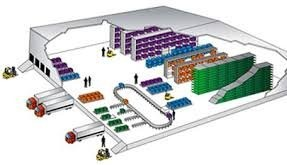
Uma imagem com texto

Descrição gerada automaticamente

Programação orientada objetos

LEI

*Manual técnico- Projeto Centro de Distribuição*



*Ano letivo: 2022/2023*

Bernardo Vaz (202200278) (202200278@estudantes.ips.pt)

Tiago Ramada (202200354) (202200354@estudantes.ips.pt)

Índice

[Introdução 3](#_Toc139118896)

[Breve descrição do programa 3](#_Toc139118897)

[Classes e relações entre as mesmas 4](#_Toc139118898)

[Armazem 4](#_Toc139118899)

[Local 4](#_Toc139118900)

[Armazenamento 4](#_Toc139118901)

[Venda 4](#_Toc139118902)

[Veiculo 4](#_Toc139118903)

[AutomaticGuidedCard 4](#_Toc139118904)

[Trailer 5](#_Toc139118905)

[TransportCar 5](#_Toc139118906)

[UnitLoadCarrier 5](#_Toc139118907)

[Centro Distribuição 5](#_Toc139118908)

[Bag 5](#_Toc139118909)

[Box 6](#_Toc139118910)

[CardBox 6](#_Toc139118911)

[Pallet 6](#_Toc139118912)

[Product 6](#_Toc139118913)

[TypeOfProduct 6](#_Toc139118914)

[Sensors 6](#_Toc139118915)

[Ultrasonic 6](#_Toc139118916)

[Camera 7](#_Toc139118917)

[Lidar 7](#_Toc139118918)

[Posicionaveis 7](#_Toc139118919)

[Shelf 7](#_Toc139118920)

[ArmazemImport 7](#_Toc139118921)

[FileInputReader 7](#_Toc139118922)

[LocaisImport 8](#_Toc139118923)

[ProductsImport 8](#_Toc139118924)

[ShelfesImport 8](#_Toc139118925)

[VeiculosImport 8](#_Toc139118926)

[Main 8](#_Toc139118927)

[Conclusão 9](#_Toc139118928)

[2º Fase – Introdução 9](#_Toc139118929)

[Descrição do programa 9](#_Toc139118930)

[Conclusão (2º Fase) 11](#_Toc139118931)

# Introdução

O objetivo deste projeto era o desenvolvimento de uma simulação do funcionamento de um centro de distribuição de produtos, onde os produtos seriam recebidos num local específico do armazém, embalados e depois seriam transportados para os locais de armazenamento.

Deste modo desenvolvemos uma aplicação para o desejado onde implementamos um conjunto de classes que permitisse representar a logica de aplicação.

# Breve descrição do programa

Este é um programa que como dito anteriormente é suposto simular o funcionamento de um centro de distribuição, onde tentamos implementar várias funcionalidades desde a criação do armazém até a entrega dos produtos de modo a mostrar na consola os resultados. No tópico seguinte explicaremos as classes e as suas relações.

# Classes e relações entre as mesmas

## Armazem

Vamos começar pela classe Armazem que vai ter a responsabilidade de guardar os atributos do nome e limites do armazém, e esses limites fornecidos no ficheiro .csv depois vão ser usados para estruturar um “mapa” tendo um conjunto de métodos para manipular e retornar informação sobre o mesmo, que vai ser usado para adicionar/remover os objetos posicionáveis como os veículos e as prateleiras (shelfs).

## Local

Esta classe vai ser a superclasse do armazenamento e da venda, sendo uma das subclasses da classe posicionáveis. E é usada pelas classes filhas para criar respetivamente através dos métodos que ela contém os espaços para cada uma, dentro do armazém, daí ter como atributo o armazém.

## Armazenamento

Esta classe vai ser uma subclasse do Local e através da superclasse cria o seu espaço e um arraylist que vai permitir armazenar a informação do conteúdo nela presente (acerca dos pacotes nela guardados). Consegue ainda receber pacotes do exterior e enviar para o interior do armazém.

