INF008 – Programação Orientada a Objetos

1ª. Lista de Exercícios

Questão 1: Sistema de Gerenciamento de Contatos

Implemente uma classe Contact com os seguintes requisitos:

- Atributos privados: name (String), phone (String), email (String)
- Dois construtores sobrecarregados: um que recebe apenas name e phone, outro que recebe todos os três atributos
- Métodos públicos getters e setters para todos os atributos
- Um método displayInfo() que retorna uma string formatada com todas as informações
- Um método updateContact(String phone, String email) que atualiza os dados

Desafio adicional: Crie uma classe AddressBook que gerencia um array de 10 objetos do tipo Contact e implemente métodos para:

- Adicionar contato (passando objeto Contact como parâmetro). Valide se há espaço para inserir um novo contato
- Buscar contato por nome
- Listar todos os contatos

Questão 2: Sistema de Caixa com Operações Financeiras

Implemente uma classe CashRegister que simula um caixa registrador:

Requisitos:

- Atributos privados: currentBalance (double), transactionCount (int), registerId (String)
- Construtores sobrecarregados (pelo menos 3 variações com diferentes combinações de parâmetros)
- Métodos públicos para operações:
    - processPayment(double amount) - processa um pagamento
    - processRefund(double amount) - processa um reembolso
    - getDailyReport() - retorna relatório do dia

Sobrecarga de métodos: Implemente versões sobrecarregadas dos métodos de pagamento que aceitam diferentes parâmetros (valor em double, valor em int, valor com descrição)

Questão 3: Sistema de Geometria com Sobrecarga

Implemente uma classe GeometryCalculator com métodos sobrecarregados:

1. **Calcular área:**
    - calculateArea(double radius) - círculo
    - calculateArea(double base, double height) - retângulo/triângulo
    - calculateArea(double side1, double side2, double side3) - triângulo pela fórmula de Heron
2. **Calcular perímetro:**
    - calculatePerimeter(double... sides) - método varargs para polígonos (pesquise sobre isso...)
3. **Construtores sobrecarregados:**
    - Construtor padrão
    - Construtor que recebe uma precisão (número de casas decimais)
    - Construtor que chama o outro usando this()

Desafio: Use o princípio DRY (Don't Repeat Yourself) evitando duplicação de código nos cálculos.

Questão 4: Gerenciamento de Estoque de Produtos

Implemente uma classe InventoryItem com:

- Atributos privados: itemCode, description, unitPrice, quantityInStock, minimumStockLevel
- Construtores sobrecarregados (pelo menos 3 variações)
- Validações nos setters (preço não pode ser negativo, estoque não pode ser negativo, etc.)
- Métodos de negócio:
    - applyDiscount(double percentage)
    - updateStock(int quantity) - com sobrecarga para diferentes cenários
    - isBelowMinimumStock() - verifica se está abaixo do estoque mínimo

Desafio: Implemente um construtor que chama outro construtor da mesma classe usando this() para inicializar valores padrão e eliminar duplicação.

Questão 5: Sistema de Configuração de Aplicação

Implemente uma classe AppConfig que gerencia configurações de aplicação:

Requisitos:

- Atributos privados: appName, version, maxConnections, timeoutSeconds, isDebugMode
- Múltiplos construtores sobrecarregados para diferentes cenários de inicialização
- Métodos para manipulação de configuração:
    - updateSettings(int maxConnections, int timeoutSeconds)
    - updateSettings(boolean isDebugMode) - versão sobrecarregada
    - validateConfig() - valida se as configurações são consistentes

Desafio principal: Implemente a invocação de construtores usando this() de modo que um construtor mais complexo chame construtores mais simples, eliminando duplicação de código de inicialização.