Lista de Exercícios: Herança e Design de Hierarquias

1. Gestão de Frota na "Mobilidade Urbana"

A locadora de veículos "Mobilidade Urbana" precisa de um sistema para gerenciar sua frota. A análise inicial identificou que todos os veículos, sejam carros ou motos, compartilham características essenciais como marca, modelo e ano. Sua tarefa é criar a base para este sistema.

- **Parte A: A Base Comum (Classe Veiculo):** Crie uma classe Veiculo com os atributos private String marca, private String modelo e private int ano.
- **Parte B: As Especializações (Carro e Moto):** Crie as classes Carro e Moto, que devem herdar de Veiculo (use a palavra-chave extends). A classe Carro deve ter um atributo adicional private int numeroDePortas, e a Moto, private int cilindradas.
- **Parte C: Construção Segura (Construtores):** Adicione construtores a todas as classes. Os construtores de Carro e Moto devem receber todos os dados necessários (os herdados e os específicos) e usar super() para delegar a inicialização dos atributos comuns à superclasse Veiculo.
- **Parte D: Teste de Frota (main):** Em uma classe Main, crie uma instância de Carro e uma de Moto e imprima seus dados para verificar se a hierarquia foi construída corretamente.

2. Refatorando para o Futuro no Sistema Acadêmico

O sistema da Unicesumar está crescendo. A gestão percebeu que, além de Aluno, eles em breve precisarão cadastrar Professor e FuncionarioAdministrativo. Todos eles são tipos de Pessoa e compartilham dados como nome e cpf. Em vez de duplicar código no futuro (uma má prática), você foi encarregado de refatorar a classe Aluno existente para uma nova arquitetura baseada em Herança.

- **Sua Tarefa:** Crie uma superclasse Pessoa com os atributos private String nome e private String cpf e um construtivo para inicializá-los. Em seguida, modifique sua classe Aluno existente para que ela herde de Pessoa, removendo os atributos duplicados e ajustando seu construtor para usar a chamada super().

3. Pacientes da Clínica Veterinária "PetSaúde"

A clínica veterinária "PetSaúde" solicitou um software para gerenciar a ficha de seus pacientes. A análise de requisitos mostrou que eles atendem principalmente cães e gatos, que são tipos de Animal. Todo animal na clínica tem um nome e uma idade. Cães têm uma raça específica, enquanto gatos são diferenciados pela corDoPelo.

- **Sua Tarefa:** Modele e implemente essa hierarquia. Crie uma superclasse Animal e as subclasses Cachorro e Gato. Garanta que todas as classes sejam encapsuladas (private) e possuam construtores para inicialização segura dos dados, usando super() onde for apropriado.

4. Modelando Produtos em um E-commerce

Um e-commerce vende diferentes tipos de produtos. Existem Produtos genéricos, que possuem id, nome e preco. No entanto, existem categorias específicas com atributos próprios. Por exemplo, um Eletronico é um tipo de Produto que também possui voltagem e garantiaEmMeses. Um Movel é um tipo de Produto que possui material e cor.

- **Sua Tarefa:** Implemente a hierarquia de classes Produto (superclasse), Eletronico (subclasse) e Movel (subclasse). Todas as classes devem ser encapsuladas e ter construtores adequados que utilizem a chamada super() corretamente.

5. O Teste do "É-UM" (Análise de Design)

Como arquiteto de software, você precisa decidir se a relação entre os conceitos abaixo deve ser modelada com Herança ("é-um-tipo-de") ou se seria melhor usar Composição ("tem-um"). Em um comentário no seu código, justifique sua escolha para cada par:

- a) NotaFiscal e ItemDaNotaFiscal
- b) Funcionario e Endereco
- c) IngressoVIP e Ingresso
- d) Pedido e Cliente
- e) Carro e Pneu

6. Prevendo a Ordem da Construção (Leitura de Código)

Considere o código da hierarquia abaixo. Sem compilar, preveja a ordem exata das mensagens que serão impressas no console quando a linha new SubSubClasse(); for executada no main. Justifique sua resposta em um comentário.

```
class SuperClasse {
  public SuperClasse() {
    System.out.println("1. Construtor da SuperClasse");
  }
}
class SubClasse extends SuperClasse {
  public SubClasse() {
    super(); // Chamada explícita (mas seria implícita de qualquer forma)
    System.out.println("2. Construtor da SubClasse");
  }
}
class SubSubClasse extends SubClasse {
  public SubSubClasse() {
    super();
    System.out.println("3. Construtor da SubSubClasse");
  }
}
```

7. Desafio de Design: Formas Geométricas

O software de design gráfico "CanvasPro" precisa de uma estrutura de classes para representar formas geométricas. Todas as formas possuem uma cor (String). Círculos são um tipo de forma, definidos por um raio. Retângulos são outro tipo, definidos por base e altura.

- **Sua Tarefa:** Projete e implemente uma hierarquia de classes que modele essa situação. Crie a superclasse FormaGeometrica e as subclasses Circulo e Retangulo. Implemente construtores apropriados para todas as classes.

8. Caça ao Bug: O Construtor Esquecido

O código abaixo não compila. Analise as classes Pai e Filha. Identifique o erro de compilação na classe Filha e explique em um comentário por que ele ocorre. Em seguida, corrija o código para que ele compile e funcione.

```
// CÓDIGO COM ERRO
class Pai {
   String nome;
   public Pai(String nome) {
      this.nome = nome;
   }
}

class Filha extends Pai {
   int idade;
   // O erro de compilação está aqui! Por quê?
   public Filha(int idade) {
      this.idade = idade;
   }
}
```

9. Pensando em Relacionamentos (Conceitual)

Você está modelando um sistema universitário. Você tem os conceitos Universidade e Departamento (ex: "Departamento de Computação"). Um Departamento "é-um-tipo-de" Universidade (Herança) ou uma Universidade "tem-muitos" Departamentos (Composição)? Justifique sua escolha de design.

10. Uma Prévia do Futuro: Coleções de Tipos Comuns

Usando a hierarquia de Veiculo que você criou, experimente o seguinte no seu método main:

1. Crie um ArrayList que possa guardar Veiculos: List<Veiculo> minhaFrota = new ArrayList<>();
2. Crie um objeto Carro e um objeto Moto.
3. Adicione ambos os objetos à lista minhaFrota.

O código compilou e funcionou? Em um comentário, explique por que o Java permite que você coloque um Carro e uma Moto dentro de uma lista de Veiculo. (Dica: pense na relação "é-um-tipo-de").