PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso de Bacharelado em Engenharia de Software

Disciplina	Departamento	Turno	Período
Programação Modular	Ciência da Computação	Manhã/Noite	2°

Professor
Hugo de Paula (hugo@pucminas.br)

Matrícula:	Aluno:

Exercício 3

1. Por que é possível afirmar que, do ponto de vista da programação modular, a herança rompe a barreira do encapsulamento e aumenta o acoplamento?

2. Implemente um programa em Java com os requisitos a seguir:

   (a) Uma classe chamada `Carro`, com os seguintes atributos e métodos:

   ```
   int velocidade;
   double preco;
   String cor;
   double getPrecoVenda();
   ```

   (b) Uma subclasse de `Carro` chamada `Caminhao`, com os seguintes atributos e métodos:

   ```
   int carga;
   double getPrecoVenda();
   // se (carga > 2000) 10% desconto, senao 20% desconto
   ```

   (c) Uma subclasse de `Carro` chamada `Fiat`, com os seguintes atributos e métodos:

   ```
   int descontoDeFabrica;
   double getPrecoVenda();
   // subtrai o desconto do preco do veiculo
   ```

   (d) Uma subclasse de `Carro` chamada `Sedan`, com os seguintes atributos e métodos:

   ```
   int comprimento;
   double getPrecoVenda();
   //  se (comprimento > 6 metros) 5% desconto, senao 10% desconto
   ```

   (e) Crie uma classe `AgenciaDeVeiculos` com o método `main`. O método `main` deve instanciar objetos das classes descritas anteriormente e exibir os dados dos objetos inicializados.

3. Baseado a classe `Pessoa` desenvolvida no Exercício 1, crie as subclasses de `Pessoa` – `PessoaFísica` e `PessoaJuridica` – que especificam o comportamento de `Pessoa`. Essas classes devem:

   (a) Adicionar os atributos (encapsulados) pertinentes às suas novas funções.

   (b) Possuir métodos para consultar e alterar esses novos atributos.

   (c) Possuir construtores com parâmetros que permitam a construção de objetos com valores diferentes do padrão.

Matrícula:	Aluno:

Em seguida, Crie a classe `Empresa` que armazena uma lista de `Clientes` e de `Funcionarios`. Clientes podem ser pessoas físicas ou jurídicas, mas funcionários podem ser apenas pessoas físicas. Clientes possuem um limite de crédito, enquanto funcionários possuem cargo e salário.

Desenvolva o diagrama UML que representa essa aplicação. Desenvolva os métodos necessários para que as relações entre as classes seja funcional. Utilize a abordagem de desenvolvimento dirigido por testes.

4. Em relação à Questão 3, explique do ponto de vista teórico e de implementação, as implicações de se utilizar herança para implementar as classes `Cliente` e `Funcionario`.

5. Considere agora que, tanto a classe `Pessoa` e seus derivados, quanto a classe `Empresa`, possuem um endereço, com

$$nome, endereco, telefone, cep, cidade, UF$$

. Adicione esse novo requisito sem produzir código repetido. Atualize o diagrama UML para refletir esse novo requisito.

6. Suponha agora que a `Empresa` deseje armazenar em atributo próprio o seu `Presidente`, sendo que o `Presidente` da empresa é também um `Funcionário`. Desenhe o diagrama UML resultante. Adicione comentários no diagrama para explicar as suas decisões. Não é necessário implementar.