Centro Universitário de Excelência
Sistemas de Informação

Food Delivery

Autores: Guilherme Couto Almeida e Almeida

Itabuna, 2025

Agenda

Objetivo da apresentação

1. Introdução : Introdução à programação orientada a objetos com java.

2. Fundamentação teórica: Fundamentos de POO.

3. Metodologia: Diagrama de classes.

4. Classes: Classes, seus objetos e métodos.

5. Referências.

Introdução

- POO(Programação orientada a objetos) é um **paradigma de programação** que organiza o software em **objetos**, que representam entidades do mundo real. Cada objeto combina **dados** (atributos) e **comportamentos** (métodos).

- **Objetivos principais da POO**

Modelar o mundo real → facilitar o entendimento do sistema representando coisas como objetos.
Reuso de código → com herança e polimorfismo, reaproveitar implementações.
Modularidade → dividir o sistema em partes independentes (classes/objetos).
Manutenção facilitada → mudanças ficam mais localizadas.
Maior qualidade e produtividade → código mais organizado, claro e reutilizável.

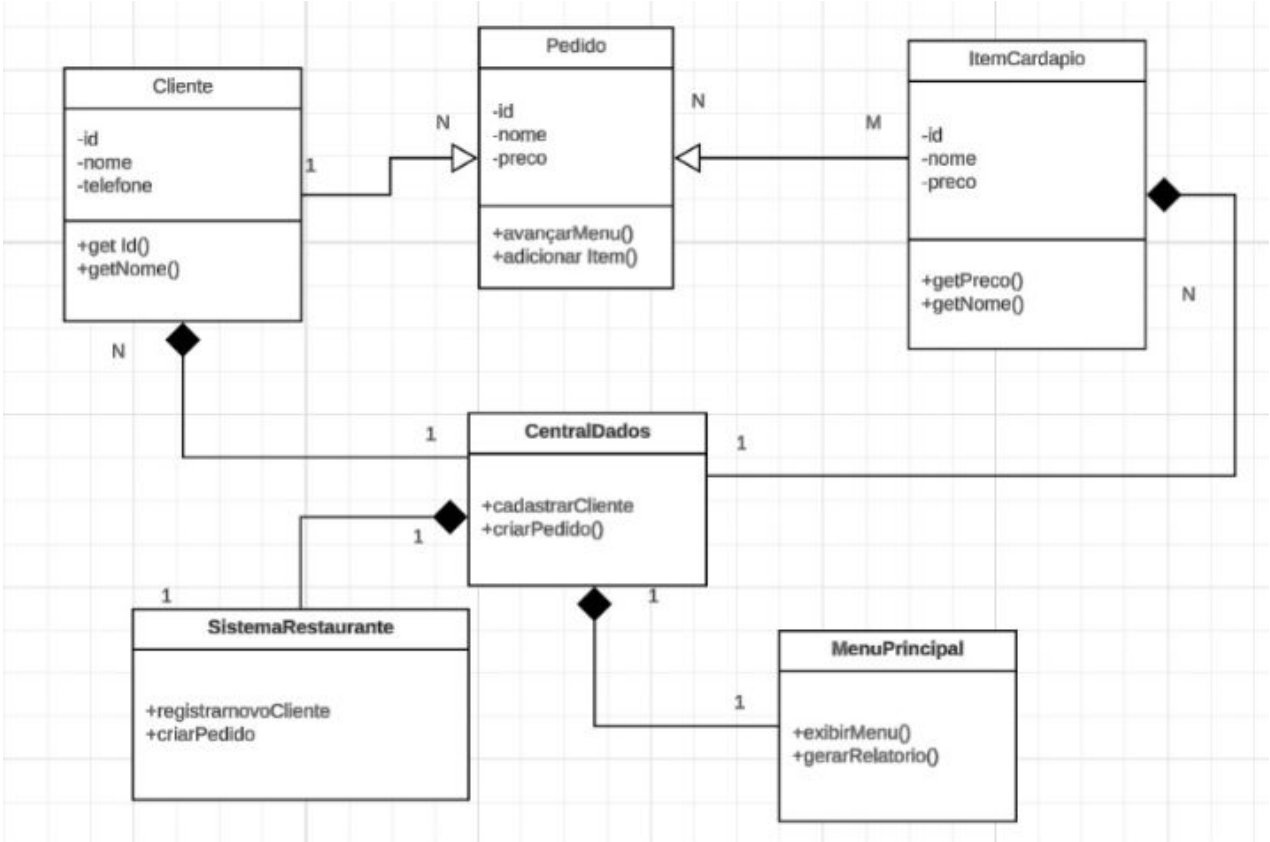
Fundamentação

Conceitos básicos da POO:

- **Classes:** Modelos que definem objetos.
- **Atributos:** Características ou dados do objeto.
- **Métodos:** Ações ou comportamentos do objeto.
- **Objetos:** Instâncias criadas a partir das classes.

