**UNIVERSIDADE FEDERAL DE PELOTAS**
**CENTRO DE DESENVOLVIMENTO TECNOLÓGICO**
**CURSOS DE CIÊNCIA E ENGENHARIA DE COMPUTAÇÃO**
**Prof.: Felipe de Souza Marques (felipem @inf.ufpel.edu.br)**

## 1. Objetivo:

Exercitar as habilidades e conceitos de programação orientada a objetos desenvolvidos ao longo da disciplina pela implementação de uma aplicação em Java, executada por um grupo de **1 ou 2** alunos.

O programa deve ser estruturado de forma a receber um conjunto de entradas (simultaneamente ou durante o uso do programa), cuja consistência deve ser verificada, processá-las, e fornecer uma ou mais saídas.


## 2. Avaliação

Conteúdos: A aplicação desenvolvida deverá demonstrar os seguintes conteúdos:

a. (3 pontos) Habilidade em utilizar abstração para uma definição correta das classes envolvidas na aplicação.

b. (2 pontos) Documentação de programas (identação, utilização de nomes de variáveis, abstração dos procedimentos para obter maior clareza, uso de comentários no código).

c. (2 ponto) Domínio na utilização de tipos de dados simples e estruturados.

d. (1 ponto) Formatação e controle de entrada e saída, com construção de interfaces que orientem corretamente o usuário sem instruções ou intervenção adicional do programador.

e. (2 pontos) Atendimento aos requisitos do enunciado do programa: modelo de estrutura de dados, de interação e de relatórios, opções do programa e salvamento intermediário de estado do sistema.

## 3. Contextualização

Uma empresa que gerencia estacionamentos resolveu fazer o controle de forma automatizada. No sistema informatizado de controle, o operador (usuário) recebe uma requisição de vaga para estacionar um determinado veículo, verifica se existe alguma vaga desocupada adequada para as características do veículo e realiza a alocação.

O objetivo geral do trabalho será desenvolver o simulador deste sistema de controle. Basicamente, o programa elaborado deverá oferecer as funcionalidades básicas deste sistema, que serão detalhadas em seguida.

## 4. Noções importantes

O objetivo principal do programa é alocar **vagas** de estacionamento para **veículos,** onde**:**

- Cada veículo tem um modelo, um determinado peso (em Kg), um determinado comprimento (em metros), uma determinada largura (em metros) e uma determinada altura (em metros). Todo o veículo é identificado pelo seu número de chassi (um número inteiro).

- Cada vaga é identificada por um número no sistema, suporta um certo limite de peso (em kg), um determinado comprimento máximo (em metros), uma largura e altura máximas (em metros).

## 5. O programa

O programa deverá inicialmente carregar os dados de 2 arquivos texto, fornecidos ao aluno. O arquivo **`veiculos.txt`** conterá as informações iniciais sobre os veículos que estão à espera de uma vaga no estacionamento, previamente cadastrados pelo operador. O arquivos **`vagas.txt`** conterá as informações iniciais sobre as vagas disponíveis no estacionamento. Ou seja, as informações sobre veículos e vagas serão dadas, o programa apenas deverá processar essas informações.

Cada arquivo corresponde a um conjunto de registros separados entre si por uma quebra de linha. A estrutura de cada registro em cada arquivo é dada como segue:

**`veiculos.txt`**

```
<modelo>,<numero do chassi>,<peso>,<altura>, <comprimento>,<largura>
```

**`vagas.txt`**

```
<identificação>,<peso máximo>,<altura máxima>,<comprimento
máximo>,<largura máxima>
```

## 6. Requisitos

- A interface deve exibir os veículos que estão em espera e as vagas livres no estacionamento. De posse dessas informações, o usuário deve digitar na linha de comando o chassi do veículo e o identificador da vaga para indicar que o veículo foi estacionado na vaga. Quando um veículo é estacionado em uma vaga, o status da vaga deve ser alterado para ocupada e o veículo não deve mais aparecer na lista de espera mostrada ao usuário. Quando um automóvel sai do estacionamento, o operador deve ser capaz de atualizar esta informação, alterando o status da vaga para desocupado.

- Para estacionar um veículo em uma vaga é preciso respeitar as restrições (de peso, altura, largura e comprimento) que a vaga apresenta, de modo que os limites não possam ser excedidos.

- É preciso fornecer a possibilidade de salvar a simulação em um determinado ponto e recuperar esta simulação depois, de modo que ela possa ser continuada. Para isso, o programa deverá ser capaz de salvar e carregar os estados das estruturas de interesse.

