



Curso de Computação 2 Programação em Java

Exercício 2 - Herança

Veiculo

- + marca: string
- + modelo: string
- + dono: string
- tanque_capacid: double
- tanque_atual: double
- autonomia_km_l: double
- km_rodados: double
- + Veiculo(marca: string, modelo: string, dono: string, tanque_capacid: double, autonomia_km_l: double)
- + lerTanqueAtual(): double
- + lerTanqueCapacid(): double
- + lerAutonomia(): double
- + lerRodagem(): double
- + alterarAutonomia(autonomia: double): boolean
- + abastecerTanque(litros: double): boolean
- + fazerViagem(km: double): boolean
- validarProp(prop: double): boolean
- + alterarKMRodados(km_rodados: double): void



Caminhao

- + qnt_eixos: integer
- capacid_carga: double
- + Caminhao(marca: string, modelo: string, dono: string, tanque_capacid: double, autonomia_km_l: double, capacid_carga: double qnt_eixos: integer)
- + Caminhao(marca: string, modelo: string, dono: string, tanque_capacid: double, autonomia_km_l: double, km_rodados: double, capacid_carga: double, qnt_eixos: integer)
- + lerCapacidCarga(): double
- + fazerViagem(km: double, carga: double): boolean

Utilizando o IntelliJ IDEA, estenda a classe pública *Veiculo*, que foi criada no exercício anterior, por meio da classe pública *Caminhao*. Implemente a classe *Caminhao* em um arquivo .java separado e de forma que as propriedades e métodos que iniciam com + sejam públicas e com – sejam privadas.





Curso de Computação 2 Programação em Java

Em relação à implementação da classe *Veiculo* realizada no Exercício 1, será necessário criar um novo método chamado *alterarKMRodados* que vai alterar a propriedade privada *km_rodados* com o novo valor informado.

As propriedades da classe Caminhao possuem os seguintes significados:

- qnt_eixos se refere à quantidade de eixos do caminhão como, por exemplo, 2, 3, 4 ou 6 eixos.
- capacid carga é a capacidade de transporte de carga do caminhão em toneladas.

Por sua vez, os métodos da classe devem ser implementados para as seguintes ações:

- Há dois métodos de construção em overload. Em ambos, o método de construção da superclasse deve ser chamada com super(). A diferença entre os métodos é o parâmetro km_rodados, que deve ser atribuído à propriedade de mesmo nome por meio do método alterarKMRodados definido na classe Veiculo. Caso km_rodados não seja informado, o objetivo deve ser iniciado com o valor 0. Dica: use o método de construção da superclasse para inicializar corretamente suas variáveis privadas.
- lerCapacidCarga deve retornar o valor da propriedade privada capacid_carga.
- fazerViagem é um overload, mas não um override, do método de mesmo nome da classe Veiculo. Esse método da classe Caminhao deve testar se capacid_carga >= carga. Se não passar no teste, retorne false. Se passar no teste, deve se fazer o teste já implementado no método da superclasse fazerViagem e retornar seu valor booleano.

Recomenda-se que se crie uma segunda classe *Main* com a estrutura básica abaixo e conforme ensinado em aula para testar com cuidado todo o comportamento solicitado para as classes *Veiculo* e *Caminhao*.

```
public class Main {
    public static void main(String args[]) {
        // faça aqui seus testes da classe Veículo aqui para ter certeza de que está certo!
    }
}
```

Você deve entregar pelo Google Classroom <u>SOMENTE</u> os arquivos Veiculo.java e Caminhao.java como as propriedades e métodos descritos acima. <u>Insira um comentário na primeira linha de cada arquivo com o seu nome e DRE.</u>