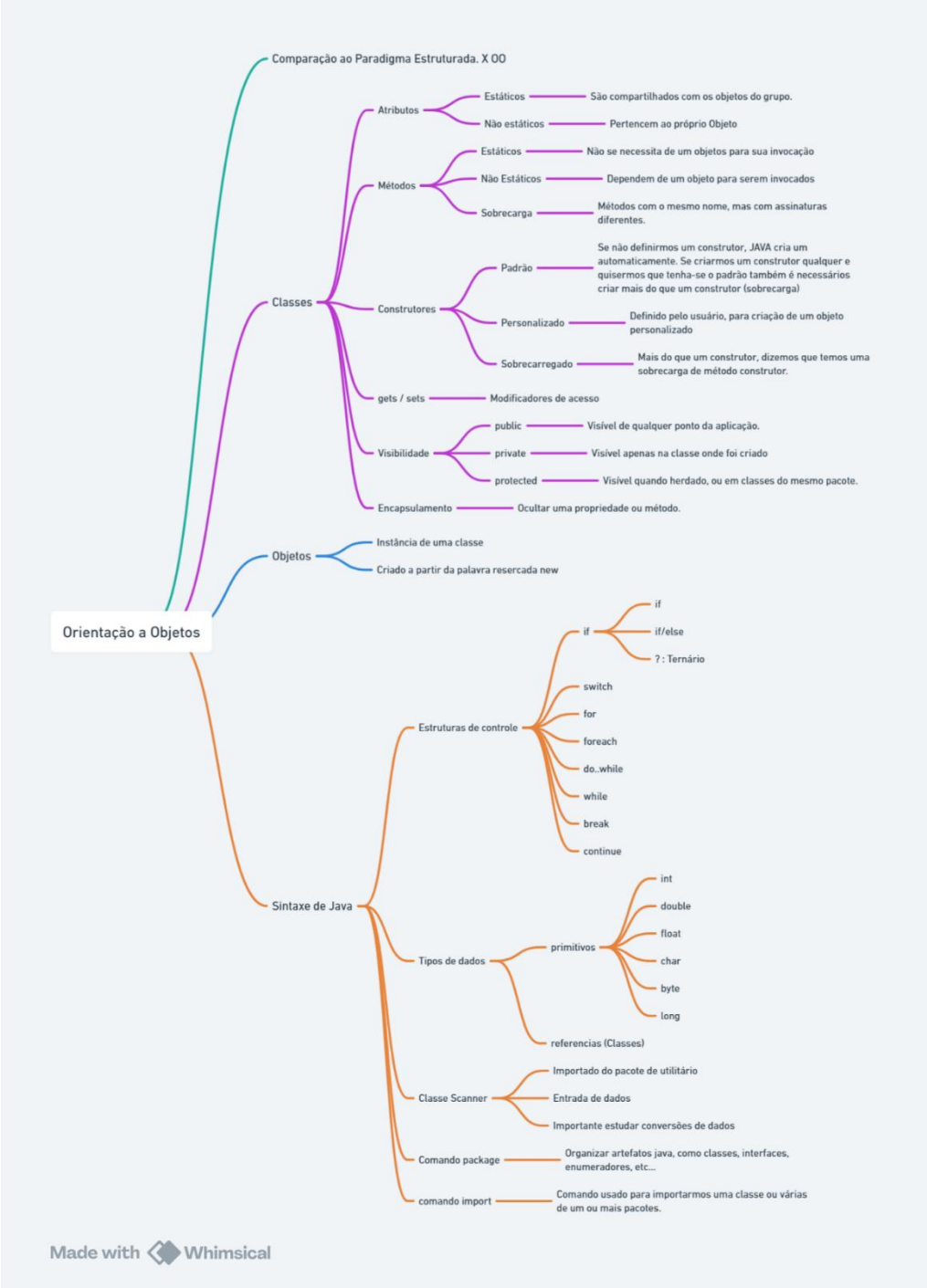
Programação Orientada à Objetos

Prof. Maromo

Tópicos a estudar:



Material 01 ao Material 05 (Conteúdo para estudo)

Exercícios de revisão para prática de POO.

Exercício 1 – Projeto de Classe com Atributos e Métodos

Uma empresa deseja desenvolver um sistema para gerenciar seus funcionários. Cada funcionário possui nome, matrícula e salário. A empresa quer calcular o salário líquido aplicando um desconto fixo de 10%.

Solicitação:

Implemente a classe Funcionario, contendo os atributos necessários e um método calcularSalarioLiquido(). Explique o papel dos atributos e métodos definidos, e descreva como o sistema deve instanciar objetos dessa classe para representar diferentes funcionários.

Exercício 2 – Utilização de Atributo Estático

Uma fábrica de carros deseja acompanhar quantos carros já foram produzidos. Para isso, cada vez que um novo carro for criado, um contador global deve ser incrementado.