A POO facilita a organização do código, o reuso e a manutenção dos programas.

Diagrama de classe



Classe Menu Principal

```
=== MENU PRINCIPAL ===
1. Cadastrar Cliente
2. Cadastrar Item do Cardápio
3. Listar Clientes
4. Listar Itens do Cardápio
5. Criar Pedido
6. Avançar Status de Pedido
7. Consultar Pedidos por Status
8. Gerar Relatório Simplificado
9. Gerar Relatório Detalhado
0. Sair
Escolha uma opção: 1
```

- A classe `MenuPrincipal` é a **interface de linha de comando** do sistema de restaurante.
- Ela organiza e exibe o menu de opções, captura entradas do usuário e delega as operações para a camada de serviço (`SistemaRestaurante`) e de relatórios.

Classe Menu Principal

O que ela faz:

Atributos principais

- `sistema` → instância de `SistemaRestaurante` usada para executar as operações de negócio.
- `scanner` → leitor de entrada do console para capturar as escolhas e dados do usuário.

Construtor

- Inicializa os componentes de interação (`SistemaRestaurante` e `Scanner`) para permitir a navegação e execução das funcionalidades via console.

Métodos auxiliares (privados)

- `listarPedidosNaoEntregues()` → retorna apenas os pedidos cujo `status` ainda não é `ENTREGUE`, auxiliando a tela de avanço de status.

Classe Menu Principal

O que ela faz:

Métodos principais (fluxo do menu)

- `exibirMenu()` → laço principal que mostra as opções, lê a escolha e direciona para o método correspondente.
- `cadastrarCliente()` → solicita nome/telefone, registra o cliente e exibe o ID gerado.
- `listarClientes()` → imprime a lista de clientes cadastrados.
- `cadastrarItem()` → solicita nome/preço e registra um novo item no cardápio.
- `listarItens()` → imprime os itens disponíveis do cardápio.
- `criarPedido()` → escolhe cliente, adiciona itens/quantidades e cria o pedido, mostrando o número gerado.
- `avancarStatus()` → lista pedidos não entregues, recebe o número e avança para o próximo status.
- `consultarPorStatus()` → recebe um status e lista os pedidos filtrados com total e cliente.
- `gerarRelatorio(RelatorioVendas)` → delega a geração do relatório (simplificado ou detalhado) com base na lista de pedidos.

Classe Cliente

- A classe `Cliente` representa a **entidade cliente** dentro do sistema de restaurante.
- Ela armazena informações básicas de cada cliente de forma imutável, servindo como modelo para cadastro e consultas.

O que ela faz:

Atributos principais

- `id` → identificador único do cliente.
- `nome` → nome completo do cliente.
- `telefone` → contato telefônico.

Construtor

- Permite criar um cliente informando `id`, `nome` e `telefone`.

Métodos de acesso (getters)

- `getId()` → retorna o identificador.
- `getNome()` → retorna o nome.
- `getTelefone()` → retorna o telefone.

Classe ItemCardapio

- A classe `ItemCardapio` representa a **entidade item do cardápio** dentro do sistema de restaurante.
- Ela armazena informações básicas de cada produto de forma imutável, servindo como modelo para consultas e criação de pedidos.

O que ela faz:

Atributos principais

- `id` → identificador único do item.
- `nome` → nome do prato ou produto.
- `preco` → valor monetário do item.

Construtor

- Permite criar um item informando `id`, `nome` e `preco`.

Métodos de acesso (getters)

- `getId()` → retorna o identificador.
- `getNome()` → retorna o nome.
- `getPreco()` → retorna o preço.

Classe ItemPedido

- A classe `ItemPedido` representa a **associação entre um item do cardápio e a quantidade escolhida** dentro de um pedido.
- Ela serve como base para calcular o valor parcial de cada item, compondo o total do pedido.

