

Introdução a Programação Orientada a Objetos



Conceudista: Prof. Esp. Alexander Gobbato P. Albuquerque

Revisão Textual: Prof.^a M.^a Rosemary Toffoli

Objetivos da Unidade:

- Estudar os conceitos básicos para a criação e entendimento dos conceitos da orientação;
- Mostrar a diferença entre programação estruturada e a programação orientada a objetos.

Contextualização

Material Teórico

Material Complementar

Referências



Contextualização

Um paradigma de programação fornece (e determina) a visão que o programador possui sobre a estruturação e execução do programa. Assim como ao resolver um problema podemos adotar uma entre variadas metodologias, ao criar um programa podemos adotar um determinado paradigma de programação para desenvolvê-lo.

Os sistemas projetados atualmente são maiores, mais complexos e sujeitos a constantes alterações e adaptações aos diversos ambientes computacionais. Através do **encapsulamento** de informações, a **reutilização** de esforços empregados em projetos anteriores e a modificação do sistema se tornaram mais fáceis.

Material Teórico

Introdução

A metodologia Orientação a Objetos é baseada em “objetos do mundo real”, e por este motivo é mais intuitiva ao ser humano, tais como: objetos e atributos, classes e membros, estruturas e componentes, ação e reação. Os métodos de desenvolvimento de *software* anteriores ao surgimento desse paradigma organizam a especificação de um sistema de acordo com suas funções ou com os dados manipulados. Geralmente, esses métodos apresentam dificuldades na transição da representação do sistema em uma fase para outra do processo de desenvolvimento (da Análise para o Projeto e, do Projeto para a Implementação).

Em um sistema orientado a objetos, os dados e todas as operações, que manipulam esses dados, são agrupados em uma única estrutura: “as classes”. Desde o início do desenvolvimento desses sistemas e, em todas as suas fases, o analista trabalha com o mesmo elemento de abstração, os objetos.

Estudo das técnicas de programação utilizando o paradigma da orientação a objetos, bem como ferramentas em laboratório para o desenvolvimento de aplicações científicas e comerciais.

Importância do Software

Dependência generalizada da vida moderna em sistemas de computação.

O *software* é parte cada vez maior dos custos e do sucesso desses sistemas.

Producir *software* é uma atividade inherentemente complexa:

- Independente de leis físicas, restrições de materiais e processos de manufatura;
- Requer métodos disciplinados de desenvolvimento.

Histórico

A orientação a objetos (OO) foi utilizada primeiramente na década de 70.

A Origem na linguagem SIMULA-67 (década de 60 – Noruega), que já implementava alguns conceitos da Orientação a Objetos.

SIMULA-68 foi a primeira linguagem a suportar os conceitos da Orientação a Objetos.

Smalltalk, criada pela Xerox, popularizou e incentivou a Orientação a Objetos.

Outras linguagens de Orientação a Objetos: C++, Object Pascal (Delphi), C#, Java.

Java, de fato, popularizou a Orientação a Objetos.

Programa Estruturada versus Programa Orientado a Objetos

- **Estruturado:** Ênfase nos procedimentos, implementados em blocos estruturados, com comunicação entre procedimentos por passagem de dados;
- **Orientado a objetos:** Dados e procedimentos fazem parte de um só elemento básico (objeto). Os elementos básicos comunicam-se entre si, caracterizando a execução do programa à Dados e procedimentos agrupados em um só elemento.

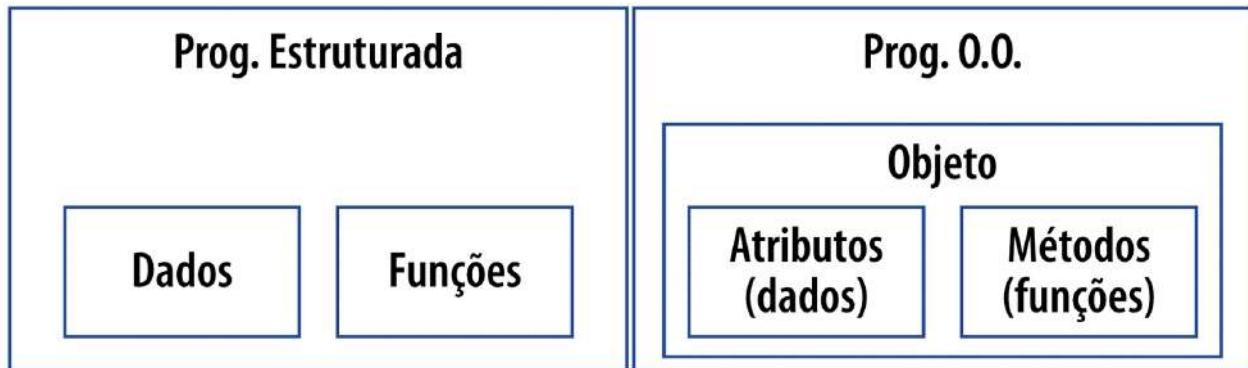


Figura 1

Por que a Tecnologia de Objetos?