## Venda

Esta classe também subclasse do Local vai através da superclasse criar o seu espaço e criar 2 arraylist que vão armazenar a quantidade de produtos e pacotes para posteriormente através de métodos de receber (dos carros) poderá vender produtos, pacotes e pode ainda mostrar a informação do conteúdo dos array na consola.

## Veiculo

Esta classe abstrata é filha da classe posicionáveis e superclasse dos 4 tipos de veículos existentes. É uma classe abstrata porque assim com isto criamos métodos abstratos que sendo comuns nestas classes todas, pudéssemos desta forma mais tarde implementar em cada uma delas os mesmos.

## AutomaticGuidedCard

Esta classe é subclasse da classe veículo por isso tivemos de implementar os métodos abstratos da superclasse. Este tipo de veículo vai ter vários atributos como o peso máximo da carga transportada e tipos específicos de carga a transportar verificada por métodos, direção, posições (x e y) os 3 tipos de sensor, uma velocidade e ainda o peso atual da carga. Com os métodos desenvolvidos aqui podemos adicionar/remover carga ao carro através de métodos de apoio que fazem a verificação acerca da carga a transportar, e receber informação do estado do veículo e mover o veículo.

## Trailer

Esta classe é filha da classe veículo por isso tivemos de implementar os métodos abstratos da superclasse. Este tipo de veículo vai ter vários atributos como um tipo de carro TransportCar já que não transporta carga, mas sim carros deste tipo, direção, posições (x e y) os 3 tipos de sensor e uma velocidade. Com os métodos desenvolvidos aqui podemos mover o veículo, adicionar/remover veículos a rebocar através de métodos de apoio que fazem a verificação se é possível rebocar um veículo ou se já contém um a ser rebocado neste momento, e receber informação do estado do veículo.

## TransportCar

Esta classe é filha da classe veículo por isso tivemos de implementar os métodos abstratos da superclasse. Este tipo de veículo vai ter vários atributos como o peso máximo da carga transportada e tipos específicos de carga a transportar verificada por métodos, direção, posições (x e y) os 3 tipos de sensor, uma velocidade e ainda o peso atual da carga, apesar de ele não se movimentar implementamos na mesma os métodos. Com os métodos desenvolvidos aqui podemos adicionar/remover carga ao carro através de métodos de apoio que fazem a verificação acerca da carga a transportar, e receber informação do estado do veículo.

## UnitLoadCarrier

Esta classe é subclasse da classe veículo por isso tivemos de implementar os métodos abstratos da superclasse. Este tipo de veículo vai ter vários atributos como a carga (neste caso palete) e tipos específicos de carga a transportar verificada por métodos, direção, posições (x e y) os 3 tipos de sensor, uma velocidade. Com os métodos desenvolvidos aqui podemos adicionar/remover carga ao carro através de métodos de apoio que fazem a verificação acerca da carga a transportar, e receber informação do estado do veículo e mover o veículo.

## Centro Distribuição

Esta classe é a superclasse dos 4 tipos de embalagens sendo eles (Bag, Box, CardBox e Pallet), que é responsável por fornecer o código ás embalagens conforme o seu tipo, guardando o tipo e número de embalagem.

## Bag

Esta classe é a classe filha do centro distribuição que utiliza o construtor da superclasse para obter um código para a sua embalagem, aqui é possível adicionar/remover produtos ás embalagens de acordo com as restrições de produtos que podem ser embalados neste tipo de embalagem com a ajuda de métodos auxiliares criados, e obter informações acerca da embalagem como o peso e código.

## Box

Esta classe é a classe filha do centro distribuição que utiliza o construtor da superclasse para obter um código para a sua embalagem, aqui é possível adicionar/remover produtos ás embalagens de acordo com as restrições de produtos que podem ser embalados neste tipo de embalagem com a ajuda de métodos auxiliares criados, e obter informações acerca da embalagem como o peso e código.

## CardBox

Esta classe é a classe filha do centro distribuição que utiliza o construtor da superclasse para obter um código para a sua embalagem e determinado a partida que tipo de produto vai ser armazenado (grande ou pequeno) de modo a facilitar posteriormente no processo de transporte e embalamento, aqui é possível adicionar/remover produtos ás embalagens de acordo com as restrições de produtos que podem ser embalados neste tipo de embalagem com a ajuda de métodos auxiliares criados, e obter informações acerca da embalagem como o peso e código.

