Assuntos abordados:

- Herança
- Classes / métodos abstratos
- Exception

Cenário

Uma revenda trabalha com três tipos de veículos: motos, veículos de passeio e veículos de carga.

Todos os veículos possuem placa, ano de fabricação e um valor.

Sobre as motos é importante saber a categoria (texto) e a quantidade de cilindradas.

Sobre os veículos de passeio deve-se manter o registro da quantidade de passageiros máxima e a capacidade do porta-malas (em litros).

Sobre os veículos de carga é necessário saber a capacidade máxima de carga (em toneladas), quantidade de eixos e a capacidade do tanque de combustível (em litros).

Sobre os veículos em geral, também é importante saber os seguintes dados do comprador: cpf, nome, endereço (CEP, logradouro, número e bairro) e o telefone. Uma mesma pessoa pode comprar diversos veículos, mas um veículo pode ter apenas um comprador.

Com base neste cenário, construa um programa em Java com as seguintes características.

1. [0,5] Crie uma exceção para tratar o caso de um dos valores numéricos seja negativo. Os atributos de valor do veículo, quantidade de cilindradas, quantidade de passageiros máxima, capacidade do porta malas, quantidade máxima de carga, de eixos e do tanque de combustível devem ser todas positivas. A correta seleção da superclasse desta exceção (Exception, RuntimeException e Error) faz parte da avaliação. O nome da exceção deve ser NegocioException, sendo que a mensagem do problema encontrado deve ser enviada como parâmetro para a exceção.

2. [4,5] Defina uma hierarquia e crie as classes necessárias ao cadastro de veículos, observando as seguintes recomendações:

    2.1. Lembre-se de que apenas motos, veículos de passeio e veículos de carga

podem ser cadastrados, ou seja, não deve ser possível instanciar um veículo que não seja de um desses tipos;

2.2. As classes devem ter construtores parametrizados com todos os seus atributos para facilitar a instanciação das mesmas:

2.3. As classes devem ter métodos getters e setters para viabilizar alteração em seus atributos;

2.4. As classes devem ter implementação do método toString para facilitar a listagem de dados;

2.5. Um método que receba um valor numérico como parâmetro e caso este valor seja negativo, deve lançar a exceção NegocioException, com a mensagem: <nome-do-atributo> + " não pode ser negativo(a).". Por exemplo, se for passada uma quantidade máxima de carga negativa, deve ser lançada uma instância da exceção NegocioException com a mensagem "A quantidade máxima de carga não pode ser negativo(a).";

2.6. Os métodos que definirem os atributos valor do veículo, quantidade de cilindradas, quantidade de passageiros máxima, capacidade do porta malas, quantidade máxima de carga, de eixos e do tanque de combustível, devem utilizar o método do item "2.5" para realizar a validação dos mesmos;

3. [1,0] Crie uma classe VeiculoDAO, com os métodos estáticos necessários ao atendimento das demandas da camada de apresentação (TUI). Você deve ter uma única lista, que deverá armazenar todos os tipos de veículo;

4. [1,0] Crie uma classe PessoaDAO, com os métodos estáticos necessários ao atendimento das demandas da camada de apresentação (TUI). Você deve ter uma lista, que deverá armazenar as pessoas (compradores);

5. Crie uma classe VeiculoTUI:

   5.1. Métodos estáticos:

   a) [1,0] cadastrarMoto: este método deve obter os dados necessário para instanciar uma moto e utilizar o método adequado do VeiculoDAO para persistir esta instância;

   b) [0,5] cadastrarVeiculoPasseio: este método deve obter os dados necessário para instanciar um veículo de passeio e utilizar o método adequado do VeiculoDAO para persistir esta instância;

c) [0,5] cadastrarVeiculoCarga: este método deve obter os dados necessário para instanciar um veículo de carga e utilizar o método adequado do VeiculoDAO para persistir esta instância;

6. Crie uma classe PessoaTUI:

6.1. Métodos estáticos:

a) [1,0] cadastrarPessoa: este método deve obter os dados necessário para instanciar uma pessoa e utilizar o método adequado do PessoaDAO para persistir esta instância;

Atenção:

1. **Nessa implementação, as validações devem ficar nas classes de domínio e a classe TUI deve acessar diretamente a classe de DAO, NÃO devendo ser implementada uma classe de negócio;**
2. **NÃO deve ser implementada um menu para acesso às funcionalidades.**

Arquitetura definida para a atividade: