UNIVERSIDADE FEDERAL DO CEARÁ

Lista - 2
Disciplina de Técnicas de Programação

Departamento de Engenharia da Computação

Prof. Thiago Iachiley

entrega até às 23h:59 do dia 17/07/2025

1. Uma empresa precisa gerenciar informações básicas de seus funcionários. Cada funcionário possui um nome, um cargo e um salário. O salário não deve ser acessado ou modificado diretamente de fora da classe para garantir a integridade dos dados. Crie uma classe Funcionario em Java com os seguintes atributos: nome (String), cargo (String) e salario (double). O atributo salario deve ser privado. Implemente um construtor que inicialize todos os atributos. Adicione métodos públicos (getters) para todos os atributos e um método setSalario(double novoSalario) que permita modificar o salário, mas apenas se o novoSalario for maior que zero. No método main, crie um objeto Funcionario, defina um salário inválido (negativo) e depois um válido, mostrando como o encapsulamento funciona.

2. Desenvolva um sistema simples para calcular a área de diferentes formas geométricas. Crie uma classe Retangulo com atributos largura (double) e altura (double). Implemente um construtor. Adicione um método público chamado calcularArea() que retorne a área do retângulo. No método main, instancie Retangulo, defina seus valores e imprima a área calculada.

3. Uma locadora de veículos precisa organizar seus dados. Todos os veículos têm uma marca e um ano. Carros, além disso, possuem um número de portas. Crie uma classe base Veiculo com atributos marca (String) e ano (int). Implemente um construtor e métodos getters para ambos. Em seguida, crie uma subclasse Carro que herde de Veiculo e adicione o atributo numeroPortas (int). O construtor de Carro deve inicializar seus próprios atributos e chamar o construtor da superclasse. No método main, instancie um Carro, atribua valores e imprima todos os seus atributos, demonstrando a herança.

4. Em um zoológico virtual, diferentes animais emitem sons distintos. Crie uma classe abstrata Animal com um método abstrato emitirSom(). Crie duas subclasses concretas: Cachorro e Gato. Ambas as subclasses devem sobrescrever

o método emitirSom() para imprimir "Latido" e "Miado", respectivamente. No método main, crie uma lista de Animal e adicione instâncias de Cachorro e Gato. Itere sobre a lista e chame o método emitirSom() para cada animal, demonstrando o polimorfismo.

5. Um sistema bancário precisa gerenciar contas correntes, onde o saldo não pode ser negativo. Crie uma classe ContaCorrente com um atributo privado saldo (double). Implemente um construtor que inicialize o saldo. Crie um método público depositar(double valor) que adicione o valor ao saldo. Crie um método público sacar(double valor) que subtraia o valor do saldo, mas apenas se o saldo for suficiente (o saldo não pode se tornar negativo). Se o saque não for possível, imprima uma mensagem de erro. Adicione um getter para o saldo. No main, simule operações de depósito e saque para testar as regras de negócio.

6. Em um sistema de figuras geométricas, queremos calcular a área de um círculo. Crie uma classe FormaGeometrica com um método calcularArea() que retorna 0.0 (ou lança uma exceção, indicando que deve ser sobrescrito). Crie uma subclasse Circulo que herde de FormaGeometrica e adicione um atributo raio (double). O Circulo deve sobrescrever o método calcularArea() para calcular a área de um círculo ($\pi \cdot raio2$). Use Math.PI para o valor de Pi. No main, instancie um Circulo, defina o raio e chame calcularArea(), demonstrando a sobrescrita.

7. Um sistema de processamento de pedidos pode lidar com diferentes tipos de itens. Crie uma classe base ItemPedido com um método calcularPreco() que retorna 0.0. Crie duas subclasses: Produto (com atributos nome e precoUnitario) e Servico (com atributos descricao e horasTrabalhadas e valorHora). Ambas as subclasses devem sobrescrever calcularPreco() para retornar o preço total (precoUnitario para Produto e horasTrabalhadas * valorHora para Servico). Crie uma classe ProcessadorPedidos com um método público processar(ItemPedido item) que recebe um ItemPedido e imprime seu preço, demonstrando o polimorfismo em parâmetros. No main, crie instâncias de Produto e Servico e passe-as para o método processar().

8. Uma aplicação precisa gerenciar usuários com nome de usuário e senha, mas a senha nunca deve ser exposta diretamente. Crie uma classe Usuario com atributos privados nomeUsuario (String) e senha (String). O construtor deve receber ambos. Adicione um método público autenticar(String senhaDigitada) que retorna true se a senhaDigitada coincidir com a senha armazenada, e false caso contrário. Não adicione um getter para a senha. No main, crie um Usuario e teste o método autenticar com senhas corretas e incorretas, reforçando o encapsulamento.

9. Em um sistema de gerenciamento de biblioteca, tanto livros quanto revistas são publicações e compartilham algumas características. Crie uma classe base Publicacao com atributos titulo (String) e anoPublicacao (int), e um método exibirDetalhes() que imprime esses detalhes. Crie uma subclasse Livro que herde de Publicacao e adicione um atributo autor (String). A subclasse Livro deve reutilizar o método exibirDetalhes() da superclasse e adicionar a impressão do autor. No main, instancie um Livro e chame exibirDetalhes(), mostrando a reutilização e extensão.

10. Um sistema de gerenciamento de recursos humanos precisa calcular o bônus de diferentes tipos de colaboradores. Crie uma classe base Colaborador com um atributo nome (String) e um método abstrato calcularBonus() que retorna um double. Crie duas subclasses concretas: Desenvolvedor (com atributo salarioBase) e Gerente (com atributo salarioBase e quantidadeProjetosGerenciados).

- Desenvolvedor deve sobrescrever calcularBonus() para retornar 10% do salarioBase.
- Gerente deve sobrescrever calcularBonus() para retornar 15% do salarioBase mais R$ 100 por quantidadeProjetosGerenciados.

No método main, crie um array ou ArrayList de Colaborador e adicione instâncias de Desenvolvedor e Gerente. Itere sobre a coleção e imprima o nome e o bônus de cada colaborador, demonstrando o polimorfismo com coleções.

Instruções para os Alunos:

- Para cada questão, crie um novo arquivo .java com a classe principal (public class NomeDaClasseProblema { ... }) e as classes auxiliares (class OutraClasse { ... }).
- Certifique-se de que o código compile e execute sem erros.
- Os resultados da execução (saídas no console) devem demonstrar a correta implementação dos conceitos pedidos.
- Zip todos os arquivos, com todos os códigos e faça o upload no Sigaa.
- Será atribuída nota 0 aos programas implementados de forma igual a outros colegas.