## Pallet

Esta classe é a subclasse do centro distribuição que utiliza o construtor da superclasse para obter um código para a sua embalagem sendo que assumimos através do enunciado que a máxima capacidade de cada pallet seria de 5 cardbox. Aqui é possível adicionar/remover cardbox a estas pallets, e obter o código.

## Product

Esta classe vai conter um nome, id e peso e vai ainda colaborar com a classe TypeOfProduct para escolher um tipo de produto validando o peso, e podemos com esta classe obter as informações do produto.

## TypeOfProduct

Esta classe é um enumerado com os diversos tipos de produtos que podem existir neste armazém e contém um método que retorna o tipo de produto de um dado produto, e usamos em comparações para a implementação dos métodos auxiliares para saber se o produto é compatível com o tipo de embalagem.

## Sensors

Esta classe é a superclasse dos 3 tipos de sensores existentes e este recebe um angulo e uma distância de visualização sendo que é usado para as classes derivadas desta puderem usar o construtor já que é comum a todas diferenciando apenas os valores.

## Ultrasonic

Esta classe é a subclasse de Sensors que fornece os dados ao construtor e contém um método que verifica se existe obstáculos de acordo com os valores fornecidos e objetos desta classe são diretamente implementados na criação de cada veículo. Pode-se ainda obter a informação dos valores fornecidos.

## Camera

Esta classe é a subclasse de Sensors que fornece os dados ao construtor e contém um método que verifica o tipo do obstáculo se encontra de acordo com os valores fornecidos e objetos desta classe são diretamente implementados na criação de cada veículo. Pode-se ainda obter a informação dos valores fornecidos.

## Lidar

Esta classe é a subclasse de Sensors que fornece os dados ao construtor e contém um método que verifica se existe obstáculos de acordo com os valores fornecidos e objetos desta classe são diretamente implementados na criação de cada veículo. Pode-se ainda obter a informação dos valores fornecidos.

## Posicionaveis

Esta classe é a superclasse de Local, Shelf, Veículo, ou seja, acaba por ser superclasse também de as subclasses de cada uma destas mencionada. Esta classe armazena todos os veículos e prateleiras criadas (automaticamente adicionados aqui ao criar os objetos). Contem métodos que nos permitem verificar se os veículos estão nos locais (armazenamento ou venda) além de puder procurar por veículo e prateleiras em coordenadas fornecidas pelo utilizador, pode ainda adicionar ou remover manualmente veículos e prateleiras. Esta classe contém ainda um tipo enumerado Directions que é usado pelos objetos que estendem desta classe para ajudar na movimentação e/ou colocação nas posições.

## Shelf

Esta classe é uma subclasse de Posicionaveis, onde pudemos através desta classe adicionar/remover produtos embalados tendo atributos de direção e coordenadas (x e y), podemos também nesta classe aceder a informação acerca da prateleira. Contém também um método de apoio para saber se o veículo está a encarar a prateleira de modo a pudermos colocar os produtos embalados do carro para a prateleira.

*Passamos agora para as classes de leitura dos ficheiros csv*

## ArmazemImport

Esta classe vai ter um atributo onde guarda o caminho para o ficheiro com os dados do armazém a serem usados na simulação, onde a mesma irá ler o ficheiro com os dados guardados fazer a separação dos dados por virgulas e guardar num array de modo a depois fornecer os dados de forma correta e ordenada para funcionamento da simulação, tudo isto com a colaboração da classe FileInputReader.

## FileInputReader

Esta classe vai servir de apoio para as classes de leitura de ficheiros csv, pois vai ser aqui que com o método existente criado, vai ser aberta uma stream, onde colocar o ficheiro que pretendemos ler, lemos o conteúdo desse ficheiro e colocamos tudo num array de strings para mais tarde nas outras classes separar e obter os dados para a simulação.

