Atividade 03 – Herança

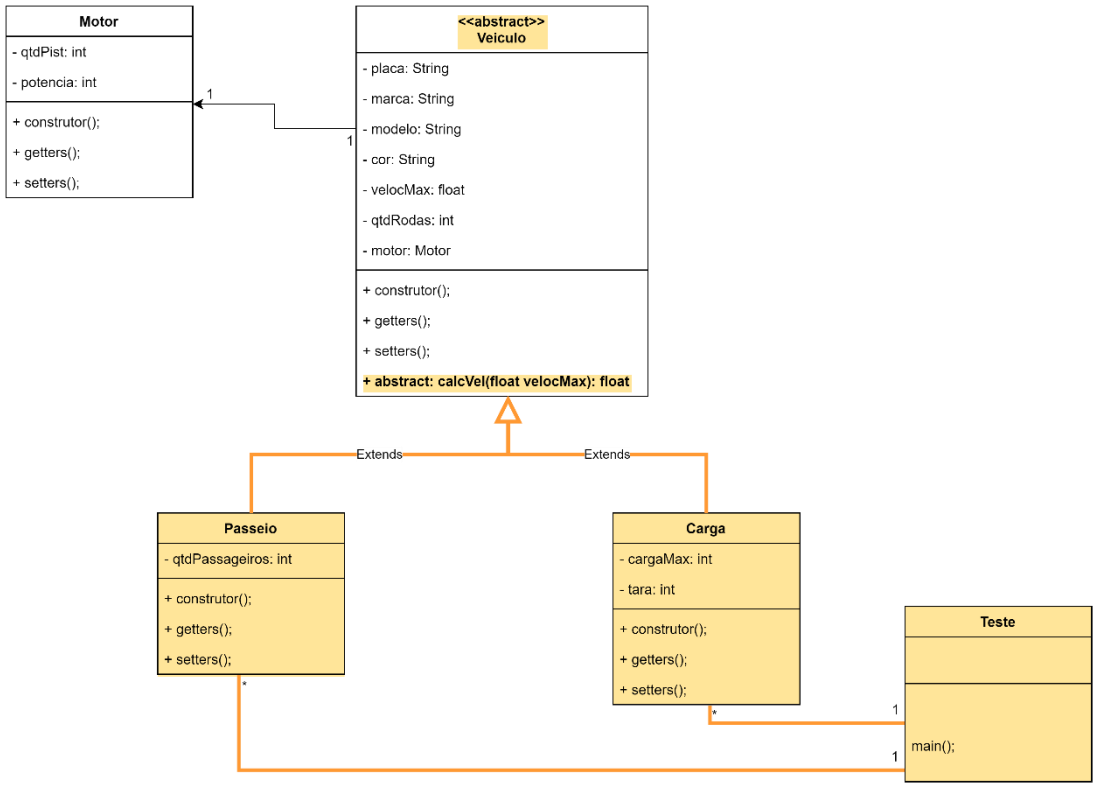
➔ **Este exercício trata-se de uma evolução da Atividade 01.**

Embora a Atividade 3 trate do conceito e aplicação do mecanismo de Herança, ela também pode nos ajudar compreender a realidade das empresas e instituições que buscam intensificar o uso de padrões de projetos desenvolvidos, de maneira *ad hoc,* por elas mesmas, com intuito de padronizar a construção de seus softwares.

Por exemplo, ao definir-se como "final" um método "set", indicando que não poderá ser sobrescrito, garantimos a perpetuação de uma regra de negócio restritiva sobre as características possíveis de um objeto (um atributo deste), como quando não queremos que um atributo receba um valor fora de uma escala prevista.

1) Observe o seguinte Diagrama de Classes

Obs.: *As alterações e novos elementos referentes à atividade encontram-se marcadas em amarelo e laranja (relacionamentos) no diagrama.*



2) Utilize o código desenvolvido na Atividade 01 e desenvolva os novos elementos apresentados no digrama acima. Abaixo, segue a lista de requisitos a serem seguidos:

a. A “entrada” da velocidade (atributo *velocMax*) sempre será dada em km/h porém, a exibição destes dados ocorrerá na classe Teste e da seguinte forma:

   i. A velocidade do veículo de **passeio** deverá ser calculada em m/h;

1 kilometer/hour = 1000 meter/hour

ii. A velocidade do veículo de **carga** deverá ser calculada em cm/h;

1 kilometer/hour = 100000 centimeter/hour

Use o método *calcVel(**float velocMax**)*, para fazer este cálculo. O método deve ser herdado da classe Veiculo (mãe).