O que ela faz:

Atributos principais

- `item` → referência ao objeto `ItemCardapio` escolhido.
- `quantidade` → número de unidades do item selecionado.

Construtor

- Permite criar um item de pedido informando `item` e `quantidade`.

Métodos de acesso e calculo

- `getItem()` → retorna o item do cardápio associado.
- `getQuantidade()` → retorna a quantidade escolhida.
- `getSubtotal()` → calcula o valor total do item multiplicando preço × quantidade.

Classe Pedido

- A classe `Pedido` representa a **entidade pedido** dentro do sistema de restaurante.
- Ela organiza os itens escolhidos por um cliente, armazena o valor total, controla o status do pedido e registra sua data e hora de criação.

O que ela faz:

Atributos principais

- `numero` → identificador numérico do pedido.
- `cliente` → referência ao cliente que fez o pedido.
- `itens` → lista de itens (`ItemPedido`) que compõem o pedido.
- `valorTotal` → valor acumulado do pedido.
- `status` → estado atual do pedido (ex.: ACEITO, PREPARANDO, ENTREGUE).
- `dataHora` → momento em que o pedido foi aceito.
- `listeners` → lista de ouvintes para monitorar mudanças de status.

Construtor

- Permite criar um pedido informando `numero` e `cliente`.
- Inicializa a lista de itens e define o status e data/hora como nulos.

Classe Pedido

O que ela faz:

Métodos de manipulação

- `adicionarItem(ItemCardapio, quantidade)` → adiciona um item ao pedido e atualiza o valor total.
- `aceitarPedido()` → define o status como **ACEITO**, registra a data/hora e notifica os ouvintes.
- `avancarStatus()` → avança o pedido para o próximo status (de ACEITO até ENTREGUE).
- `adicionarListener(PedidoListener)` → inscreve ouvintes para acompanhar mudanças.
- `notificarListeners()` → avisa os ouvintes quando o status muda.

Métodos de acesso (getters)

- `getNumero()` → retorna o número do pedido.
- `getCliente()` → retorna o cliente associado.
- `getItens()` → retorna a lista de itens.
- `getValorTotal()` → retorna o valor acumulado.
- `getStatus()` → retorna o status atual.
- `getDataHora()` → retorna a data/hora de aceitação.

Interface PedidoListener

- A interface `PedidoListener` define um **contrato para monitorar mudanças no status de um pedido**.
- Ela é usada para implementar o padrão *Observer*, permitindo que outras classes sejam notificadas automaticamente quando o estado de um pedido for alterado.

O que ela faz:

Atributo/Comportamento principal

- Não possui atributos próprios, apenas o método que deve ser implementado.

Método definido

- `onStatusAlterado(Pedido pedido)` → é chamado sempre que o status de um pedido muda, recebendo como parâmetro o objeto `Pedido` atualizado.

Enumeração StatusPedido

- A enumeração `StatusPedido` representa os **estágios de um pedido** dentro do sistema de restaurante.
- Ela define um ciclo de vida que começa na aceitação do pedido e termina na sua entrega ao cliente.

O que ela faz:

Constantes definidas

- `ACEITO` → pedido confirmado pelo sistema.
- `PREPARANDO` → pedido está sendo preparado na cozinha.
- `FEITO` → preparo concluído, aguardando próximo passo.
- `AGUARDANDO_ENTREGADOR` → pedido pronto, esperando ser retirado para entrega.
- `SAIU_PARA_ENTREGA` → pedido saiu para entrega ao cliente.
- `ENTREGUE` → pedido foi entregue ao cliente.

Classe RelatorioDetalhado

- A classe `RelatorioDetalhado` gera um **relatório completo de vendas** no sistema de restaurante.
- Ela exibe cada pedido individualmente, mostrando dados do cliente, itens comprados, valores parciais, valor total e o status do pedido.

O que ela faz:

Atributos principais

- Não possui atributos próprios, apenas implementa o comportamento definido pela interface `RelatorioVendas`.

Método principal

- `gerar(List<Pedido> pedidos)` → percorre todos os pedidos recebidos e imprime um relatório detalhado com:
  - Número do pedido.
  - Nome do cliente.
  - Lista de itens com quantidade e subtotal.
  - Valor total do pedido.
  - Status atual do pedido.

