, foi implementar um *ArrayList* de itens e, a cada vez que o usuário digita o tipo de item seguido da quantidade dele, eu acho uma maneira de adicionar esta mesma quantidade, só que de vezes, este mesmo item no *ArrayList*. Desta maneira, para contar a quantia de itens específicos, basta percorrer o a lista que obterei a informação.

Com este código, percebi que não preciso calcular a carga com todos aqueles *ifs*, pois, consigo calcular percorrendo o *Arraylist* também. Pensei em uma forma de simplificar e deixar este método menos “poluído” fazendo um *switch*.

Durante o desenvolvimento, fiquei estagnado devido ao pensamento de implementar para que consiga fazer com mais de 1 cidade de destino. Porém, este pensamento estava acabando com todas as possibilidades do meu código, então, comecei a implementar para cadastrar o transporte com apenas um destino, depois eu me preocuparia em destinos diferentes.

Notei que com a existência do *ArrayList* de itens, não seria necessário a existência do atributo carga no construtor do transporte, então pensei em removê-lo. Ainda estou em dúvidas se adiciono um método que calcula o peso da carga ou coloco o *ArrayList* como parâmetro e cálculo na frota.

Durante a implementação dos *enums*, tive muita dúvida pois havia esquecido as características dele, por exemplo, não sabia/não lembrava se era possível ter um *ArrayList*

de *enum*, havia esquecido até que podia instanciar um *enum*. Para dar uma revisada, pedi para um colega o material de *enum* do professor Yamaguti, que me ensinou isso ano passado e desta maneira consegui aprender a utilizar novamente.
Para testar esta primeira tentativa, tentei fazer uma expressão lambda para poupar linhas e simplificar o código, mas não consegui (nomeei o ArrayList de itens como "carga" e deletei o *double* carga).
Portanto, fiz um *forEach* com *ifs* para implementar o contador de itens que eu desejava.
O *forEach* com o contador de itens dentro do *ArrayList* me causou um problema que eu nunca havia visto antes no código: *java heap space error*. Ao pesquisar, entendi que é um erro de uso completo da memória. Eu já estava pensando que este método de colocar 300 itens de celular em um *ArrayList* estava sendo uma grande má prática, porém, não sabia que ia chegar a comprometer a memória. O momento em que lotou a memória foi após contar todos os itens, foi quando o programa chegava na contagem dos caminhões, que nem era tão grande.
Sendo assim, fiz de um modo diferente em que eu apenas adicionava 1 tipo de item no *ArrayList* e criei uma variável peso que multiplica o *qtdItem* com o peso dele. Assim, não precisaria adicionar vários elementos no *ArrayList* e pude seguir com o desenvolvimento.
Passei muito tempo tentando fazer apenas o transporte para uma cidade, devido as preocupações em citar os itens que estão sendo transportados, no qual pensei que seria necessário citar a quantidade deles. Mas, após ler novamente o enunciado e perceber que não era necessário citar a quantia de cada item que está sendo transportado, foi só questão de tempo para implementar. As grandes dificuldades desta primeira parte de testes, com certeza, foi a confusão que meu código deixou, tive que ir editando conforme testava e isso me deixou um pouco perdido. Porém, descansei e no outro dia com a cabeça leve consegui ter a calma e concentração que precisava para programar corretamente.
Sendo sincero, não gostei do primeiro escopo. Penso que o código não está pratico, lotado de ifs e confuso. Como está perto da data da entrega, penso que o código poderia estar um pouco melhor se estivesse programando com mais calma. Ainda não sei se conseguirei acabar esta funcionalidade e a próxima a tempo para ter calma de rever o código e tentar simplifica-lo, mas, dentre as dificuldades, estou feliz que o programa implementado está funcionando da maneira que desejei e agora tentarei fazê-lo com mais de 1 destino disponível.
Para fazer com mais de 1 destino, pensei em algumas possibilidades, mas o maior problema era a questão da quantidade de itens. A variável *qtdItem* servia para apenas 1 item e eu precisaria armazenar a quantidade de todos os itens. Portanto, pesquisei se era possível, de alguma forma, adicionar algo que varie dentro do *enum* de item. Procurei apenas pelo motivo de não saber mais o que fazer, pois estava a um tempo pensando em alguma solução. Até que achei um site que explicava que sim, era possível. Assim como em uma classe, bastava implementar um atributo sem colocá-lo no construtor. Consegui implementar e agora possuo uma variável que me diga a quantia de itens daquele tipo que havia no meu transporte.
Com isso, bastava eu implementar no switch do método *cadastraTransporte*() um *setQuantidade*(*qtdItem*), assim eu teria armazenada a informação de quantos itens daquele tipo haviam no transporte. Também adicionei uma variável chamada *custoMedio* para controlar a média dos itens, pois, achei interessante que não apareceu na requisição da funcionalidade, mas apareceu no exemplo então me preocupei em deixar registrado também este número. No fim do programa, ele divide pelo tamanho do *ArrayList* de itens, assim sabendo a média.