Solicitação:

Crie uma classe Carro com um atributo estático quantidadeProduzida. Explique como esse atributo é manipulado dentro da classe para contar os carros produzidos. Diferencie o atributo estático dos demais atributos comuns e justifique seu uso neste contexto.

Exercício 3 – Métodos Estáticos vs. Métodos de Instância

Considere uma classe Conversor com dois métodos:

Um método estático converterCelsiusParaFahrenheit(double celsius)

Um método não estático exibirConversao(double celsius) que utiliza o método estático internamente para imprimir o resultado da conversão.

Solicitação:

Descreva o papel de cada método e porque um deles é estático e o outro não. Em que momento cada um deles deve ser chamado e como isso se relaciona com a instanciação de objetos?

Exercício 4 – Sobrecarga de Métodos em Situação Real

Você está desenvolvendo um sistema bancário e precisa implementar uma classe **Pagamento** com um método **pagar**. Este método pode aceitar diferentes formas de pagamento:

1. Um pagamento com valor em dinheiro.
2. Um pagamento com valor e código do cartão.
3. Um pagamento com valor, cartão e número de parcelas.

Solicitação:

Implemente a ideia de sobrecarga de métodos na classe Pagamento. Explique a vantagem da sobrecarga neste caso e como a linguagem Java trata múltiplos métodos com o mesmo nome.

Exercício 5 – Análise de Instanciação e Reutilização

Em um sistema de controle acadêmico, a classe Aluno foi definida com os atributos **nome** e **curso**. No entanto, o método que imprime os dados do aluno (**imprimirDados**) está sendo chamado diretamente sem instanciar o objeto.

Solicitação:

Explique por que isso causa erro de compilação. Em seguida, refaça o trecho de código para mostrar a forma correta de instanciação de um objeto Aluno e uso do método imprimirDados. Comente sobre a diferença entre métodos estáticos e de instância no contexto deste problema.

Exercício 6 – Encapsulamento e Proteção dos Dados

Considere a classe abaixo:

```
public class Produto {
    public String nome;
    public double preco;
}
```

Um aluno utilizou essa classe para criar objetos em outra parte do programa e modificou diretamente o preço do produto, atribuindo um valor negativo.

Solicitação:

a) Explique por que o código acima viola o princípio do encapsulamento e quais riscos essa violação pode trazer para a integridade dos dados da aplicação.

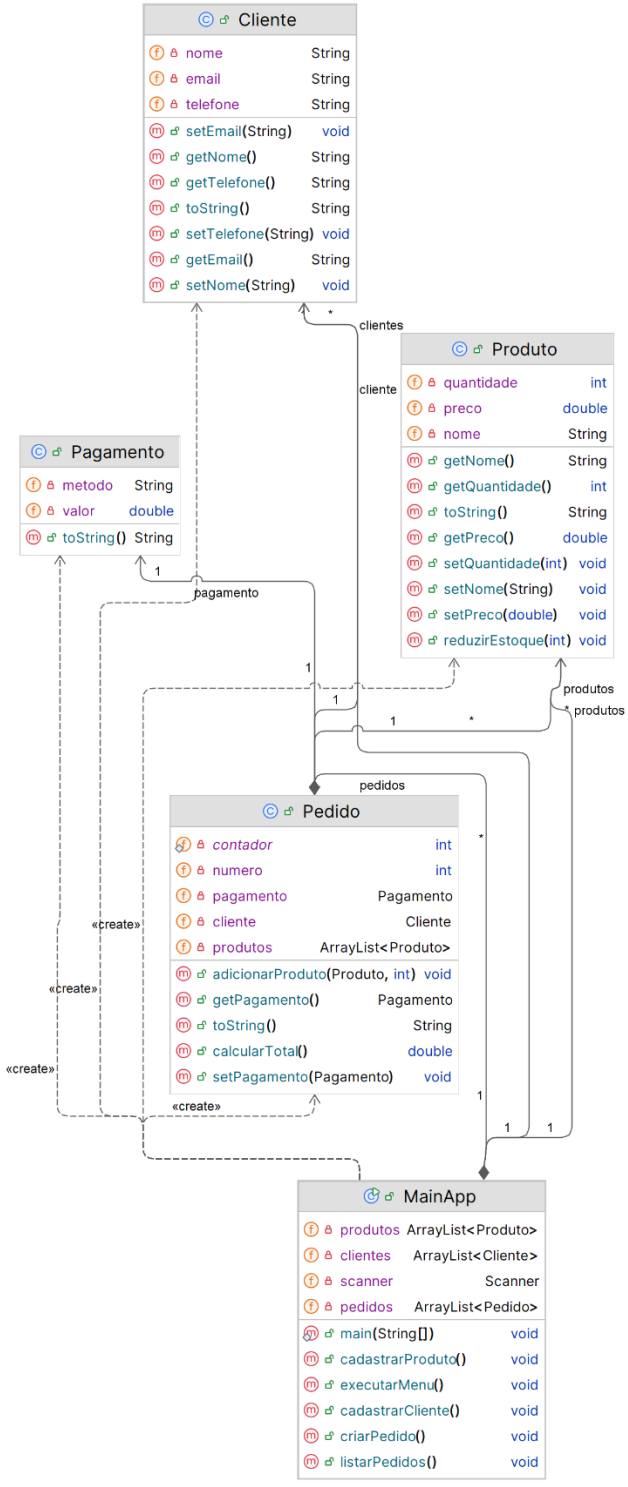
b) Reescreva a classe Produto aplicando corretamente o encapsulamento, utilizando modificadores de acesso e métodos get e set.

c) No método setPreco, implemente uma validação que impeça a definição de preços negativos. Explique como essa abordagem protege a lógica da aplicação.

