



UNIVERSIDADE FEDERAL DO CEARÁ
Campus de Quixadá
Prof. Thiago Werlley
QXD0007- Programação Orientada Objeto

Prática
2022.2

Lista de Exercícios 04 – Herança

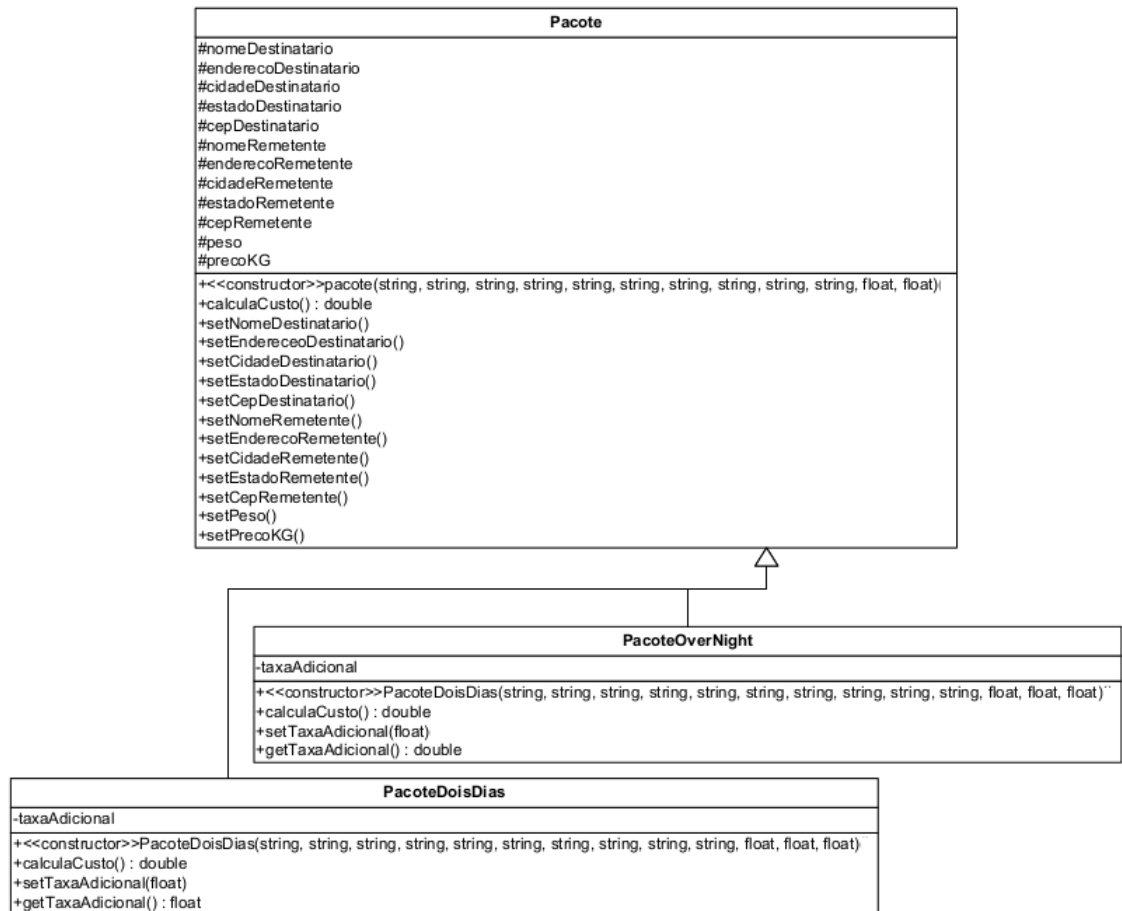
Instruções

- Todos os exercícios que envolvem programas devem ser resolvidos por programas em linguagem C++;
 - Na solução dos exercícios, devem ser utilizados os conceitos listados no cabeçalho desta lista;
 - Eventuais dúvidas podem ser sanadas com o professor.
1. Crie uma classe Empresa capaz de armazenar os dados de uma empresa (Nome, Endereço, Cidade, Estado, CEP e Telefone). Inclua um construtor sem argumentos e um com argumentos para inicialização dos atributos. Crie métodos que funcionem como getter e setter e print.

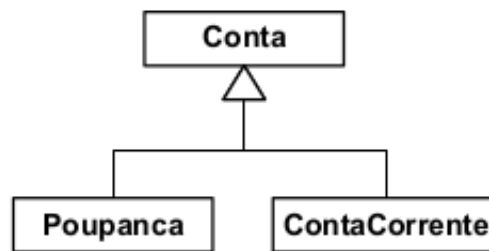
Utilize a classe Empresa como base para criar a classe Restaurante. Esta classe derivada deve conter atributos que representem o tipo de comida e o preço médio de um prato. Crie um construtor para esta classe que chame explicitamente o construtor da classe Empresa, um getter e um setter, além de um método print, que utiliza o método print da classe base. Crie um driver para testar sua aplicação.
 2. Crie a classe Veiculo, contendo o peso, a velocidade máxima, e o preço. Inclua um construtor sem argumentos e um com argumentos para inicialização dos atributos. Crie métodos que funcionem como getter e setter e print.

