Centro Universitário de Excelência
Sistemas de Informação

IMPLEMENTAÇAO DAS CLASSES, OBJETOS E RELACIONAMENTOS EM JAVA

Autores:
<Elisson Bastos Oliveira Menezes Junior>
<André Valter Leite>
<Ian da Silva Borges>
<Victor Tadeu Araújo Augusto?
<Joao Felipe Santos Costa>

Itabuna, 2025

Agenda

Objetivo da apresentação:Compartilhar a nossa experiencia e apredizado com esse projeto

1. Item 1: Introdução

2. Item: Fundamentação teórica

3. Item: Metodologia.

4. Item: Resultados

Introdução

O projeto serviu como um experimento prático para a implementação dos pilares da Programação Orientada a Objetos (POO) em Java:

O foco principal foi a modelagem de:

Classes ,Objetos e Relacionamentos

Solidificando a compreensão teórica através do desenvolvimento de uma aplicação concreta, simulando um sistema de delivery.

Fundamentação teórica

1. Conceito Central: Classe e Objeto
Classe: abstração que define um modelo (blueprint) para criação de objetos.

Objeto: instância concreta de uma classe, com:

Estado: atributos (variáveis).

Comportamento: métodos (funções).

2. Construtor
Método especial invocado no momento da instanciação.

Garante que o objeto seja criado em um estado inicial válido.

3. Princípios Fundamentais da POO
Encapsulamento:
Agrupa dados e comportamentos em uma unidade coesa (classe), controlando o acesso a eles.

Herança:
Permite que classes herdem atributos e métodos de outras, promovendo reuso e hierarquia.

Polimorfismo:
Objetos de classes diferentes podem responder à mesma mensagem de formas distintas.

Fundamentação teórica PT.2

4. UML (Unified Modeling Language)
Linguagem de modelagem para documentar e visualizar a arquitetura de software.

Representa graficamente:

Classes, atributos, métodos.

Relacionamentos entre classes (herança, associação, etc.).

5. Resumo: POO em Prática
A POO organiza o código em entidades (objetos) que combinam dados e comportamentos, seguindo princípios como encapsulamento, herança e polimorfismo, com suporte de ferramentas como UML para modelagem.

Objetos de classes diferentes podem responder à mesma mensagem de formas distintas.

Metodologia

O projeto adotou uma metodologia de desenvolvimento orientada a objetos, iniciando pela modelagem do domínio através de um diagrama de classes UML que definiu as entidades principais (Cliente, ItemCardapio, Pedido) e a classe associativa ItemPedido para resolver relações muitos-para-muitos.

A persistência em memória foi garantida pela classe BancoDeDados, seguindo o padrão Singleton para uma única instância global. A lógica de negócio foi centralizada na camada de serviço através do padrão Facade (FoodDeliveryFacade), oferecendo uma interface unificada para operações do sistema.

A interface com o usuário foi implementada via linha de comando (AplicacaoCLI), enquanto a classe Main coordenou a inicialização e fluxo da aplicação.

Esta abordagem em camadas (domínio, persistência, serviço e apresentação) promoveu separação de preocupações, resultando em baixo acoplamento, alta coesão, facilidade de manutenção e preparação para evoluções futuras.

Resultados

O sistema FoodDelivery foi implementado com sucesso, demonstrando capacidade para:

Gerenciar cadastros de clientes e itens do cardápio de forma eficiente.

Processar pedidos completos, incluindo associação de itens e quantidades.

Gerar relatórios básicos sobre pedidos e operações do sistema.

Manter integridade dos dados através de persistência em memória com Singleton.

Oferecer uma interface intuitiva via linha de comando (CLI) para interação com o usuário.

Conclusão

O projeto demonstrou com sucesso a aplicação prática dos conceitos de programação orientada a objetos em Java, implementando uma arquitetura escalável com padrões como Singleton e Facade. A separação clara entre domínio, persistência, serviço e apresentação resultou em um sistema coeso, de fácil manutenção e preparado para futuras evoluções, validando a eficácia da modelagem UML e dos princípios de POO.