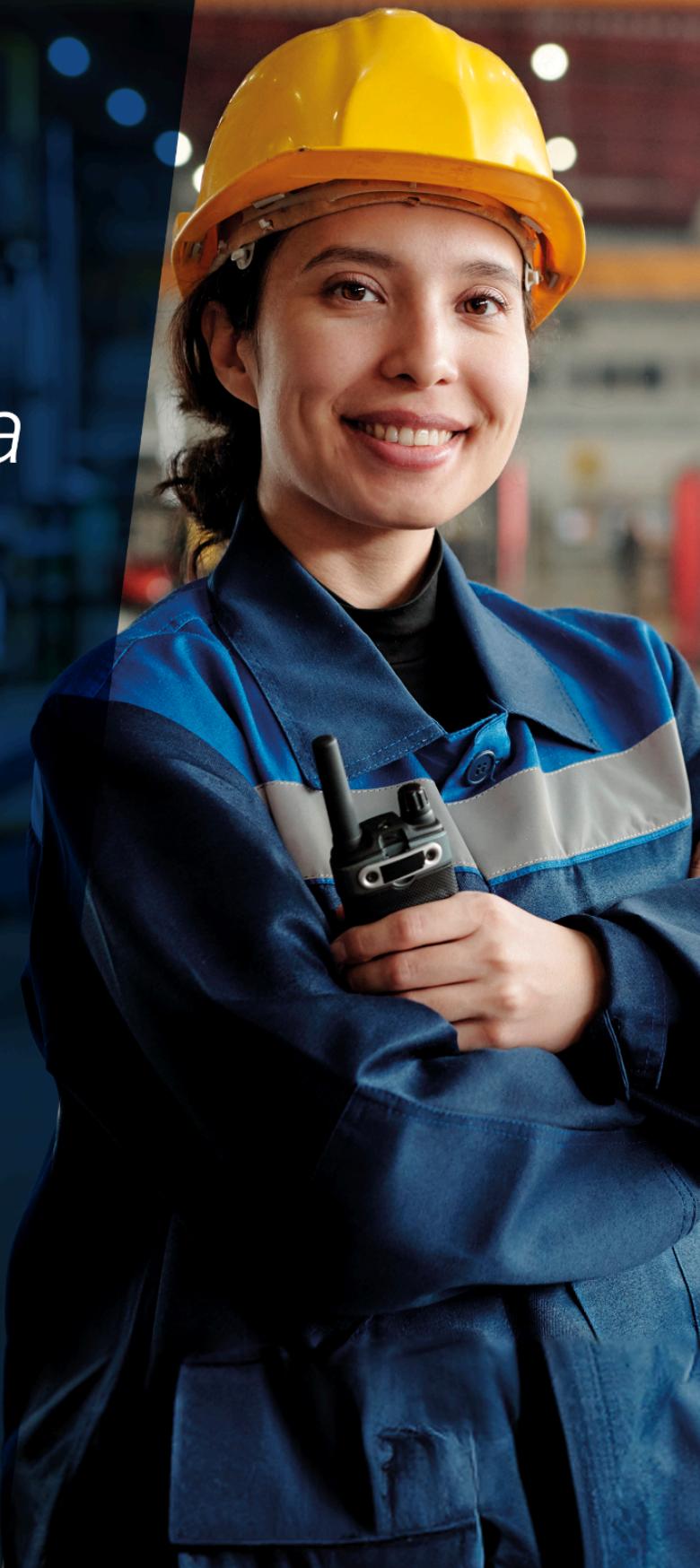


*Formando
quem forma
a indústria*



SENAI

› Educação
Profissional



Atividade 2: Refatoração para o Princípio Aberto-Fechado (OCP)

Unidade Curricular: Arquitetura de Sistemas

Professor: Lucas Santos

Cenário: Sistema de Cálculo de Descontos OCP

Contextualização

A empresa fictícia "TechStore" (a mesma do cenário anterior) precisa que seu sistema de gerenciamento de **descontos** seja robusto. O time de marketing lança promoções frequentemente, exigindo a introdução constante de novos tipos de desconto (ex: Desconto de Aniversário, Desconto por Volume, Desconto de Black Friday).

