# UMA CONCEITUAÇÃO DIDÁTICA SOBRE ORIENTAÇÃO A OBJETOS

**REIS, Davi Silvestre Moreira dos, Especialista***

* Faculdade de Tecnologia de Praia Grande
Departamento de Informática para Gestão de Negócios
Pça. 19 de Janeiro, 144, Boqueirão, Praia Grande / SP, CEP: 11700-100
Fone (13) 3591-1303
davismdosreis@yahoo.com.br

## RESUMO

Este breve artigo predispõe-se a apresentar, de forma clara e sucinta, o conceito do paradigma da Orientação a Objetos, em voga atualmente em todas as etapas do desenvolvimento de um sistema, desde seu início, no levantamento de requisitos, até o final da fase de programação – ou codificação –, quando é enfatizado, passando pela análise e modelagem dos dados e atualmente sendo encontrado, inclusive, na própria persistência dos dados, sejam estes inseridos ou processados, que é o banco de dados. Este pequeno ensaio objetivará apresentar aos iniciantes em programação, e aos desconhecedores da Orientação a Objetos, uma abordagem didática e simples, visando facilitar a compreensão e o aprendizado desta metodologia de desenvolvimento tão solidificada e difundida nos dias de hoje.

**PALAVRAS-CHAVE:** Orientação a Objetos, Polimorfismo, Herança, Encapsulamento.

## *ABSTRACT*

*This brief article is to present, clearly and succinctly, the concept of the paradigm of the Objects Oriented, en vogue currently in all the stages of the development of a system, since its beginning, in the survey of requirements, until the end of the phase of programming - or codification -, when it is emphasized, passing by the analysis and modeling of the data and currently being found, also, in the proper*

*persistence of the data, inserted or processed, which are the database. This small essay will objectify to present to the beginners in programming, and the ones that don't know about Objects Oriented, a didactic and simple boarding, aiming at facilitating the understanding and the learning of this methodology of development so solid and spread out nowadays.*

***KEY-WORDS:*** *Objects-Oriented, Polymorphism, Inheritance, Encapsulation.*

## INTRODUÇÃO

Um dos objetivos do paradigma da orientação a objetos é tentar representar, através de programas, ou seja, no mundo do software, objetos supostamente existentes no mundo real, utilizando, para tanto, uma característica não encontrada nos modelos de desenvolvimento tradicionais: a incorporação de "ações" e "dados" por meio de um mesmo objeto. Como afirma Ingals (1981): "nós iremos poupar tempo se tornarmos nossos computadores compatíveis com a mente, ao invés do caminho contrário". Além disso, a orientação a objetos está baseada na elaboração de partes independentes que, unidas, formam um todo, sendo que se uma dessas "partes" (que seriam os objetos) for danificada, ela poderá facilmente ser reparada, sem causar dano para o "todo" (que seria o sistema). Os objetos podem ser reutilizados para compor outros sistemas, e podem ter seus códigos recombinados, o que permite a criação de novos objetos, facilitando e diminuindo o trabalho de desenvolvimento.

## 1 ORIGEM

As opiniões de alguns autores divergem sobre a origem exata do conceito de Orientação a Objetos. Por exemplo, de acordo com Booch (1996, p.4):

> O uso da orientação a objeto como metodologia básica para o desenvolvimento de sistemas abrangendo todo o ciclo, desde a análise até a construção de códigos, é uma prática bem recente. Apenas na década de 80 é que surgiram os primeiros estudos sobre o uso da Orientação a Objeto para especificação em projetos de sistemas.

Porém, outros autores compartilham a opinião de que esse conceito bem-sucedido de desenvolvimento de sistemas de informática tenha surgido há mais tempo, entre o final da década de 60, através do surgimento da linguagem Simula, na Noruega, e principalmente em meados da década de 70, quando do desenvolvimento da linguagem Smalltalk, que seria a primeira totalmente orientada a objetos (INGALLS, 1981).

É ponto comum, entretanto, considerar a linguagem Simula e, mais efetivamente, a linguagem Smalltalk, como origem do conceito de Orientação a Objetos. Mas, enfim, quais seriam esses conceitos?

A seguir, observaremos quais são esses importantes conceitos que compõem o paradigma da orientação a objetos.

## 2 CONCEITOS

Também encontram-se algumas divergências de opiniões com relação à quantidade, e quais seriam, os conceitos básicos que integram e dão suporte, ou melhor, "dão vida" à Orientação a Objetos.

De acordo com Cantú (1998), por exemplo, a Orientação a Objetos é baseada em três conceitos fundamentais: classes, herança e polimorfismo.

Já segundo Ambler (1998), os conceitos fundamentais são em maior número: classes, objetos, instância, atributos, métodos, abstração, encapsulamento, herança, persistência, relacionamento entre instâncias, acoplamento, entre outros.