Após a execução do switch, o usuário já registrou todos os itens em que deseja transportar, agora, é só armazenar as informações que obtive na classe frota e registrar os custos médios.
Para imprimir na tela do usuário, utilizei o *printf* para conseguir colocar tudo na mesma linha. Entre os *printfs*, coloquei um *forEach* tanto de itens, quanto de caminhões que havia no transporte para retratar ao usuário quais tipos de cada um havia. Como foi dito nas páginas anteriores, tentei implementar a expressão lambda mas não deu certo, então tive que fazer do jeito que "polui" o código.
Resolvido tal problema, pensei em colocar todo o desenvolvimento que tinha feito até agora neste método dentro de um *if*(*qtdCidades* == 0) e, quando *qtdCidades* != 0, significa que haverá pelo menos 2 destinos diferentes. Coloquei esta possibilidade dentro de um *else*, que haveria outro switch, desta vez com itens que o usuário gostaria de descarregar. Ao selecionar o item em específico, ele dizia a quantidade que desejava retirar e esta quantidade ficaria armazenada no *qtdItem*. Pensando em uma forma de subtrair a quantidade de itens em cada transporte, implementei mais um método no *enum* Item que recebia um certo número e este número subtraia a quantidade de itens que obtinha. Com este método eu conseguia subtrair a quantia de itens que iam ser descarregados na primeira parada. Caso a subtração resultasse um valor menor que 1 (0), o item era removido do *ArrayList*.
Após este switch, o código fica similar ao código que fiz após o switch anterior, registrar as informações e dizer ao usuário. Ao fim do for, o usuário já fez tudo que precisava fazer e agora é repassado a ele as informações totais do transporte.
Agora que estou com o código feito, vou implementar o tratamento de exceções. Uso o mesmo modelo de tratamento de exceções em todos os scanners de *int* que estão no programa, que consiste em criar um do e dentro dele repetir a frase requisitada enquanto a resposta esperada não é digitada pelo usuário, juntamente com as *exceptions*. Tentei criar uma exceção que tratava caso o usuário digitasse um número menor que 1 na quantia de cidades.
Infelizmente, não consegui. Porém, implementei um *if* dentro do *do* que cobria esta possibilidade de inserção e deu certo.
Para fazer o usuário digitar a cidade da maneira correta, pensei em implementar um método *consultaCidade*() na frota que confere se há a existência da cidade em que o usuário digitou, faço isso dentro de um do que reinicia até ele escrever uma cidade que exista. Porém, me deparei com um problema:
Mesmo digitando a palavra, ele reiniciava o do. Pensei que poderia ser algo escrito errado no if, então troquei por um *.equals* e deu certo. Devido eu pedir 2 valores diferentes para usar no switch, ele também só recomeça depois que digita 2 valores. Mas creio que deixarei desta maneira, para mudá-lo não penso em nenhuma forma que não tenha que mexer em todo o código, por falta de tempo e não comprometer o código, não tentarei mudar isso.
Após colocar todos os tratamentos de exceções/ifs, o programa está completo. Se houver tempo após o término da 3ª funcionalidade, calcularei os custos unitários médios para checar se estão corretos, pois não fiz isso.