Os principais problemas do software hoje são:

- Diminuir o custo e o tempo da mudança;
- Aumentar a capacidade e facilidade de adaptação.

Objetos são especialmente bons para:

- Reduzir o tempo necessário para adaptar um sistema existente (reação mais rápida à mudanças no seu ambiente de negócio);
- Reduzir o esforço, a complexidade e os custos associados à mudança.

Orientação a Objetos – Fase e Responsabilidade

- **Análise:**
 - Entender o domínio do problema;
 - Definir a estrutura estática da aplicação;
 - Criar vocabulário comum de conceitos.
- **Projeto:**
 - Apresentar a melhor solução para problema;
 - Definir a estrutura dinâmica da aplicação;
 - Fazer especificação.
- **Implementação:**
 - Construir a aplicação em conformidade com a especificação;
 - Fazer testes e validações;
 - Fazer implantação.

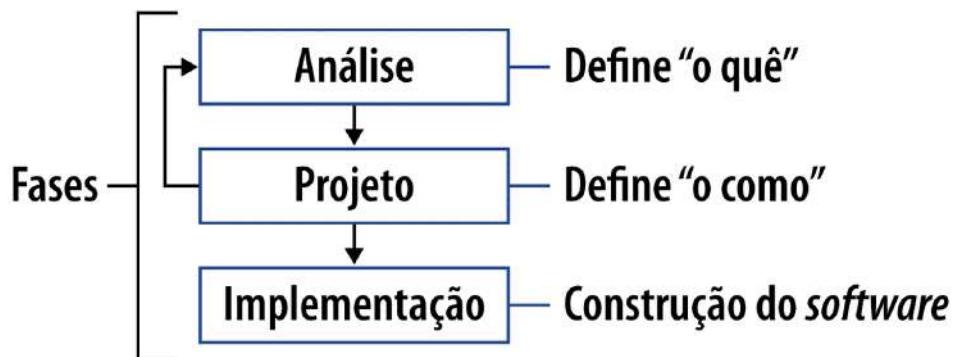


Figura 2

Conceito da Análise e Projeto Orientado a Objetos

A metodologia de Análise e Projeto Orientado a Objetos é um processo que contém suporte a notação para definição de *software* baseado no conceito de objetos que combinam estrutura e comportamento na mesma entidade.

Este processo geralmente não é linear, mas é previsível, cíclico, e passível de teste e rastreamento.

A notação *OOA&D* (*Object-oriented Analysis and Design*) é definição visual comum que permite as pessoas compartilharem conhecimento sobre sistema – *OOA&D* consiste em três partes:

- Processo (São as atividades);
- Notação (demonstra a interação dos objetos);
- Regras (descreve como o sistema deve trabalhar).

Principais Conceitos

As duas principais características da Orientação a objetos são Classes e Objetos.

O que é uma classe?

Saiba Mais

Classe – [Do lat. *classe*.] S. f. 1.: Numa série ou num conjunto, grupo ou divisão que apresenta características semelhantes;

21. Inform. Em programação ou modelagem orientada a objetos (v. orientação a objetos), categoria descritiva geral, que abrange o conjunto de objetos que compartilham uma ou mais características quanto a seus itens de dados e procedimentos associados. 22. Lóg. Agrupamento de objetos que têm uma ou mais características em comum.

“Uma **classe** é uma entidade descreve um conjunto de objetos com propriedades e comportamentos semelhantes e com relacionamentos comuns com outros objetos”.

As classes são as partes mais importantes de qualquer sistema orientado a objetos.

Usamos as **classes para capturar o vocabulário do sistema** que está em desenvolvimento. Essas classes podem incluir abstrações que são parte do domínio do problema, assim como as classes que fazem uma implementação. Podemos usar ainda as classes para representar itens de *software*, de *hardware* e até itens que sejam somente conceituais.

Contexto: Venda de flores.

Chris precisa enviar flores para o amigo Robin que mora em outra cidade.

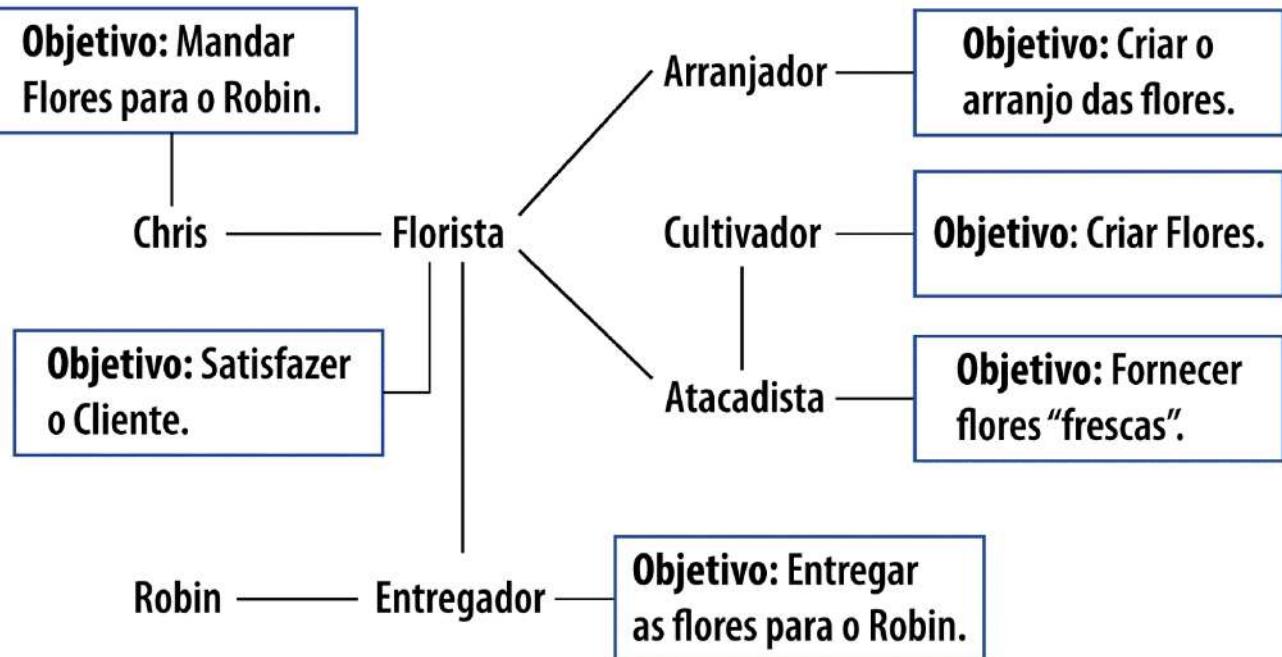


Figura 3 – Venda de Flores

Fonte: Adaptada de BUDD, 2002

Agentes e Comunidades

- Chris passa uma mensagem para o florista contendo uma requisição (entregar flores para Robin em outra cidade);
- É responsabilidade do florista satisfazer esta requisição;
- Chris não sabe o método que o florista irá utilizar para entregar flores em outra cidade e também desconhece os detalhes desta operação.
 - POO é estruturada como uma comunidade de agentes que interagem chamados de objetos;
 - Cada objeto tem um papel a ser executado;
 - Cada objeto fornece serviços ou executam ações que podem ser utilizadas por outros membros da comunidade.



Figura 4

Fonte: Adaptada de BUDD, 2002

Mensagens e Métodos

- Uma ação é a transmissão de uma mensagem para um agente (objeto) responsável pela ação;
- Quando um objeto aceita a mensagem ele é responsável por tratar a ação;
- Em resposta a esta mensagem o receptor executará algum método para atender a solicitação.

Responsabilidades

- Descreve o comportamento em termos de responsabilidades;
- O conjunto de responsabilidades de um objeto é descrito através e protocolos

Classes e Instâncias

- O Florista é um exemplo de classe, pois representa todos os floristas. O Florista Pitoco, e o Bodjo são instâncias desta classe;
- Um outro exemplo de classes e instâncias é forma de gelatina e as gelatinas.

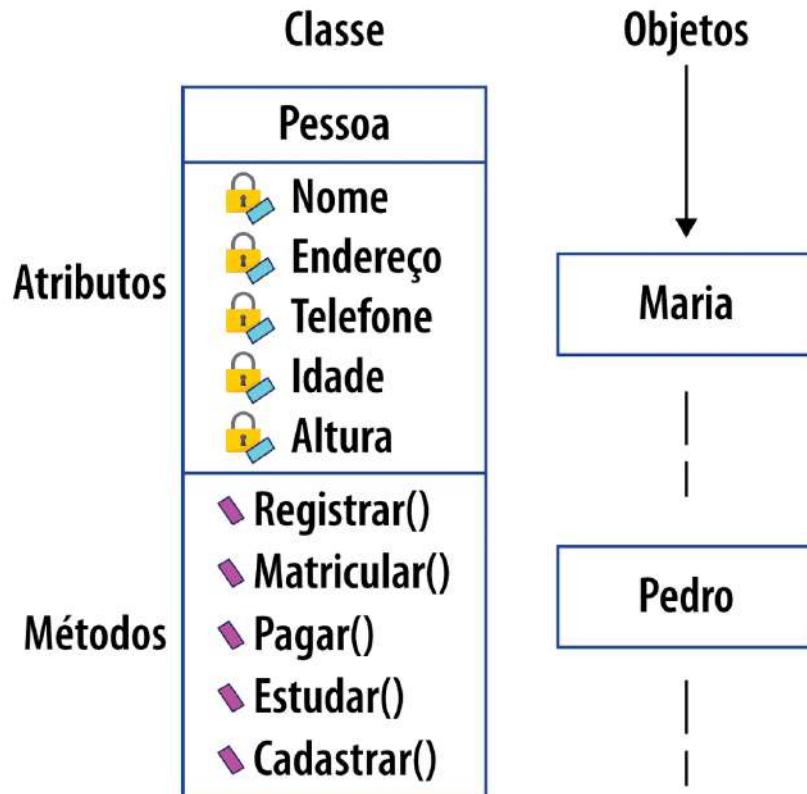


Figura 5

Notação UML para Classe e Objetos

A identificação para as classes em UML é um retângulo dividido em três partes. Caso não queira exibir os detalhes da classe você pode suprimir alguns detalhes como atributo e os métodos, conforme mostram as Figuras a seguir:

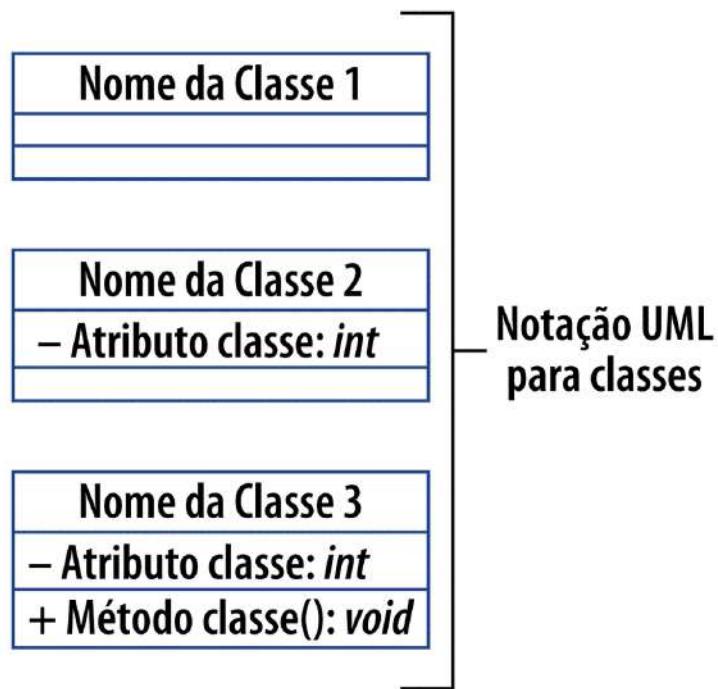


Figura 6

A seção superior é reservada para a identificação da classe. Essa seção é obrigatório o preenchimento. O nome da classe deve conter somente letras e palavras concatenadas e a primeira letra deve ser escrita em maiúsculo.

Exemplo de nomes:

- Aluno;
- Empregado;
- Curso.

Na segunda seção são exibidos os atributos da classe, esses atributos são variáveis membro da classe que podem ser usadas pelos métodos tanto para acesso ou escrita.

O atributo pode ser criado com as seguintes notações:

- + Acesso público, tanto funções membros da classe e funções externas podem ter acesso ao atributo público;
- - Acesso privado, somente funções membros da classe tem acesso a esses atributos;

- # Acesso protegido, o atributo também é privado, mas as funções membros das classes derivadas também tem acesso aos atributos.

Na última seção são exibidos os métodos da classe, ou seja, as funções que a classe possui.

Os métodos também seguem a mesma notação que os atributos.

- + Acesso público, tanto funções membros da classe e funções externas podem ter acesso ao atributo público;
- - Acesso privado, somente funções membros da classe tem acesso a esses atributos;
- # Acesso protegido, o atributo também é privado mas as funções membros das classes derivadas também tem acesso aos atributos.



Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Livros

Aprenda Programação Orientada a Objetos em 21 Dias

SINTES, A. *Aprenda Programação Orientada a Objetos em 21 Dias*. Editora: Pearson. 1. ed.

Java: Como Programar

DEITEL, P. J.; DEITEL, H. M. *Java: Como Programar*. Editora: Pearson. 8. ed.

Core Java – Volume 1

HORSTMAN, C. S.; CORNELL, G. *Core Java: Volume 1*. Editora: Pearson.

Leitura

Orientação a Objetos – Simples Assim!

Clique no botão para conferir o conteúdo.

ACESSE

 Referências

DEITEL, H. M. *Java: Como Programar*. 6. ed. Porto Alegre: Bookman, 2007.

GAMMA, E. et al. *Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos*. Porto Alegre: Bookman, 2007.