De modo geral, podemos dizer que os conceitos fundamentais, dos quais surgem os demais, são: abstração, classe, objeto, encapsulamento, herança, polimorfismo e identidade.

## 2.1 Abstração

Por uma conceituação simplista, pode-se dizer que abstração é a capacidade natural do ser humano utilizada para tratar de situações complexas, tentando "abstrair" semelhanças dessas situações ou processos. É o reconhecimento de similaridades entre objetos, situações e processos do mundo real, concentrando os estudos nessas similaridades, nesses aspectos essenciais, e ignorando, apenas nesse momento, suas diferenças, ou as características menos importantes.

Um mesmo problema pode ter abstrações diferentes. Isso está diretamente relacionado com a perspectiva de quem o observa.

Toma-se como exemplo uma criança.


**Figura 1 – Perspectiva de observação**

De acordo com a figura 1, em casa, para sua mãe, a criança é tida como um filho; já no ambiente escolar, é vista como um aluno. Porém, o ponto comum que podemos enxergar em ambos os contextos é que se trata de uma "criança". Vejamos que foram feitas observações diferentes ("filho" e "aluno") para um mesmo objeto de estudo ("criança"). Portanto, podemos dizer que "filho" e "aluno" são objetos e a abstração que se dá entre eles resulta em "criança". Uma outra abstração, de acordo com as ilustrações, poderia resultar em "menina".

## 2.2 Classe

A classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos – que serão criados a partir dela –, através de métodos (ações), e quais estados eles serão capazes de manter, através de atributos (propriedades). Para se diferenciar entre métodos e atributos, pode-se dizer que métodos são definidos como verbos (i.e. "abrir", "fechar", "obter") e atributos são

definidos como substantivos ou adjetivos (i.e. “nome”, “largura”, “estado”).

De uma forma possivelmente mais clara, classe é o objeto abstrato (como no item anterior) construído computacionalmente. Quando o programa contendo a classe é executado, essa classe é utilizada para “dar vida” ao objeto (que será nosso próximo tópico de estudo). A classe é construída uma única vez, em um determinado trecho do código, e, a partir de então, são declaradas variáveis dessa classe, contendo todas suas características. Essas variáveis serão instâncias dessa classe, e a essas instâncias damos o nome de objetos, como veremos mais adiante. De uma forma mais prática, é comum dizermos que “criamos classes” e, a partir daí, “instanciamos objetos”.

Em suma, pode-se dizer que a classe seria uma forma, um molde, e a partir desse molde construímos diversos objetos.

Há ainda, um outro conceito dentro de classes: a de classe-mãe (ou superclasse) e de classe-filha (subclasse ou classe derivada). A superclasse é hierarquicamente superior à(s) subclasse(s). Estas, aliás, são criadas a partir de uma superclasse, “ganhando” (ou melhor, herdando, como se verá mais adiante) todos seus métodos e atributos. As subclasses podem, ainda, implementar novos métodos e atributos que não existiam inicialmente na superclasse.

Utilizando-se ainda o exemplo ilustrativo da figura 01, podemos dizer que uma superclasse seria “criança” e duas classes derivadas (subclasses) seriam “filho” e “aluno”. Estas herdariam todos os métodos (i.e. “brincar”, “comer”, “correr”, “dormir”, “sorrir”) e atributos (i.e. “nome”, “idade”, “sexo”, “data de nascimento”) da superclasse “criança” e ainda implementariam mais métodos e atributos de acordo com suas necessidades. A subclasse “filho”, por exemplo, poderia possuir os atributos “pai” e “mãe”, para armazenar os nomes de seus pais, enquanto a subclasse “aluno” poderia ter o método “estudar”, para dizer de uma ação que ela promove diferentemente de “filho” e “criança”.

## 2.3 Objetivo

Sinteticamente falando a partir do item anterior, podemos dizer que um objeto uma instância de uma classe. Um objeto é capaz de

armazenar estados através de seus atributos e reagir a mensagens enviadas a ele; assim como se relacionar e enviar mensagens a outros objetos. Objeto, por definição, pode ser considerado tudo que existe. Pode ser uma pessoa, um carro ou mesmo uma janela.

Tomando-se novamente o exemplo da figura 01, podemos "instanciar" vários objetos a partir da subclasse "filho" e outros a partir da subclasse "aluno". Podemos, ainda, instanciar outros objetos a partir da classe-mãe ("criança"). Cada um desses objetos teria todas as características e comportamentos da classe que o originou. Poderíamos, pois, ter três objetos a partir de "filho" para representar os três filhos de um casal, enquanto poderíamos ter vinte objetos criados a partir de "aluno" para representar os alunos de uma sala de aula. Quanto aos objetos criados a partir de "criança", pode-se dizer que estes teriam sido criados para representar outras crianças que não fossem, nem filhos de um possível casal em questão, e nem alunos da sala de aula supostamente exemplificada.