## LocaisImport

Esta classe vai ter um atributo onde guarda o caminho para o ficheiro com os dados dos locais a serem usados na simulação, onde a mesma irá ler o ficheiro com os dados guardados fazer a separação dos dados por virgulas e guardar num array de modo a depois fornecer os dados de forma correta e ordenada para funcionamento da simulação, tudo isto com a colaboração da classe FileInputReader.

Nota: Esta seria a implementação correta que não conseguimos concretizar.

## ProductsImport

Esta classe vai ter um atributo onde guarda o caminho para o ficheiro com os dados dos produtos a serem usados na simulação, onde a mesma irá ler o ficheiro com os dados guardados fazer a separação dos dados por virgulas e guardar num array de modo a depois fornecer os dados de forma correta e ordenada para funcionamento da simulação, tudo isto com a colaboração da classe FileInputReader.

## ShelfesImport

Esta classe vai ter um atributo onde guarda o caminho para o ficheiro com os dados das prateleiras a serem usados na simulação, onde a mesma irá ler o ficheiro com os dados guardados fazer a separação dos dados por virgulas e guardar num array de modo a depois fornecer os dados de forma correta e ordenada para funcionamento da simulação, tudo isto com a colaboração da classe FileInputReader.

## VeiculosImport

Esta classe vai ter um atributo onde guarda o caminho para o ficheiro com os dados dos veículos a serem usados na simulação, onde a mesma irá ler o ficheiro com os dados guardados fazer a separação dos dados por virgulas e guardar num array, além de ler o tipo do veículo para puder colocar os dados no tipo certo de veículo, de modo a depois fornecer os dados de forma correta e ordenada para funcionamento da simulação, tudo isto com a colaboração da classe FileInputReader.

*Agora por fim passemos a classe main que será onde ocorre a simulação*

## Main

Esta classe main vai ser responsável por usar os diversos métodos criados nos ficheiros de leitura para ir buscar a informação ao criar os diversos objetos para a simulação.

# Conclusão

Começamos por reconhecer que foi um projeto trabalhoso, onde sentimos alguma dificuldade, mas que apesar de tudo temos connosco um sentimento de que cumprimos os objetivos para com este enunciado.

Temos a limitação de que não conseguimos resolver um problema na movimentação dos veículos em que eles acabam por simplesmente passar por cima dos obstáculos invés de os contornarem. Apesar desta limitação que não conseguimos resolver, reforçamos que temos presente um sentimento de sucesso, e que com este projeto foi-nos proporcionado uma hipótese para consolidar a matéria que nos foi ensinada e que certamente aprendemos imenso.

# 2º Fase – Introdução

Nesta segunda fase o que é proposto é a criação da interface gráfica com o utilizador onde através do desenvolvimento em Java FX e reutilizando o código da primeira fase, se possa observar o que mostrávamos na consola na primeira fase, agora em janelas do Java FX.

# Descrição do programa

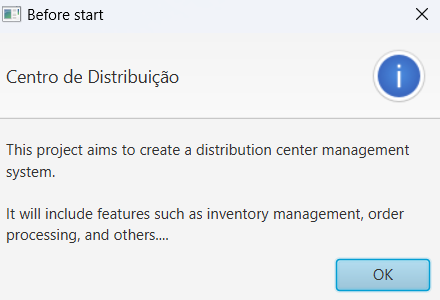
Tudo começa com a apresentação da janela inicial, onde se demonstra o titulo do projeto e apresenta um botão para iniciar, após clicar no mesmo é mostrada uma janela de pop-up onde diz o pretendido do projeto e alerta que após o fecho deste pop-up se vai então dar inicio a simulação pretendida no enunciado, de seguida passamos para a janela onde vai ocorrer a simulação, que contém as listas com os veículos disponíveis, e as prateleiras disponíveis, e ainda mais 2 listas conectadas ás mesmas onde irá mostrar a informação ao momento das informações sobre os mesmos/mesmas, contém também um botão para mover (ou seja dar 1 passo na simulação) e de stop (que coloca os veículos parados, ou seja velocidade = 0). No lado esquerdo desta janela da simulação teremos então o “mapa” onde se vai visualizar todas estas funcionalidades ao momento.

Devido ao tamanho das fotos, encontram-se na página seguinte.