3ª funcionalidade

Estou pensando em implementar esta funcionalidade de um jeito similar a 2ª, colocar como variáveis tudo o que é necessário em questão de números e palavras, conseguindo-os através da frota.
Na implementação desta funcionalidade, pensei em fazer um *forEach* de transportes para pegar todos os transportes que estão registrados. Para os números mais simples como consultar custo através do trecho, não foi muito difícil, pois já havia sido feito para a 2ª funcionalidade, foi apenas uma questão de reutilização do método que está na classe Frota.
Porém, a grande dificuldade que tive nesta funcionalidade foram conseguir o total de itens transportados e a média deles. Pensei por muito tempo em como implementar, mas não achei outro jeito além de atribuir mais um atributo na classe Transporte. Além disso, implementei mais um método na classe Frota que recebia o transporte que precisava ser lido. Consiste no seguinte: o transporte já obtém o custo e o item, através disso obtenho a quantidade desde item e, assim, consigo a média por produto. Adicionei-os em um *HashMap* de *String* e Double, desta maneira, era só implementar e colocá-los na tela do usuário. Eu achava que o *ArrayList* não poderia me ajudar neste quesito, então pesquisei e aprendi a usar o *HashMap*. Uma experiência muito boa, visto que nunca tinha utilizado ele para nada além de exemplos de aula.
Para conseguir a quantia total de caminhões, foi um pouco mais simples. Fiz um *ArrayList* e coloquei em ordem a quantia de caminhões de pequeno, médio e grande porte. Só tive o trabalho de cuidar para pegar o número do endereço correto.
Este foi o final da minha programação no desafio, vou revisar, tentar deixar o código o mais bem visível possível e corrigir os erros.

TESTES (aqui você deverá colar capturas de tela de todas as funcionalidades desenvolvidas e realizar comentários, use o espaço que julgar necessário)

FUNCIONALIDADE 1 – Consultar trechos x modalidade

```
============================================================
       Sistema de transporte interestadual de cargas
[1] Consultar trechos disponíveis e modalidades de transporte
[2] Cadastrar novo transporte
[3] Exibir dados estatísticos
[4] Terminar o programa
============================================================

Opção desejada:
1
Estes são os trechos disponíveis e a distância entre eles:
------------------------------------------------------------------------------
{ id= '0' origem='ARACAJU', destino='ARACAJU', distancia='0}
{ id= '0' origem='ARACAJU', destino='BELEM', distancia='2079}
{ id= '0' origem='ARACAJU', destino='BELO HORIZONTE', distancia='1578}
{ id= '0' origem='ARACAJU', destino='BRASILIA', distancia='1652}
{ id= '0' origem='ARACAJU', destino='CAMPO GRANDE', distancia='2765}
{ id= '0' origem='ARACAJU', destino='CUIABA', distancia='2775}
{ id= '0' origem='ARACAJU', destino='CURITIBA', distancia='2595}
{ id= '0' origem='ARACAJU', destino='FLORIANOPOLIS', distancia='2892}
{ id= '0' origem='ARACAJU', destino='FORTALEZA', distancia='1183}
{ id= '0' origem='ARACAJU', destino='GOIANIA', distancia='1848}
{ id= '0' origem='ARACAJU', destino='JOAO PESSOA', distancia='611}
{ id= '0' origem='ARACAJU', destino='MACEIO', distancia='294}
{ id= '0' origem='ARACAJU', destino='MANAUS', distancia='5215}
{ id= '0' origem='ARACAJU', destino='NATAL', distancia='788}
{ id= '0' origem='ARACAJU', destino='PORTO ALEGRE', distancia='3296}
{ id= '0' origem='ARACAJU', destino='PORTO VELHO', distancia='4230}
{ id= '0' origem='ARACAJU', destino='RECIFE', distancia='501}
{ id= '0' origem='ARACAJU', destino='RIO BRANCO', distancia='4763}
{ id= '0' origem='ARACAJU', destino='RIO DE JANEIRO', distancia='1855}
{ id= '0' origem='ARACAJU', destino='SALVADOR', distancia='356}
{ id= '0' origem='ARACAJU', destino='SAO LUIS', distancia='1578}
{ id= '0' origem='ARACAJU', destino='SAO PAULO', distancia='2187}
{ id= '0' origem='ARACAJU', destino='TERESINA', distancia='1142}
{ id= '0' origem='ARACAJU', destino='VITORIA', distancia='1408}
{ id= '0' origem='BELEM', destino='ARACAJU', distancia='2079}
{ id= '0' origem='BELEM', destino='BELEM', distancia='0}
```

