Centro Universitário de Excelência
Sistemas de Informação

Implementação do Padrão de Projeto Facade no Sistema TiaLuDelivery

Autores: Maciel Lopes Francisco
Aeverton Santos de Oliveira
Carlos Henrique Santos de Carvalho
Daniel Barbosa dos Reis
Allan Brito Barreto

Itabuna, 2025

Agenda

O objetivo desta apresentação é:
Demonstrar a implementação e os benefícios do padrão de projeto Facade para simplificar a interação com os subsistemas do projeto TiaLuDelivery.

1. Introdução:
O contexto do projeto TiaLuDelivery

2. O problema:
A complexidade de um sistema com múltiplos componentes

3. A solução:
O que é o Padrão Facade?

4. Nossa Implementação**:**
Apresentação do código `SistemaPedidosFacade`

5. Resultados e conclusão:
O impacto da nossa solução

Introdução ao Projeto

- O projeto **TiaLuDelivery** visa refatorar um sistema de delivery, aplicando boas práticas e padrões de projeto.
- Nossa tarefa (Equipe Assunção) foi criar um ponto de acesso único e simplificado para as operações principais do sistema.
- O objetivo era **desacoplar** a camada de interface (CLI) das regras de negócio internas.

O Problema: Complexidade

- Tópicos a serem citados :

- Sem um ponto central de acesso, a camada de interface (CLI) precisaria conhecer e interagir com **várias classes** do sistema: `Cliente`, `Pedido`, `Cardapio`, `Pagamento`, etc.
    - Isso gera um **alto acoplamento**, dificultando a manutenção e a evolução do código.
    - Qualquer mudança em uma classe do subsistema poderia quebrar a interface do usuário.

A Solução: O Padrão Facade

- Como o Facade resolve:
  - O Facade (ou "Fachada") fornece uma **interface única e simplificada** para um conjunto de classes de um subsistema.
  - Ele funciona como um "porteiro" ou "atendente": o cliente faz um pedido simples, e o Facade se encarrega de realizar todas as operações complexas internamente.
  - **Benefícios:** Reduz o acoplamento, esconde a complexidade e torna o sistema mais fácil de usar.

Nossa Implementação

- Solução:

- Criamos a classe `SistemaPedidosFacade.java`.
- Ela expõe métodos simples e diretos para as operações principais do sistema.
- O código está preparado para se integrar com os padrões das outras equipes (Builder, State, etc.).

```
    // Registrar novo cliente
    public Customer registrarNovoCliente(String nome, String telefone) {
        // ... lógica complexa escondida aqui ...
    }

    // Criar novo pedido
    public Order criarPedido(Customer cliente, List<OrderItem> itens) {
        // ... lógica complexa escondida aqui ...
    }

    // Listar cardápio
    public List<MenuItem> listarCardapio() {
        // ... lógica complexa escondida aqui ...
    }
}
```

Resultados

- A classe `SistemaPedidosFacade` foi integrada com sucesso ao projeto principal.
- A camada de interface (CLI) agora se comunica **apenas** com a nossa fachada, simplificando o código e reduzindo o acoplamento.
- O teste de integração inicial confirmou que a fachada está acessível e pronta para uso.
- **Saída do Teste:** `[OK] Facade instanciado e acessível (método ainda não implementado).`

Considerações finais

- O padrão Facade se mostrou a escolha ideal para atender aos requisitos do projeto.
- Nossa implementação cria uma **barreira de proteção** que esconde a complexidade interna do sistema.
- O trabalho da Equipe Assunção serve como uma **base sólida** para a integração dos demais padrões de projeto, garantindo um sistema mais organizado e de fácil manutenção.

Referências

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Documentação da OAT 1.2 - Refatorando Código com Padrões de Projeto.
- Código-fonte do repositório TiaLuDelivery (GitHub da disciplina)