Sistema de Gestão de Veículos de Transporte

Uma empresa de transporte deseja desenvolver um sistema para gerenciar seus veículos e viagens. O sistema deve utilizar os princípios da Programação Orientada a Objetos (POO), como encapsulamento, herança, polimorfismo e classes abstratas, para representar diferentes tipos de veículos e seus comportamentos.

A classe Veiculo servirá como base para todos os tipos de veículos do sistema. Ela deve conter atributos privados, garantindo o encapsulamento, e métodos que permitam o controle e a exibição de suas informações.

Os atributos são:

- placa: identifica o veículo;
- modelo: nome ou tipo do veículo;
- capacidade: número máximo de passageiros;
- combustivel: quantidade atual de combustível em litros.

Os métodos definidos em Veiculo incluem:

- abastecer(double litros) → adiciona combustível ao tanque, somando o valor informado à quantidade atual;
- viajar(double distancia) → calcula o consumo de uma viagem baseado na distância, e caso o consumo seja inferior ou igual a quantidade de combustível você deve exibir uma mensagem que deu certo realizar a viagem e debitar o consumo da variável combustivel. Este método é abstrato, e deverá ser implementado de maneira diferente por cada tipo de veículo;
- exibirDados() → apresenta as informações principais sobre o veículo.

A classe também deve conter métodos getters e setters para permitir o acesso controlado aos atributos, respeitando o encapsulamento.

A partir da classe Veiculo, deverão ser criadas três subclasses: Onibus, Caminhao e Carro, cada uma representando um tipo específico de veículo com comportamentos próprios.

A classe Onibus adiciona o atributo quantidadePassageiros, que indica quantos passageiros estão a bordo. O método viajar(double distancia) deve calcular o consumo de combustível considerando 0,5 litro por quilômetro percorrido, mais 0,01 litro por passageiro. Caso o consumo calculado seja maior que a quantidade de combustível disponível, o sistema não deve permitir a viagem e deve exibir uma mensagem informando que o combustível é insuficiente, além de não alterar o valor atual de combustível. O método exibirDados() deve ser sobrescrito para incluir também o número de passageiros.

A classe Caminhao possui o atributo cargaAtual, que representa o peso da carga transportada em toneladas. O método viajar(double distancia) deve calcular o

consumo total com base em 1,2 litro por quilômetro, somado a 0,2 litro por tonelada transportada. Assim como nas demais subclasses, se o consumo calculado for maior que a quantidade de combustível disponível, a viagem não deve ocorrer e uma mensagem informando “Combustível insuficiente para o caminhão” deve ser exibida e o nível de combustível permanece inalterado. O método exibirDados() é sobrescrito para mostrar, além das informações básicas, o peso atual da carga transportada.

A classe Carro adiciona o atributo booleano arCondicionadoLigado, indicando se o ar-condicionado está em uso. O método viajar(double distancia) deve calcular o consumo base de 0,15 litro por quilômetro, e caso o ar-condicionado esteja ligado, esse valor deve ser aumentado em 10%. Se o combustível não for suficiente para a viagem, o método deve exibir a mensagem “Combustível insuficiente para o carro” e a viagem não deve ser realizada e o combustível, portanto, não deve ser alterado. O método exibirDados() deve ser sobrescrito para incluir a informação sobre o estado do ar-condicionado (ligado ou desligado).

A classe SistemaTransporte é responsável por gerenciar todos os veículos cadastrados no sistema. Ela deve conter uma lista de objetos do tipo Veiculo, permitindo armazenar diferentes tipos (ônibus, caminhões e carros), o que demonstra o uso de polimorfismo.

Os principais métodos são:

- registrarVeiculo(Veiculo v) → adiciona um novo veículo à lista de veículos;
- realizarViagens(double distancia) → percorre toda a lista e chama o método viajar() de cada veículo. Aqui ocorre o polimorfismo, pois o comportamento real do método depende do tipo específico do objeto (ônibus, caminhão ou carro);
- exibirRelatorio() → exibe os dados atualizados de todos os veículos cadastrados, chamando o método exibirDados() que foi sobrescrito nas subclasses.