Atividade Prática: Criando o ProjetoPedidos na IntelliJ IDEA Ultimate

Objetivo

Desenvolver um sistema orientado a objetos chamado **ProjetoPedidos**, com foco em modelagem, encapsulamento, composição, uso de listas, métodos e atributos estáticos. Conforme modelagem e descrição abaixo:



Etapa 1 – Criar o Projeto no IntelliJ

1) Abra a **IDE IntelliJ IDEA Ultimate**.
2) Vá em **File > New > Project**.
3) Selecione **Java**.
4) Nomeie o projeto como: ProjetoPedidos.
5) Finalize clicando em **Finish**.

Etapa 2 – Estrutura de Pacotes

1) No painel lateral, clique com o botão direito sobre src.
2) Crie os seguintes pacotes:
   - model ⟶ conterá as classes do sistema.
   - app ⟶ conterá a classe principal **MainApp**.

Etapa 3 – Modelagem das Classes

Classe Cliente (pacote: model)

Atributos:

- nome (privado)
- email (privado)
- telefone (privado)

Métodos:

- **Construtor:** recebe nome, e-mail e telefone.
- **Getters e Setters:** para todos os atributos.
- **toString()**: retorna uma representação legível do cliente.

Classe Produto (pacote: model)

Atributos:

- nome (privado)
- preco (privado)
- quantidade (privado)

Métodos:

- **Construtor:** recebe nome, preço e quantidade.
- **Getters e Setters:** para os três atributos.
- **reduzirEstoque(int qtd)**: reduz a quantidade em estoque.
- **toString()**: retorna nome, preço e quantidade disponíveis.

Classe Pagamento (pacote: model)

Atributos:

- valor (privado)
- metodo (privado)

Métodos:

- **Construtor:** recebe valor e método de pagamento (ex: "PIX", "Cartão").
- **Getters:** para os atributos, se necessário.
- **toString()**: exibe as informações do pagamento.

Classe Pedido (pacote: model)

Atributos:

- static contador: usado para gerar um número único de pedido.
- numero (privado): número do pedido.
- cliente (privado): referência a um objeto da classe Cliente.
- produtos (lista): lista de produtos comprados.
- pagamento (privado): objeto da classe Pagamento.

Métodos:

- **Construtor:** recebe um cliente e inicializa o número do pedido com base no contador.
- **adicionarProduto(Produto produto, int quantidade)**: adiciona o produto à lista, verificando estoque.
- **calcularTotal()**: soma o valor total dos produtos.
- **setPagamento(Pagamento)**: associa um pagamento ao pedido.
- **getPagamento()**: retorna o objeto Pagamento.
- **toString()**: exibe resumo do pedido (número, cliente, valor total).

Etapa 4 – Classe MainApp (pacote: app)

Responsável por executar e orquestrar o sistema.

Atributos:

- clientes: lista de clientes cadastrados.
- produtos: lista de produtos disponíveis.
- pedidos: lista de pedidos realizados.
- scanner: para entrada de dados no console.

Métodos:

- **main(String[] args)**: ponto de entrada da aplicação.
- **executarMenu()**: exibe o menu com opções e executa conforme a escolha.
- **cadastrarCliente()**: solicita dados e adiciona cliente à lista.
- **cadastrarProduto()**: solicita dados e adiciona produto à lista.

- **criarPedido()**: seleciona um cliente, escolhe produtos, define pagamento e registra o pedido.
- **listarPedidos()**: exibe todos os pedidos registrados.

Etapa 5 – Execução e Testes

1) Implemente cada classe conforme a modelagem.
2) Na classe **MainApp**, crie um menu interativo com opções como:
    1. Cadastrar cliente
    2. Cadastrar produto
    3. Criar pedido
    4. Listar pedidos
    5. Sair
3) Execute o programa e teste os fluxos.

Conceitos Aplicados no Projeto

- Definição e uso de **classes e objetos**
- Aplicação de **encapsulamento**
- **Composição** entre classes (Pedido possui Cliente, Produto, Pagamento)
- Uso de **listas (ArrayList)** para agrupar dados
- Implementação de **atributos estáticos**
- Criação de **métodos com responsabilidade clara**
- Construção de **interfaces de entrada com o usuário (Scanner)**
- Princípios de **reutilização e organização de código**