INF008 – Programação Orientada a Objetos

2ª. Lista de Exercícios

Questão 1: Sistema Financeiro - Contas Bancárias

```
package br.com.banco.model;

public class ContaBancaria {
    private String numero;
    private double saldo;
    private String tipo; // "CORRENTE" ou "POUPANCA"

    public ContaBancaria(String numero, String tipo) {
        this.numero = numero;
        this.tipo = tipo;
        this.saldo = 0.0;
    }

    public ContaBancaria(String numero, String tipo, double saldoInicial) {
        // TODO: Implementar chamada a outro construtor
        // TODO: Validar saldoInicial >= 0
    }

    // TODO: Criar métodos sobrecarregados para depositar
    // depositar(double valor) e depositar(double valor, String descricao)

    private boolean validarValor(double valor) {
        return valor > 0;
    }

    // TODO: Criar getters APENAS para os atributos necessários
}
```

Pergunta: Implemente os construtores e métodos seguindo rigorosamente: (1) Um construtor deve chamar o outro usando this, (2) Sobrecarregue o método depositar mantendo a validação centralizada, (3) Use private para o método de validação, (4) Exponha apenas getters necessários.

Questão 2: Rede Social - Gerenciamento de Posts

```
package com.rede.social.entity;


public class Post {
    private String conteudo;
    private String autor;
    private int curtidas;
    private boolean publico;

    public Post(String conteudo, String autor) {
        this(conteudo, autor, true);
    }

    public Post(String conteudo, String autor, boolean publico) {
        // TODO: Implementar com validações
    }

    // TODO: Sobrecarregar curtir(): void e curtir(int quantidade): void
    // Ambos devem usar um método privado para validar quantidade

    private void validarConteudo(String conteudo) {
        if (conteudo == null || conteudo.trim().isEmpty()) {
            throw new IllegalArgumentException("Conteúdo não pode ser vazio");
        }
    }
}
```

Pergunta: Complete a classe implementando: (1) Construtores que evitem duplicação usando *this*, (2) Métodos curtir sobrecarregados que compartilhem lógica de validação através de método privado, (3) Validações adequadas nos construtores.

Questão 3: Sistema de Matrícula - Gestão de Alunos

```
package edu.universidade.core;

public class Aluno {
    private String matricula;
    private String nome;
    private int idade;
    private String curso;

    // TODO: Criar 3 construtores sobrecarregados:
    // 1. Apenas matricula e nome
    // 2. Matricula, nome e idade
    // 3. Matricula, nome, idade e curso
    // Um deve chamar o outro para evitar duplicação

    public String getMatricula() { return matricula; }
    public String getNome() { return nome; }
    // TODO: Definir quais getters são realmente necessários

    // TODO: Criar método privado validarMatricula(String matricula)
}
```

Pergunta: Implemente os construtores encadeados e determine quais getters devem ser públicos. Justifique a escolha de visibilidade baseado no princípio de menor privilégio.