|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Código: P318**  **Disciplina: Programação Orientada a Objeto** N1 | 2º bimestre| Curso: EC | Turma: <3>14/05/2023 - 23h55 Prof.(ª): Eduardo R. Marcelino | Coord.: Rodrigo Fontes | |  |
| Aluno (a): ­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­­ | Nº: | | RA: | Nota: |
|  |  | |  |  |
| Orientações: | | | |  |
| *Data da vista da prova:*  / /  *- horário:*  h .  Declaro estar de acordo com a nota e com a revisão e correção realizada pelo professor.  O não comparecimento do aluno na vista dos instrumentos avaliativos, na data prevista, significa a concordância tácita com as notas atribuídas, desobrigando a instituição de deferir eventuais pedidos de revisão de notas (após a divulgação de notas no portal do aluno, dentro do semestre letivo ou em situações futuras). | | | | Rubrica do aluno: |

Observações:

* Atividade deverá se feita em dupla. **Ambos devem postar no moodle a atividade**.
* A aplicação deve ser feita em Windows Forms
* Deve-se criar uma área de notificações onde mensagens serão exibidas durante a execução da aplicação.
* Todas as capacidades estão em kg.
* Todas as distâncias e velocidades estão em km/h
* Todos os métodos devem exibir na área de notificações uma mensagem avisando que o método foi executado. **Inclua na mensagem a identificação do veículo**. EX:
  + Liga/Desliga Limpador() -> exibe notificação: ***Limpador do veículo XXXXX ligado.*** 
    - *Aonde* ***XXXXX*** *é a identificação do veículo*
  + *Atracar() -> Exibe notificação: Veículo* ***XXXXX*** *atracado.*
  + *Acelelar(x) -> Exibe notificação: Veículo* ***XXXXX*** *foi acelerado em X*

Veiculo:

Todo veículo deve possuir os seguintes atributos e métodos:

* Identificação (String)
* Modelo (classe Modelo)
* Acelera() (incrementa em 1 a velocidade)
* Desacelera() (diminui em 1 a velocidade)
* Velocidade atual (int)
* Peso: Peso total do veículo, totalmente descarregado.

Deseja-se modelar os seguintes tipos veículos:

* Carro
  + Quantidade portas (int)
  + Liga/Desliga Limpador()
  + Capacidade de passageiros (int)
  + PagarPedagio() (fixo 7,00)
  + Veículo Oficial (booleano) (isento de pedágio)
* Caminhão
  + Liga/Desliga Limpador()
  + Peso carregado (kg) (double)
  + Quantidade de eixos (int)
  + Capacidade de passageiros (int)
  + PagarPedagio() (8,50 por eixo)
  + Carregar(double peso) // carrega o caminhão com um determinado peso, acumulando.
  + Descarregar() // retira todo o peso do caminhão
  + Capacidade máxima para Carregar (kg) (double)
    - Se ultrapassada, o veículo não deve acelerar, sendo indicado gerar uma uma exceção.
* Ônibus
  + Liga/Desliga Limpador()
  + Quantidade de eixos (int)
  + Capacidade de passageiros (int)
  + PagarPedagio() (8,50 por eixo)
  + Leito (booleano)
* Moto
  + Empinar()
  + PagarPedagio() (3,00)
  + Capacidade de passageiros (int)
* Avião
  + Pousar()
  + Arremeter()
  + Decolar()
  + Capacidade de passageiros (int)
  + Liga/Desliga Limpador()
* Avião de guerra
  + Pousar()
  + Arremeter()
  + Decolar()
  + Atacar()
  + Ejetar()
  + Capacidade de passageiros (int)
* Trem
  + Quantidade de vagões (int)
  + Capacidade de passageiros (int)
  + Liga/Desliga Limpador()
* Navio
  + Atracar()
  + Capacidade de passageiros (int)
* Navio de guerra
  + Atracar()
  + Atacar()
  + Capacidade de passageiros (int)

Deseja-se cadastrar separadamente as marcas e os modelos.

* Marca:
  + Código (int)
  + Descrição (string)
* Modelo:
  + Código
  + Descrição (string)
  + Marca (Marca)

Ex:

MARCA

m1 = {1 - VW }

m2 = {2 – GM}

MODELO

Mo1= {1 – GOL – m1}

Mo2= {2 – Celta – m2}

* Crie o diagrama de classes
* Deve ser possível executar todas as ações de todos os tipos de veículos.
* O sistema deve ter uma opção para instanciar, automaticamente, ao menos uma instância de cada veículo.
* Com base nas classes acima, faça as seguintes consultas:
  + Acione o limpador de todos os veículos
  + Atraque todos os navios
  + Ataque com todos os veículos
  + Empine todos os veículos
  + Exibir o total de pedágio pago
  + Exibir todos os veículos

Há também a figura do Pedágio, uma entidade que irá cobrar do veículo um valor. Cada veículo sabe como o valor é pago. O pedágio também deve acumular o total de pedágio pago de todos os veículos que passaram por ele. O sistema deve permitir o cadastro de pedágios. Não há limite para o cadastro de pedágios. Deve haver uma forma de selecionar o pedágio no momento da cobrança (Exemplo: caixa combo contendo os pedágios cadastrados). Caso seja um veículo isento de pedágio, deve-se liberar o veículo sem a cobrança, porém o mesmo deve ser registrado com o valor zero.

* Pedágio
  + Identificação: string
  + Localização : string
  + void Receber (veículos que pagam pedágio)
  + Total de pedágio recebido (double) (acumulado dos pagamentos)
* Deve haver uma opção para listar todos os pedágios cadastrados e os valores recebidos por cada um deles.
* O pedágio também deve manter um histórico de todos os pagamentos recebidos. Deve-se armazenar a data/hora, placa e valor recebido dos veículos que passaram pelo pedágio, com a opção de listar esse histórico na tela. Veículos isentos devem ser registrados com o valor zerado.
* Deve haver uma opção para cobrar, automaticamente e de forma aleatória (na escolha do veículo e da praça de pedágio), o pedágio de alguns veículos previamente cadastrados. Esta opção visa facilitar o teste da aplicação.

Persistência dos dados

* Opção para salvar e recuperar os dados cadastrados de um arquivo texto. Pode ser utilizado o formato JSON.
* Deve-se persistir não só os veículos cadastrados como também o histórico de cobranças do pedágio, caso este tenha sido implementado.

Observações: