7

TEXTO BASE

ANÁLISE E MODELAGEM DE SISTEMAS

Faculdade IMPACTA

Texto base

7

O.O. Conceitos

Paradigma Orientado a Objetos

Prof. Renato de Tarso Silva

Resumo

Este texto traz breves conceitos sobre (1) objetos, (2) análise comparativa entre paradigmas de programação, (3) diferenciação de classes, e (4) as principais características de programação orientada a objetos. Também, iniciam-se as definições de classes, elementos essenciais na continuidade dos estudos desta disciplina.

7.1. Início

Um atraente começo para melhor se entender o paradigma orientado a objetos (O.O), e suas vantagens, é criar uma comparação com o paradigma estruturado - conhecido também como procedural - ainda muito encontrado em softwares de mercado, principalmente em sistemas legados. Note o comparativo feito na tabela 7.1.

Orientado a Objetos (P.O.O)	Procedural/Estruturado
Métodos	*Procedimentos e funções*
Instâncias de variáveis	*Variáveis*
Mensagens	*Chamadas a procedimentos e funções*
Classes	*Tipos de dados definidos pelo usuário*
Herança	*Não disponível*
Polimorfismo	*Não disponível*

Tabela 7.1. Procedural x Orientado a Objetos

O paradigma estruturado contém características - nem melhores nem piores que os da programação O.O - que, dependendo do projeto, propiciam ou não seu uso no desenvolvimento. Neste paradigma, o estruturado, há uma escrita de códigos mais "simples", e o fluxo dos algoritmos criado é mais visível e tangível. Um código sendo procedural contém basicamente uma sequência de decisões e iterações. Por ter menor potencial de reuso e manutenção, é impensável sua utilização em sistemas complexos.

O P.O.O. contém linhas de programação mais “limpas” e coesas. Seus fluxos de algoritmos são mais abstraídos, o que aumenta drasticamente a organização e a escalabilidade de aplicações. Seu alto poder de reutilização e manutenibilidade o faz ser atraente em qualquer grandeza de escopo em projetos.

A evolução das linguagens de programação no paradigma O.O pôs em pauta a necessidade de se aperfeiçoar linguagens de modelagem que compreendessem a análise de projetos O.O. A UML é a linguagem mais utilizada e indicada para modelar objetos em projetos de software, pois foi aprimorada, ao longo do tempo, para compreender representações de qualquer conceito fundado no paradigma O.O.

7.2. Conceitos do Paradigma O.O

O objetivo de se trabalhar com objetos é poder levar entidades e conceitos do mundo real, como objetos computacionais manipuláveis em programação. A necessidade, que os precursores do paradigma orientado a objetos tinham, era a de contemplar de maneira coerente a realidade do cotidiano e seus elementos em formas de códigos de programação. Assim, admitindo implementações de sistema mais orgânicos, permitindo se criar entidades como “organismos”: com características, comportamentos e relacionamentos entre si.

Serão abordados 6 (seis) dos mais importantes conceitos O.O., ilustrados na figura 7.2. Duas definições - de classe e de objeto - são essenciais para se continuar o restante dos conceitos, e super disseminados nesta matéria. As outras 4 (quatro) definições: abstração, polimorfismo, herança e encapsulamento, trazem importantes técnicas das mais usuais na dinamização do trabalho com objetos.

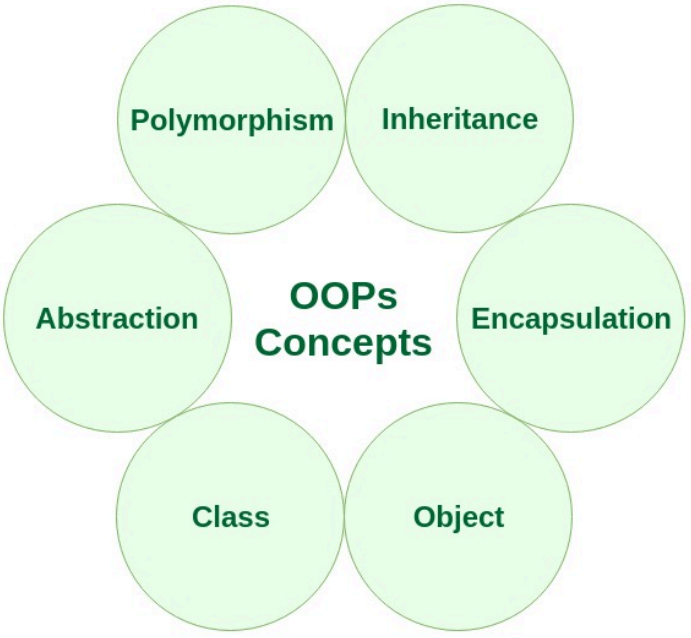

Figura 7.2. Conceitos O.O.

7.2.1. Classe

Uma classe define um conjunto comum e estrutural de dados (atributos) que, além de ser capaz manter associações (relacionamentos) com outras classes, dispõe de serviços (operações / métodos) capazes de processar seus dados. Pode-se dizer que as classes são padrões (formas / moldes / templates) que permitem formar um ou mais objetos que compartilhem os mesmos atributos, operações e relacionamentos. Cada classe contém os tipos de dados necessários para executar operações exigidas por objetos que ela forme; assim, um objeto, para existir na aplicação, sempre parte de uma classe. Veja a figura 7.2.1, em que um molde padroniza a criação de objetos diferentes.

Figura 7.2.1. Classe é um padrão, um molde.

7.2.2. Objeto

Um objeto é um membro de uma classe. Isso quer dizer que seus atributos e métodos são disponibilizados pela estrutura declarada da classe. Um objeto, é uma instância de uma classe, ou seja, um elemento que contém suas individualidades, mas respeita a forma da classe por onde foi instanciado, se comportando de acordo com o que a classe formata para processar seus tipos de dados. A figura 1.2.2 apresenta a classe “professor”, pela qual objetos do tipo professor podem ser instanciados.

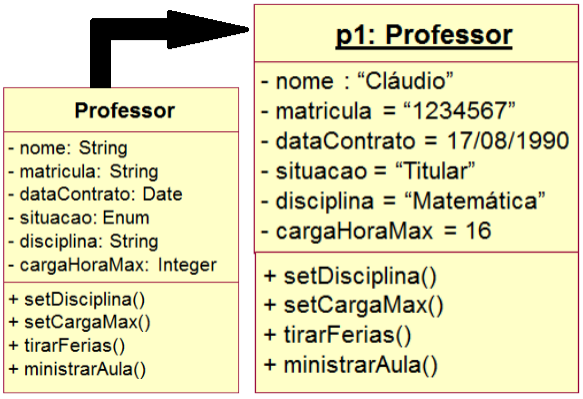

Figura 7.2.2. Objeto P1: Instância da classe Professor.

O entendimento das definições de Classe e Objeto permite o avanço para conceitos mais avançados e de ampla utilidade na orientação a objetos.

7.2.3. Abstração

Quando se diz que algo é abstrato, entende-se que é algo confuso, apenas idealizado ou inteligível. Podem ser conceitos que a maioria das pessoas têm dificuldade de compreender ou alcançar. Então, em O.O., a abstração tem o intuito de eliminar ou reduzir a complicação que determinados conceitos podem apresentar. Através da abstração de objetos, pode-se, por exemplo, concentrar a representação de vários conceitos em apenas um; variar funcionalidades, o nível de abstração e o ponto de vista em um único elemento.

A aplicação de uma abstração depende do observador, ou seja, quem irá consumir recursos do elemento abstraído, permitindo focar em aspectos essenciais ao observador e ignorando aspectos irrelevantes. Assim é exposto o que se faz, e não como se faz. A figura 7.2.3 representa um componente digital que, sem dúvidas, tem alta complexidade em suas funcionalidades, mas contém uma interface frontal, que facilita interações com outros sistemas e viabiliza sua utilização.

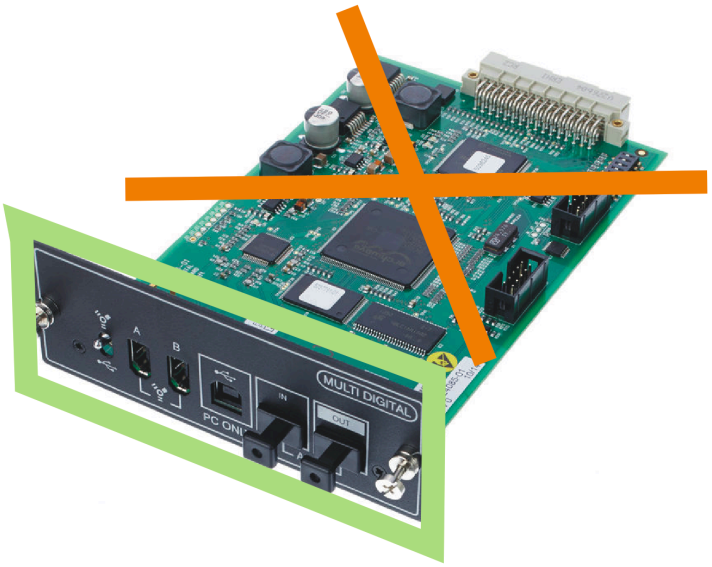

Figura 7.2.3. Uma interface Abstrai funcionalidades

7.2.4. Herança

Assim como na biologia, em O.O., um elemento “Pai” pode repassar seus valores e comportamentos para outros elementos dele oriundos - que herdarão deste “pai” o que ele contiver. Ao se definir classes, é possível herdar recursos (características e métodos) de outra classe, em que uma superclasse é um tipo de classe cujos recursos são herdados (uma classe pai/mãe); e uma subclasse é um tipo de classe que herda (uma classe derivada/filha), mas que pode ter sua estrutura própria e distinta.

Subclasses podem ter várias superclasses, e podem herdar (atributos e métodos) de todas as suas superclasses adjacentes em sua cadeia de herança, assim como um pessoa herda características de seus avós.