Atenção:

- O método *calcVel(**float velocMax**)* **NÃO** deve alterar o valor do atributo velocMax, **apenas convertê-lo** e retornar o valor convertido para que seja exibido na tela por meio da classe Teste;

b. Os métodos construtores default **das novas classes** também deverão iniciar com 0 (zeros) atributos que sejam de tipos numerais (int, double, float, etc.) e com espaço em branco os que forem de tipo literais (char, String e etc.).

c. Garanta que nunca ocorra:

i. As classes **Passeio** e **Carga jamais** deverão ser estendidas (herdadas);

ii. Nenhum método "set" (de nenhuma classe) poderá ser sobrescrito;

d. A classe "Teste" deve ser construída de forma a testar todas as funcionalidades do programa (entrada, saída e cálculos), propiciando assim "trocas de mensagens" entre os objetos das classes Teste←→Passeio e Teste←→Carga. *Por meio dela deverá ser possível instanciar 5 veículos de cada tipo (Passeio/Carga)*.

3) O QUE SERÁ AVALIADO

a. Construção das classes, com os atributos e métodos conforme descritos no diagrama de classe do item 01.

b. Relacionamento de herança entre as classes.

c. Cada uma das solicitações presentes no item 2.

d. Implementação da relação entre as classes Teste -> Passeio, Teste -> Carga, conforme solicitado no item **2.d.**

e. Uso do encapsulamento.

Importante!

- Atenha-se aos nomes dos elementos (classes, atributos e métodos) conforme apresentados no diagrama.
- Novos métodos poderão ser criados, caso julgue necessário.
- Os itens avaliados são os solicitados no enunciado. Elementos extras NÃO renderão pontos a mais.
- O não cumprimento do que foi solicitado acarretará no decréscimo da nota de acordo com a gravidade da falta.
- A justificativa para qualquer desconto será colocada, pelo avaliador, no campo de feedback de cada Atividade.

4) Exemplo de entrada de dados e saída esperada.

Entrada de dados

```java
public static void main(String[] args) {

    Passeio veiculoPasseio = new Passeio();
    veiculoPasseio.setMarca("Nissan");
    veiculoPasseio.setModelo("March");
    veiculoPasseio.setPlaca("ABC 1234");
    veiculoPasseio.setVelocMax(200);
    veiculoPasseio.setQtdePassageiros(5);
    veiculoPasseio.getMotor().setPotencia(70);
    veiculoPasseio.getMotor().setQtdPist(3);
```

```java
public static void main(String[] args) {

    Carga veiculoCarga = new Carga();
    veiculoCarga.setMarca("Fiat");
    veiculoCarga.setModelo("Strada");
    veiculoCarga.setPlaca("BBB 3333");
    veiculoCarga.setVelocMax(164);
    veiculoCarga.setCargaMax(705);
    veiculoCarga.setTara(1084);
    veiculoCarga.getMotor().setPotencia(88);
    veiculoCarga.getMotor().setQtdPist(7);
```

Obs.: A entrada de dados pode ser feita por meio dos métodos setters (como nos exemplos acima) ou utilizando os construtores de cada classe.

Saída de dados esperada

```
============= VEICULO PASSEIO ============

Marca := Nissan
Modelo := March
Placa := ABC 1234
Velocidade Maxima := 200000,000
Qtd Passageiros := 5
Potencia do Motor := 70
Qtd Pistoes Motor := 3

==========================================
```

```
============== VEICULO CARGA =============

Marca := Fiat
Modelo := Strada
Placa := BBB 3333
Velocidade Maxima := 16400000,000
Tara := 1084
Carga Maxima := 705
Potencia do Motor := 70
Qtd Pistoes Motor := 3

==========================================
```

Obs.: Embora os exemplos acima mostrem apenas 1 saída para cada tipo de veículo, a aplicação desenvolvida deve ter a saída para os CINCO veículos de CADA TIPO, conforme solicitado no enunciado acima,