Possui todos os trajetos disponíveis, de acordo com o arquivo enviado

```
{ id= '0' origem='VITORIA', destino='CUIABA', distancia='2119}
{ id= '0' origem='VITORIA', destino='CURITIBA', distancia='1300}
{ id= '0' origem='VITORIA', destino='FLORIANOPOLIS', distancia='1597}
{ id= '0' origem='VITORIA', destino='FORTALEZA', distancia='2397}
{ id= '0' origem='VITORIA', destino='GOIANIA', distancia='1428}
{ id= '0' origem='VITORIA', destino='JOAO PESSOA', distancia='2001}
{ id= '0' origem='VITORIA', destino='MACEIO', distancia='1684}
{ id= '0' origem='VITORIA', destino='MANAUS', distancia='4476}
{ id= '0' origem='VITORIA', destino='NATAL', distancia='2178}
{ id= '0' origem='VITORIA', destino='PORTO ALEGRE', distancia='2001}
{ id= '0' origem='VITORIA', destino='PORTO VELHO', distancia='3575}
{ id= '0' origem='VITORIA', destino='RECIFE', distancia='1891}
{ id= '0' origem='VITORIA', destino='RIO BRANCO', distancia='4109}
{ id= '0' origem='VITORIA', destino='RIO DE JANEIRO', distancia='521}
{ id= '0' origem='VITORIA', destino='SALVADOR', distancia='1202}
{ id= '0' origem='VITORIA', destino='SAO LUIS', distancia='2607}
{ id= '0' origem='VITORIA', destino='SAO PAULO', distancia='882}
{ id= '0' origem='VITORIA', destino='TERESINA', distancia='2171}
{ id= '0' origem='VITORIA', destino='VITORIA', distancia='0}
--------------------------------------------------------------------------------
Digite o nome da cidade origem que deseja realizar seu transporte:
brasilia
Agora, digite o nome da cidade destino:
fortaleza
Por fim, digite a sigla que corresponde o tamanho do caminhão em que deseja consultar o preço de transporte:

Caminhao de pequeno porte [P]
Caminhao de medio porte [M]
Caminhao de grande porte [G]
m
de BRASILIA para FORTALEZA, utilizando um caminhao de porte [M], a distância é de 2200km e o custo será de R$26224,00.
```

```
Digite o nome da cidade origem que deseja realizar seu transporte:
porto alegre
Agora, digite o nome da cidade destino:
sao paulo
Por fim, digite a sigla que corresponde o tamanho do caminhão em que deseja consultar o preço de transporte:

Caminhao de pequeno porte [P]
Caminhao de medio porte [M]
Caminhao de grande porte [G]
p
de PORTO ALEGRE para SAO PAULO, utilizando um caminhao de porte [P], a distância é de 1109km e o custo será de R$5400,83.
```

```
Digite o nome da cidade origem que deseja realizar seu transporte:
natal
Agora, digite o nome da cidade destino:
cuiaba
Por fim, digite a sigla que corresponde o tamanho do caminhão em que deseja consultar o preço de transporte:

Caminhao de pequeno porte [P]
Caminhao de medio porte [M]
Caminhao de grande porte [G]
g
de NATAL para CUIABA, utilizando um caminhao de porte [G], a distância é de 3543km e o custo será de R$97219,92.
============================================================
       Sistema de transporte interestadual de cargas
[1] Consultar trechos disponíveis e modalidades de transporte
[2] Cadastrar novo transporte
[3] Exibir dados estatísticos
[4] Terminar o programa
============================================================

Opção desejada:
```

FUNCIONALIDADE 2 – Cadastrar transporte

```
==========================================================
       Sistema de transporte interestadual de cargas
[1] Consultar trechos disponíveis e modalidades de transporte
[2] Cadastrar novo transporte
[3] Exibir dados estatísticos
[4] Terminar o programa
==========================================================

Opção desejada:
2
Digite a quantia de cidades que você deseja transportar seus itens:
2
Qual a cidade de origem?
porto alegre

Digite o destino do transporte que partirá da cidade PORTO ALEGRE:
florianopolis

Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
1
Quantas unidades deste item você deseja transportar?
300

Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
```

```
Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
2
Quantas unidades deste item você deseja transportar?
50

Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
3
Quantas unidades deste item você deseja transportar?
70

Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
5
Quantas unidades deste item você deseja transportar?
2000
```

```
Selecione o tipo de item que deseja transportar:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
7

de PORTO ALEGRE para FLORIANOPOLIS, a distância a ser percorrida é de 476km.
Para o transporte dos produtos celular, geladeira, freezer, luminaria, será necessário utilizar
[2] caminhões de pequeno porte
[0] caminhões de médio porte
[1] caminhões de grande porte
de forma a resultar no menor custo de transporte por km rodado. O custo do transporte destes itens é R$ 17697,68 e o custo unitário médio é R$7,31
Após a chegada em FLORIANOPOLIS, qual será o próximo destino?
```



```
curitiba

Selecione a carga que você deseja descarregar na cidade FLORIANOPOLIS, antes de transportá-la até CURITIBA:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
2
Quantas unidades deste item você deseja descarregar?
25

Selecione a carga que você deseja descarregar na cidade FLORIANOPOLIS, antes de transportá-la até CURITIBA:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
3
Quantas unidades deste item você deseja descarregar?
50

Selecione a carga que você deseja descarregar na cidade FLORIANOPOLIS, antes de transportá-la até CURITIBA:
[1] Celular
[2] Geladeira
[3] Freezer
[4] Cadeira
[5] Luminaria
[6] Lavadora de roupa
[7] Já selecionei todos os itens
1
Quantas unidades deste item você deseja descarregar?
100
```

```
7

de FLORIANOPOLIS para CURITIBA, a distância a ser percorrida é de 300km.
Para o transporte dos produtos celular, geladeira, freezer, luminaria, será necessário utilizar
[0] caminhões de pequeno porte
[1] caminhões de médio porte
[0] caminhões de grande porte
de forma a resultar no menor custo de transporte por km rodado. O custo do transporte destes itens é R$ 3576,00 e o custo unitário médio é R$20,43
-----------------------------------------------------------------
O custo total para este transporte é R$21273,68
A distância total que este transporte percorrerá é 776km
O custo unitário médio total é R$13,87
-----------------------------------------------------------------
```

FUNCIONALIDADE 3 – Dados estatísticos

Cadastrei o cenário 1 de cadastra suporte e este são os dados estatísticos.

```
Opção desejada:
3

Transporte número: 1
----------------------------------------
Custo do trecho: R$17697,68
Custo médio por km do trecho 1: R$37,18
Custo por trecho total: R$17697,68
Custo médio por km total: 37,18

Custo por modalidade de caminhão:
[P] 4636,24
[M] 0,00
[G] 13061,44

Custo médio por tipo de produto:
[geladeira] 707,91
[freezer] 884,88
[celular] 88,49
[luminaria] 8,85
----------------------------------------
Transporte número: 1
----------------------------------------
Custo do trecho: R$3576,00
Custo médio por km do trecho 2: R$11,92
Custo por trecho total: R$21273,68
Custo médio por km total: 27,41
```

```
Custo por modalidade de caminhão:
[P] 0,00
[M] 3576,00
[G] 0,00

Custo médio por tipo de produto:
[geladeira] 143,04
[freezer] 178,80
[celular] 17,88
[luminaria] 1,79
----------------------------------------
Custo total: R$21273,68
Total de itens transportados: 2420
Número total de veículos transportados: 4
```

AUTOAVALIAÇÃO

Você concluiu a implementação de 100% das funcionalidades solicitadas?

(X) Sim ( ) Não

Para as 3 principais funcionalidades solicitadas, como você avalia a sua solução?

Marque um 'X'.

	Inexistente/ Insuficiente	Pouco satisfeito(a)	Satisfeito(a)	Muito satisfeito(a)
Funcionalidade 1				X
Funcionalidade 2			X	
Funcionalidade 3			X	

Principais dificuldades

As principais dificuldades foram:

- ➔ Arquivo *.csv* em formato de matriz. Nunca havia trabalhado com isso antes, demorei um bom tempo até conseguir chegar no ponto de implementar todas as cidades e a distância entre elas. Os estudos sobre leitura de arquivos me ajudaram muito.
- ➔ Cadastrar mais de um destino nos transportes. O método ficou bastante extenso e creio que foi onde mais levei tempo para desenvolver, principalmente pelo conteúdo que foi requisitado para dizer ao usuário.
- ➔ Tempo. Me adiantei para realizar o desafio com calma, mas imprevistos acontecem e eu perdi um grande amigo justamente na semana do desafio. Tiveram alguns dias que não tive condições de programar nada, isso apertou muito e fez eu programar com mais pressa e menos capricho. Felizmente, consegui ter o foco necessário e acabei o desafio a tempo!

Desempenho Geral

Gostei do meu desempenho, mas creio que poderia ter sido melhor. No começo do desafio eu estava convicto em utilizar a interface gráfica, infelizmente, não consegui por falta de tempo. Por outro lado, gostei muito do meu desempenho em questões isoladas do código, como a leitura de arquivos. Eu não conhecia como funcionava para um programa ler um arquivo, aprendi na faculdade através de uma decoreba de códigos. Este desafio me obrigou a pesquisar e entender sobre para conseguir implementar o *.csv* que eu tanto queria, foi um grande aprendizado para mim!
Fico feliz de ter desenvolvido o programa em linguagem Java e aprender novos macetes, gosto desta linguagem e o desafio me mostrou novas possibilidades do que fazer em algumas situações, que não sabia quando estudei.

Obrigado por participar deste processo seletivo.
Salve o documento em PDF com o seu nome completo.