- A execução do programa deve ser representada em um log armazenado em arquivo de texto. O log deve registrar o horário de início da simulação, cada uma das entradas de veículos em vagas, cada uma das saídas de veículos das vagas, e, por fim, o horário de fim de execução do programa. Cada registro do log deverá ter o seguinte formato:

<”entrada”/”saída”>,<”sucesso”/”falha”>,<chassi do veículo>,<identificador da vaga>,<horário da tentativa de alocação>

## 7. As opções do menu do programa

**Abrir:**
O usuário deve escolher se quer iniciar uma nova simulação , ou continuar uma simulação salva.

a. Para iniciar uma solução nova, o programa deve:

- Abrir o arquivo **`veiculos.txt`** em modo somente para leitura e carregar seu conteúdo para um array de estruturas que representam cada um dos veículos em espera, com seus respectivos dados.
- Abrir o arquivo **`vagas.txt`** em modo somente para leitura e carregar seu conteúdo para um array de estruturas que representam cada uma das vagas disponíveis, com seus respectivos estados.
- É importante que o programa seja capaz de carregar um número variado (não fixo) de vagas e veículos, por isso deve-se generalizar o programa para carregar qualquer número de corridas.
- Mostrar os veículos em espera na respectiva área da interface.
- Mostrar as vagas disponíveis na respectiva área da interface.
- Deve-se ter cuidado para não mostrar mais dados do que a interface é capaz de comportar. Assim, em caso do número de informações exceder a capacidade da interface, mostra-se apenas os primeiros automóveis da lista .

b. Para iniciar uma simulação salva, o programa deve abrir os arquivos binários com os registros anteriores e carregar esses dados de modo que a simulação possa transcorrer do ponto salvo anteriormente.

**Entrar:**
Nesta opção o usuário aloca uma vaga desocupada do estacionamento para um determinado automóvel. Para isso, exibe-se a lista de veículos em espera, e de vagas disponíveis, respeitando os requisitos acima descritos. Em seguida, o usuário deve informar o número do chassi de um carro e o identificador de uma vaga ao sistema. Se a vaga estiver ocupada, ou se não suportar as características do veículo o sistema deve informar que esta alocação não é possível, em caso contrário, a alocação é realizada, o status da vaga é alterado para ocupada e as listas de veículos em espera e de vagas disponíveis devem ser atualizadas na interface. Independentemente do sucesso ou falha desta operação, é gerado um registro no log de execução.

**Pesquisar:**
Nesta opção o usuário pode informar ao sistema a identificação de um veículo a ser estacionado, e o sistema deve retornar a vaga mais adequada para este veículo. A vaga mais adequada é aquela que primeira satisfaz os requisitos do veículo (em termos de peso e dimensões).

**Sair:**
Nesta opção o usuário desaloca a vaga deixada pelo veículo que está saindo. É exibida na tela a lista de vagas ocupadas e, em seguida, o usuário deve digitar o código de identificação de

uma delas em linha de comando. Independentemente do sucesso ou falha desta operação, é gerado um registro no log de execução.

**Salvar:**

O programa deve salvar o estado atual da aplicação de modo que seja possível continuar a simulação deste ponto em uma outra ocasião. O estado atual do sistema corresponde ao estado das estruturas de interesse (registros de veículos ainda em espera e de vagas). A estrutura do arquivo binário que armazenará o estado da aplicação deve ser definida pelo aluno.

**Relatórios:**

O programa deve ser capaz de salvar em arquivo texto os seguintes relatórios:

- Para cada vaga, a quantidade total de veículos que estacionaram nela hoje e o total de tentativas falhas de estacionar um veículo nela.

- O total de veículos estacionados hoje, longos, curtos, pesados, leves, altos, baixos, largos, estreitos. Um veículo é pesado se tiver peso igual ou superior a 2500 kg. Um veículo é longo se tiver comprimento igual ou superior a 2,5 metros. Um veículo é largo se tiver largura maior ou igual a 1,6 metros. Um veículo é alto se tiver altura maior ou igual a 1,7 metros.

- A lista de veículos que já estacionaram hoje, em ordem decrescente de peso, altura, comprimento e largura, nesta ordem. Ou seja, se dois veículos, a e b, pesarem 3500 kg, mas a tem 1,6 m de altura e b tem 2 m de altura, b deve aparecer antes que a.

**Fim:**

- Pergunta se o usuário realmente deseja encerrar a execução e, em caso positivo, fecha todos os arquivos abertos e fecha o sistema.

## 8. A interface

O aluno pode desenvolver interfaces gráficas se quiser, mas isto NÃO É um requisito para este trabalho (iremos estudar a construção de interfaces gráficas mais adiante).
A interface deve apresentar uma estrutura semelhante a esta:

| **Abrir Entrar Pesquisar Sair Salvar Relatórios Fim** |
|---|

Menu

| <N. Chassi1><br><N. Chassi2><br>... | <Peso 1><br><Peso 2><br>... | <Altura 1><br><Altura 2><br>... | <Comprimento 1><br><Comprimento 2><br>...<br>... | <Largura1><br><Largura2><br>… |
|---|---|---|---|---|

Lista dos veículos em espera

| >> <Número chassi> <identificador da vaga> |
|---|

Linha de comando

| <ID vaga1><br><ID vaga 2><br>... | <peso max1><br><peso max2><br>... | <altura max 1><br><altura max 2><br>... | <comprimento max1><br><comprimento max2><br>... | <Larg max1><br><Larg max2><br><br>... |
|---|---|---|---|---|

Lista de vagas disponíveis

A estrutura acima (ou semelhante a ela) deve ser exibida sempre que o usuário escolhe a opção **Entrar**. Na opção **Sair** , deve-se exibir uma lista das vagas ocupadas, com o respectivo código de identificação . Depois de cada entrada, saída, relatório ou salvamento, a interface deve permitir que o usuário possa continuar selecionando uma das opções do menu.