Posto isto, pode-se observar que todos os objetos possuem "nome" (atributo herdado da superclasse "criança"), mas apenas os instanciados a partir de "filho" armazenariam as informações sobre seus pais, enquanto somente os objetos instanciados a partir de "aluno" promoveriam a ação "estudar".

## 2.4 Encapsulamento

Um dos conceitos básicos por trás da orientação a objetos é o encapsulamento.

O Encapsulamento consiste na separação de aspectos internos e externos de um objeto.

Este mecanismo é amplamente utilizado como proteção, para impedir o acesso direto ao estado de um objeto (seus atributos) ou mesmo alguns métodos, disponibilizando externamente apenas os métodos necessários para codificação que alteram estes estados. Ele também permite acesso a qualquer tipo de comunicação com o objeto através, por exemplo, de mensagens.

Recorrendo mais uma vez ao exemplo da figura 01, podemos raciocinar que um atributo que poderia ser "encapsulado" de forma a ninguém ter acesso poderia ser, por exemplo, o "tamanho do pé", que

armazenaria informações sobre o número que a criança calça. Podemos observar que essa informação, apesar de importante, não precisa ficar exposta a todos que acessarem os objetos instanciados a partir das classes "criança", "filho" e "aluno". Essa informação (número do calçado que a criança utiliza) provavelmente deveria existir, mas seria mais conveniente que sua manipulação e seu processamento ficassem restritos internamente à codificação das classes (ou da classe-mãe, que provavelmente é o lugar onde o atributo que armazena tal informação foi criado).

## 2.5. Herança

De acordo com Furlan (1998, p.12), a herança: "É o mecanismo de reutilização de atributos e operações definidos em classes gerais para classes mais específicas, podendo ser usado para expressar tanto generalização como associação." Ou, numa definição um pouco mais abrangente de Coad (1992, p.26), podemos dizer que herança "é o compartilhamento de atributos e operações entre classes, baseado em um relacionamento hierárquico. Uma classe pode ser definida de maneira abrangente, e depois ser refinada em subclasses. Os elementos de uma superclasse não precisam ser repetidos em suas subclasses, que automaticamente herdam estes elementos. A reutilização que isto proporciona é uma das principais características da orientação a objetos".

Em outras palavras, pode-se dizer que herança é o mecanismo que permite criar uma nova classe aproveitando os métodos e atributos de outra classe. Quando uma classe "B" é criada a partir de uma classe "A", ela recebe, "herda", seus métodos e atributos. "B" automaticamente passa a ter todos os métodos e as propriedades de "A". Há, ainda, a idéia de herança múltipla, quando uma subclasse possui mais de uma superclasse.

Vale ressaltar uma interessante observação: a classe que herda pode definir outros novos atributos e métodos, mas não pode redefinir atributos e métodos que ela tenha herdado. Os métodos redefinidos na subclasse podem ter número e tipos dos parâmetros diferentes do método correspondente da superclasse.

Ainda tomando como exemplo a figura 01 anterior, podemos facilmente enxergar que a classe "criança" seria a superclasse, enquanto

as classes “filho” e “aluno” seriam as subclasses criadas a partir da superclasse “criança”. Como já citado, estas herdariam todos os métodos (p.e. “brincar”, “comer”, “correr”, “dormir”, “sorrir”) e atributos (i.e. “nome”, “idade”, “sexo”, “data de nascimento”) da superclasse “criança”. E por mais que “filho” e “aluno” implementassem métodos e atributos que necessitassem, elas não teriam o poder ter alterar os métodos (i.e. “brincar”, “comer”, etc) e atributos herdados (i.e. “nome”, “idade”, etc).

## 2.6 Polimorfismo

De acordo com Ferreira e Jarabeck (1991), um exemplo bem didático para o polimorfismo é dado por um simples moedor de carne. Esse equipamento tem a função de moer carne, produzindo carne moída para fazer bolinhos. Desse modo, não importa o tipo (classe) de carne alimentada; o resultado será sempre carne moída, não importa se de boi, de frango ou de qualquer outro tipo. As restrições impostas pelo processo estão no próprio objeto, definidas pelo seu fabricante e não pelo usuário do produto.

O termo polimorfismo é utilizado em biologia para definir variações em forma e função de membros de uma mesma espécie. Por analogia, podemos dizer que “polimorfismo”, em Orientação a Objetos, é a capacidade de permitir tratar objetos semelhantes de uma maneira uniforme; é a capacidade de permitir a um objeto se comportar de acordo com sua classe. Pode, ainda, ser definido como sendo a capacidade que objetos diferentes têm de reagirem, segundo a sua função, a uma mesma ordem padrão. O comando “abre”, por exemplo, faz um objeto entrar em ação, seja ele uma janela, uma porta ou uma tampa de garrafa. Para todos ocorrerá uma abertura, mas essa abertura se dará de modo diferente, de acordo com cada objeto. Cabe, aqui, uma observação importante: o polimorfismo, para ser implementado exige a utilização do conceito de herança e aplica-se apenas aos métodos da classe.