Crie a classe Motor, contendo o número de cilindros e a potência. Inclua um construtor sem argumentos e um com argumentos para inicialização dos atributos. Crie métodos que funcionem como getter e setter e print.

Crie a classe CarroPasseio, derivada das classes Veiculo e Motor. Inclua atributos como modelo e cor. Crie um construtor para esta classe que chame explicitamente o construtor das classes base, um getter e um setter, além de um método print, que utiliza o método print da classe base. Crie um driver para testar sua aplicação.
 3. Crie a classe Caminhao, derivada das classes Motor e Veiculo. Inclua os atributos toneladas, altura máxima e comprimento. Crie um construtor para esta classe que chame explicitamente o construtor das classes base, um getter e um setter, além de um método print, que utiliza o método print da classe base. Crie um driver para testar sua aplicação.
 4. Existem diferentes tipos de opções de entregas de pacotes, cada um com custos específicos associados. Crie uma hierarquia de herança para representar várias formas de entrega de pacotes de acordo com o diagrama UML abaixo. Todos os construtores devem realizar testes de consistência quanto aos dados fornecidos (por exemplo, garanta que o peso e o preço por quilo não serão menores ou iguais a zero). O cálculo dos custos deve ser realizado como especificado abaixo:
 - a. Classe Pacote: peso x preço por quilo;
 - b. Classe PacoteDoisDias: resultado do cálculo anterior somado de uma taxa adicional;
 - c. Classe PacoteOvernight: resultado do item a somado a uma taxa adicional multiplicada pelo peso do pacote.



5. Crie uma hierarquia de herança que um banco possa utilizar para representar dois tipos de conta: poupança e conta corrente. Todos os clientes deste banco podem depositar e sacar dinheiro de suas contas.



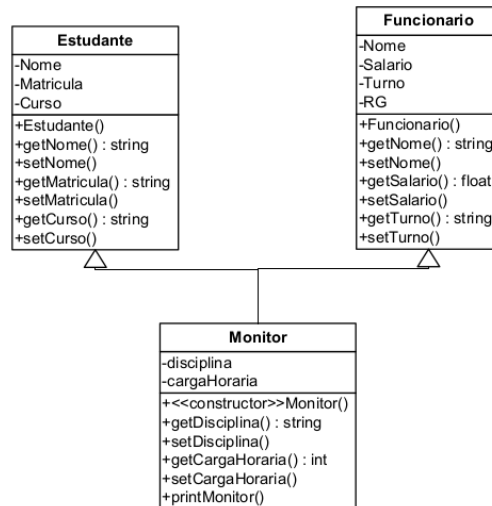
A classe Conta deve possuir um atributo que represente o saldo da conta. Este atributo deve ser inicializado através de um construtor parametrizado que valide o valor enviado como parâmetro. Devem ser criados métodos para mostrar o saldo, para crédito e para débito na conta. Note que se o valor de débito for maior que o saldo, deve ser impressa uma mensagem de erro. Crie um getter e um setter para o atributo.

A classe Poupanca deve possuir um atributo relacionado à variação (rendimento), com métodos getter, setter e construtor. Crie também um método CalculaRendimento, que informa o valor do saldo multiplicado pela taxa de rendimento.

A classe ContaCorrente deve incluir um atributo que represente a taxa cobrada por cada transação de crédito/débito, com getter, setter e construtor. Redefina os métodos de crédito e débito para descontar o valor de tal taxa a cada transação bem sucedida. Os métodos originais da classe Conta devem ser invocados na redefinição. Crie um driver para testar sua hierarquia.

Os exercícios abaixo não consideram rigor taxonômico. São apenas exercícios simples para fixação de herança múltipla.

6. Crie classes Humano, Inseto e SuperHeroi. A classe Humano possui atributos Nome, Sexo, Idade, Lingua e Etnia. A classe Inseto possui atributos Nome, Venenoso, Alado, Ferrao (os três últimos são booleanos). A classe SuperHeroi herda das duas classes anteriores, e adiciona os atributos Codinome, Trauma e Poderes. Todos os atributos possuem getters e setters. Crie a hierarquia de herança múltipla, um construtor e um método print na classe derivada. Crie um driver para testar sua hierarquia.
7. Implemente a hierarquia de herança múltipla definida pelo diagrama UML abaixo:



8. Implemente a hierarquia de herança múltipla definida pelo diagrama UML abaixo. Em cada classe, defina um construtor e defina três atributos, com getter e setter. Implemente apenas a classe VeiculoAnfibio como classe derivada e crie um método print, que imprime todos os dados de um veículo anfíbio.