A aplicação de herança traz enorme vantagem, pois possibilita um altíssimo reaproveitamento na implementação de funcionalidades. Note a figura 1.2.4, que representa classes mais generalizadas - mais abstraídas - de onde classes mais especializadas podem, delas, herdar coisas. Ex: todo Vendedor é um Empregado, e todo Empregado por sua vez é uma Pessoa. Repare: Cliente também é uma Pessoa, mas não é Empregado, menos ainda um Vendedor; nem todo Empregado tem "Taxa de Comissão", somente Vendedor. Assim, nota-se que cada subclasse pode ter sua particularidade e, ainda assim, herdar de superclasses em comum.

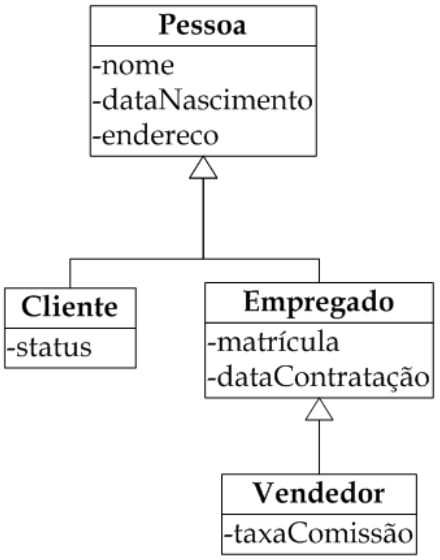

Figura 7.2.4. Classes herdam de outras classes

7.2.5. Polimorfismo

É dos recursos mais utilizados em O.O., e vem do latim poli(várias)+morfis(formas), também conhecido como "sobrescrita de operação". É um recurso para fazer mutações de comportamentos (capacidade de se adaptar), criando operações polimórficas para uma mesma assinatura (nome de método/operação). Apesar de terem a mesma assinatura, uma operação da subclasse redefine sua implementação para comportar-se diferentemente.

Assim, subclasses podem criar interfaces, pois contêm métodos comuns a vários solicitantes diferentes que usem a mesma operação. O solicitante não precisa saber qual classe implementa sua funcionalidade e a mesma assinatura pode resultar distintos processamentos. Note a figura 7.2.5, a classe Vendedor herda, e sempre utiliza, o método "obterSalárioBase()" da classe Funcionário, mas a operação "*obterPagamento*()" da classe Vendedor precisa responder um cálculo diferenciado, com cálculo de comissão. Assim, utilizará o próprio método e não "*obterPagamento*()" da classe funcionário, sobrescrevendo o método da superclasse.

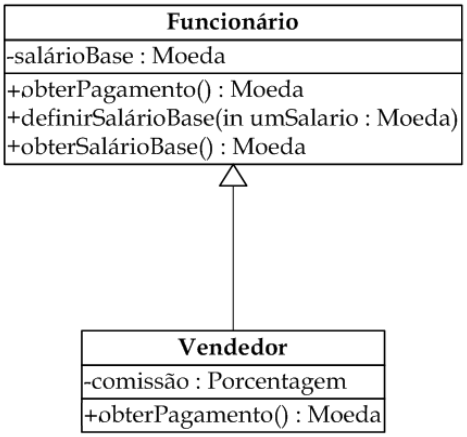

Figura 7.2.5. Polimorfismo - Sobrescrita de Operação

7.2.6. Encapsulamento

É o poder de esconder, ou proteger, as propriedades e as operações de uma classe. Assim, um objeto desta classe pode disponibilizar o que ele faz sem expor como ele faz. A técnica protege a estrutura de uma classe das outras classes, por estar apenas internamente acessíveis e implementadas. Ao encapsular partes de uma classe, os objetos terão total controle sobre tais partes. Por exemplo, em um automóvel, somente através de um painel/ignição pode-se ligar o seu motor; logo, encapsulamento é proteção e segurança em um ambiente O.O.

As visibilidades das propriedades das classes são o que permitem aplicar um encapsulamento. Note na figura 7.2.6, todos atributos na classe Professor são privados(-), enquanto os métodos de Professor são todos públicos(+). O sinal de privado(-), posto antes de uma parte (atributo/método), diz que aquela parte somente é acessível pela própria classe Professor, e quando há sinal de público(+), permite aquela parte ser acessível por qualquer outra classe.

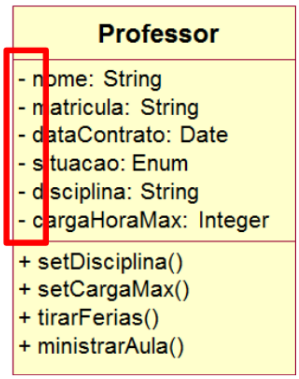

Figura 7.2.6. Encapsulamento - Visibilidade

Os conceitos de O.O. abordados são algumas das ferramentas mais poderosas existentes no paradigma para lidar com as dificuldades de implementação de sistemas complexos que representam problemas do mundo real com suas peculiaridades.

Referências

Booch G., Rumbaugh J., Jacobson I (2005). “The Unified Modeling Language user Guide” 2ª Edição.