Classe RelatorioSimplificado

- A classe `RelatorioSimplificado` gera um **resumo geral das vendas** no sistema de restaurante.
- Ela apresenta apenas informações consolidadas, sem detalhar cada pedido individualmente.

Atributos principais

- Não possui atributos próprios, apenas implementa o comportamento da interface `RelatorioVendas`.

O que ela faz:

Método principal

- `gerar(List<Pedido> pedidos)` → recebe a lista de pedidos e exibe:
  - Quantidade total de pedidos.
  - Valor total arrecadado (soma de todos os pedidos).

Interface RelatorioVendas

- A interface `RelatorioVendas` define o **contrato para geração de relatórios** no sistema de restaurante.
- Ela estabelece um método que deve ser implementado por diferentes tipos de relatórios, garantindo flexibilidade e padronização.

O que ela faz:

Método definido

- `gerar(List<Pedido> pedidos)` → recebe uma lista de pedidos e deve produzir um relatório de vendas com base nesses dados.

Classe CentralDados

- A classe `CentralDados` funciona como um **repositório central de informações** no sistema de restaurante.
- Ela utiliza o padrão **Singleton**, garantindo que exista apenas uma instância responsável por armazenar clientes, itens do cardápio e pedidos.

O que ela faz:

Atributos principais

- `instance` → instância única da classe (`Singleton`).
- `clientes` → lista de todos os clientes cadastrados.
- `itensCardapio` → lista dos itens do cardápio disponíveis.
- `pedidos` → lista de todos os pedidos criados.
- `nextClienteId`, `nextItemId`, `nextPedidoId` → contadores automáticos para gerar identificadores únicos.

Construtor

- É privado (`private CentralDados()`), impedindo a criação direta de objetos.
- A instância é acessada pelo método estático `getInstance()`.

Classe CentralDados

O que ela faz:

Métodos de acesso (getters)

- `getClientes()` → retorna a lista de clientes.
- `getItensCardapio()` → retorna a lista de itens do cardápio.
- `getPedidos()` → retorna a lista de pedidos.

Métodos principais

- `cadastrarCliente(String nome, String telefone)` → cria e adiciona um novo cliente.
- `cadastrarItem(String nome, double preco)` → cria e adiciona um novo item ao cardápio.
- `criarPedido(int clienteId)` → cria um pedido associado a um cliente existente.
- `buscarClientePorId(int id)` → retorna um cliente pelo identificador.
- `buscarItemPorId(int id)` → retorna um item pelo identificador.
- `buscarPedidoPorId(int id)` → retorna um pedido pelo identificador.

Classe SistemaRestaurante

- A classe **`SistemaRestaurante`** atua como a **camada de serviços do sistema**, intermediando a interação entre o menu principal e os dados armazenados na `CentralDados`.
- Ela fornece métodos para cadastrar clientes e itens, criar pedidos, gerenciar status e consultar informações.

O que ela faz:

Atributos principais

- `dados` → instância única de `CentralDados`, utilizada para acessar e manipular os dados do sistema.

Métodos de cadastro e criação

- `registrarNovoCliente(String nome, String telefone)` → cadastra um cliente.
- `registrarNovoItem(String nome, double preco)` → cadastra um novo item do cardápio.
- `criarPedido(int clienteId)` → cria um pedido associado a um cliente.

Classe SistemaRestaurante

O que ela faz:

Métodos de manipulação

- `adicionarItemAoPedido(int pedidoId, int itemId, int quantidade)` → adiciona um item em um pedido existente.
- `avancarStatusPedido(int pedidoId)` → avança o status de um pedido para a próxima etapa.

Métodos de consulta

- `consultarPedidosPorStatus(StatusPedido status)` → retorna pedidos filtrados por status.
- `listarClientes()` → retorna todos os clientes cadastrados.
- `listarItensCardapio()` → retorna todos os itens do cardápio.
- `listarTodosPedidos()` → retorna todos os pedidos cadastrados.

Referências

- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. The Unified Modeling Language User Guide. 2. ed. Boston: Addison-Wesley, 2005.
- GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994.
- LARMAN, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. 3. ed. New Jersey: Prentice Hall, 2004.
- PRESSMAN, R. S. Engenharia de Software: Uma Abordagem Profissional. 8. ed. Porto Alegre: AMGH, 2019.
- SOMMERVILLE, I. Engenharia de Software. 10. ed. São Paulo: Pearson, 2018.