Novamente tomando por base a figura 01, podemos imaginar que o método “entrar”, quando dito a um objeto instanciado a partir de “filho” possa se referir ao ato de entrar para casa, para tomar banho, jantar e dormir, por exemplo; já quando nos referimos “entrar” para um objeto instanciado de “aluno”, podemos dizer que ele adentrará a sala

de aula para estudar e aprender. O nome do método, "entrar", é o mesmo, mas as ações a serem tomadas serão um pouco diferentes, se comportando de acordo com a classe a qual pertencer.

### 2.7 Identidade

Apesar de todos os conceitos anteriores serem relativamente de fácil compreensão, o conceito de "identidade" talvez seja o mais simples de ser explicado, exemplificado e compreendido. Identidade é a característica que cada objeto tem de ser único, de possuir uma única identificação e uma única forma de ser representado, chamado ou acessado dentro de todo o sistema, mesmo sendo criado a partir de uma classe que deu origem a outros tantos objetos – nesse caso, que é muito comum, cada objeto é único dentro do sistema.

E, como cada objeto tem sua própria identidade, mesmo tendo sendo criados a partir de uma mesma classe e mesmo que seus atributos e seus métodos sejam idênticos, dois ou mais objetos existentes no sistema serão sempre diferentes.

Tomando outra vez o conceito ilustrativo da figura 01, mesmo que tenhamos três objetos instanciados a partir da classe "filho", e, portanto, mesmo que esses objetos tenham os mesmos atributos e métodos, cada objeto "filho" será diferente e único no sistema, como o são na vida real.

## CONCLUSÃO

Pode-se observar que o paradigma da Orientação a Objetos é, conceitualmente, muito simples de ser compreendido. Há, contudo, observações e detalhes importantes aos quais devemos nos ater no momento de desenvolver um sistema utilizando-se essa metodologia.

É fácil visualizar a grande ambição da metodologia de Orientação a Objetos: trazer para o mundo do *software* determinadas formas de enxergar certos "objetos", seus comportamentos, situações e propriedades, que são, originalmente, inerentes ao nosso mundo real, passível de compreensão a todas as pessoas.

Trata-se de uma metodologia muito interessante, eficaz e eficiente para desenvolvimento de sistemas, tão utilizada atualmente que é fácil vislumbrarmos sua empregabilidade não mais restrita à informática, como fora concebida, mas ampliando seu campo de atuação para outras áreas de desenvolvimento de projetos, bastando, para tanto, que haja a possibilidade de se utilizar uma abordagem voltada à construção de regras ou moldes padrões, como as "classes" para que, a partir dessas, sejam criados os "objetos", que serão manipulados, trabalhados e que, enfim, realizarão suas ações.

## REFERÊNCIAS BIBLIOGRÁFICAS


AMBLER, Scott W. **Análise e projeto orientados a objeto.** Rio de Janeiro: IBPI Press, 1998.

BOOCH, G. ***Software architecture and the UML.*** 1999. (apresentação).

AMBLER, Scott W; JACOBSON, I.; RUMBAUGH, J. ***UML*** *– unified modeling language version 1.1 – rational*. September 1996.

CANTÚ, Marco. **Dominando o delphi 4.** São Paulo: Makron Books, 1998.

COAD, P.; YOURDON, E. **Análise baseada em objetos.** Rio de Janeiro: Campus, 1992.

FERREIRA, Marcelo; JARABECK, Flávio. **Programação orientada ao objeto com clipper 5.0.** São Paulo: Makron Books, 1991.

FURLAN, Jose David. **Modelagem de objetos através da UML.** São Paulo: Makron Books, 1998.

INGALLS, Daniel H. H. ***Design principles behind smalltalk*****.** BYTE Magazine, August 1981 (Online). Disponível em: <http://users.ipa.net/%7edwighth/smalltalk/byte_aug81/design_principles_behind_smalltalk.html>. Acesso em: 14/08/2006.

LARMAN, Craig. **Utilizando UML e padrões:** uma introdução à análise orientada a objetos. Porto Alegre: Bookman, 2000.

MARTIN, James; ODELL, James J. **Análise e projeto orientados a objeto.** São Paulo: Makron Books, 1996.

PRESSMAN, Roger S. **Engenharia de *software*.** São Paulo: Makron Books, 1995.

RUMBAUGH, J.; Blaha, Michael; PREMERLANI, William; EDDY, Frederick; LORENSEN, William. **Modelagem e projetos baseados em objetos.** Rio de Janeiro: Campus, 1997.