📘 Apostila – Projeto de Sistemas Orientado a Objetos (OO)

---

1. Introdução

O **projeto de sistemas orientado a objetos (OO)** organiza o software em torno de **classes** e **objetos**, em vez de funções e procedimentos. Essa abordagem promove maior **modularidade**, **reuso de código** e **flexibilidade** para manutenção e evolução do sistema.

Processo de desenvolvimento de sistemas OO: 1. **Análise de sistemas OO** → levantamento e análise de requisitos.
2. **Projeto de sistemas OO** → foco na arquitetura e organização do sistema.
3. **Implementação, testes e implantação** → materialização da solução em código e entrega final.

---

2. Conceitos Fundamentais da Orientação a Objetos

- **Classe** → modelo ou molde que define atributos (dados) e métodos (comportamentos).
- **Objeto** → instância de uma classe, representando um elemento real do sistema.
- **Abstração** → capacidade de focar apenas nos aspectos essenciais de uma entidade, ocultando detalhes irrelevantes.
- **Modularidade** → divisão do sistema em partes menores (módulos) independentes e coesos.
- **Coesão** → grau em que um módulo é especializado em uma única responsabilidade. Quanto maior, melhor.
- **Acoplamento** → grau de dependência entre módulos. Quanto menor, melhor.
- **Herança** → relação entre classes onde uma herda atributos e métodos de outra. Facilita reuso e extensibilidade.
- **Agregação** → relação “tem-um” onde o objeto agregado pode existir independentemente (ex.: carro – motor).
- **Composição** → relação “parte-de” onde o objeto depende do todo para existir (ex.: pedido – item).

---

3. Fases do Projeto OO

1. **Projeto de dados** → definição de modelos de dados a partir dos requisitos.
2. **Projeto arquitetural** → organização das classes em **componentes** e definição da estrutura global.
3. **Projeto de interfaces** → descrição das interações internas (entre módulos) e externas (com usuários ou outros sistemas).
4. **Projeto de componentes** → detalhamento técnico, refinando a arquitetura para implementação prática.

Benefícios da arquitetura de software: - Melhora a comunicação entre os envolvidos.
- Facilita manutenção e evolução do sistema.

- Permite reuso de componentes.
- Reduz tempo e custo de desenvolvimento.

Processo na arquitetura de software: - Pré-projeto: levantamento e análise de requisitos.
- Análise e modelagem: definição das classes e relacionamentos.
- Desenvolvimento: implementação dos módulos definidos.
- Testes: validação das funcionalidades.

Aspectos humanos: - **Gerente** → coordena e garante alinhamento.
- **Arquiteto** → define os caminhos técnicos e estratégicos.

---

4. Modelos de Dados e de Classes

- **Conceitual** → visão abstrata do sistema, sem considerar restrições tecnológicas.
- **Projeto** → detalhamento do modelo conceitual, adicionando aspectos da solução.
- **Implementação** → definição das classes em uma linguagem de programação específica.

Exemplo de classes e relações: - Pessoa (classe geral) → Cliente, Fornecedor (subclasses via herança).
- Pedido → composto por Itens (composição).
- Produto agregado a Estoque (agregação).

---

5. UML (Unified Modeling Language)

A **UML** é a linguagem padrão para modelagem de sistemas orientados a objetos. Ela fornece diagramas que auxiliam na especificação, visualização e documentação.

Principais diagramas: - **Casos de uso** → mostram interações entre usuário e sistema.
- **Classes** → descrevem atributos, métodos e relacionamentos.
- **Objetos** → instâncias específicas de classes em determinado momento.
- **Diagramas de sequência** → mostram a troca de mensagens entre objetos.
- **Diagramas de atividades** → representam fluxos de processos.

Relacionamentos representados em UML: - Associação (ligação entre classes).
- Herança (generalização/especialização).
- Agregação (símbolo de losango branco).
- Composição (símbolo de losango preto).

---

6. Práticas de Modelagem

- **Requisitos típicos**: locais, lotes, movimentações, inventários, produtos/itens.
- **Dicas práticas de modelagem:**
- Começar pelo ponto central (ex.: produtos).
- Mapear o processo de requisitos.
- Simular consultas no banco de dados para validar o modelo.

Exemplo prático de modelagem: - **Controle de estoque** → entidades Produto, Estoque, Fazenda.
- **Gestão de projetos** → entidades Atividades, Recursos, Calendário, Apontamentos.

---

7. Conclusão

O **projeto orientado a objetos** fornece uma base sólida para o desenvolvimento de software, garantindo maior organização, qualidade e capacidade de manutenção. Conceitos como **abstração, modularidade, coesão, acoplamento e reuso** são fundamentais. A utilização de **UML** e boas práticas de modelagem aumenta a clareza e a comunicação entre equipe e stakeholders, reduzindo falhas e custos.

✅Este material consolida os principais pontos das suas anotações em uma **apostila estruturada e completa** sobre Projeto de Sistemas OO.