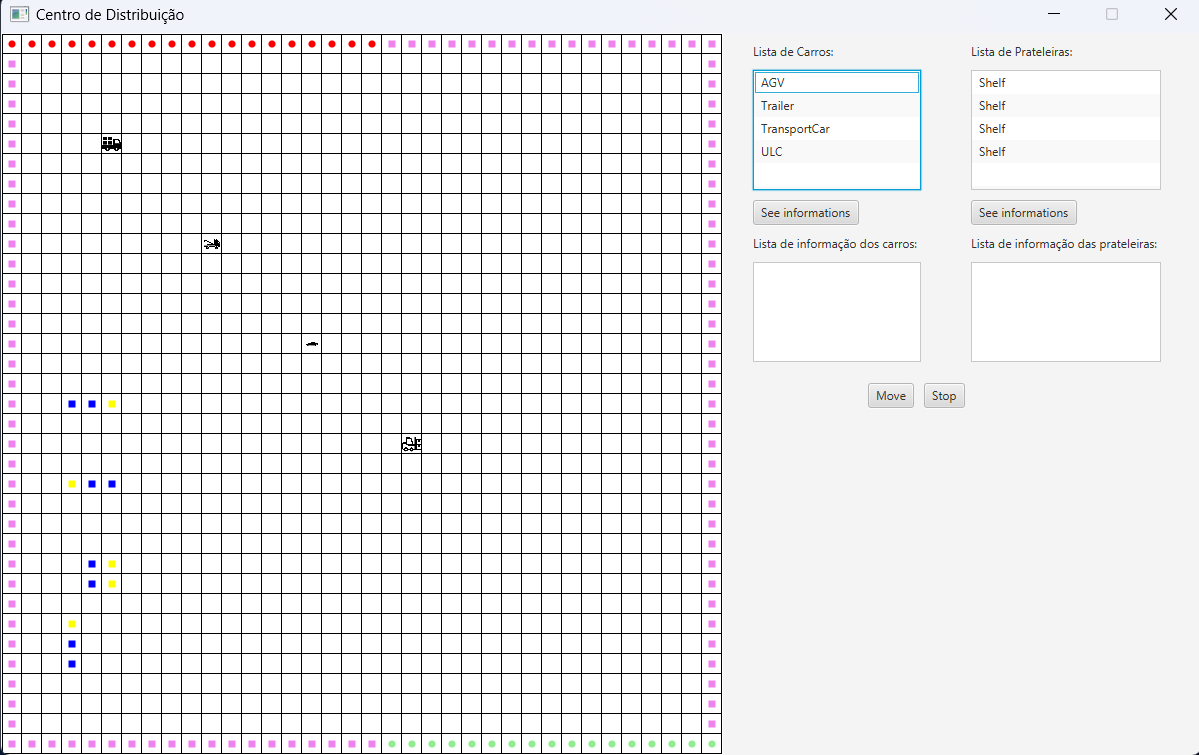
A janela inicial é a seguinte:



A janela de pop-up:



A janela de simulação é a seguinte:



Legenda:

* A linha de posições a rosa são as paredes do armazém
* A linha de posições de intervalo a vermelho e a verde são os locais de armazenamento e venda
* As posições a azul escuro são as prateleiras e a amarelo são as zonas de possível carregamento da carga de acordo com a direção da prateleira
* Os quatro veículos são distinguidos pelas imagens nomeadamente se lermos de cima para baixo respetivamente, Trailer, Transport Car, Automatic Guided Car e por último Unit Load Carrier em que cada um altera a sua figura quando transporta carga para ser visivelmente fácil perceber se tem carga ou não.

Para mover os carros basta selecionar o veiculo pretendido e clicar mover. Caso se tente fazer isso com as prateleiras obviamente não funcionará.

# Conclusão (2º Fase)

Após da primeira fase do projeto tivemos agora a oportunidade de experimentar criar uma interface gráfica utilizando o Java FX, sendo que foi um processo desafiante e complexo principalmente na criação do gridpane (“mapa”) em que tivemos alguma dificuldade, porém conseguimos ultrapassar e após começarmos a perceber a lógica achamos que fizemos o necessário para alcançar os objetivos necessários para esta 2º fase.