O Problema: Violação do OCP

O código atual (na classe CalculadoraDeDesconto) lida com diferentes tipos de desconto usando condicionais (if/else ou switch) com base em um enum ou um tipo. O problema é que, toda vez que um novo tipo de desconto é adicionado, o desenvolvedor é obrigado a modificar a classe CalculadoraDeDesconto para incluir a nova regra.

Essa necessidade de modificar o código central para estender a funcionalidade viola o Princípio Aberto-Fechado (OCP), que afirma: uma entidade de software deve ser aberta para extensão, mas fechada para modificação.

Código de Partida (Violação do OCP)

A seguir, a estrutura de classes que será utilizada como ponto de partida:

```
package OCP_VIOLACAO;

// Classe Entidade
public class Pedido {
    private String id;
    private double valorBruto;
    private int quantidadeItens;
    private boolean isPago;
    private String emailCliente;
    private TipoDesconto tipoDesconto; // Tipo de desconto a ser aplicado

    public Pedido(String id, double valorBruto, int quantidadeItens, String emailCliente, TipoDesconto tipoDesconto) {
        this.id = id;
        this.valorBruto = valorBruto;
        this.quantidadeItens = quantidadeItens;
        this.isPago = false;
        this.emailCliente = emailCliente;
        this.tipoDesconto = tipoDesconto;
    }

    // --- GETTERS ---
    public String getId() { return id; }
    public double getValorBruto() { return valorBruto; }
    public int getQuantidadeItens() { return quantidadeItens; }
    public boolean isPago() { return isPago; }
    public String getEmailCliente() { return emailCliente; }
    public TipoDesconto getTipoDesconto() { return tipoDesconto; }

    // --- SETTERS ---
    public void setValorBruto(double valorBruto) { this.valorBruto = valorBruto; }
    public void setQuantidadeItens(int quantidadeItens) { this.quantidadeItens = quantidadeItens; }
    public void setPago(boolean pago) { isPago = pago; }
    public void setTipoDesconto(TipoDesconto tipoDesconto) { this.tipoDesconto = tipoDesconto; }

    // Enum para Tipos de Desconto
    enum TipoDesconto {
        CUPOM, VIP, SAZONAL, ANIVERSARIO // ANIVERSARIO é o novo desconto que causará
        a modificação
    }

    // CLASSE QUE VIOLA O OCP
    public class CalculadoraDeDesconto {

        public double aplicarDesconto(Pedido pedido) {
            double valor = pedido.getValorBruto();
```

```
TipoDesconto tipo = pedido.getTipoDesconto();

if (tipo == TipoDesconto.CUPOM) {
    // Lógica para CUPOM: 10%
    return valor * 0.90;
} else if (tipo == TipoDesconto.VIP) {
    // Lógica para VIP: 15%
    return valor * 0.85;
} else if (tipo == TipoDesconto.SAZONAL) {
    // Lógica para Sazonal: 5%
    return valor * 0.95;
}
// AQUI ESTÁ A VIOLAÇÃO: SE O NOVO DESCONTO (ANIVERSARIO) FOSSE
ADICIONADO:
/* else if (tipo == TipoDesconto.ANIVERSARIO) {
    // Lógica para Aniversário: 20%
    return valor * 0.80;
} */
// ESSA MODIFICAÇÃO NO CÓDIGO EXISTENTE (A CLASSE CalculadoraDeDesconto)
VIOLA O OCP!

    return valor;
}
```

Seu Desafio: Aplicação do OCP (Estratégia)

Devem refatorar o código para que o sistema de descontos atenda ao Princípio Aberto-Fechado, utilizando o Padrão Strategy (Estratégia).

1. **Explique o Mau Design (OCP):** Descreva sucintamente por que a introdução de novos descontos obriga a quebrar o OCP no código de partida.
2. **Crie a Solução OCP (Refatoração):**
 - Defina uma interface (EstrategiaDeDesconto) que represente o contrato para a função de cálculo de desconto.
 - Crie quatro classes concretas que implementem essa interface (para CUPOM, VIP, SAZONAL e a nova regra ANIVERSARIO).
 - Refatore a classe CalculadoraDeDesconto para que ela receba a estratégia de desconto como argumento, eliminando completamente os blocos if/else internos.
3. **Demonstre a Extensão:** Mostre o código da nova classe DescontoAniversario e como a classe principal é utilizada para aplicar essa nova regra, provando que o sistema é aberto para extensão (nova classe) e fechado para modificação (a classe CalculadoraDeDesconto não muda).